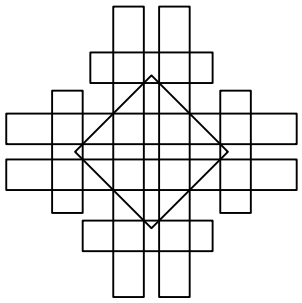
41st International Symposium on Computational Geometry

SoCG 2025, June 23–27, 2025, Kanazawa, Japan

Edited by

Oswin Aichholzer

Haitao Wang



LIPICS

Editors

Oswin Aichholzer
Graz University of Technology, Austria
oswin.aichholzer@tugraz.at

Haitao Wang
University of Utah, Salt Lake City, Utah, USA
haitao.wang@utah.edu

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

ISBN 978-3-95977-370-6

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/978-3-95977-370-6.

Publication date
June, 2025

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication 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.2025.0

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)
- Sławomir Lasota (University of Warsaw, PL)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN – Chair)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Pierre Senellart (ENS, Université PSL, Paris, France)
- Alexandra Silva (Cornell University, Ithaca, US)

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Oswin Aichholzer and Haitao Wang 0:xi–0:xii

Conference Organization
..... 0:xiii–0:xv

Additional Reviewers
..... 0:xvii–0:xxii

Regular Papers

Reconfiguration of Unit Squares and Disks: PSPACE-Hardness in Simple Settings
Mikkel Abrahamsen, Kevin Buchin, Maike Buchin, Linda Kleist, Maarten Löffler, Lena Schlipf, André Schulz, and Jack Stade 1:1–1:18

The Maximum Number of Digons Formed by Pairwise Intersecting Pseudocircles
Eyal Ackerman, Gábor Damásdi, Balázs Keszegh, Rom Pinchasi, and Rebeka Raffay 2:1–2:14

Convexity Helps Iterated Search in 3D
Peyman Afshani, Yakov Nekrich, and Frank Staals 3:1–3:14

A Subquadratic Algorithm for Computing the L_1-Distance Between Two Terrains
Pankaj K. Agarwal, Boris Aronov, Olivier Devillers, Christian Knauer, and Guillaume Moroz 4:1–4:17

Optimal Motion Planning for Two Square Robots in a Rectilinear Environment
Pankaj K. Agarwal, Mark de Berg, Benjamin Holmgren, Alex Steiger, and Martijn Struijs 5:1–5:17

A Sparse Multicover Bifiltration of Linear Size
Ángel Javier Alonso 6:1–6:18

Single-Source Shortest Path Problem in Weighted Disk Graphs
Shinwoo An, Eunjin Oh, and Jie Xue 7:1–7:15

Rapid Mixing of the Flip Chain over Non-Crossing Spanning Trees
Konrad Anand, Weiming Feng, Graham Freifeld, Heng Guo, Mark Jerrum, and Jiaheng Wang 8:1–8:18

Geometric Realizations of Dichotomous Ordinal Graphs
Patrizio Angelini, Sabine Cornelsen, Carolina Haase, Michael Hoffmann, Eleni Katsanou, Fabrizio Montecchiani, Raphael Steiner, and Antonios Symvonis 9:1–9:16

Steinhaus Filtration and Stable Paths in the Mapper
Dustin L. Arendt, Matthew Broussard, Bala Krishnamoorthy, Nathaniel Saul, and Amber Thrall 10:1–10:21

When Alpha-Complexes Collapse onto Codimension-1 Submanifolds
Dominique Attali, Mattéo Clémot, Bianca B. Dornelas, and André Lieutier 11:1–11:19

Higher-Order Color Voronoi Diagrams and the Colorful Clarkson–Shor Framework
Sang Won Bae, Nicolau Oliver, and Evanthia Papadopoulou 12:1–12:19

41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The Erdős–Szekeres Conjecture Revisited
Jineon Baek and Martin Balko 13:1–13:15

Extremal Betti Numbers and Persistence in Flag Complexes
Lies Beers and Magnus Bakke Botnan 14:1–14:18

When Distances Lie: Euclidean Embeddings in the Presence of Outliers and Distance Violations
Matthias Bentert, Fedor V. Fomin, Petr A. Golovach, M. S. Ramanujan, and Saket Saurabh 15:1–15:16

Signotopes with Few Plus Signs
Helena Bergold, Lukas Egeling, and Hung P. Hoang 16:1–16:14

Sparse Bounded Hop-Spanners for Geometric Intersection Graphs
Sujoy Bhore, Timothy M. Chan, Zhengcheng Huang, Shakhar Smorodinsky, and Csaba D. Tóth 17:1–17:15

Finding a Shortest Curve That Separates Few Objects from Many
Therese Biedl, Éric Colin de Verdière, Fabrizio Frati, Anna Lubiw, and Günter Rote 18:1–18:16

Polychromatic Coloring of Tuples in Hypergraphs
Ahmad Biniaz, Jean-Lou De Carufel, Anil Maheshwari, Michiel Smid, Shakhar Smorodinsky, and Miloš Stojaković 19:1–19:17

Polynomial-Time Algorithms for Contiguous Art Gallery and Related Problems
Ahmad Biniaz, Anil Maheshwari, Magnus Christian Ring Merrild, Joseph S. B. Mitchell, Saeed Odak, Valentin Polishchuk, Eliot W. Robson, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou, Thomas Shermer, Jack Spalding-Jamieson, Rolf Svenning, and Da Wei Zheng 20:1–20:21

Structure and Independence in Hyperbolic Uniform Disk Graphs
Thomas Bläsius, Jean-Pierre von der Heydt, Sándor Kisfaludi-Bak, Marcus Wilhelm, and Geert van Wordragen 21:1–21:16

Transforming Dogs on the Line: On the Fréchet Distance Under Translation or Scaling in 1D
Lotte Blank, Jacobus Conradi, Anne Driemel, Benedikt Kolbe, André Nusser, and Marena Richter 22:1–22:16

On the Twin-Width of Smooth Manifolds
Édouard Bonnet and Kristóf Huszár 23:1–23:16

An 11/6-Approximation Algorithm for Vertex Cover on String Graphs
Édouard Bonnet and Paweł Rzążewski 24:1–24:15

Approximating Klee's Measure Problem and a Lower Bound for Union Volume Estimation
Karl Bringmann, Kasper Green Larsen, André Nusser, Eva Rotenberg, and Yanheng Wang 25:1–25:16

Geometric Spanners of Bounded Tree-Width
Kevin Buchin, Carolin Rehs, and Torben Scheele 26:1–26:15

Computing Oriented Spanners and Their Dilation
Kevin Buchin, Antonia Kalb, Anil Maheshwari, Saeed Odak, Carolin Rehs, Michiel Smid, and Sampson Wong 27:1–27:17

Small Triangulations of 4-Manifolds and the 4-Manifold Census
Rhuaidi Antonio Burke, Benjamin A. Burton, and Jonathan Spreer 28:1–28:16

Sparsification of the Generalized Persistence Diagrams for Scalability Through Gradient Descent
Mathieu Carrière, Seunghyun Kim, and Woojin Kim 29:1–29:17

Computation of Toroidal Schnyder Woods Made Simple and Fast: From Theory to Practice
Luca Castelli Aleardi, Eric Fusy, Jyh-Chwen Ko, and Razvan-Stefan Puscasu 30:1–30:19

A Linear Time Algorithm for the Maximum Overlap of Two Convex Polygons Under Translation
Timothy M. Chan and Isaac M. Hair 31:1–31:16

Faster Algorithms for Reverse Shortest Path in Unit-Disk Graphs and Related Geometric Optimization Problems: Improving the Shrink-And-Bifurcate Technique
Timothy M. Chan and Zhengcheng Huang 32:1–32:14

On Zarankiewicz's Problem for Intersection Hypergraphs of Geometric Objects
Timothy M. Chan, Chaya Keller, and Shakhar Smorodinsky 33:1–33:14

Simplification of Trajectory Streams
Siu-Wing Cheng, Haoqiang Huang, and Le Jiang 34:1–34:14

A Theory of Sub-Barcodes
Oliver A. Chubet, Kirk P. Gardner, and Donald R. Sheehy 35:1–35:16

Strongly Sublinear Separators and Bounded Asymptotic Dimension for Sphere Intersection Graphs
James Davies, Agelos Georgakopoulos, Meike Hatzel, and Rose McCarty 36:1–36:16

Nearest Neighbor Searching in a Dynamic Simple Polygon
Sarita de Berg and Frank Staals 37:1–37:15

An Algorithm for Tambara-Yamagami Quantum Invariants of 3-Manifolds, Parameterized by the First Betti Number
Colleen Delaney, Clément Maria, and Eric Samperton 38:1–38:15

Tiling with Three Polygons Is Undecidable
Erik D. Demaine and Stefan Langerman 39:1–39:16

Apex Representatives
Tamal K. Dey, Tao Hou, and Dmitriy Morozov 40:1–40:16

Decomposing Multiparameter Persistence Modules
Tamal K. Dey, Jan Jendrysiak, and Michael Kerber 41:1–41:19

Range Counting Oracles for Geometric Problems
Anne Driemel, Morteza Monemizadeh, Eunjin Oh, Frank Staals, and David P. Woodruff 42:1–42:16

On Spheres with k Points Inside
Herbert Edelsbrunner, Alexey Garber, and Morteza Saghafian 43:1–43:12

A Minor-Testing Approach for Coordinated Motion Planning with Sliding Robots
Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan 44:1–44:15

Higher Order Bipartiteness vs Bi-Partitioning in Simplicial Complexes
Marzieh Eidi and Sayan Mukherjee 45:1–45:12

Non-Euclidean Erdős–Anning Theorems
David Eppstein 46:1–46:15

Shelling and Sinking Graphs on the Sphere
Jeff Erickson and Christian Howard 47:1–47:18

Exact Algorithms for Minimum Dilation Triangulation
Sándor P. Fekete, Phillip Keldenich, and Michael Perk 48:1–48:18

On Sparse Covers of Minor Free Graphs, Low Dimensional Metric Embeddings, and Other Applications
Arnold Filtser 49:1–49:16

Immersions and Albertson's Conjecture
Jacob Fox, János Pach, and Andrew Suk 50:1–50:10

Quantum Combine and Conquer and Its Applications to Sublinear Quantum Convex Hull and Maxima Set Construction
Shion Fukuzawa, Michael T. Goodrich, and Sandy Irani 51:1–51:15

Uniform Bounds on Product Sylvester-Gallai Configurations
Abhibhav Garg, Rafael Oliveira, and Akash Kumar Sengupta 52:1–52:15

A PTAS for TSP with Neighbourhoods over Parallel Line Segments
Benyamin Ghaseminia and Mohammad R. Salavatipour 53:1–53:15

The Fréchet Distance Unleashed: Approximating a Dog with a Frog
Sariel Har-Peled, Benjamin Raichel, and Eliot W. Robson 54:1–54:13

A Practical Algorithm for Knot Factorisation
Alexander He, Eric Sedgwick, and Jonathan Spreer 55:1–55:15

Sublinear Data Structures for Nearest Neighbor in Ultra High Dimensions
Martin G. Herold, Danupon Nanongkai, Joachim Spoerhase, Nithin Varma, and Zihang Wu 56:1–56:15

Snap Rounding: A Cautionary Tale
John Hershberger 57:1–57:14

Tracking the Persistence of Harmonic Chains: Barcode and Stability
Tao Hou, Salman Parsa, and Bei Wang 58:1–58:16

Incremental Planar Nearest Neighbor Queries with Optimal Query Time
John Iacono and Yakov Nekrich 59:1–59:15

Improved Approximation Algorithms for Three-Dimensional Knapsack
Klaus Jansen, Debajyoti Kar, Arindam Khan, K. V. N. Sreenivas, and Malte Tutas 60:1–60:18

k-Dimensional Transversals for Fat Convex Sets
Attila Jung and Dömötör Pálvölgyi 61:1–61:12

On Approximability of ℓ_2^2 Min-Sum Clustering
Karthik C. S., Euiwoong Lee, Yuval Rabani, Chris Schwiegelshohn, and Samson Zhou 62:1–62:18

The Maximum Clique Problem in a Disk Graph Made Easy
J. Mark Keil and Debajyoti Mondal 63:1–63:16

Super-Polynomial Growth of the Generalized Persistence Diagram
Donghan Kim, Woojin Kim, and Wonjun Lee 64:1–64:20

Recognizing 2-Layer and Outer k-Planar Graphs
Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff 65:1–65:16

Lipschitz Decompositions of Finite ℓ_p Metrics
Robert Krauthgamer and Nir Petruschka 66:1–66:14

Hard Diagrams of Split Links
Corentin Lunel, Arnaud de Mesmay, and Jonathan Spreer 67:1–67:17

Persistent (Co)Homology in Matrix Multiplication Time
Dmitriy Morozov and Primoz Skraba 68:1–68:16

Computing Betti Tables and Minimal Presentations of Zero-Dimensional Persistent Homology
Dmitriy Morozov and Luis Scoccola 69:1–69:15

Computing Geomorphologically Salient Networks via Discrete Morse Theory
Tim Ophelders, Anna Schenfisch, Willem Sonke, and Bettina Speckmann 70:1–70:17

Banana Trees for the Persistence in Time Series Experimentally
Lara Ost, Sebastiano Cultrera di Montesano, and Herbert Edelsbrunner 71:1–71:13

Geometric Bipartite Matching Based Exact Algorithms for Server Problems
Sharath Raghvendra, Pouyan Shirzadian, and Rachita Sowle 72:1–72:15

Embedding Graphs as Euclidean kNN-Graphs
Thomas Schibler, Subhash Suri, and Jie Xue 73:1–73:14

The Point-Boundary Art Gallery Problem Is $\exists\mathbb{R}$-Hard
Jack Stade 74:1–74:23

Levels in Arrangements: Linear Relations, the g-Matrix, and Applications to Crossing Numbers
Elizaveta Streltsova and Uli Wagner 75:1–75:18

A Note on the No-$(d+2)$-On-a-Sphere Problem
Andrew Suk and Ethan Patrick White 76:1–76:8

Dynamic Maximum Depth of Geometric Objects
Subhash Suri, Jie Xue, Xiongxin Yang, and Jiumu Zhu 77:1–77:13

Efficient Greedy Discrete Subtrajectory Clustering
Ivor van der Hoog, Lara Ost, Eva Rotenberg, and Daniel Rutschmann 78:1–78:20

CG Challenges

Computing Non-Obtuse Triangulations with Few Steiner Points
Mikkel Abrahamsen, Florestan Brunck, Jacobus Conradi, Benedikt Kolbe, and André Nusser 79:1–79:13

Incremental Algorithm and Local Search for Minimum Non-Obtuse Triangulations
Taehoon Ahn, Jaegun Lee, Byeonguk Kang, and Hwi Kim 80:1–80:8

Media Expositions

AmoebotSim 2.0: A Visual Simulation Environment for the Amoebot Model with Reconfigurable Circuits and Joint Movements
Matthias Artmann, Tobias Maurer, Andreas Padalkin, Daniel Warner, and Christian Scheideler 81:1–81:5

Software for the Thompson and Funk Polygonal Geometry
Hridhaan Banerjee, Carmen Isabel Day, Auguste H. Gezalyan, Olga Golovatskaia, Megan Hunleth, Sarah Hwang, Nithin Parepally, Lucy Wang, and David M. Mount 82:1–82:6

French Onion Soup, Ipelets for Points and Polygons
Klint Faber, Auguste H. Gezalyan, Adam Martinson, Aniruddh Mutnuru, Nithin Parepally, Ryan Parker, Mihil Sreenilayam, Aram Zaprosyan, and David M. Mount 83:1–83:6

Incremental and Interactive PQ- and PC-Trees
Simon D. Fink and Dominik Peters 84:1–84:4

Finding Shortest Reconfiguration Sequences for Modular Robots
UML Modular Robotics Group, Hugo A. Akitaya, Andrew Clements, Sam Downey, Jonathan Eisenbies, Soham Samanta, Gabriel Shahrouzi, and Frederick Stock 85:1–85:5

Preface

The 41st International Symposium on Computational Geometry (SoCG 2025) was held in Kanazawa, Japan, June 23–27, 2025, as part of the Computational Geometry Week (CG Week 2025).

The symposium received 225 submissions, the second highest in the symposium's history, and after a thorough review process, in which each paper was evaluated by three or more independent reviewers, the program committee accepted 78 papers for presentation. 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, the full paper is available at a public repository and referenced in the corresponding extended abstract.

The **Best Paper Award** of SoCG 2025 went to the paper "A Linear Time Algorithm for the Maximum Overlap of Two Convex Polygons Under Translation" by Timothy M. Chan and Isaac M. Hair; this paper has been invited to submit an extended version to the Journal of the ACM. The **Best Student Paper Award** of SoCG 2025 went to the paper "A Sparse Multicover Bifiltration of Linear Size" by Ángel Javier Alonso. This was the second year bestowing a best student paper award; all authors of the paper needed to be students at the time of submission to be eligible. The **Best Student Presentation Award** was determined and announced at the symposium, based on ballots cast by the attendees. For the second time since last year, a paper was selected for submission to the newly-established diamond open access journal TheoretiCS: "Levels in Arrangements: Linear Relations, the g-Matrix, and Applications to Crossing Numbers" by Elizaveta Streltsova and Uli Wagner. Additionally, a selection of papers was invited to submit extended versions to forthcoming special issues of Discrete & Computational Geometry and the Journal of Computational Geometry dedicated to the symposium.

The **SoCG Test of Time Awards** of this year went to "Linear Programming and Convex Hulls Made Easy" by Raimund Seidel, which was published in SoCG 1990, and to "Stability of Persistence Diagrams" by David Cohen-Steiner, Herbert Edelsbrunner, and John Harer, which was published in SoCG 2005.

The scientific program of CG Week 2025 was enriched by two distinguished invited speakers. An invited talk, entitled "Computational Origami for Art, Science, and Industry", was given by Tomohiro Tachi from The University of Tokyo. A second invited talk, entitled "Parameterized Computational Geometry: Visibility Problems, Geometric Intersection Graphs and Graph Drawing", was delivered by Meirav Zehavi from Ben-Gurion University of the Negev. We thank the plenary speakers for kindly accepting our invitation.

In addition to the technical papers, there were six submissions to the multimedia exposition. Submissions were reviewed and five 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://www.computational-geometry.org`.

The 7th Computational Geometry Challenge was part of CG Week 2025. The challenge problem was about minimum non-obtuse triangulations. This year, there were ten teams participating in the challenge, and these proceedings contain contributions by the two top-placed teams describing their winning approaches.

We thank the authors of all submitted works. We are most grateful to the members of the SoCG Program Committee, the Media Exposition 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

41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of reviewers; without their help, it would have been impossible to run the selection process. Finally, we thank Philipp Kindermann for meticulous work as Proceedings Chair.

Many other people contributed to the success of SoCG 2025 and the entire CG Week. We are very grateful to the local organization committee for their work in organizing the event. Finally, we thank all the members of the Test of Time Award, Workshop, and Young Researchers Forum Committees, the CG Challenge Advisory Board, and the CG Week Steering Committee.

- Oswin Aichholzer
  SoCG Program Committee Co-Chair
  Graz University of Technology
- Haitao Wang
  SoCG Program Committee Co-Chair
  University of Utah

Conference Organization

SoCG Program Committee

- Mikkel Abrahamsen (University of Copenhagen)
- Oswin Aichholzer (co-chair; Graz University of Technology)
- Hugo Akitaya (University of Massachusetts Lowell)
- Mark de Berg (Eindhoven University of Technology)
- Sujoy Bhore (Indian Institute of Technology Bombay)
- Ahmad Biniaz (University of Windsor)
- Håvard Bakke Bjerkevik (SUNY Albany)
- Gerth Stølting Brodal (Aarhus University)
- Hsien-Chih Chang (Dartmouth College)
- Siu-Wing Cheng (Hong Kong University of Science and Technology)
- Vida Dujmović (University of Ottawa)
- David Eppstein (University of California, Irvine)
- Emily Fox (University of Texas at Dallas)
- Jie Gao (Rutgers University)
- Dan Halperin (Tel Aviv University)
- Tao Hou (University of Oregon)
- Francis Lazarus (Université Grenoble Alpes)
- Chih-Hung Liu (National Taiwan University)
- Daniel Lokshtanov (University of California Santa Barbara)
- Anna Lubiw (University of Waterloo)
- Amir Nayyeri (Oregon State University)
- Eunjin Oh (Pohang University of Science and Technology)
- Tim Ophelders (Utrecht University and TU Eindhoven)
- Irene Parada (UPC BarcelonaTech)
- Rahul Saladi (Indian Institute of Science)
- Patrick Schnider (University of Basel and ETH Zürich)
- Raimund Seidel (Saarland University)
- Don Sheehy (North Carolina State University)
- Shakhar Smorodinsky (Ben-Gurion University)
- Jonathan Spreer (University of Sydney)
- Takeshi Tokuyama (Kwansei Gakuin University)
- Torsten Ueckerdt (Karlsruhe Institute of Technology)
- Pavel Valtr (Charles University)
- Kasturi Varadarajan (University of Iowa)
- Haitao Wang (co-chair; University of Utah)
- Jinhui Xu (University at Buffalo)
- Jie Xue (New York University Shanghai)

SoCG Proceedings Chair

- Philipp Kindermann (Universität Trier)

41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

CG Media Exposition Committee

- Aaron T. Becker (University of Houston)
- Brittany Fasy (Montana State University)
- Anirban Ghosh (University of North Florida)
- Philipp Kindermann (Trier University)
- Fabian Klute (Universitat Politècnica de Catalunya)
- Tamara Mchedlidze (Utrecht University)
- Martin Nöllenburg (chair; TU Wien)
- Christian Scheffer (HS Bochum)
- Willem Sonke (TU Eindhoven)
- Jules Wulms (co-chair; TU Eindhoven)

CG Challenge Committee

- Sándor Fekete (TU Braunschweig)
- Phillip Keldenich (TU Braunschweig)
- Dominik Krupke (TU Braunschweig)
- Stefan Schirra (Universität Madgeburg)

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)

Workshop Committee

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

Young Researchers Forum Program Committee

- Eyal Ackerman (University of Haifa at Oranim)
- Sayan Bandyapadhyay (Portland State University)
- Chaya Keller (Ariel University)
- Benjamin Raichel (University of Texas at Dallas)
- Salman Parsa (DePaul University)
- Eunjin Oh (chair; Pohang University of Science and Technology)
- Frank Staals (Utrecht University)
- Lori Ziegelmeier (Macalester College)

Local Organizing Committee

- Tonan Kamata (JAIST)
- Yoshio Okamoto (The University of Electro-Communications)
- Yota Otachi (Nagoya University)
- Toshiki Saitoh (Kyushu Institute of Technology)
- Tomoko Taniguchi (JAIST)
- Ryuhei Uehara (chair; JAIST)
- Giovanni Viglietta (University of Aizu)

Steering Committee (2024–2026)

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

Additional Reviewers

Mohammad Ali Abam
Ahmed Abdelkader
Mahmoud Abo Khamis
Karim Abu-Affash
Eyal Ackerman
Henry Adams
Peyman Afshani
Akanksha Agrawal
Hee-Kap Ahn
Ángel Javier Alonso
Greg Aloupis
Enrique G. Alvarado
Shinwoo An
Alexandr Andoni
Patrizio Angelini
Sunil Arya
Vikrant Ashvinkumar
Sepehr Assadi
Martin Aumüller
Sang Won Bae
Martin Balko
Ruben Ballester
Aritra Banik
Agnese Barbensi
Saugata Basu
Alexander Belov
Robin Belton
Thijs Beurskens
Michael Bilevich
Mitchell Black
Yossi Bokor Bleile
Prosenjit Bose
Magnus Botnan
Marie-Charlotte Brandenburg
Bruce Brewer
Florestan Brunck
Kevin Buchin
Maike Buchin
Benjamin Burton
Sergio Cabello
Sarah Cannon
Alessandra Caraceni
Charlie Carlson
Erin Chambers
Timothy Chan
Abhranil Chatterjee
Tianzhi Chen
Victor Chepoi
Man Kwun Chiu
Kyungjin Cho
Oliver A. Chubet
Nate Clause
David Cohen-Steiner
Éric Colin de Verdière
Jacobus Conradi
Jonathan Conroy
Justin Curry
Giordano Da Lozzo
Ovidiu Daescu
Justin Dallant
Minati De
Jean-Lou De Carufel

41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Arnaud de Mesmay
Oliveros Deborah
Vincent Delecroix
Chengyuan Deng
Vincent Despré
Prathamesh Dharangutte
Giuseppe Antonio Di Luna
Hu Ding
Michael Dobbins
Andrew Draganov
Anne Driemel
Nathan Dunfield
Matthijs Ebbens
Eduard Eiben
Jeff Erickson
Louis Esperet
Guillermo Esteban Pascual
Stefan Felsner
Qilong Feng
Arnold Filtser
Omrit Filtser
Benedikt Fluhr
Efi Fogel
Fabrizio Frati
Florian Frick
Daniel Frishberg
Niloufar Fuladi
Radoslav Fulek
Ameet Gadekar
Waldo Gálvez
Bernd Gärtner
Tzvika Geft
Matt Gibson-Lopez
Barbara Giunti
Chad Giusti
Marc Glisse
Xavier Goaoc
Miriam Goetze
Daniel Gonçalves
Mayank Goswami
Lee-Ad Gottlieb
Aziz Burak Guelen
Andrea Guidolin
Xiangyu Guo
Purvi Gupta
Siddharth Gupta
Rohit Gurjar
Rachana Gusain
Thekla Hamm
Sariel Har-Peled
Sebastian Haslebacher
Alexander He
Qizheng He
Dan Hefetz
Petr Hliněný
Michael Hoffmann
Gary Hoppenworth
Haoqiang Huang
Junyu Huang
Zhengcheng Huang
Ziyun Huang
Alfredo Hubard
Marius Huber
Kristóf Huszár
John Iacono
Tanmay Inamdar

Yuni Iwamasa
Riko Jacob
Mahya Jamshidian
Bruno Jartoux
Vít Jelínek
Attila Jung
Matthew Kahle
Roman Karasev
Matthew J. Katz
Chaya Keller
Michael Kerber
Woojin Kim
Philipp Kindermann
David Kirkpatrick
Sándor Kisfaludi-Bak
Michal Kleinbort
Linda Kleist
Katharina Klost
Fabian Klute
Stephen Kobourov
Matias Korman
Jan Kratochvil
Stephan Kreutzer
Bala Krishnamoorthy
Dominik Krupke
Marc Lackenby
Justin Lanier
Lap Chi Lau
Sylvain Lazard
Hung Le
Seunghun Lee
Mingxi Lei
Michael Lesnick
Amit Levi
André Lieutier
William Lochet
Maarten Löffler
Kevin Lu
Ben Lund
Kelin Luo
Jayson Lynch
Alexander Lytchak
Joseph Maher
Anil Maheshwari
Clement Maria
Leonardo Martínez
William Maxwell
Bradley McCoy
Pooran Memari
Facundo Memoli
Laura Merker
Magnus Christian Ring Merrild
Majid Mirzanezhad
Pranabendu Misra
Hendrik Molter
Debajyoti Mondal
Fabrizio Montecchiani
Pat Morin
Sonoko Moriyama
Guillaume Moroz
Dmitriy Morozov
Wolfgang Mulzer
Liz Munch
Anurag Murty Naredla
Cameron Musco
Soumi Nandi

Additional Reviewers

Abhinandan Nath
Piotr Nayar
Tom Needham
Ofer Neiman
Yakov Nekrich
Linh Nguyen
André Nusser
Yoshio Okamoto
Yota Otachi
Nina Otter
Arnau Padrol
Fahad Panolan
Evanthia Papadopoulou
Salman Parsa
Ori Parzanchevski
Pavel Paták
Zuzana Patáková
Michael Payne
Matteo Pegoraro
Sarah Peluse
Sarah Percival
Jonathan Perry
Timothe Picavet
Matt Piekenbrock
Michał Pilipczuk
Rom Pinchasi
Tim Planken
Valentin Polishchuk
Siddharth Pritam
Sergey Pupyrev
Sharath Raghvendra
Benjamin Raichel
Nikhilesh Rajaraman
Mathias Rav
Rahul Raychaudhury
Bastien Rivier
Vanessa Robins
Liam Roditty
Jonathan Rollin
Ron Rosenthal
Günter Rote
Bodhayan Roy
Natan Rubin
Casper Moldrup Rysgaard
Paweł Rzążewski
Rachel Saban
Arsenii Sagdeev
Morteza Saghafian
Shreyas Samaga
Eric Samperton
Rik Sarkar
Saket Saurabh
Radmila Sazdanovic
Sara Scaramuccia
Marcus Schaefer
Anna Schenfisch
Thomas Schibler
Christiane Schmidt
Samuel Schneider
Jens Kristian Refsgaard Schou
Eric Sedgwick
Carlos Segovia
Yuan Sha
Liren Shan
Micha Sharir
Roohani Sharma

Siddharth Sheth
Suraj Shetiya
Jonathan Shewchuk
Pouyan Shirzadian
Anatasios Sidiropoulos
Rodrigo I. Silveira
Stavros Sintos
Primoz Skraba
Lucas Slot
Michiel Smid
Yaar Solomon
Jack Spalding-Jamieson
Bettina Speckmann
Piyush Srivastava
Frank Staals
Jack Stade
Frederick Stock
Aditya Subramanian
Martin Suderland
Andrew Suk
Vaishali Surianarayanan
Rolf Svenning
Konrad Swanepoel
Yufei Tao
Cuong Than
Stephan Tillmann
Raphaël Tinarrage
Géza Tóth
Csaba D. Tóth
Konstantinos Tsakalidis
Mark Tygert
Ivor Van Der Hoog
Thijs van der Horst
Kevin Verbeek
Constantin Vernicos
Antoine Vigneron
Giorgio Vinciguerra
N. V. Vinodchandran
Hubert Wagner
Erik Waingarten
Bartosz Walczak
Daochen Wang
Elena Wang
Hung-Lung Wang
Minghua Wang
Qingsong Wang
Qisheng Wang
Simon Weber
Alexandra Weinberger
Andreas Wiese
Marcus Wilhelm
Mathijs Wintraecken
Michał Włodarczyk
Paul Wollan
Sampson Wong
William Wu
Jiayi Xian
Cheng Xin
Shuyi Yan
Chao Yang
Xiongxin Yang
Sang Duk Yoon
Ji Zeng
Hanwen Zhang
Yiming Zhao
Da Wei Zheng
Jiumu Zhu

Reconfiguration of Unit Squares and Disks: PSPACE-Hardness in Simple Settings

Mikkel Abrahamsen
University of Copenhagen, Denmark

Kevin Buchin
TU Dortmund, Germany

Maike Buchin
Ruhr-Universität Bochum, Germany

Linda Kleist
Universität Potsdam, Germany

Maarten Löffler
Utrecht University, The Netherlands

Lena Schlipf
Universität Tübingen, Germany

André Schulz
FernUniversität Hagen, Germany

Jack Stade
University of Copenhagen, Denmark

Abstract

We study well-known reconfiguration problems. Given a start and a target configuration of geometric objects in a polygon, we wonder whether we can move the objects from the start configuration to the target configuration while avoiding collisions between the objects and staying within the polygon. Problems of this type have been considered since the early 80s by roboticists and computational geometers. In this paper, we study some of the simplest possible variants where the objects are labeled or unlabeled unit squares or unit disks. In unlabeled reconfiguration, the objects are identical, so that any object is allowed to end at any of the targets positions. In the labeled variant, each object has a designated target position. The results for the labeled variants are direct consequences from our insights on the unlabeled versions.

We show that it is PSPACE-hard to decide whether there exists a reconfiguration of (unlabeled/labeled) unit squares even in a simple polygon. Previously, it was only known to be PSPACE-hard in a polygon with holes for both the unlabeled and labeled version [Solovey and Halperin, Int. J. Robotics Res. 2016]. Our proof is based on a result of independent interest, namely that reconfiguration between two satisfying assignments of a formula of Monotone-Planar-3-Sat is also PSPACE-complete. The reduction from reconfiguration of Monotone-Planar-3-Sat to reconfiguration of unit squares extends techniques recently developed to show NP-hardness of packing unit squares in a simple polygon [Abrahamsen and Stade, FOCS 2024]. We also show PSPACE-hardness of reconfiguration of (unlabeled/labeled) unit disks in a polygon with holes. Previously, it was known that unlabeled reconfiguration of disks of two different sizes was PSPACE-hard [Brocken, van der Heijden, Kostitsyna, Lo-Wong and Surtel, FUN 2021].

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Complexity theory and logic

Keywords and phrases reconfiguration, unit square, unit disk, unlabeled, labeled, simple polygon, polygon

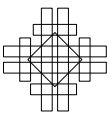
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.1

Related Version *Full Version*: `https://arxiv.org/abs/2412.21017`

Funding *Mikkel Abrahamsen*: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.
Jack Stade: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.

Acknowledgements This work was initiated at the Dagstuhl seminar 24072: Triangulations in Geometry and Topology. We thank the organizers and the participants for the fruitful atmosphere.

© Mikkel Abrahamsen, Kevin Buchin, Maike Buchin, Linda Kleist, Maarten Löffler, Lena Schlipf, André Schulz, and Jack Stade;
licensed under Creative Commons License CC-BY 4.0



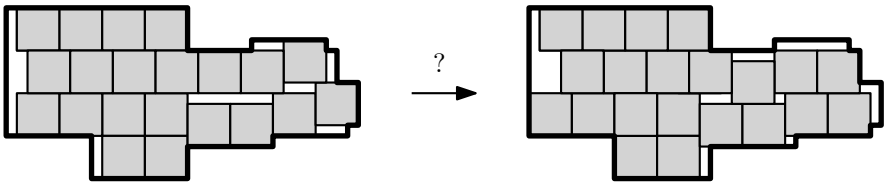


Figure 1 Can the left configuration of unit squares in a simple polygon be reconfigured to the right configuration of unit squares?

1 Introduction

In *geometric reconfiguration* we are given a start configuration of geometric *objects* and a desired target configuration. A particularly well-studied geometric configuration problem is *multi-robot motion planning*. The goal is to move the robots (or objects) from the start position to the target configuration while avoiding collisions. Problems of this type have been considered since the early 80s by roboticists and computational geometers. Hopcroft, Schwartz and Sharir [14] showed already in 1984 that reconfiguration of labeled rectangles moving in a rectangle is PSPACE-hard. The motivation of their work was "to find moving systems (necessarily involving arbitrarily many degrees of freedom) whose geometry is *as simple as possible*, but for which the motion planning problem is still PSPACE-hard".

In this paper, we study some of the simplest possible variants where the objects are unlabeled or labeled axis-aligned congruent squares or unit disks. In case of squares, the objects have to stay axis-aligned during the reconfiguration. For an example, consider Figure 1. Obstacles are modeled by the workspace in which the objects lie. At any time, the objects have to avoid the complement of the workspace. Depending on the variant, the workspace is given as a polygon, a simple polygon, or a grid polygon, that is an orthogonal polygon whose vertex coordinates are integers. In unlabeled reconfiguration, the objects are indistinguishable, so that any object is allowed to end at any of the target positions. In labeled reconfiguration the target position for every object is explicitly given. Table 1 presents an overview of the known results and the new results of this paper.

Table 1 Summary of results in motion planning for multiple objects.

Contribution	Labeled	Complexity	Objects	Workspace
[14]	yes	PSPACE-hard	rectangles	rectangle
[12]	yes	PSPACE-hard	1×2 rectangles	rectangle
[19]	yes	PSPACE-hard	unit squares	polygon (with holes)
this work	yes	PSPACE-hard	unit squares	simple polygon
[16]	yes	P	unit squares	grid polygon
[19]	no	PSPACE-hard	unit squares	polygon (with holes)
this work	no	PSPACE-hard	unit squares	simple polygon
easy exercise	no	trivial "yes"	unit squares	grid polygon
[21]	yes	strongly NP-hard	disks of three sizes	simple polygon
this work	yes	PSPACE-hard	unit disks	polygon (with holes)
[4]	no	PSPACE-hard	disks of two sizes	polygon (with holes)
this work	no	PSPACE-hard	unit disks	polygon (with holes)

Recent results of PSPACE-hardness of geometric reconfiguration problems have been proven by a reduction from Nondeterministic Constrained Logic (NCL) [11, 13]. This holds for instance for the works [4, 5, 12, 19]. Here, a polygon is built "on top of" an instance of NCL, which is in itself a plane graph (the problem NCL will be described in detail in Section 2). This technique inevitably leads to a polygon with holes, because the bounded faces of the NCL-instance are turned into holes of the polygon.

A similar phenomenon appears in many proofs of NP-hardness of two-dimensional geometric problems where the input is a polygon: Such reductions are often from a variant of Planar-3-Sat and work by building a polygon on top of the Planar-3-Sat-instance we are reducing from. Again, this leads to a polygon with a hole for every bounded face of the graph representing the Planar-3-Sat-formula.

Abrahamsen and Stade [1] recently overcame this obstacle by making a novel reduction from Planar-3-Sat in order to obtain NP-hardness of the well-known problem of packing unit squares in a simple polygon; a problem that had been known to be NP-hard for polygons with holes since 1981 [8]. Interestingly, Abrahamsen and Stade [1] used a non-planar geometric realization of a planar formula, where variables are represented as horizontal rows and edges are vertical columns. These rows and columns are allowed to cross, but the planarity of the graph ensures that the endpoints of all rows and columns are incident to the outer face of the drawing. This makes it possible to construct a simple polygon, following the boundary of the outer face, that "implements" the functionality of the rows and columns. The rows and columns of the drawing are represented by rows and columns of squares, and these likewise cross each other in the interior of the polygon. The crucial observation is that movement in one direction does not influence movement in the other direction, so binary values can be "transported" independently in both directions through a crossing.

Our contribution. In this paper we show PSPACE-hardness of reconfiguration of unit squares in a simple polygon. To this end, we first show a result of independent interest. The solution space of boolean formulas is given by an hypercube graph, where edges are formed between assignments which differ in exactly one variable. The satisfying assignments of a formula ϕ define a subgraph of this graph, which we call $G(\phi)$. We call two satisfying assignments *connected* if they are connected in $G(\phi)$. The reconfiguration problem for boolean formulas asks if two satisfying assignments are connected. We show the following result.

▶ **Theorem 1.** *The reconfiguration problem for Monotone-Planar-3-Sat formulas is* PSPACE*-complete.*

For some other versions of 3-Sat, the reconfiguration problem has likewise been shown to be PSPACE-hard, such as Monotone-Planar-Not-All-Equal-3-SAT [6]. In Table 2, we give a summary of some variants of 3-Sat and the complexity of their satisfyability and reconfiguration problems. Interestingly, most satisfiability and reconfiguration problems are hard, but there are some exceptions that can be solved in polynomial time.

Using Theorem 1, we show PSPACE-hardness for reconfiguration of squares.

▶ **Theorem 2.** *Unlabeled reconfiguration of* 8×8 *squares in simple grid polygons is* PSPACE*-complete.*

This implies PSPACE-hardness for reconfiguration of unit squares in simple polygons. We also remark that the size of the coordinates of the constructed polygon is $O(n^2)$, where n is the number of corners.

In our construction, the squares can only move locally. Hence, we exactly know the start and end position of each square and also obtain the same result for the labeled version.

Table 2 Overview of computational complexity for the satisfiability and reconfiguration problems of fundamental 3-Sat variants.

	Satisfiability	Reconfiguration
3-Sat	NP-complete [15]	PSPACE-complete [10]
Planar-3-Sat	NP-complete [17]	PSPACE-complete [6]
Positive-1-in-3-SAT	NP-complete [9]	P ('no' iff assignments differ)
Positive-Not-All-Equal-3-SAT	NP-complete [18]	PSPACE-complete [10]
Mon.-Planar-Not-All-Eequal-3-SAT	P [18]	PSPACE-complete [6]
Monotone-Planar-3-Sat	NP-complete [7]	PSPACE-complete [This work]

▶ **Corollary 3.** *Labeled reconfiguration of* 8×8 *squares in simple grid polygons is* PSPACE*-complete.*

Theorem 2 and Corollary 3 strengthen the results of Solovey and Halperin [19], who showed that the problem is PSPACE-hard for polygons with holes in the unlabeled and labeled variants. In contrast, in grid polygons, it is an easy exercise to show that reconfiguration of unlabeled unit squares is always doable (if the number of squares agrees) and reconfiguration of labeled unit squares is polynomial time decidable [16]. Hence, our result is close to the border of polynomial-time decidability.

Lastly, we consider the reconfiguration of unit disks.

▶ **Theorem 4.** *Reconfiguration of unlabeled unit disks in a polygon (possibly with holes) is* PSPACE*-hard.*

Before, it was only known that reconfiguration of unlabeled disks of two different sizes was PSPACE-hard, as shown by Brocken, van der Heijden, Kostitsyna, Lo-Wong and Surtel [4]. For disks of the same size so far only positive results were known under the assumption that there is sufficient separation between the start and target positions of the disks in a simple polygon [2, 3] and also in a polygon with holes [20]. The main technical challenge here is to artificially discretize the continuous space of disk configurations, to avoid allowing global states in which many gadgets are locally in an "in-between" state which would break the reduction. We overcome this challenge by designing a special edge gadget that enforces the consistent orientation of an edge, and argue that only one edge may be reoriented at a time. Thus, our results in particular show that an assumption such as the separation assumption in [20] is necessary if $\mathsf{P} \neq \mathsf{PSPACE}$.

In fact, our constructed polygon guarantees that no two disks may swap their positions. Therefore, hardness for the labeled version follows immediately.

▶ **Corollary 5.** *Reconfiguration of labeled unit disks in a polygon (possibly with holes) is* PSPACE*-hard.*

Previously, Spikaris and Yap [21] showed NP-hardness for labeled disks of three different sizes.

Organization of the paper. The rest of the paper is structured as follows. In Section 2, we briefly introduce NCL and and present the general idea of the NP-hardness proof for Monotone-Planar-3-Sat. In Section 3, we prove that the reconfiguration of unit squares in a simple polygon is PSPACE-hard, by reducing from reconfiguration of Monotone-Planar-3-Sat. Finally, in Section 4, we show that the reconfiguration of unit disks in a

polygon with holes is PSPACE-hard, by a reduction from NCL. We conclude with a discussion in Section 5. The proofs of all statements marked by ($\star$) and the full proof of Theorem 1 are presented in the full version.

2 Reconfiguration of Monotone-Planar-3-Sat

Here we sketch a proof of the PSPACE-hardness where we reduce from NCL. We start by reviewing the configuration-to-configuration variant of the NCL problem. An instance is given by an oriented planar cubic graph G and an reorientation G' of G. Each edge is colored red or blue, such that each vertex is incident to either three blue edges, or one blue edge and two red edge. The orientations of G and G' need to be *feasible*, that is, the indegree of every vertex is at least two, where blue edges count with multiplicity 2. An NCL-instance is positive, if and only if there is a sequence of edge reversals that transforms G to G' such that each intermediate orientation is feasible.

For the reduction we start with an embedding of G and modify G such that it becomes the vertex-clause-incidence graph of a formula ϕ. Any satisfying assignment of the variables of ϕ encodes a feasible orientation of G and vice versa. Furthermore, if two feasible orientations of G differ by the orientation of a single edge, the corresponding satisfying variable assignments differ by two variable flips. Moreover, any variable flip corresponds to two feasible orientations of G that differ by at most one edge reorientation. A key idea is to replace all vertices of G with one or two clause nodes and every edge with a variable node as sketched in Figure 2. A variable x_e for an edge e is `true`, if and only if the current orientation of e matches the orientation of e in G.

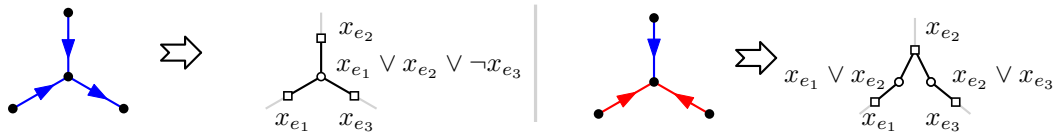


Figure 2 Mapping an NCL-Instance to an embedding of a planar 3-SAT formula. Variable nodes are drawn as squares, clause nodes are drawn as disks.

To prove that the embedding of the variable-clause incidence graph has the required properties we have to extend ϕ. In particular, we have to ensure that on every face in the planar embedding of the clause-variable incidence graph of ϕ positive and negative clauses alternate. See the full version for all details and a full proof of Theorem 1.

3 Unit squares in simple polygons

In this section, we prove Theorem 2 by reducing MONOTONE-PLANAR-3-SAT to the unlabeled reconfiguration of squares in simple grid polygons. We start with a MONOTONE-PLANAR-3-SAT formula ϕ and two satisfying assignments σ_0 and σ_1 of ϕ. In our reduction, we create a grid polygon P and use the assignments σ_0 and σ_1 to define two configurations c_0 and c_1 of 8×8 squares in P. We then show that there is a reconfiguration from c_0 to c_1 in P if and only if there is a reconfiguration from σ_0 to σ_1 in ϕ. The squares are axis-aligned and may move by translations only.

3.1 Reference centers

As a preliminary step we provide a helpful tool for verifying the correctness of the reduction. Let P be a grid polygon. A set $\mathcal{R}$ of k non-overlapping 4×4 squares, each contained in P, is called a set of *reference centers* for P if, for any packing of k 8×8 squares in P, each square in the packing contains one square in $\mathcal{R}$. Such a configuration of 8×8 squares in P is called *perfect*. For every $y \in \mathbb{Z}$, let b_y be an integer in $[0, 8)$. For $x, y \in \mathbb{Z}$, let $S_{xy} = [8x + b_y, 8x + b_y + 4] \times [8y, 8y + 4]$. Suppose that each S_{xy} is either fully contained in P or is interior-disjoint with P. Let $\mathcal{R}$ be the set of those S_{xy} that are fully contained in P. An example is shown in Figure 3. We will show that $\mathcal{R}$ is a set of reference centers for P.

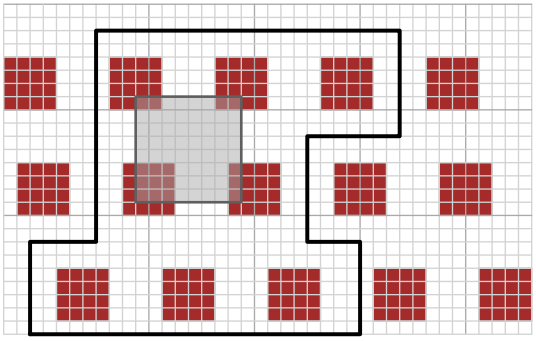


Figure 3 A grid polygon that has reference centers with $b_0 = 1, b_1 = 6, b_2 = 5$ and an 8×8 square intersecting four reference centers; the area of intersection is 16. (Note that in this example there exists no perfect configuration.)

▶ **Lemma 6.** ($\star$) *Let S be an 8×8 square placed in P. Then the area of the intersection of S with $\mathcal{R}$ is* 16.

▶ **Lemma 7.** ($\star$) *$\mathcal{R}$ is a set of reference centers for P.*

By Lemma 7, in any reconfiguration from one perfect configuration to another, each square contains the same reference center the whole time, i.e., the squares merely shift slightly sideways and up and down around the same reference center. In order to describe the mechanics of our intended reconfigurations, we only need to consider positions of squares that are extreme or centered in each direction with respect to their reference centers. This leaves us with the nine possible positions shown in Figure 4a. In Section 3.7, we prove that also arbitrary continuous reconfigurations in P correspond to reconfigurations of ϕ, but for understanding the key ideas of the reduction, it suffices to consider these nine discrete positions.

When the top (resp. bottom) edge of a square aligns with that of its reference center, we say that the square is *down* (resp. *up*). Otherwise, the midpoints of the reference center and the square have the same y-coordinate, in which case we say that the square is *vemid* (abbreviation of vertically in the middle). When the left (resp. right) edge of the square aligns with that of the reference center, the square is *right* (resp. *left*). Otherwise, the midpoints of the reference center and the square have the same x-coordinate, in which case we say that the square is *homid* (abbreviation of horizontally in the middle).

For most squares, several of the nine possible positions can be excluded, for instance due to being pushed by the boundary of P or another square. In our figures, we use a color scheme for the reference centers, to make the reader aware of what possible positions a square

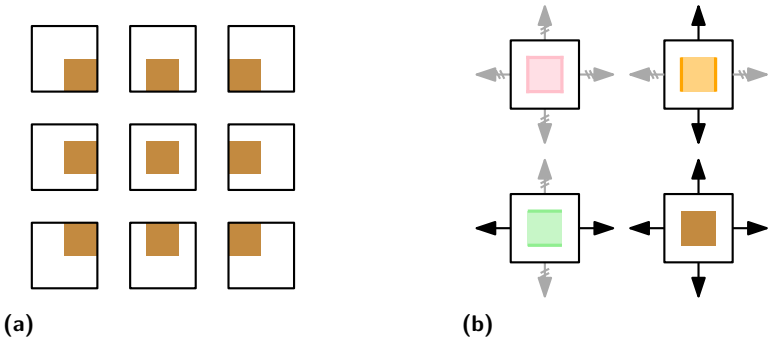


Figure 4 (a) The up to nine important placements of a square for a given reference center. (b) Examples of colored reference centers and how the squares containing them may move.

has. The colors have the following meanings: the square can not move (*pink*), only move vertically (*orange*), only move horizontally (*green*), move vertically and horizontally (*brown*). There might exist additional restrictions on the positions that are not indicated by the colors.

3.2 Schematics of the construction

On a big scale, our construction follows the ideas of Abrahamsen and Stade [1]. Therefore, we start by reviewing the high level ideas. Let ϕ be an instance of Monotone-Planar-3-Sat with variables $x_1, \ldots, x_k$ and consider the variable-clause-incidence graph G with a suitable embedding. We now describe schematically the construction of a grid polygon P. Each feasible solution to ϕ corresponds to a configuration of 8×8 squares in P, and there is a reconfiguration between two configurations if and only if there is a reconfiguration between the two solutions of ϕ.

In the first step, we introduce a *segment representation* that consists of horizontal segments for vertices and clauses, and of vertical segments for the edges of G, see in Figure 5b. The idea behind this representation is as follows: The horizontal segments will correspond to one or several rows of squares which transmit information horizontally. Similarly, the vertical segments will correspond to columns of squares that transmit information vertically. The endpoints of each segment should be on the outer face of the drawing, so that the smallest enclosing orthogonal polygon runs through all of the endpoints.

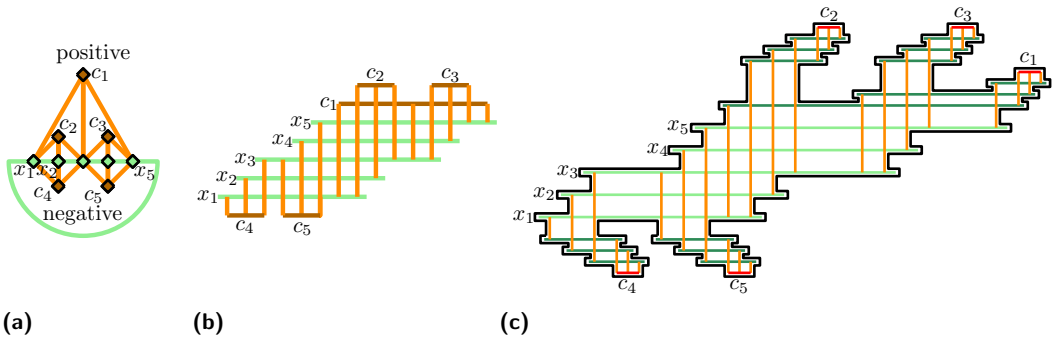


Figure 5 Schematic construction for transforming an instance of Monotone-Planar-3-Sat to a polygon. (a) a variable-clause incidence graph. (b) first step for constructing the segment representation of G (c) second step for constructing the segment representation of G: introducing the auxiliary variable rows and OR gadgets.

To construct the segment representation, we introduce a horizontal segment for each variable $x_1, \ldots, x_k$ in this order bottom-up, where the x-coordinates of the left and right endpoints are increasing, see Figure 5. Furthermore, the right endpoint of x_1 is to the right of the left endpoint of x_k, so that all the segments have a common horizontal overlap. These are the *primary* variable rows. The positive clauses are represented by horizontal segments above x_k and the negative clauses by horizontal segments below x_1. We say that a clause c' is *nested* in a clause c, if in G the clause node of c' lies inside a cycle defined by two incident edges of the node of c closed with a path between variable nodes. For instance, in Figure 5a, c_2 and c_3 are nested inside c_1. Note that the property of being nested defines a partial order. If a positive clause c_i is nested inside c_j, then we place the segment of c_i above that of c_j in the segment representation. If instead the clauses are negative, the segment of c_i is below that of c_j. Each edge $x_i c_j$ of G is realized by a vertical segment connecting the segments of x_i and c_j. All vertical segments to the positive clauses are placed to the right of all those to the negative clauses. The left-to-right ordering of the edges to the positive clauses along the variables $x_1, \ldots, x_k$ in G is preserved by the corresponding vertical segments, as well as the ordering of the edges to the negative clauses. The segment of a positive (resp. negative) clause c_j starts at the top (resp. bottom) endpoint of the vertical segment corresponding to the leftmost edge incident to c_j in G and ends at the top (resp. bottom) endpoint of the rightmost edge.

In the next step, illustrated in Figure 5c, we replace each clause segment by three *auxiliary* variable segments, one for each of the variables connected to the clause. In the right side of the auxiliary segments, we connect them using vertical segments to an OR gadget, represented by a red horizontal segment. As sketched in the figure, we construct (guided by the partial nested-order of the positive and negative clauses) a polygon whose boundary approximately follows the outer face of the resulting representation, but the finer details will be given later. The following lemma was proved in [1].

▶ **Lemma 8.** *There is a segment representation as described, where the segments representing the OR gadgets are not crossed by any vertical segments, and all segment endpoints are incident to the outer face of the drawing.*

Algebraically, the introduction of an auxiliary variable row for a clause c_t corresponds to the following equivalence:

$$(x_i \vee x_j \vee x_k) \Leftrightarrow \Big(\exists y_{t,i}, y_{t,j}, y_{t,k} : (y_{t,i} \Rightarrow x_i) \wedge (y_{t,j} \Rightarrow x_j) \wedge (y_{t,k} \Rightarrow x_k) \wedge (y_{t,i} \vee y_{t,j} \vee y_{t,k})\Big)$$

Here, x_i, x_j, x_k are the primary (original) variables and $y_{t,i}, y_{t,j}, y_{t,k}$ are the auxiliary variables introduced for that particular clause. We remark that the formula is not monotone anymore, but the only non-monotone clauses are implications. Note that the equivalence also preserves reconfiguration: If, say, x_i changes to `false`, then we change $y_{t,i}$ to `false` before changing x_i. If x_i changes to `true`, then we change $y_{t,i}$ to `true` after x_i. For a negative clause c_t , we use an analogous equivalence:

$$(\neg x_i \vee \neg x_j \vee \neg x_k) \Leftrightarrow$$
$$\Big(\exists y_{t,i}, y_{t,j}, y_{t,k} : (\neg y_{t,i} \Rightarrow \neg x_i) \wedge (\neg y_{t,j} \Rightarrow \neg x_j) \wedge (\neg y_{t,k} \Rightarrow \neg x_k) \wedge (\neg y_{t,i} \vee \neg y_{t,j} \vee \neg y_{t,k})\Big)$$

3.3 Components and gadgets

We informally distinguish between *gadgets* and *components* in our constructions, where a gadget is given by a set of segments on the boundary creating some simple functionality, and a component is thought of as a whole region of the polygon that can involve an arbitrary number of gadgets.

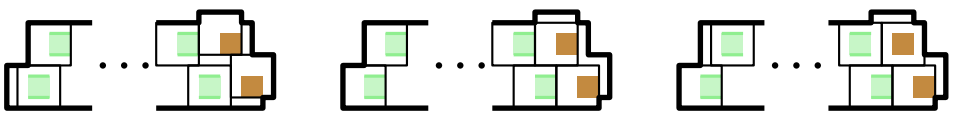
On a very high level the truth assignment of a variable of ϕ is encoded by a configuration of a so-called *variable component*; see Figure 6. The variable component (used to represent primary and auxiliary variables) has essentially three possible states, namely *plus*, *minus* and *transitional*. The first two correspond to the values `true` and `false` of a binary variable, while the truth value of the transitional state is defined to be the `true`/`false` value of the previous plus/minus state. In order to split the information represented in a variable component of a primary variable and transfer it to a variable component of an auxiliary variable we use PUSH gadgets. In particular, these gadgets are used to model the implication-clauses and they come in two versions: PUSH-UP-IF-MINUS and PUSH-DOWN-IF-PLUS. The former is used to model implications of the form $y \Rightarrow x$, the later for $\neg y \Rightarrow \neg x$. The gadgets use a vertically moving stack of square rows with growing width (called pyramid) to transmit information from primary to auxiliary variables, see for example Figure 8. For a PUSH-UP-IF-MINUS gadget representing an implication $y \Rightarrow x$ a pyramid that is raised enforces that the auxiliary variable component of y is set to minus, if the variable component of x is set to minus. Similarly, a PUSH-DOWN-IF-PLUS gadget modeling $\neg y \Rightarrow \neg x$ has the functionality that a pyramid that is lowered enforces that the auxiliary variable component of y is set to plus, if the variable component of x is set to plus. We will make sure that the vertically moving pyramid structure do not collide with the vertically operating variable components they "cross". To only allow satisfying assignments for every clause we place an *OR gadget* that is connected with the corresponding auxiliary variable components. We reuse the ideas for the PUSH Gadgets. For a clause $(y_i \vee y_j \vee y_k)$ the auxiliary variables will push a pyramid up into the clause gadget if the corresponding auxiliary variable is set to minus. The OR gadget has only room for two of these pyramids to push up. The clauses $(\neg y_i \vee \neg y_j \vee \neg y_k)$ are handled symmetrically. Here, an auxiliary variable set to plus would result in a pyramid pushing down and the OR gadget can only incorporate two lowered pyramids.

We continue with describing the gadgets and components in detail.

3.4 The variable component

The variable component has two *main* rows of squares and an arbitrary number of PUSH gadgets, ensuring dependency between the primary and auxiliary variables and between the auxiliary variables and the OR gadgets. The PUSH gadgets require the introduction of *helper rows* padded along the main rows, above the top row or below the bottom row. The construction of PUSH gadgets and helper rows will be explained in later sections.

The construction of the main rows of a variable component is shown in Figure 6. A perfect configuration of the component has three possible states, as defined by the horizontal position of the two rightmost squares (covering the brown reference centers). Note that none of the two squares can push right, and if one is homid, the other needs to be right.



(a) plus state **(b)** transitional state **(c)** minus state

Figure 6 A variable component and its three states, defined by the horizontal positions of the two rightmost squares with brown reference centers.

If the bottom is in the homid position, the state is *plus*. If the top is in the homid position, the state is *minus*. Otherwise, the state is *transitional*.

Now we describe what happens when a block of raised or lowered squares "crosses" a variable component. As discussed before, pyramids made of such blocks are used to create dependencies between two variable components, but it is important that they can cross other variable components without interacting with them.

When a stack of raised squares crosses a row of squares with a different horizontal alignment, the stack of raised squares will grow in width. Hence, the stack will be pyramid shaped if all rows have a different alignment than the neighboring rows. It is important that we know the width of a pyramid when it reaches the target variable component where it will end. Thus, we need to know exactly how many times the width is increased and which squares are raised in the top row. To this end, and in order to create a necessary amount of vertical spacing, we introduce *static* rows in between all neighboring pairs of variable components. A static row consists of squares in the beginning and in the end, which cannot move, and some squares in the middle, which can move vertically up and down, see Figure 7. So in particular, all squares in a static row have a fixed horizontal position.

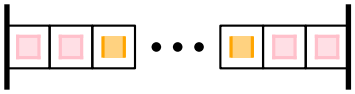


Figure 7 A static row. The squares have fixed horizontal positions, but some of the middle squares will be able to move up or down because of pyramids.

We ensure that the fixed horizontal position is never aligned with the rows of the variable component. Furthermore, the top row of a variable component is never aligned with the bottom row, see Figure 8. Hence, when a pyramid crosses a variable component, the specific horizontal positions of the rows do not affect which squares are raised/lowered on the other side of the variable. This is expressed in the following lemma.

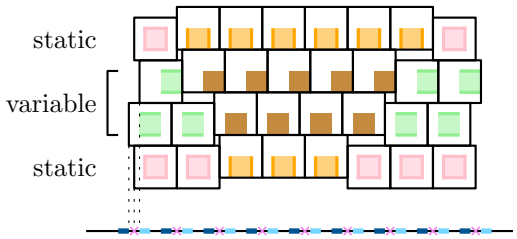


Figure 8 A pyramid crossing a variable component. The scale at the bottom shows the intervals of the possible x-coordinates of the left corners of the upper and lower squares in the variable rows (light and dark blue, respectively) and those of the squares in the static rows (pink). Since these three groups of intervals are pairwise disjoint, the pyramid grows in width by one square per row. Hence, the width of the pyramid always grows by three squares when crossing a variable component, regardless of the state of the variable component.

▶ **Lemma 9.** ($\star$) *Consider a perfect configuration. If a block of consecutive squares in a static row below (above) a variable component are up (down), then the corresponding squares in the static row above (below) the variable component must also be up (down) as well as two extra squares on the right (left) and one on the left (right).*

Recall that there may be helper rows above or below the main rows of a variable component, which will be explained in more detail in Section 3.5. Conceivably, these could be unaligned with the main rows, which would cause a pyramid to grow more in width when crossing the variable. However, if they are aligned (which is our "intended behavior" of the helper rows), then the width will not grow beyond what is expressed by Lemma 9.

3.5 PUSH gadgets and helper rows

Figures 9 and 10 show the PUSH-UP-IF-MINUS and PUSH-DOWN-IF-PLUS gadgets, respectively. The gadgets are mirror images of each other by a horizontal axis. In order to make PUSH gadgets, we need a square that is pushed up or down only when it is left. To this end, we introduce *helper rows*, which are extra rows above and below the main variable rows. Consider the PUSH-UP-IF-MINUS gadget, shown in Figure 9. If the squares of the lower main row are left, the squares of the helper row will be left as well, because the square a presses down b, which thus pushes c to the left. Therefore the square d will be up, and a pyramid of raised squares is created.

Figure 9 The plus and minus states of PUSH-UP-IF-MINUS. In minus, square d is pushed up.

The PUSH-DOWN-IF-PLUS gadget works by analogous mechanics. If the squares in the upper main row are left, two squares of the lower row are pushed down, see Figure 10.

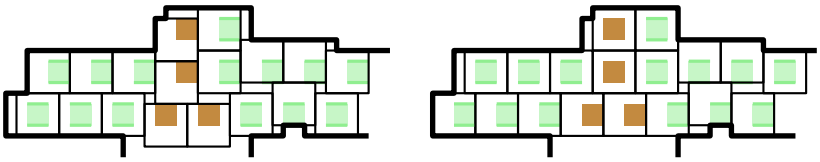


Figure 10 The plus and minus states of PUSH-DOWN-IF-PLUS. In plus, the squares with brown reference centers are pushed down.

All helper rows are created at the right end of the variable gadget and end at a PUSH gadget. Figure 11 shows schematically where the helper rows are placed, and Figure 12 shows a detailed example with multiple helper rows. Note that with multiple helper rows, the same principle is used as with a single helper row in order to ensure that when the lower main row is left, then all helper rows below are also left, so they all create pyramids of raised squares.

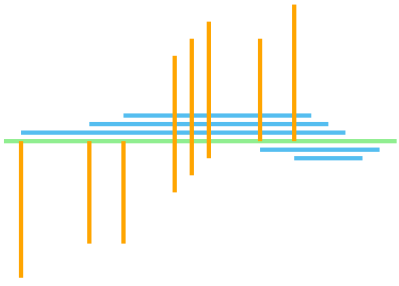


Figure 11 Schematic showing the horizontal span of the helper rows (blue) along a principal variable component. The variable connects to three auxiliary variables below and two above, so there are three helper rows above and two below. In between the two groups, pyramids from other principal variable components can cross, indicated as three orange vertical segments in the middle.

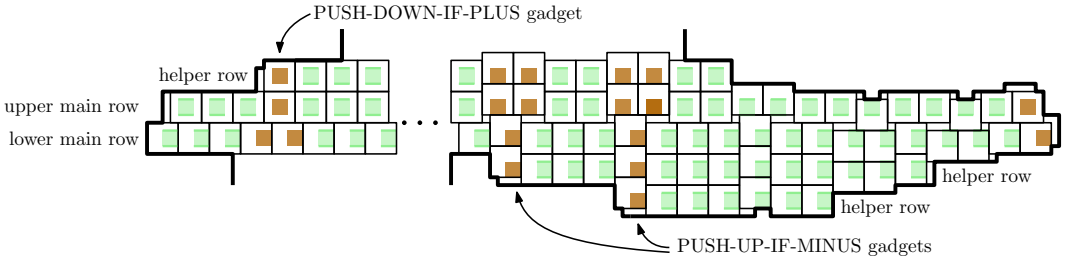


Figure 12 A variable component in the minus position with two PUSH-UP-IF-MINUS gadgets and one PUSH-DOWN-IF-PLUS gadget. Thus, there are two helper rows at the bottom and one at the top. Other pyramids may cross the variable in the region indicated by the dots.

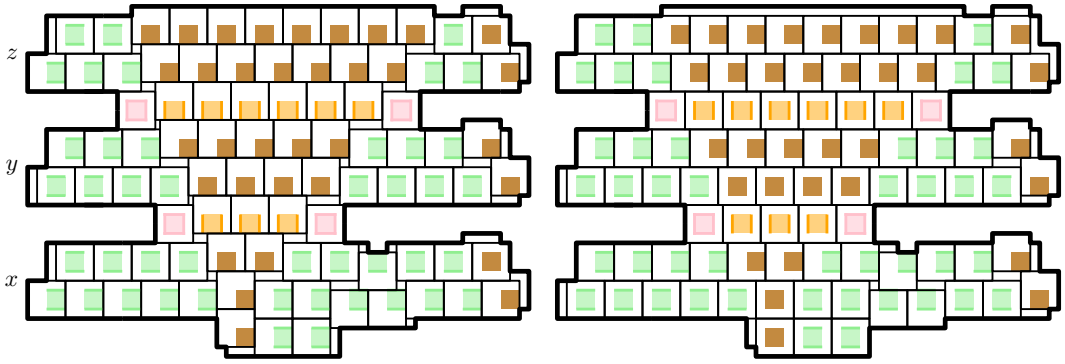


Figure 13 There is a PUSH-UP-IF-MINUS gadget on the bottom variable x, which connects x to the top variable z. When x is minus, a pyramid of raised squares is created. The boundary of the polygon at z forces z to also be minus. The state of y has no influence. When x is plus, the pyramid does not have to raise, so z is not restricted.

We now describe how a pyramid ends at an auxiliary variable component; see Figure 13 for a concrete example to support the description. When we make a PUSH-UP-IF-MINUS gadget from a primary variable x to an auxiliary variable y, we create a pocket in the boundary of P along the upper primary row of y to enable the squares of the pyramid to raise. We place the pocket so that the squares can only raise if they are homid (and if the pyramid has grown in width by exactly three squares every time it crossed another variable component). Thus, the rightmost square of the top row of y is also homid, so y must be minus. We have thus made the implication $y \Rightarrow x$.

In minus, pyramids are created to the positive auxiliary variables, which in turn create pyramids to the positive clauses. Likewise, in plus, pyramids are created to the negative clauses. In transitional, pyramids are created to all clauses. As we will see in Section 3.6, there will only be room for two pyramids for each clause. Hence, each clause ensures that one of the variables is in the plus/minus state that represents the `true`/`false` value that satisfies the clause. This gives the following lemma.

▶ **Lemma 10.** *In every perfect configuration, the following holds. If a variable is not plus (minus), a pyramid of raised squares appears from any PUSH-UP-IF-MINUS (PUSH-DOWN-IF-PLUS) gadget, and if the pyramid is received by an auxiliary variable, that variable must be minus (plus). As a consequence, we can realize the implications $y \Rightarrow x$ and $\neg y \Rightarrow \neg x$, i.e., ensure that the values of variables encoded by the configuration satisfy the implications.*

3.6 OR gadget

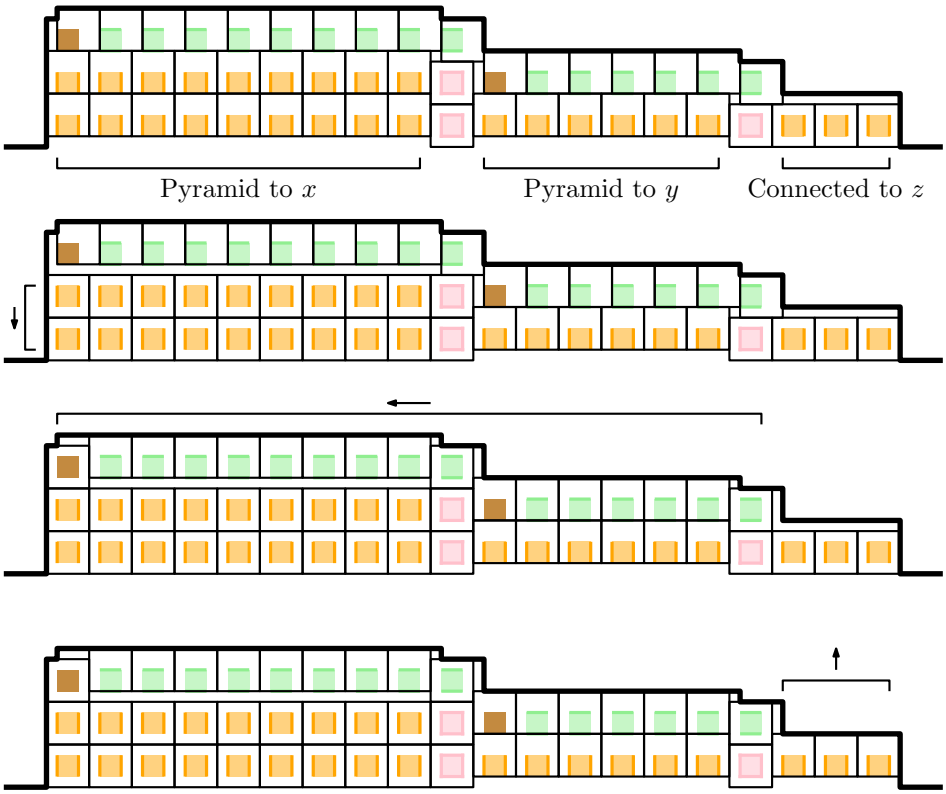


Figure 14 The OR gadget for a clause $x \vee y \vee z$. Initially, there is a pyramid from x and y. The pyramid from x is removed, and it is shown how the top squares reconfigure so that a pyramid from z can raise.

Figure 14 shows the positive OR gadget. The gadget is connected to three auxiliary variable components. Using PUSH-UP-IF-MINUS gadgets, pyramids can raise from the variables and end in the OR gadget. The gadget allows for two pyramids to be raised, but not all three at once. Hence, one variable must be plus. When at most one pyramid is raised, the squares in the top of the gadget can reconfigure so that there is room for any of the other two pyramids to raise. An example is shown in the figure – the reader can easily check that it is likewise possible in the other five cases where one pyramid is raised and we need to make room for one of the other two to raise as well. This corresponds to a reconfiguration of the formula ϕ: If $x \vee y \vee z$ is a clause and one of the variables, say z, changes from positive to negative, then x or y must have been positive before z changes (otherwise the clause would not be satisfied after z changing). Hence, there was only one pyramid, so the squares in the gadget can reconfigure to accommodate raising the pyramid from z.

Figure 15 shows how the OR gadget is connected to the variable components. The negative OR gadget is obtained by reflection along a horizontal axis.

▶ **Lemma 11.** *In a positive OR gadget, at least one of the connected auxiliary variables is plus. In a negative OR gadget, at least one of the connected auxiliary variables is minus.*

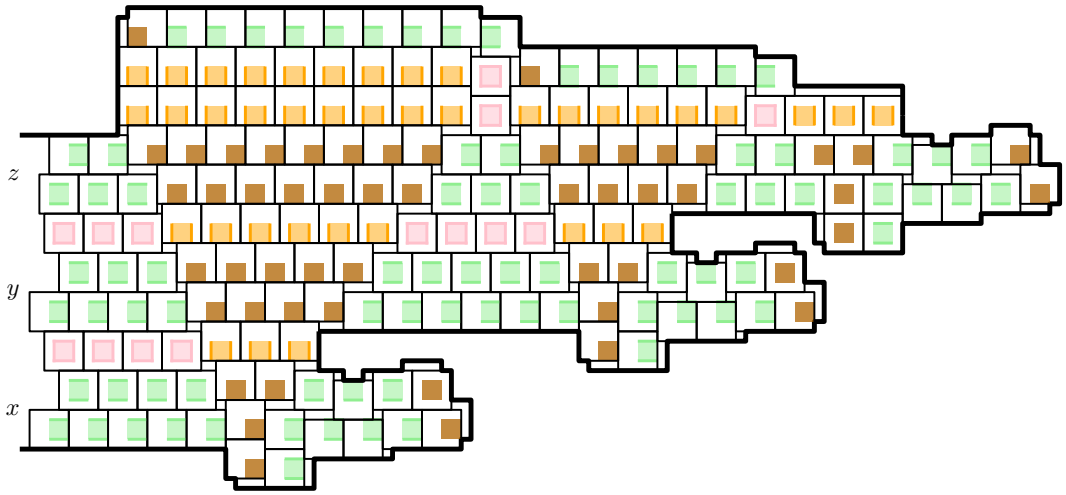


Figure 15 The positive OR gadget connected to the auxiliary variable components for x, y and z. Here, x and y are minus, so they create pyramids, but z is plus, which allows for a reconfiguration.

3.7 Verification

Recall that we start with a Monotone-Planar-3-Sat formula ϕ and two satisfying assignments σ_0 and σ_1 of ϕ. In our reduction, we create a polygon P as described. We define start and target configurations c_0 and c_1 according to σ_0 and σ_1: We place squares in all primary and auxiliary variable components in c_i in the plus/minus state corresponding to the value of the variable in σ_i. By construction there is room for the pyramids created by PUSH gadgets. We now need to verify that there is a reconfiguration between σ_0 and σ_1 in ϕ if and only if there is one between c_0 and c_1 in P. This is expressed by the following two lemmas.

▶ **Lemma 12.** ($\star$) *If ϕ has a reconfiguration between two satisfying assignments σ_0 and σ_1, then there is also a reconfiguration between the corresponding start and target configurations c_0 and c_1 in P.*

▶ **Lemma 13.** ($\star$) *If there is a reconfiguration in P from the start configuration c_0 to the target c_1 in P, then there is a reconfiguration in the formula ϕ from the satisfying assignment σ_0 to σ_1.*

Finally, we prove the containment in PSPACE.

▶ **Lemma 14.** ($\star$) *Unlabeled reconfiguration of 8×8 squares in simple grid polygons is contained in* PSPACE.

Theorem 2 result now follows from Theorem 1 and Lemmas 12–14.

As we only consider perfect configurations in our reduction, by Lemma 7 each square always contains its reference center. Thus, we can easily label each square and obtain hardness for the labeled variant, i.e., Corollary 3.

4 Unit disks in polygons with holes

We now present the ideas for showing PSPACE-hard of reconfiguration of unit disks in a polygon possibly with holes. As a first step, we prove it for an *arc-gon*, that is, a shape whose boundary consists of straight line segments and circular arcs.

▶ **Proposition 15.** *Reconfiguration of unlabeled unit disk robots in an arc-gon is* PSPACE*-hard.*

Proof-Sketch. As in Section 2, we reduce from the configuration-to-configuration variant of Nondeterministic Constrained Logic (NCL) [11, 12, 13]. From a given NCL instance, specified by two orientations of a plane cubic graph G and a reorientation G', we construct an arc-gon and two configurations of unit disks such that the NCL instance is a yes-instance if and only of the two configurations of unit disks can be reconfigured into one another.

Similar to [4], we lay out G using five types of *tiles* as illustrated in Figure 16. Here we ensure that each edge has a at least one straight tile.

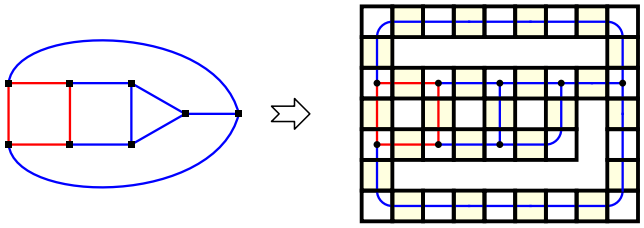
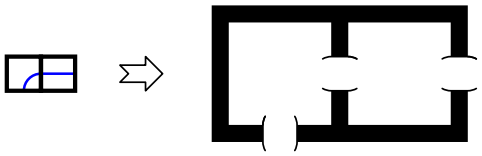
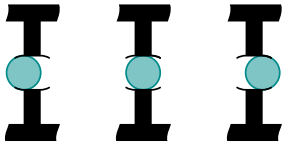


Figure 16 A layout of an NCL instance using a limited set of tiles.

Each tile is then realized as a *room*, where small "hooks" limit the movement of the disks in the opening of the room which is called the *door* of the room, see Figures 17a and 17b.



(a) Each tiles becomes a room, and an edge through tile sides implies a door.



(b) Each door has a (green) *door disk* that must stay within the door opening.

Figure 17

The interiors of the rooms mimic the behavior of the vertices and edges of G during a reconfiguration, see Figure 18. In particular, it is essential to guarantee that an edge can only have one orientation at a time; we implement this in our *straight* tile gadget where we guarantee that at least one door disks is "pushed out" and implies that the AND and OR tiles behave as required, see Figure 19. ◀

Then, we turn the arc-gon constructed in the proof of Proposition 15 into a polygon with holes and thereby complete the proof of Theorem 4. Observing that no two disks may swap within the polygon implies PSPACE-hardness for the labeled variant, i.e., Corollary 5.

5 Discussion

We proved that reconfiguration of Monotone-Planar-3-Sat is PSPACE-hard, and thereby complemented a sequence of recent results establishing the relationships between 3-Sat variants and their reconfiguration counterparts. This result was a stepping stone towards showing PSPACE-hardness of reconfiguration of 8×8-squares in a simple grid polygon. While the reduction reuses ideas from Abrahamsen and Stade [1], we also needed several new ideas. Most importantly, the reduction to the packing problem uses very rigid configurations whereas our setting inherently requires to allow for some movements. While the computational complexity of reconfiguring $k \times k$-squares in simple grid polygons is trivial for $k = 1$, the cases $k \in \{2, \ldots, 7\}$ remain an interesting open problem.

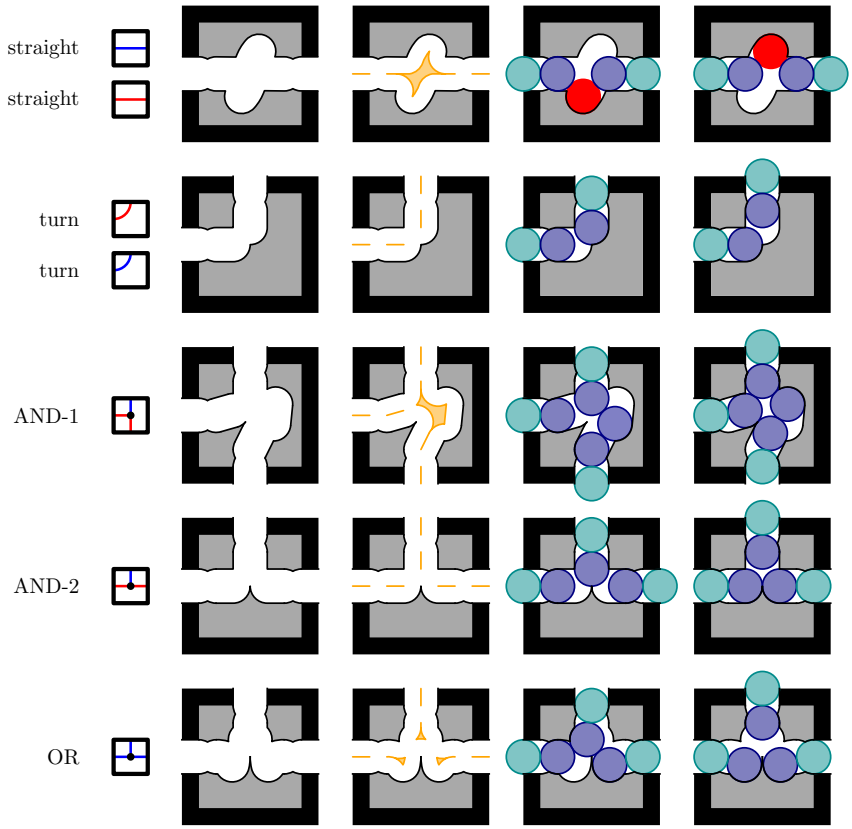


Figure 18 Geometric realizations of the five tile types. Marked in orange is the *dilation* of the shapes by 1; these are possible locations of disk centers. On the right two different valid configurations of disks with different extremal positions of the door disks.

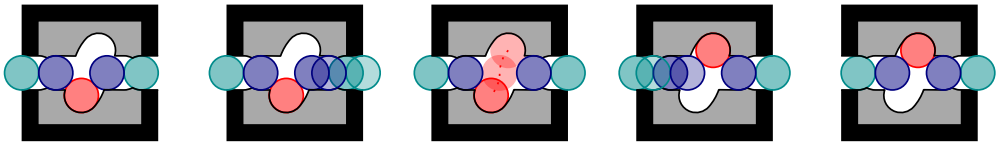


Figure 19 A sequence of moves that imitates the reorientation of an edge.

We have also shown that reconfiguration of unit disks in a polygon with holes is PSPACE-hard. A natural next question is unit disks in a simple polygon. At a first glance, it seems difficult to extend our insights in the square setting to this case, because it is troublesome to propagate movement independently in multiple directions in a large region filled with disks. Therefore, we expect that a PSPACE-hardness proof, if it exists, will likely require different techniques.

References

1 Mikkel Abrahamsen and Jack Stade. Hardness of packing, covering and partitioning simple polygons with unit squares. In *Annual Symposium on Foundations of Computer Science (FOCS 2024)*, 2024. doi:10.48550/arXiv.2404.09835.

2 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Trans Autom. Sci. Eng.*, 12(4):1309–1317, 2015. doi:10.1109/TASE.2015.2470096.

3 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 *International Symposium on Computational Geometry, SoCG 2022*, volume 224 of *LIPIcs*, pages 12:1–12:16, 2022. doi:10.4230/LIPICS.SOCG.2022.12.

4 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 *10th International Conference on Fun with Algorithms, FUN 2021*, volume 157 of *LIPIcs*, pages 15:1–15:16, 2021. doi:10.4230/LIPICS.FUN.2021.15.

5 Josh Brunner, Lily Chung, Erik D. Demaine, Dylan H. Hendrickson, Adam Hesterberg, Adam Suhl, and Avi Zeff. 1×1 rush hour with fixed blocks is PSPACE-complete. In *10th International Conference on Fun with Algorithms, FUN 2021*, volume 157 of *LIPIcs*, pages 7:1–7:14, 2021. doi:10.4230/LIPICS.FUN.2021.7.

6 Jean Cardinal, Erik D. Demaine, David Eppstein, Robert A. Hearn, and Andrew Winslow. Reconfiguration of satisfying assignments and subset sums: Easy to find, hard to connect. *Theoretical Computer Science*, 806:332–343, 2020. doi:10.1016/j.tcs.2019.05.028.

7 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geom. Appl.*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

8 Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.

9 M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. Freeman, 1979. URL: https://books.google.nl/books?id=fjxGAQAAIAAJ.

10 Parikshit Gopalan, Phokion G. Kolaitis, Elitza Maneva, and Christos H. Papadimitriou. The connectivity of boolean satisfiability: Computational and structural dichotomies. *SIAM Journal on Computing*, 38(6):2330–2355, 2009. doi:10.1137/07070440X.

11 Robert A. Hearn and Erik D. Demaine. The nondeterministic constraint logic model of computation: Reductions and applications. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002*, volume 2380 of *LNCS*, pages 401–413. Springer, 2002. doi:10.1007/3-540-45465-9_35.

12 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005. doi:10.1016/J.TCS.2005.05.008.

13 Robert A. Hearn and Erik D. Demaine. *Games, puzzles, and computation*. CRC Press, 2009.

14 J.E. Hopcroft, J.T. Schwartz, and M. 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. doi:10.1177/027836498400300405.

15 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.

16 D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Symposium on Foundations of Computer Science (FOCS)*, pages 241–250, 1984. doi:10.1109/SFCS.1984.715921.

17 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

18 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, 1978. doi:10.1145/800133.804350.

19 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *Int. J. Robotics Res.*, 35(14):1750–1759, 2016. doi:10.1177/0278364916672311.

20 Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems XI, Sapienza University of Rome*, 2015. doi:10.15607/RSS.2015.XI.011.

21 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.

The Maximum Number of Digons Formed by Pairwise Intersecting Pseudocircles

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

Gábor Damásdi
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary
ELTE Eötvös Loránd University, Budapest, Hungary

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

Rom Pinchasi
Technion – Israel Institute of Technology, Haifa, Israel

Rebeka Raffay
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Abstract

In 1972, Branko Grünbaum conjectured that any nontrivial arrangement of $n > 2$ pairwise intersecting pseudocircles in the plane can have at most $2n-2$ digons (regions enclosed by exactly two pseudoarcs), with the bound being tight. While this conjecture has been confirmed for cylindrical arrangements of pseudocircles and more recently for geometric circles, we extend these results to any simple arrangement of pairwise intersecting pseudocircles.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases pairwise intersecting arrangement, arrangement of pseudocircles, counting digons, tangencies, Grünbaum's conjecture

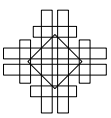
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.2

Funding *Gábor Damásdi*: Research supported by ERC grant No. 882971, "GeoScape" and by the Eköp-24 university excellence scholarship program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.
Balázs Keszegh: Research supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, by the National Research, Development and Innovation Office – NKFIH under the grant K 132696 and FK 132060, by the ÚNKP-23-5 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund and by the ERC Advanced Grant "ERMiD". This research has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the ELTE TKP 2021-NKTA-62 funding scheme.

1 Introduction

In this paper we focus on families of pairwise intersecting pseudocircles. A *family of pseudocircles* is a set $\mathcal{F}$ of simple closed Jordan curves in the plane such that every two of them are either disjoint, meet at exactly one point in which they touch or intersect at exactly two points in which they properly cross each other. The *arrangement* $\mathcal{A}(\mathcal{F})$ is the cell complex into which the plane is decomposed by the pseudocircles in $\mathcal{F}$. It consists of vertices, edges and faces. The arrangement is called *trivial* if there are two points that lie

© Eyal Ackerman, Gábor Damásdi, Balázs Keszegh, Rom Pinchasi, and Rebeka Raffay;
licensed under Creative Commons License CC-BY 4.0



on every pseudocircle in $\mathcal{F}$. If there is no point that lies on three pseudocircles, then both the arrangement and $\mathcal{F}$ are called *simple*. Note that every simple arrangement of more than two pseudocircles is nontrivial. The arrangement is called *pairwise intersecting* if any two pseudocircles intersect in exactly two points.

A *digon* is a face in the arrangement that is bounded by exactly two edges. If the two edges of a digon are contained in pseudocircles c_1 and c_2, respectively, then we say that c_1 and c_2 *form* that digon and that each of c_1 and c_2 *supports* it.

A trivial arrangement of n pseudocircles contains $2n$ digons for $n > 1$. In his 1972 monograph "Arrangements and Spreads" [11] Branko Grünbaum conjectured that for nontrivial arrangements of pairwise intersecting pseudocircles the maximum number of digons is $2n - 2$.

▶ **Conjecture 1** (Grünbaum's digon conjecture [11, Conjecture 3.6])**.** *Every nontrivial arrangement of $n > 2$ pairwise intersecting pseudocircles has at most $2n - 2$ digons.*

This conjecture was accompanied by a construction with exactly $2n - 2$ digons, see Figure 1 for an example with six pseudocircles. This example can be easily extended by adding new pseudocircles to the four in the middle, which illustrates that the bound in Conjecture 1 is tight. With some care, this construction can even be realized with geometric circles. A different construction appears in [8].

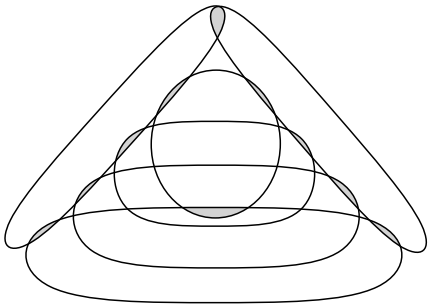


Figure 1 A family of 6 pseudocircles forming 10 digons.

Conjecture 1 was shown to be true in several special cases (assuming simple arrangements). Agarwal et al. [3] showed that Conjecture 1 is true for *cylindrical* arrangements, these are arrangements for which there is a point that is surrounded by each of the pseudocircles. More recently, Conjecture 1 was studied and advertised by Felsner, Roch and Scheucher [8] who showed that it holds for arrangements in which there are three pseudocircles such that every two of them form a digon. This new interest in Grünbaum's conjecture has also motivated us to study the problem. First, we were able to show in [1] that Grünbaum's conjecture is true for simple arrangements of pairwise intersecting geometric circles. Here, using ideas from [1] and [3] we prove:

▶ **Theorem 2.** *Every simple arrangement of $n > 2$ pairwise intersecting pseudocircles has at most $2n - 2$ digons. This bound is tight.*

Our proof is inspired by the proof technique introduced in [3] for bounding the number of digons in cylindrical arrangements of n pairwise intersecting pseudocircles. In that paper, a bipartite graph is associated to a cylindrical arrangement in a way that each pseudocircle is represented by a single point which is its intersection with a common fixed transversal line. The edges of this graph correspond to digons and are drawn according to a simple yet clever rule such that every two independent edges cross an even number of times. The Strong Hanani-Tutte Theorem (see below) then implies that the drawn (bipartite) graph is planar and hence by Euler's formula has at most $2n - 4$ edges.

For the general case of Grünbaum's conjecture, we combine a modification of this graph drawing technique and a graph *doubling* technique that was used for the case of geometric circles in [1]. Namely, a bipartite graph is drawn such that every pseudocircle is represented by *two* points which are its intersection with a common transversal pseudocircle. Furthermore, every digon is represented by *two* edges that are drawn following a rule in the spirit of [3] such that every two independent edges cross an even number of times. Therefore, the drawn (bipartite) graph is planar and hence has at most $2(2n) - 4 = 4n - 4$ edges, representing at most $2n - 2$ digons.

Note that our result can be applied for bounding the number of *touching points* in simple arrangements where any two pseudocircles intersect or touch. In any such arrangement of $n > 2$ pseudocircles, one can turn each touching point into exactly one digon between the two pseudocircles that support the touching. Therefore, for simple arrangements, an upper bound on the number of digons translates directly to a bound on the number of touching pairs or equivalently (in this case) touching points. One can also turn each digon into exactly one touching point in the case of simple arrangements if $n > 3$, but for $n = 3$, two pseudocircles may touch the third one at separate points forming a single digon (the region bounded by the entire third pseudocircle), see Figure 2. In this scenario, turning the digon into a touching point would reduce the third pseudocircle to a single point, which is degenerate, and the final arrangement would violate the assumptions on the number of crossings.

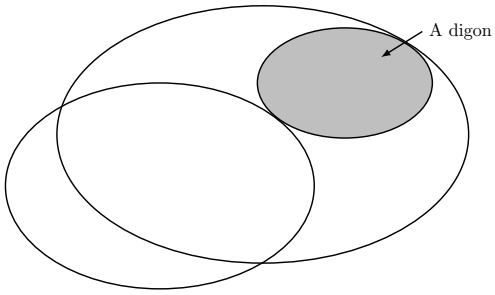


Figure 2 If we allow touchings and $n = 3$, then a digon might be surrounded by only one pseudocircle.

The transformation from digons to touching points or vice versa keeps their sum constant at every step (except for $n = 2$ or in the above-mentioned case), therefore they are equivalent for $n > 3$. From Theorem 2, we have:

▶ **Corollary 3.** *Every simple arrangement of pseudocircles where any two pseudocircles intersect or touch has at most $2n - 2$ touching pairs of pseudocircles. For $n > 3$, the sum of digons and touching pairs is at most $2n - 2$.*

1.1 Outline

In Section 2 we collect useful observations and results that we need for the main proof. In Section 3 we prove Theorem 2, and in Section 4 we discuss some open questions.

2 Some terminology and tools

We begin with some simple definitions and observations, some of which also appear in [1] in the context of circles.

Let $\mathcal{F}$ be a simple family of $n > 2$ pairwise intersecting pseudocircles in the plane. We may assume (and we will assume from now on) that every pseudocircle in $\mathcal{F}$ supports at least one digon. Otherwise, we can remove any pseudocircle from $\mathcal{F}$ that does not satisfy this condition and argue for the remaining family of pseudocircles.

As touchings are forbidden in a pairwise intersecting family, each digon is supported by exactly two pseudocircles. A digon is called a *lens* if it is surrounded by both supporting pseudocircles, and it is called a *lune* otherwise. A lune must be surrounded by precisely one of its supporting pseudocircles, while lying outside the region surrounded by the other (see Figure 3). Here we ignore the case where the unbounded face is a digon. This indeed can be avoided by means of a simple inversion of the plane.

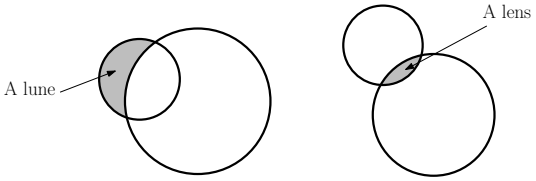


Figure 3 A lens and a lune.

A pseudocircle α is *internal* if it supports a digon that is surrounded by α and it is called *external* if it supports a digon (necessarily a lune) that is not surrounded by α. The following simple observation is important for the proof.

▶ **Proposition 4.** *A pseudocircle in $\mathcal{F}$ cannot be both internal and external.*

Proof. Assume to the contrary that there is a pseudocircle $c \in \mathcal{F}$ which is both external and internal. This means that there exist $c_1, c_2 \in \mathcal{F}$ such that c and c_1 form a digon not surrounded by c while c and c_2 form a digon that is surrounded by c (see Figure 4).

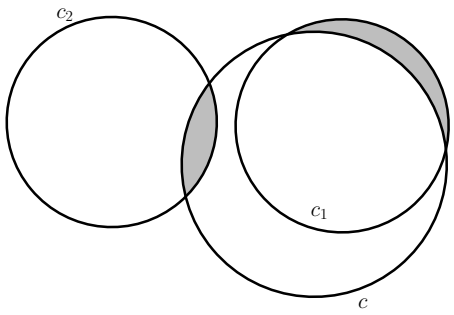


Figure 4 A pseudocircle cannot be both internal and external.

In such a case c_1 and c_2 cannot intersect, contradicting our assumption that we have an arrangement of pairwise intersecting pseudocircles. Indeed, c_1 and c_2 cannot cross in the region surrounded by c because that subarc of c_2 is an edge of a digon. Similarly, c_1 and c_2 cannot intersect in the complementary region of the plane because that subarc of c_1 is an edge of an other digon. ◀

Any simple closed Jordan curve in the plane divides the plane into two parts, one bounded (the *interior*) and the other unbounded (the *exterior*). We will refer to these two parts as the *regions* of the curve. For a pseudocircle $\alpha \in \mathcal{F}$, we will call the region that contains the digons supported by α the *digon-region* of α. By Proposition 4 and because we assume

that every pseudocircle supports at least one digon, every pseudocircle in $\mathcal{F}$ has exactly one digon-region. One advantage of this definition is that we can treat lunes and lenses in a uniform way: every digon is just the intersection of the digon-regions of the two corresponding pseudocircles supporting it.[1] This will simplify the case-analysis in some of the proofs.

It is well-known that any family of pairwise intersecting pseudocircles can be extended to a larger family of pairwise intersecting pseudocircles, and the following lemma shows that we can also do it without destroying the existing digons.

▶ **Lemma 5.** *Let $\mathcal{F}$ be a simple family of $n > 2$ pairwise intersecting pseudocircles. Then there is a closed curve c such that $\mathcal{F} \cup \{c\}$ is a simple family of pairwise intersecting pseudocircles, and furthermore c does not intersect any of the digons in $\mathcal{A}(\mathcal{F})$.*

Proof. Let α be a pseudocircle in $\mathcal{F}$. From Proposition 4 we know that one of the two regions of α (the one that is not the digon-region of α) does not contain any digon supported by α. Let c be a curve that is running very close to α outside of the digon-region of α. Then c intersects each pseudocircle in $\mathcal{F} \setminus \{\alpha\}$ exactly twice and it intersects no digon of $\mathcal{A}(\mathcal{F})$. However, c does not intersect α. In order to fix this, we notice that because $\mathcal{F}$ is a simple arrangement of pseudocircles, there is a subarc s of α that is disjoint from any digon supported by α and consequently it is disjoint from any digon in $\mathcal{A}(\mathcal{F})$. We can modify c along s such that the resulting simple closed curve crosses α at two points yielding the desired extension of $\mathcal{F}$ (see Figure 5). ◀

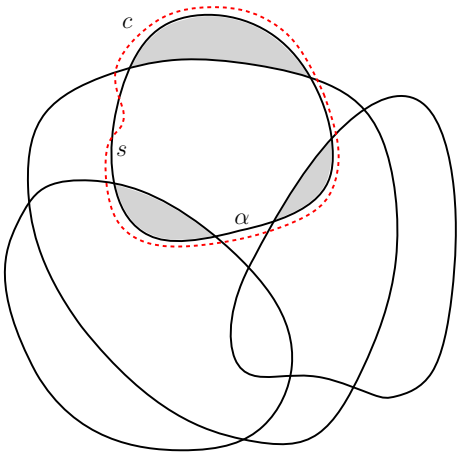


Figure 5 Extending $\mathcal{F}$ with a curve c which avoids the digons in $\mathcal{A}(\mathcal{F})$ and intersects every pseudocircle in $\mathcal{F}$ exactly twice.

For a pseudocircle α with a given orientation as a closed curve and points v_1, v_2 on α we denote by $[v_1, v_2]_\alpha$ (resp., $(v_1, v_2)_\alpha$) the closed (resp., open) segment of α from v_1 to v_2 following α in the given orientation.

▶ **Proposition 6.** *Let $\{\alpha_1, \alpha_2, c\}$ be a simple family of pairwise intersecting pseudocircles such that α_1 and α_2 form a digon. Suppose that c is oriented and let α_j^{in} (resp., α_j^{out}) be the intersection point of c and α_j in which c enters (resp., leaves) the digon-region of α_j, for $j = 1, 2$. Then $\alpha_1^{\text{in}}, \alpha_1^{\text{out}}, \alpha_2^{\text{in}}, \alpha_2^{\text{out}}$ is the cyclic order along c of the four intersection points of α_1 and α_2 with c.*

[1] Another way to think about this is to imagine that we work on a sphere, and we have caps corresponding to the digon-regions.

Proof. If $\alpha_2^{\text{in}} \in (\alpha_1^{\text{in}}, \alpha_1^{\text{out}})_c$ or $\alpha_2^{\text{out}} \in (\alpha_1^{\text{in}}, \alpha_1^{\text{out}})_c$, then we have a point from $\alpha_2 \cap c$ which lies in the digon-region of α_1. However, the intersection of α_2 with the digon-region of α_1 consists only of the subarc of α_2 which bounds the digon formed by α_1 and α_2. This implies that c intersects this digon which is impossible. Therefore, α_1^{out} must follow α_1^{in} in the cyclic order and, similarly, α_2^{out} must follow α_2^{in}. ◀

A *topological graph* is a graph drawn in the plane such that its vertices are drawn as distinct points and its edges are drawn as Jordan arcs connecting the corresponding points. Apart from its endpoints, an edge of a topological graph cannot contain any drawn vertex. Furthermore, every two edges in a topological graph intersect at a finite number of points, each of which is either a common endpoint or a crossing point. A graph is *planar* if it can be drawn as a topological graph where no pair of edges crosses. By the Strong Hanani-Tutte Theorem, it is enough to require an even number of crossings between independent edges.[2]

▶ **Theorem 7** (Strong Hanani-Tutte Theorem [20]). *A graph is planar if and only if it can be drawn as a topological graph in which every two independent edges cross an even number of times.*

3 Proof of Theorem 2

As we noted in the beginning of Section 2, we can suppose that $\mathcal{F}$ is a simple family of $n > 2$ pairwise intersecting pseudocircles in the plane such that every pseudocircle in $\mathcal{F}$ supports at least one digon. By Lemma 5 there is a simple closed curve c that intersects each pseudocircle in $\mathcal{F}$ twice but does not intersect the digons of the arrangement $\mathcal{A}(\mathcal{F})$. We fix the counterclockwise orientation on c. Whenever we traverse c or consider a cyclic order of points on it, we do it according to this orientation. Recall that c is an auxiliary pseudocircle and it is not a member of $\mathcal{F}$; in particular we do not care about digons supported by c.

Every pseudocircle $\alpha \in \mathcal{F}$ intersects c at two points such that at one of them c enters the digon-region of α and at the other c leaves this digon-region. We denote these points by α^{in} and α^{out}, respectively. See for example Figure 6. In this figure, the digon-regions of α and γ are the interior regions of these curves, whereas the digon-region of β is its exterior region.

3.1 The digon graph and its double cover

In order to bound from above the number of digons in $\mathcal{A}(\mathcal{F})$ we consider the graph G_d whose vertex set corresponds to the pseudocircles in $\mathcal{F}$ and whose edge set corresponds to pairs of pseudocircles which form a digon in $\mathcal{A}(\mathcal{F})$ (since $\mathcal{A}(\mathcal{F})$ is nontrivial and intersecting a pair of pseudocircles may not form more than one digon). We call G_d the *digon graph* of the arrangement. Bounding the number of edges in G_d is equivalent to bounding the number of digons in $\mathcal{A}(\mathcal{F})$.

For our proof, we consider the *bipartite double cover* of G_d - this graph, denote it by G, has two vertices, α^{in} and α^{out} for every vertex α of G_d, and two edges, $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\beta^{\text{out}}, \alpha^{\text{in}})$ for every edge (α, β) of G_d. Thus, the number of edges of G is twice the number of digons in $\mathcal{A}(\mathcal{F})$. By construction G is bipartite. It remains to show that G is also planar and therefore has at most $2|V(G)| - 4 = 4n - 4$ edges. This will prove Theorem 2.

[2] Two edges are *independent* if they do not share an endpoint.

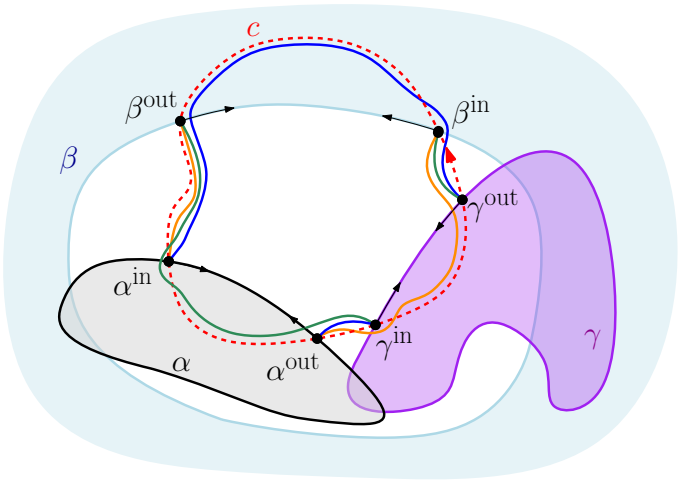


Figure 6 Three pseudocircles α, β and γ and a drawing of the corresponding bipartite double cover graph G. For example, α and γ form a digon, hence we draw the edges $(\alpha^{\text{out}}, \gamma^{\text{in}})$ and $(\gamma^{\text{out}}, \alpha^{\text{in}})$ along c. Considering the (blue) edge $(\gamma^{\text{out}}, \alpha^{\text{in}})$, starting at β^{in} (resp., β^{out}) and following β towards the interior of c, we first encounter α (resp., γ). Therefore, the drawn edge $(\gamma^{\text{out}}, \alpha^{\text{in}})$ "passes by" β^{in} (resp., β^{out}) within the exterior (resp., interior) of c. Each pseudocircle's label is placed within the digon-region associated with that pseudocircle.

3.2 Drawing G in the plane

We draw G as a topological graph in the plane. By abuse of notation, we will not distinguish between G and its drawing. The vertices of G are represented by the points $\{\alpha^{\text{in}} \mid \alpha \in \mathcal{F}\} \cup \{\alpha^{\text{out}} \mid \alpha \in \mathcal{F}\}$ as defined above. Thus, for every two pseudocircles $\alpha, \beta \in \mathcal{F}$ that form a digon in $\mathcal{A}(\mathcal{F})$ there should be two drawn edges $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\beta^{\text{out}}, \alpha^{\text{in}})$. These edges will satisfy the following properties:

- **(i)** An edge $(\alpha^{\text{out}}, \beta^{\text{in}})$ connects α^{out} and β^{in}, while avoiding all other vertices of G;
- **(ii)** every two edges intersect finitely many times; and
- **(iii)** an edge $(\alpha^{\text{out}}, \beta^{\text{in}})$ follows very closely the subarc of c starting at α^{out} and ending at β^{in}, counterclockwise along c. It may cross the curve c several times, but it does not intersect the digon-regions of neither α nor β.

Properties (i), (ii), and (iii) can be easily satisfied. For the second part of (iii) note that Proposition 6 implies that when following c from α^{out} to β^{in} we remain outside the digon-regions of α and β.

We need one more property that the drawing of $(\alpha^{\text{out}}, \beta^{\text{in}})$ should satisfy. This is the most crucial property for the proof and it has to do with how $(\alpha^{\text{out}}, \beta^{\text{in}})$ is drawn with respect to the intermediate vertices of G along the arc on c from α^{out} to β^{in}. We use a modification of the drawing rule that appears in [3].

Let $v \in V$ be an intermediate vertex of G on c along the arc from α^{out} to β^{in}. We will need to indicate whether the edge $(\alpha^{\text{out}}, \beta^{\text{in}})$ passes near v in the interior region of c or in its exterior region. In the first case, we say that the edge *passes v from the inside* whereas in the second case it *passes v from the outside*. Other than these inside/outside decisions and properties (i)-(iii), $(\alpha^{\text{out}}, \beta^{\text{in}})$ can be drawn arbitrarily.

We determine whether $(\alpha^{\text{out}}, \beta^{\text{in}})$ passes v from the inside or from the outside according to the value $d(v, (\alpha^{\text{out}}, \beta^{\text{in}}))$ that we define in the following way. Let γ be the pseudocircle in $\mathcal{F}$ that intersects c at v. It follows from Proposition 6 that γ must be different from α and β.

We follow the curve γ starting from v in the direction *entering the interior region of* c. If we intersect α sooner than we intersect β, then $d(v, (\alpha^{\text{out}}, \beta^{\text{in}})) = 1$, otherwise $d(v, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$.[3] Since $\mathcal{F}$ is a simple family of pairwise intersecting curves, $d(v, (\alpha^{\text{out}}, \beta^{\text{in}}))$ is well-defined. If $d(v, (\alpha^{\text{out}}, \beta^{\text{in}})) = 1$, then we pass v from the inside. Otherwise $d(v, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$ and we pass v from the outside. See Figure 6 for examples.

Having described the drawing of the graph G in the plane, we will conclude the proof by showing that every two independent edges of G cross an even number of times which implies that G is planar by the Strong Hanani-Tutte Theorem (Theorem 7).

3.3 Independent edges cross evenly

Let $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ be two independent edges of G. The parity of the number of crossings between two edges depends solely on how one edge passes by a vertex of the other edge. For example, if the cyclic order of their endpoints is $\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}, \beta^{\text{in}}$, then the two edges cross evenly if and only if $(\alpha^{\text{out}}, \beta^{\text{in}})$ passes by both of γ^{out} and δ^{in} from the inside or both of them from the outside. Therefore, it is enough to consider all the possible topologically different arrangements of the at most four involved pseudocircles (actually five because of c) and verify that in each of them $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ cross an even number of times. Using a computer we have verified this, still, for the remainder of the section (and of the proof) we provide a non-computer-aided proof.

We begin by observing that it is enough to consider the case that the involved pseudocircles α, β, γ, and δ are distinct. Indeed, $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ are independent edges, therefore, $\alpha \neq \gamma$ and $\beta \neq \delta$. Still, it is possible that $\alpha = \delta$ or $\beta = \gamma$. If $\alpha = \delta$ and $\beta = \gamma$ then by Proposition 6 the edges do not cross. Otherwise, we add a new pseudocircle as a substitute in the following way. Suppose without loss of generality that $\beta = \gamma$ (if $\alpha = \delta$ we switch the labels α and γ and the labels β and δ). Then we add a new pseudocircle γ' that goes very close to $\beta = \gamma$ outside of the digon-region of $\beta = \gamma$ except for near the vertices of the digon formed by $\beta = \gamma$ and δ. There, the pseudocircle γ' goes into the digon-region of $\beta = \gamma$ thus destroying the digon formed by $\beta = \gamma$ and δ and creating a new digon with δ that is fully contained in the digon that was destroyed. See Figure 7 for an example.

The edge $(\gamma'^{\text{out}}, \delta^{\text{in}})$ starts at γ'^{out} and follows the edge $(\gamma^{\text{out}}, \delta^{\text{in}})$, while $(\alpha^{\text{out}}, \beta^{\text{in}})$ passes by γ'^{in} according the above rules. The number of crossings of the two independent edges $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma'^{\text{out}}, \delta^{\text{in}})$ is equal to that of the original edges because $(\gamma'^{\text{out}}, \delta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ are drawn in precisely the same way as far as passing from the inside or from the outside of intermediate vertices of G. Note that the new vertex γ'^{in} becomes an intermediate vertex for $(\alpha^{\text{out}}, \beta^{\text{in}})$, however, it is not an endpoint of $(\gamma'^{\text{out}}, \delta^{\text{in}})$, thus it does not matter whether $(\alpha^{\text{out}}, \beta^{\text{in}})$ passes it from the inside or from the outside.

Henceforth we assume that the pseudocircles α, β, γ and δ are distinct. We proceed by considering the six possible cyclic orders of the vertices α^{out}, β^{in}, γ^{out} and δ^{in}. If the cyclic order is $(\alpha^{\text{out}}, \beta^{\text{in}}, \gamma^{\text{out}}, \delta^{\text{in}})$, then $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ do not intersect. The cyclic orders $(\alpha^{\text{out}}, \gamma^{\text{out}}, \beta^{\text{in}}, \delta^{\text{in}})$ and $(\alpha^{\text{out}}, \delta^{\text{in}}, \beta^{\text{in}}, \gamma^{\text{out}})$ are symmetric (by switching the labels α and γ and the labels β and δ), and the same goes for the cyclic orders $(\alpha^{\text{out}}, \beta^{\text{in}}, \delta^{\text{in}}, \gamma^{\text{out}})$ and $(\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}, \beta^{\text{in}})$. Therefore, it is enough to consider the cyclic orders $(\alpha^{\text{out}}, \gamma^{\text{out}}, \beta^{\text{in}}, \delta^{\text{in}})$, $(\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}, \beta^{\text{in}})$ and $(\alpha^{\text{out}}, \delta^{\text{in}}, \gamma^{\text{out}}, \beta^{\text{in}})$.

[3] Thus, the value 1 or 2 represents whether we encounter the first or the second pseudocircle among the two pseudocircles of the ordered pair $(\alpha^{\text{out}}, \beta^{\text{in}})$.

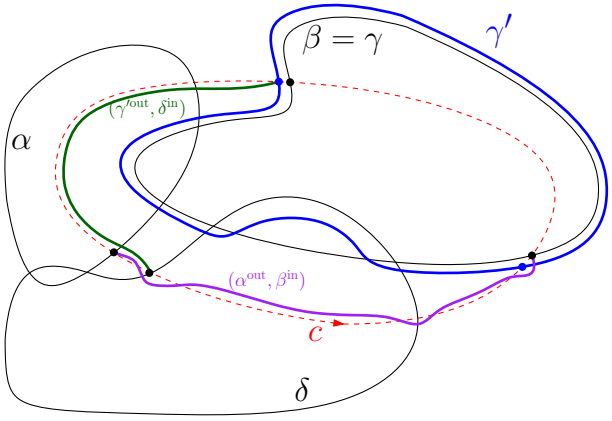


Figure 7 If $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ are edges and $\beta = \gamma$, then we can replace γ with γ' such that $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma'^{\text{out}}, \delta^{\text{in}})$ maintain the same number of crossings as the original edges. Each pseudocircle's label is placed within the digon-region associated with that pseudocircle.

Recall that we only care about how each edge passes by the vertices of the other edge. The following proposition summarizes the desired properties.

▶ **Proposition 8.** *Assume that the following conditions are satisfied:*

(A) *If $\alpha^{\text{out}}, \gamma^{\text{out}}, \beta^{\text{in}}$ and δ^{in}, appear in this cyclic order along c, then $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$;*

(B) *if $\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}$ and β^{in} appear in this cyclic order along c, then $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}}))$; and*

(C) *if $\alpha^{\text{out}}, \delta^{\text{in}}, \gamma^{\text{out}}$ and β^{in} appear in this cyclic order along c, then $d(\alpha^{\text{out}}, (\gamma^{\text{out}}, \delta^{\text{in}})) \neq d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}}))$ and $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$.*

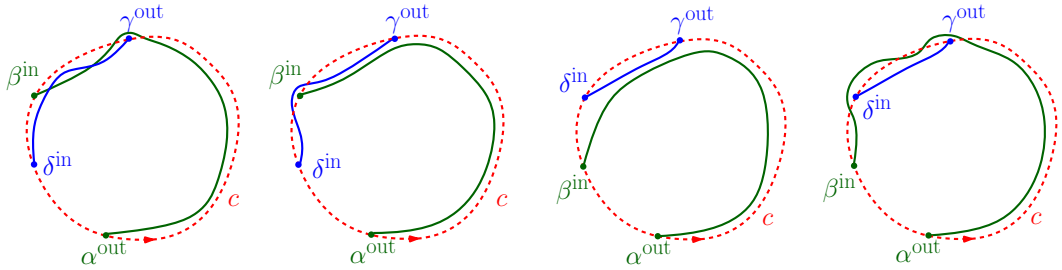
Then $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ cross an even number of times.

Proof. From properties (i), (ii) and (iii) it follows that the parity of the number of crossings between $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ depends only on the inside/outside decisions at the vertices $\alpha^{\text{out}}, \beta^{\text{in}}, \gamma^{\text{out}}$ and δ^{in}. This means that it is enough to check a single drawing for each set of inside/outside decisions that are allowed by the claim. The proof follows by a case analysis that is easy to verify by inspection, see Figure 8 for illustrations of the possible cases. ◀

It remains to prove that the drawing of G satisfies the conditions stated in Proposition 8. Conditions (A) and (C) are shown to hold in Proposition 9 while (B) is handled in Proposition 10. Note that in Condition (C) there is a symmetry between the two edges, namely, the cyclic order can also be written as $(\gamma^{\text{out}}, \beta^{\text{in}}, \alpha^{\text{out}}, \delta^{\text{in}})$. Therefore it is enough to prove, say, that $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$ holds for the given cyclic order of the endpoints, this is shown in the following Proposition 9.

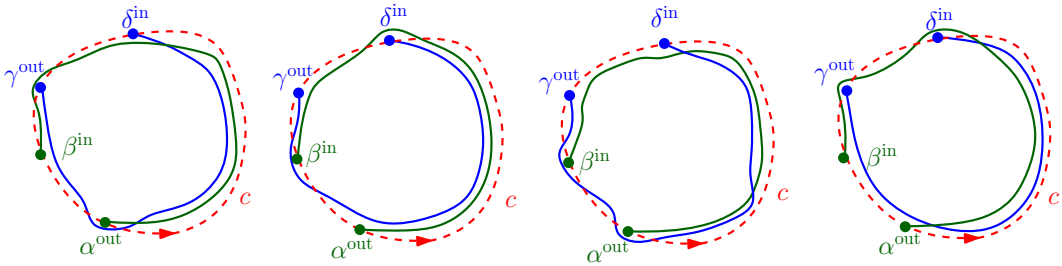
▶ **Proposition 9.** *If the cyclic of order of $\alpha^{\text{out}}, \beta^{\text{in}}, \gamma^{\text{out}}$ and δ^{in} along c is either $(\alpha^{\text{out}}, \delta^{\text{in}}, \gamma^{\text{out}}, \beta^{\text{in}})$ or $(\alpha^{\text{out}}, \gamma^{\text{out}}, \beta^{\text{in}}, \delta^{\text{in}})$, then $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$.*

Proof. Orient β and γ such that they enter the interior of c at β^{in} and γ^{out}, respectively (recall that $\beta \neq \gamma$). We will show that if $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$, then $d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}})) = 1$. A symmetric argument (reflect the arrangement and switch the role of β and γ and also of α and δ) shows that if $d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}})) = 1$, then $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$. Together, these two assertions imply the claim.



(a) The cyclic order $(\alpha^{\text{out}}, \gamma^{\text{out}}, \beta^{\text{in}}, \delta^{\text{in}})$ implies $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$.

(b) The cyclic order $(\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}, \beta^{\text{in}})$ implies $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}}))$.



(c) The cyclic order $(\alpha^{\text{out}}, \delta^{\text{in}}, \gamma^{\text{out}}, \beta^{\text{in}})$ implies $d(\alpha^{\text{out}}, (\gamma^{\text{out}}, \delta^{\text{in}})) \neq d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}}))$ and $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) \neq d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}}))$.

Figure 8 $(\alpha^{\text{out}}, \beta^{\text{in}})$ and $(\gamma^{\text{out}}, \delta^{\text{in}})$ cross evenly if the conditions of Proposition 8 hold.

Suppose that $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$. This means that as we follow γ starting from γ^{out} it intersects β before it intersects α. Let x denote this first intersection point of γ and β (see Figure 9). We will show that $[\beta^{\text{in}}, x]_\beta$ does not intersect δ, and hence $d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}})) = 1$.

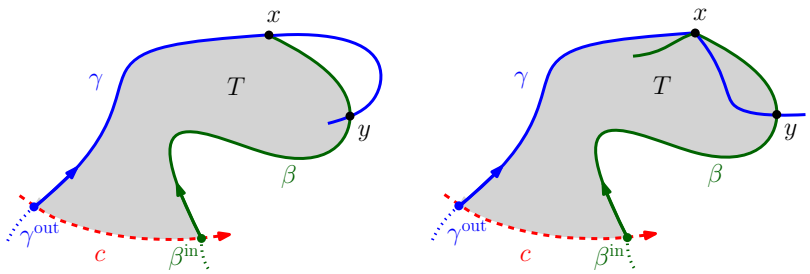
Let t be the closed curve which consists of $[\gamma^{\text{out}}, x]_\gamma$, $[\beta^{\text{in}}, x]_\beta$ and $[\gamma^{\text{out}}, \beta^{\text{in}}]_c$. We claim that t is a simple curve that is not crossed by any of α, β, and γ.

Consider first β and observe that it does not cross itself nor can it cross $[\gamma^{\text{out}}, x]_\gamma$ by the definition of x. From Proposition 6, it follows that $\beta^{\text{out}} \in (\beta^{\text{in}}, \alpha^{\text{out}})_c$ and therefore β does not cross $[\gamma^{\text{out}}, \beta^{\text{in}}]_c$, and hence it does not cross t.

Next, consider γ and observe that it does not cross itself nor can it cross $[\gamma^{\text{out}}, \beta^{\text{in}}]_c$ since it follows from Proposition 6 that $\gamma^{\text{in}} \in (\delta^{\text{in}}, \gamma^{\text{out}})_c$. Therefore, t is a simple curve (recall that β and $(\gamma^{\text{out}}, x)_\gamma$ cannot intersect by the definition of x). Let T be the "triangular" region which is bounded by t and is to the right of γ (and to the left of β and c).

Returning to γ, it remains to show that it cannot intersect $(\beta^{\text{in}}, x)_\beta$. Suppose for contradiction that it does intersect $(\beta^{\text{in}}, x)_\beta$ at a point y. Then either γ enters T at y or it leaves T at y. In the first case, following γ from y, it must leave T since the part of γ just before γ^{out} lies outside T. However, this means that γ must cross t and hence $(\beta^{\text{in}}, x)_\beta$ at another point which is impossible since β and γ already intersect at x and y, see Figure 9a. Considering the second case, if γ leaves T at y, then it implies that γ enters T at x, which in turn implies that β is also entering T at x (otherwise β and γ would touch at x which is impossible), see Figure 9b. However, following β from x, we would have to cross t since the part of β just before β^{in} lies outside T. Since we have already observed that β does not cross t we conclude that γ does not cross t either.

We now show that α cannot intersect t. By the definition of x it cannot intersect $[\gamma^{\text{out}}, x]_\gamma$. By Proposition 6 we have $\alpha^{\text{in}} \in (\beta^{\text{in}}, \alpha^{\text{out}})_c$ and it follows that α cannot intersect $[\gamma^{\text{out}}, \beta^{\text{in}}]_c$. Finally, if α crosses $[\beta^{\text{in}}, x]_\beta$, then it must cross it twice since α does not cross the other



(a) If γ enters T at y, then it must intersect $(\beta^{\text{in}}, x)_\beta$ once more.

(b) If γ leaves T at y, then β enters T at x and must cross t at some other point.

Figure 9 Illustrations for the proof of Proposition 9: when following γ into the interior of c we meet β sooner than α at a point x. Suppose for contradiction that γ intersects $[\beta^{\text{in}}, x]_\beta$ at another point y. Each pseudocircle's label is placed within the digon-region associated with that pseudocircle.

parts of t. The digon formed by α and β must lie outside of T since T lies outside the digon-region of β. Consequently, apart from its subarc that bounds this digon α is contained in T. However, this is impossible since then γ and α do not intersect.

We are now ready to show that δ cannot intersect $[\beta^{\text{in}}, x]_\beta$ which would imply that $d(\beta^{\text{in}}, (\gamma^{\text{out}}, \delta^{\text{in}})) = 1$. Assume to the contrary that δ intersects $[\beta^{\text{in}}, x]_\beta$. Notice that $\delta^{\text{in}}, \delta^{\text{out}}, \gamma^{\text{out}}$, and β^{in} appear in this cyclic order on c. This follows from Proposition 6 and the assumptions of the claim. This implies that δ does not intersect $[\gamma^{\text{out}}, \beta^{\text{in}}]_c$. It follows that the two simple closed curves δ and t must either cross twice or four times, since crossing six times or more would imply that δ intersects γ or β more than twice. If δ and t cross four times, then δ crosses twice each of $[\beta^{\text{in}}, x]_\beta$ and $[\gamma^{\text{out}}, x]_\gamma$. Therefore, the only part of δ that is outside the digon-region of γ and the digon-region of β must be in T. The curves α and δ cannot cross in the digon-region of γ, because δ and γ form a digon. Similarly, α and δ cannot cross in the digon-region of β because α and β form a digon. Since α is disjoint from T we conclude that α and δ do not intersect which is a contradiction. We reach a similar contradiction in the case where δ crosses t precisely at two points that happen to be on $[\beta^{\text{in}}, x]_\beta$, once again α and δ cannot intersect.

It remains to consider the case where δ crosses $[\beta^{\text{in}}, x]_\beta$ once and crosses $[\gamma^{\text{out}}, x]_\gamma$ once. The latter implies that either x or γ^{out} belong to the digon formed by δ and γ, which is impossible since neither β nor c intersects this digon. ◀

▶ **Proposition 10.** *If $\alpha^{\text{out}}, \gamma^{\text{out}}, \delta^{\text{in}}$ and β^{in} appear in this cyclic order along c, then we have $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}}))$.*

Proof. Orient γ and δ such that they enter the interior region of c at γ^{out} and δ^{in}, respectively. Recall that c is oriented counterclockwise and its interior region lies to its left. Thus, the digon-region of γ lies to its left whereas the digon-region of δ lies to its right. Let $\{A'_1, A'_2\} = \delta \cap \alpha$ and $\{B'_1, B'_2\} = \delta \cap \beta$, such that the cyclic order of these four points along δ is A'_1, A'_2, B'_1, B'_2 (since α and β form a digon it follows from Proposition 6 that A'_1 and A'_2 must be consecutive and so are B'_1 and B'_2). Similarly, let $\{A''_1, A''_2\} = \gamma \cap \alpha$ and $\{B''_1, B''_2\} = \gamma \cap \beta$, such that the cyclic order of these four points along γ is $A''_1, A''_2, B''_1, B''_2$.

Considering the cyclic order of A'_1, A'_2, A''_1 and A''_2 along α (under some orientation of α), it follows from Proposition 6 that the first two are consecutive and so are the last two (since γ and δ form a digon). Therefore, if we orient α such that A'_2 follows A'_1, then the cyclic order of those four points is either A'_1, A'_2, A''_1, A''_2 or A'_1, A'_2, A''_2, A''_1.

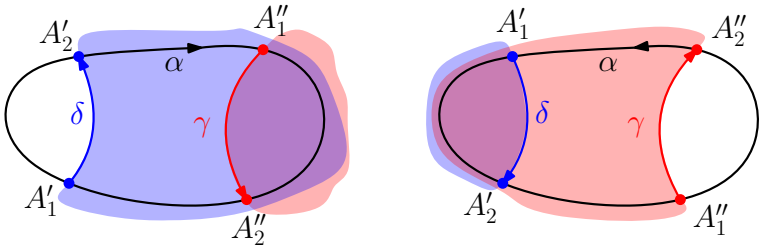


Figure 10 Suppose that A_1', A_2', A_1'', A_2'' appear in this cyclic order on α and B_1', B_2', B_1'' and B_2'' are in the exterior of α. Then the intersection of the digon-region of δ (which is to its right) and the digon-region of γ (which is to its left) contains either $[A_1'', A_2'']_\alpha$ or $[A_1', A_2']_\alpha$ which is impossible.

We claim that the former is impossible. Indeed, suppose that A_1', A_2', A_1'' and A_2'' appear in this cyclic order on α. Note that it follows from the cyclic order on δ (resp., γ) that both of B_1' and B_2' (resp., B_1'' and B_2'') must be in the same region (interior or exterior) of α. If B_1', B_2', B_1'' and B_2'' are in the exterior of α, then either $[A_1'', A_2'']_\alpha$ or $[A_1', A_2']_\alpha$ is contained in the intersection of the digon-regions of γ and δ (see Figure 10), which is impossible. By a symmetric argument, it is impossible that B_1', B_2', B_1'', and B_2'' are in the interior of α. Therefore, either B_1' and B_2' are in the digon-region of α or B_1'' and B_2'' are there. However, this implies that the subarc of β in the digon-region of α intersects either γ or δ which is impossible since α and β form a digon.

Thus $A_1', A_2'', A_2'', A_1''$ appear in this cyclic order on α, and similarly, B_1', B_2', B_2'', B_1'' appear in this cyclic order on β if it is oriented such that B_2' immediately follows B_1'. Next, we redraw α close to the digon it forms with β such that these two pseudocircles become disjoint (see Figure 11). In a similar way, we redraw δ such that δ and γ become disjoint. Note that no other intersection points but $\alpha \cap \beta$ and $\gamma \cap \delta$ are destroyed and no new intersection points are introduced. Therefore, the vertex set of $\mathcal{A}(\{\alpha, \beta, \gamma, \delta\})$ is precisely $\{A_1', A_2', A_1'', A_2'', B_1', B_2', B_1'', B_2''\}$ and its edge set consists of the edges $[A_1', A_2']_\alpha$, $[A_2', A_2'']_\alpha$, $[A_2'', A_1'']_\alpha$, $[A_1'', A_1']_\alpha$, $[B_1', B_2']_\beta$, $[B_2', B_2'']_\beta$, $[B_2'', B_1'']_\beta$, $[B_1'', B_1']_\beta$, $[A_1'', A_2'']_\gamma$, $[A_2'', B_1'']_\gamma$, $[B_1'', B_2'']_\gamma$, $[B_2'', A_1'']_\gamma$, $[A_1', A_2']_\delta$, $[A_2', B_1']_\delta$, $[B_1', B_2']_\delta$ and $[B_2', A_1']_\delta$. It is not hard to see that this implies that the face set of $\mathcal{A}(\{\alpha, \beta, \gamma, \delta\})$ consists of four digons whose boundaries are $\{[A_1', A_2']_\alpha, [A_1', A_2']_\delta\}$, $\{[A_1'', A_2'']_\alpha, [A_1'', A_2'']_\gamma\}$, $\{[B_1', B_2']_\beta, [B_1', B_2']_\delta\}$ and $\{[B_1'', B_2'']_\beta, [B_1'', B_2'']_\gamma\}$, and six faces of size four whose boundaries are (refer to Figure 11 for an illustration):
$\{[A_1', A_2']_\delta, [A_2', A_2'']_\alpha, [A_2'', A_1'']_\gamma, [A_1'', A_1']_\alpha\}$, $\{[A_1', A_2']_\alpha, [A_2', B_1']_\delta, [B_1', B_2']_\beta, [B_2', A_1']_\delta\}$,
$\{[B_1', B_2']_\delta, [B_2', B_2'']_\beta, [B_2'', B_1'']_\gamma, [B_1'', B_1']_\beta\}$, $\{[A_1'', A_2'']_\alpha, [A_2'', B_1'']_\gamma, [B_1'', B_2'']_\beta, [B_2'', A_1'']_\delta\}$,
$\{[A_1', A_1'']_\alpha, [A_1'', B_2'']_\gamma, [B_2'', B_2']_\beta, [B_2', A_1']_\delta\}$ and $\{[A_2', A_2'']_\alpha, [A_2'', B_1'']_\gamma, [B_1'', B_1']_\beta, [B_1', A_2']_\delta\}$.

Note that $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 1$ and $d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$ iff $\gamma^{\text{out}} \in (B_2'', A_2'')_\gamma$ and $\delta^{\text{in}} \in (A_2', B_2')_\delta$. Similarly, $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$ and $d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 1$ iff $\gamma^{\text{out}} \in (A_2'', B_2'')_\gamma$ and $\delta^{\text{in}} \in (B_2', A_2')_\delta$. Suppose for contradiction that one of these cases holds. Then, on the one hand, there is a curve, namely $[\gamma^{\text{out}}, \delta^{\text{in}}]_c$, which connects γ^{out} and δ^{in} while not crossing any of α, β, γ and δ (this follows from the given cyclic order and Proposition 6). However, on the other hand, there is no face of $\mathcal{A}(\{\alpha, \beta, \gamma, \delta\})$ whose boundary contains segments of both $(B_2'', A_2'')_\gamma$ and $(A_2', B_2')_\delta$ or segments of both $(A_2'', B_2'')_\gamma$ and $(B_2', A_2')_\delta$, contradicting the existence of a curve as above. ◀

This completes the proof of Theorem 2.

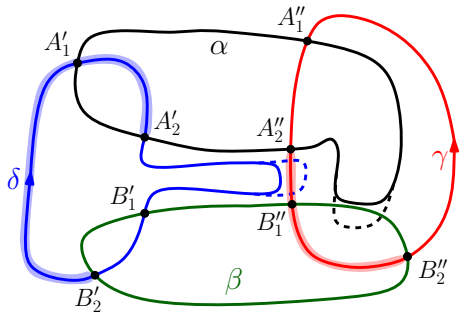


Figure 11 $\mathcal{A}(\{\alpha, \beta, \gamma, \delta\})$ after redrawing α and δ such that they do not form digons with β and γ, respectively. If $d(\gamma^{\text{out}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 2$ and $d(\delta^{\text{in}}, (\alpha^{\text{out}}, \beta^{\text{in}})) = 1$, then $\gamma^{\text{out}} \in (A_2'', B_2'')_\gamma$ and $\delta^{\text{in}} \in (B_2', A_2')_\delta$.

4 Concluding remarks

What can we say about drawings on other surfaces? For example, we can draw K_7 on the torus, which implies the existence of 7 curves such that they pairwise form a digon, giving us 21 digons in total. This suggests that we need a significantly different approach since an argument based on the embeddability of the double cover would give roughly $2n$ as an upper bound.

▶ **Problem 11.** *What is the maximum number of digons in a pairwise intersecting family of pseudocircles on a closed orientable connected surface M_g of genus g?*

We note that graphs with planar bipartite double covers were extensively studied in topological graph theory, see for example the works of Negami [14, 15]. Negami also considered general covers of graphs and formulated the following nice conjecture:

▶ **Conjecture 12** (Negami 1988). *A graph G has a finite planar cover if and only if G embeds in the projective plane.*

Indeed, with some extra work, one can show that digon graphs are embeddable in the projective plane, but we omit the proof.

Another natural question is to consider the number of triangles in intersecting pseudocircle arrangements. Felsner and Scheucher [9] showed that the maximal number of triangles is $2n^2/3+O(n)$, and this is asymptotically tight. In their proof, they use the bound $O(n)$ from [3] for the maximum number of digons in pairwise intersecting arrangements of pseudocircles. By applying Theorem 2 instead, we achieve the following improvement for simple arrangements.

▶ **Corollary 13.** *Every simple arrangement of $n \geqslant 4$ pairwise intersecting pseudocircles has at most $\frac{2}{3}n^2$ triangles.*

On the other hand, triangles are unavoidable, they showed that there are at least $2n/3$ triangles and made the subsequent conjecture.

▶ **Conjecture 14.** *Every pairwise intersecting arrangement of $n \geqslant 3$ pseudocircles has at least $n - 1$ triangles.*

On a final remark, we were able to extend our proof of Theorem 2 to nontrivial pairwise intersecting arrangements of pseudocircles. However, it involves dealing with several delicate subcases. Therefore, we have chosen to omit the full proof here.

References

1 E. Ackerman, G. Damásdi, B. Keszegh, R. Pinchasi, and R. Raffay. On the Number of Digons in Arrangements of Pairwise Intersecting Circles. In *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 3:1–14, 2024. DOI: 10.4230/LIPIcs.SoCG.2024.3.

2 E. Ackerman, G. Damásdi, B. Keszegh, R. Pinchasi, and R. Raffay. On the Number of Digons in Arrangements of Pairwise Intersecting Circles. *Combinatorica*, Submitted.

3 P. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudocircles and their applications. *J. ACM*, 51:139–186, 2004. DOI: 10.1145/972639.972641. doi:10.1145/972639.972641.

4 N. Alon, H. Last, R. Pinchasi, and M. Sharir. On the complexity of arrangements of circles in the plane. *Discrete and Computational Geometry*, 26:465–492, 2001. doi:10.1007/S00454-001-0043-X.

5 D. Archdeacon and K. Stor. Superthrackles. *Australasian Journal of Combinatorics*, 67(2):145–158, 2017. URL: http://ajc.maths.uq.edu.au/pdf/67/ajc_v67_p145.pdf.

6 G. Cairns and Y. Nikolayevsky. Bounds for generalized thrackles. *Discret. Comput. Geom.*, 23:191–206, 2000. doi:10.1007/PL00009495.

7 P. Erdős. On sets of distances of n points. *Amer. Math. Monthly*, 53:248–250, 1946. doi:10.1080/00029890.1970.11992573.

8 S. Felsner, S. Roch, and M. Scheucher. Arrangements of pseudocircles: On digons and triangles. In *Int. Symp. on Graph Drawing and Network Visualization*, volume 13764 of *Lecture Notes in Computer Science*. Springer, Cham, 2022. doi:10.1007/978-3-031-22203-0-32.

9 S. Felsner and M. Scheucher. Arrangements of pseudocircles: Triangles and drawings. *Discrete Comput. Geom.*, 65(1):261–278, 2021. doi:10.1007/S00454-020-00173-4.

10 Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of Discrete and Computational Geometry, Third Edition*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, 2018. doi:10.1201/9781315119601.

11 B. Grünbaum. *Arrangements and Spreads*, volume 10 of *CBMS Regional Conference Series in Mathematics*. AMS, 1972. doi:10.1090/cbms/010.

12 M. Katchalski and H. Last. On geometric graphs with no two edges in convex position. *Discrete Comput. Geom.*, 19(3):399–404, 1998. doi:10.1007/PL00009357.

13 A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications. *J. Combin. Theory Ser. A*, 113(4):675–691, 2006. doi:10.1016/J.JCTA.2005.07.002.

14 S. Negami. Enumeration of projective-planar embeddings of graphs. *Discrete Mathematics*, 62:299–306, 1986. doi:10.1016/0012-365X(86)90217-7.

15 S. Negami. Projective-planar graphs with even duals. *Journal Of Graph Theory*, 16:287–295, 1992. doi:10.1002/JGT.3190160402.

16 R. Pinchasi. Gallai-Sylvester theorem for pairwise intersecting unit circles. *Discrete and Computational Geometry*, 28:607–624, 2002. doi:10.1007/S00454-002-2892-3.

17 R. Pinchasi. Geometric graphs with no two parallel edges. *Combinatorica*, 28(1):127–130, 2008. doi:10.1007/S00493-008-2250-Z.

18 R. Pinchasi. A note on lenses in arrangements of pairwise intersecting circles in the plane. *Elec. J. of Comb*, Submitted.

19 R. Pinchasi and R. Radoičić. On the number of edges in a topological graph with no self-intersecting cycle of length 4. In *Towards a Theory of Geometric Graphs*, volume 342 of *Contemp. Math.*, pages 233–243. Amer. Math. Soc., Providence, RI, 2004. doi:10.1090/conm/342/06143.

20 W. Tutte. Toward a theory of crossing numbers. *J. Comb. Theory*, 8:45–53, 1970. doi:10.1016/s0021-9800(70)80007-2.

21 P. Valtr. On geometric graphs with no k pairwise parallel edges. *Discrete Comput. Geom.*, 19(3):461–469, 1998. doi:10.1007/PL00009364.

Convexity Helps Iterated Search in 3D

Peyman Afshani
Department of Computer Science, Aarhus University, Denmark

Yakov Nekrich
Department of Computer Science, Michigan Technological University, Houghton, MI, USA

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

Inspired by the classical fractional cascading technique [13, 14], we introduce new techniques to speed up the following type of iterated search in 3D: The input is a graph $\mathbf{G}$ with bounded degree together with a set H_v of 3D hyperplanes associated with every vertex of v of $\mathbf{G}$. The goal is to store the input such that given a query point $q \in \mathbb{R}^3$ and a connected subgraph $\mathbf{H} \subset \mathbf{G}$, we can decide if q is below or above the lower envelope of H_v for every $v \in \mathbf{H}$. We show that using linear space, it is possible to answer queries in roughly $O(\log n + |\mathbf{H}|\sqrt{\log n})$ time which improves trivial bound of $O(|\mathbf{H}| \log n)$ obtained by using planar point location data structures. Our data structure can in fact answer more general queries (it combines with shallow cuttings) and it even works when $\mathbf{H}$ is given one vertex at a time. We show that this has a number of new applications and in particular, we give improved solutions to a set of natural data structure problems that up to our knowledge had not seen any improvements.

We believe this is a very surprising result because obtaining similar results for the planar point location problem was known to be impossible [15].

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

Keywords and phrases Data structures, range searching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.3

Related Version *Full Version*: `https://arxiv.org/abs/2504.07545` [5]

Funding *Peyman Afshani*: Supported by DFF (Danmarks Frie Forskningsfond) of Danish Council for Independent Research under grant ID 10.46540/3103-00334B.

1 Introduction

The idea of "iterated searching" was introduced with the classical fractional cascading technique [13, 14] and it can be described as follows: consider an abstract data structure problem where given an input I, we need to preprocess it to answer a given q more efficiently. Now imagine that instead of having one input set, I, we have multiple independent input sets $I_1, I_2, \cdots$ and at the query time, we would like to answer the same query q on *some* of the data sets. This is typically modeled as follows: the input contains a catalog graph $\mathbf{G}$ and each vertex $v \in \mathbf{G}$ is associated with a different input set I_v. The goal is to preprocess the input, that is $\mathbf{G}$ and the sets I_v, so that given a query q and a connected subgraph $\mathbf{H} \subset \mathbf{G}$ we can find the answer to querying q on every data set I_v for each $v \in \mathbf{H}$. The trivial solution is to have a separate data structure for every I_v and to query them individually. Thus, the main research interest is to improve upon this trivial solution.

1.1 Fractional Cascading

Chazelle and Guibas improved upon the aforementioned "trivial solution" for the predecessor search problem. To be more precise, for each $v \in \mathbf{G}$, I_v is an ordered list of values and the query q is also a value and the goal is to find the predecessor of q in I_v for every $v \in \mathbf{H}$.

© Peyman Afshani, Yakov Nekrich, and Frank Staals;
licensed under Creative Commons License CC-BY 4.0

Chazelle and Guisbas showed how to build a data structure that uses linear space to answer such queries in $O(\log n + |\mathbf{H}|)$ time [13, 14], essentially, reducing the cost of a predecessor search to $O(1)$, after an initial investment of $O(\log n)$ time. This is clearly optimal and since its introduction has found plenty of applications.

The origin of this technique can be traced back to 1982 and to a paper by Vaishnai and Wood [22] and also the notion of "downpointers" from a 1985 paper by Willard [23] but the seminal work in this area was the aforementioned work by Chazelle and Guibas [13, 14].

1.2 2D Fractional Cascading

Due to its numerous applications, Chazelle and Liu [15] studied "2D fractional cascading" which is the iterated 2D point location problem. In particular, each vertex $v \in \mathbf{G}$ is associated with a planar subdivision, the query q is a point in $\mathbb{R}^2$, and the goal is to output for every $v \in \mathbf{H}$ the cell of the subdivion of v that contains q. Chazelle and Liu showed that in this case all hope is lost because there is an almost quadratic space lower bound for data structures with $O(\log n + |\mathbf{H}|)$ query time. Actually, their techniques prove an $\Omega(n^{2-\varepsilon})$ space lower bound for data structures with $O(\log^{O(1)} n) + o(|\mathbf{H}| \log n)$ query time. As $O(|\mathbf{H}| \log n)$ is the trivial upper bound, the lower bound "dashes all such hopes" (as Chazelle and Liu put it) for obtaining a general technique in dimensions two and beyond. However, the lower bound only applies to non-orthogonal subdivisions and only for data structures for sub-quadratic space consumption. Relatively recently, the barrier has been overcome via exploring those two directions. Afshani and Cheng [3] studied various cases involving orthogonal subdivisions and obtained almost tight results, e.g., the problem can be solved using linear space and with roughly $O(\log n + |\mathbf{H}|\sqrt{\log n})$ query time for axis-aligned subdivisions. Chan and Zheng [11] studied the non-orthogonal version but they allowed quadratic space usage and they showed that queries can be answered in $O(\log n + |\mathbf{H}|)$ time. Finally, we need to mention that a more general form of iterated search which allows $\mathbf{H}$ to be any subgraph of $\mathbf{G}$ has also been studied for 2D queries but the bounds are weaker [7].

1.3 A Brief Statement of Our Results

We introduce a general technique to solve the iterated search problem for a certain 3D query problem. Then, we show that our technique has a number of non-trivial applications.

1.3.1 A General Framework for Iterated Searching for Lower Envelopes

The exact statement of our technique requires familiarity with concepts such as shallow cuttings and therefore we postpone it to after the "Preliminaries" section. Here, we only state it informally: we study the *lower envelope iterated search (LEIS) problem*, where we are given a catalog graph $\mathbf{G}$ and each vertex $v \in \mathbf{G}$ is associated with a catalog which is a set H_v of hyperplanes. The goal is to preprocess the input into a data structure such that given a query point q and a connected subgraph $\mathbf{H} \subset \mathbf{G}$, the data structure should report whether q is below or above the lower envelope of H_v, for all $v \in \mathbf{H}$. In the shallow cutting variant of LEIS, the input has a parameter k and the data structure stores a number of lists of size $O(k)$ explicitly. Then at the query time the data structure has two options: (i) report that q is above the k-level of H_v or (ii) return a constant number, c, of pointers to c stored lists $L_1, \cdots, L_c$ such that the union of $L_1, \cdots, L_c$ contains all the hyperplanes that pass below q. In the *on-demand* version of the problem, $\mathbf{H}$ is given one vertex v at a time such that all the given vertices form a connected subgraph of $\mathbf{G}$ but the query must answer q on v before the next vertex is given. See Figure 1 for an illustration of our problem.

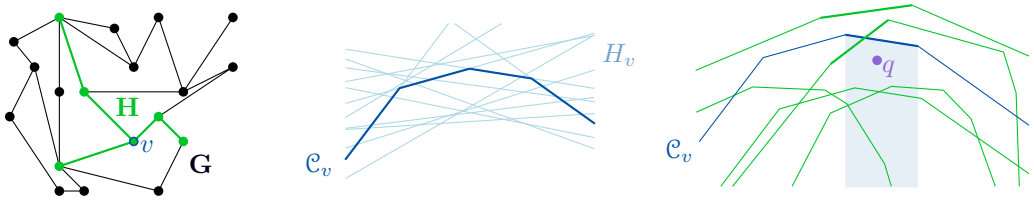


Figure 1 A schematic drawing of the LEIS Problem (with hyperplanes in $\mathbb{R}^2$ rather than $\mathbb{R}^3$). Each vertex v of $\mathbf{G}$ is associated with a set of hyperplanes H_v. For each vertex v in the query subgraph $\mathbf{H} \subset \mathbf{G}$ we wish to find the triangle in the k-shallow cutting $\mathcal{C}_v$ above the query point q.

Our main result is that all the above variants can be solved with a linear-space data structure that has $O(\log n + |\mathbf{H}|\sqrt{\log n})$ query time. This improves upon the trivial solution which results in $O(|\mathbf{H}| \log n)$ query time.

We believe this is a surprising result since a LEIS query (which is very commonly used in combination with shallow cuttings, see Lemma 3) is typically answered via a point location query and as Chazelle and Liu's lower bound [15] shows, it is hopeless to have an efficient data structure for iterated point location queries.

1.3.2 Some Applications of Our Technique

Chazelle and Liu [15] cite numerous potential applications of iterated 2D point location, if such a thing were to exist, before presenting their lower bound. The lower bound applies to some of those applications but not to all of them. In particular, it turns out that for some geometric problems, we are able to give improvements. One main problem is answering *halfspace max queries* which currently has only a basic "folklore" solution in 3D, despite being used in "blackbox" reductions in at least two general frameworks (more on this below).

▶ **Problem 1** (3D halfspace max queries). *Let P be an input set of n points in $\mathbb{R}^3$ each associated with a real-valued weight. Store them in a data structure such that given a query halfspace q one can find the point with the largest weight inside q.*

Currently, we are only aware of a "folklore" solution that uses $O(n \log n)$ space and has the query time of $O(\log^2 n)$ [6]. Using our framework, we get the following improvements.

- 3D halfspace max queries can be answered in $O(\log^{3/2} n)$ time with $O(n \log n)$ space.
- The current best solution for *3D halfspace range sampling* uses a "blackbox" data structure for weighted range sampling queries [6]. As a result, we improve the query time of the data structure in [6] from $O(\log^2 + k)$ (for sampling k points from a given query halfspace) to $O(\log^{3/2} n + k)$ by simply plugging in our solution for 3D halfspace max queries.
- Range sampling queries can be used as a blackbox to obtain solutions for other approximate aggregate queries [4] and thus our improve carries over to such queries as well.
- We get improved results for *colored halfspace reporting*, in which we store a set of n colored points so that given a query halfspace we can efficiently report all k distinct colors of the points that appear in the halfspace. We can achieve $O(\log n \log\log n + m \log\log n)$ query time using linear space, where m is the number of colors, or $O(\log n + k \log(m/k)\sqrt{\log n})$ time using $O(n \log n)$ space. This improves the query time of current solutions by a $O(\log n)$, respectively $O(\sqrt{\log n})$ factor.
- Some weighted range reporting problems can also be solved more efficiently: (*weighted halfspace range reporting*) store a given set of weighted points such that given a query halfspace q and an interval $[w_1, w_2]$, report all the points inside q whose weight is between

w_1 and w_2. (*weighted halfspace stabbing*) This is similar to the previous problem with the role of points and halfspaces swapped.

- The general framework of [19] combines weighted reporting and max reporting to give a general solution for "top-k reporting" (report the k heaviest points). Using our results, we can obtain a data structure for 3D halfspaces with $O(\log^{3/2} n + k)$ query time. This was the result "missing" in [19] where they only list new results for $d = 2$ and $d \geq 4$.
- Range reporting can also be used to obtain "approximate counting" results and a number of general reductions are available [18, 8, 17, 1] and so we can obtain improved results for "approximate counting" version of the above weighted reporting problems.
- The *offline* version of halfspace range reporting can also be improved: given a set P of n points, where a point p_i is given with an insertion time s_i and a deletion time d_i, store them in a data structure such that given a query halfspace h and a timestamp t, we can report all the points in h that exist in time t.
- Via standard lifting maps, problems involving circles in 2D can be reduced to those involving halfspaces in 3D.

2 Preliminaries

We adopt the following notations and definitions. In $\mathbb{R}^3$, a *triangle* is the convex hull of three points whereas a *simplex* is the convex hull of four points. The Z-coordinate is assumed to be the vertical direction and a *vertical prism* $\tau^{\downarrow}$ is defined by a triangle τ and it consists of all the points that are on τ or directly below τ (wrt to the Z-coordinate). For a point $q \in \mathbb{R}^3$, $q^{\downarrow}$ represents a downward ray starting from q. Given a set A of geometric objects and another geometric object r, the *conflict list* of r wrt A, denoted by $\Diamond(A, r)$, is the subset of A that intersects R. For example, if A is a set of polytopes and r is a point, then $\Diamond(A, r)$ is the set of of polytopes that contain r or if A is a set of hyperplanes and r is a point, then $\Diamond(A, r^{\downarrow})$ is the subset of hyperplanes that pass through or below r; in this latter case, $|\Diamond(A, r^{\downarrow})|$ is denoted as the *level* of r (wrt A). For a given parameter k, $1 \leq k \leq n$, the $(\leq k)$-level of A is the set of all the points in $\mathbb{R}^3$ with level at most k and the k-level of A is defined as the boundary of the $(\leq k)$-level; in particular, the 0-level of A is the lower envelope of A.

▶ **Observation 2** ([9]). *Let τ be a triangle and let $\Delta = \tau^{\downarrow}$ be the vertical prism defined by it. Let v_1, v_2, and v_3 be the vertices of τ. Any hyperplanes that intersects Δ, passes below at least one of the vertices of τ. In other words, for any set H of hyperplanes we have*

$$\Diamond(H, \Delta) \subset \Diamond(H, v_1^{\downarrow}) \cup \Diamond(H, v_2^{\downarrow}) \cup \Diamond(H, v_3^{\downarrow}).$$

We will extensively use shallow cuttings in our techniques. By now, this is a standard tool that is used very often in various 3D range searching problems.

▶ **Lemma 3** ([10, 20, 21]). *There exists a fixed constant $\alpha > 1$ such that for any given set H of n hyperplanes in 3D and a given parameter k, $1 \leq k \leq n$, the following holds: There exists a set of $O(n/k)$ hyperplanes such that their lower envelope lies above the k-level of H but below the (αk)-level of H. The lower envelope can be triangulated and for every resulting triangle τ, one can also build the conflict list $\Diamond(H, \tau^{\downarrow})$ in $O(n \log n)$ time in total (over all triangles τ). The shallow cutting $\mathcal{C}$ is taken to be the set of all the triangles and each triangle stores a pointer to its explicitly stored conflict list.*

Finally, one can store the projection of the shallow cutting in a 2D point location data structure that uses $O(n/k)$ space such that given a query point $q \in \mathbb{R}^3$, in $O(\log(n/k))$ time one can decide whether q lies below any triangle in $\mathcal{C}$ and if it does, then one can find the triangle directly above q, denoted by $\mathcal{C}(q)$. The subset of hyperplanes passing below q is a subset of the conflict list of $\mathcal{C}(q)$, i.e., $\Diamond(H, q^{\downarrow}) \subset \Diamond(H, \mathcal{C}(q)^{\downarrow})$.

Remark. The above is a "modern" statement of shallow cuttings in 3D. These were originally discovered by Matoušek [21]. However, a simple observation by Chan [9] (Observation 2) allows one to simplify the structure of shallow cuttings into the lower envelope of some hyperplanes. We will be strongly using convexity in our techniques so this treatment is extremely crucial for our purposes.

We now define the main iterated search problems that we will be studying.

▶ **Problem 4** (LEIS3D). *Let* $\mathbf{G}$ *be an input graph of constant degree, called the catalog graph, where each vertex* $v \in G$ *is associated with an input set* H_v *of hyperplanes in* $\mathbb{R}^3$*. Let* n *be the total input size and let* k*,* $1 \leq k \leq n$*, be a parameter given with the input. Store the input in a structure to answer queries that are given by a point* $q \in \mathbb{R}^3$ *and a connected subgraph* $\mathbf{H} \subset \mathbf{G}$*. For each vertex* $v \in \mathbf{G}$*, the data structure should store a* k*-shallow cutting of* H_v*. Then, for each* $v \in \mathbf{H}$ *at the query time, it must either (i) return a triangle* τ *above* q *together with a list of* c *pointers, for a constant* c*, to* c *conflict lists* $L_1, \cdots, L_c$ *in the* k*-shallow cutting of* H_v*, such that* $\Diamond(H_v, \tau) \subset \cup_{i=1}^{c} L_i$ *or (ii) indicate that* q *lies above the* k*-level of* H_v*.*

In the anchored version of the problem, for every $v \in \mathbf{G}$ *the data structure must explicitly store a set* T_v *of triangles such that the triangle* τ_v *returned at query time must be an element of* T_v*. In the on-demand version of the problem,* q *is given first but* $\mathbf{H}$ *is initially hidden and is revealed one vertex at a time at query time. The data structure must answer the query on each revealed vertex before the next vertex can be revealed. For simplicity, we assume that* $\mathbf{H}$ *is a walk, i.e., each revealed vertex is connected to the previously revealed vertex.*

Note for $k = 0$, we have a special case where during the query it suffices to simply return a triangle τ that is below the lower envelope and above q. Also the restriction that $\mathbf{H}$ must be a walk for the on-demand version is not strictly necessary in RAM models. However, removing this restriction creates some technical issues in some variants of the pointer machine model and thus it is added to avoid needless complications. Furthermore, a walk is sufficient in all the applications that we will consider. Our main technical result is the following.

▶ **Theorem 5.** *The on-demand LEIS3D problem for an input of size* n *on catalog graph* $\mathbf{G}$ *can be solved with a data structure that uses* $O(n)$ *space and has the following query time: (i)* $O(\log n \log\log n + |\mathbf{H}| \log\log n)$ *if* $\mathbf{G}$ *is a path and (ii)* $O(\log n + |\mathbf{H}|\sqrt{\log n})$ *when* $\mathbf{G}$ *is a graph of constant degree.*

3 Building the Tools

In this section, we will prove a number of auxiliary results that will later be combined to build our main results.

3.1 Point Location in 3D Convex Overlays

The following observation is our starting point. The proof is not too difficult and is available in the full version [5].

▶ **Lemma 6.** *Let* $\mathcal{P}_1, \cdots, \mathcal{P}_m$ *be* m *convex polytopes in* $\mathbb{R}^3$ *of total complexity* n*. Let* $\mathcal{A}$ *be their overlay. The overlay* $\mathcal{A}$ *has* $O(nm^2)$ *complexity.*

It is critical to observe that an equivalent lemma for 2D subdivision does not hold. For example, the overlay of two subdivisions of the plane into n "tall and thin" and "wide and narrow" slabs has $\Omega(n^2)$ size. Thus, the above lemma captures a fundamental impact of adding convexity to the picture. The next lemma shows that we can perform point location in 3D via a simple idea and a small query overhead.

▶ **Lemma 7** (basic point location). *Let $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$ be a set of m convex polytopes in $\mathbb{R}^3$ of total complexity n. We can build a data structure that uses $O(nm^2)$ space that does the following: The data structure stores a set T of "anchors" (triangles) such that each anchor is either inside a polytope or lies completely outside it. Then, given a query point q, it can output an anchor $\tau \in T$ such that $\Diamond(S, q)$ is equal to the subset of polytopes that contain τ. The time to answer the query is $O(\log n \log m)$.*

Proof sketch (see the full version [5] for the full proof). The plan is to do a binary search via point location queries. To do that, we decompose each polytope into an upper and lower hull and then turn each hull into an everywhere defined surface (a function from $\mathbb{R}^2$ to $\mathbb{R}$). We overlay them, triangulate them, and then decompose them into "levels" where each level ends up being a Z-monotone surface whose projection will be a planar decomposition which will be stored in a point location data structure. The level (and the triangle) immediately above a query point can be found with $O(\log m)$ steps of a binary search process that uses the point location queries for a total query time of $O(\log n \log m)$. ◀

For some applications, the above lemma is sufficient but for our most general result, we need to remove the factor $\log m$ in the above query time.

▶ **Lemma 8** (advanced point location). *Let $S = \{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$ be a set of m convex polytopes in $\mathbb{R}^3$ of total complexity n, where $m = O(2^{\sqrt{\log n}})$. We can build a data structure that uses $O(n2^{O(\sqrt{\log n})})$ space that does the following: The data structure stores a set T of "anchors" (triangles) such that each anchor is either inside a polytope or lies completely outside it. Then, given a query point q, it can output an anchor $\tau \in T$ such that $\Diamond(S, q)$ is equal to the subset of polytopes that contain τ. The query time is $O(\log n)$.*

The rest of this subsection is devoted to the proof of Lemma 8.

3.1.1 Preliminaries

A modified Dobkin-Kirkpatrick hierarchy. First, we describe a process of simplifying polytopes in 3D which is a modified version of building Dobkin-Kirkpartrick hierarchies [16] but in a randomized way. We refer to the standard Dobkin-Kirkpartrick simplification as DK-simplification and to our modified randomized simplification as RDK-simplification. Let $\mathcal{P}$ be a convex polytope in 3D. An *outer simplification* of $\mathcal{P}$ is obtained by considering $\mathcal{P}$ as the intersection of halfspaces that bound its faces. Then, we select a subset I of independent faces of $\mathcal{P}$ (i.e., no two adjacent faces are selected) and such that each face has $O(1)$ complexity and $|I| = \Omega(|\mathcal{P}|)$. In DK-simplification, I is simply deleted, instead in RDK-simplification, we delete each element of I with probability 0.5, independently. Observe that the deletion of I' is equivalent to attaching $|I'|$ disjoint convex cells of constant complexity to the faces of $\mathcal{P}$. To define an *inner simplification* of $\mathcal{P}$ we do a similar process but this time we consider $\mathcal{P}$ as the convex of hull of its vertices and select an independent set I of the vertices of $\mathcal{P}$ of size $\Omega(|\mathcal{P}|)$ such that each vertex has constant degree. In the standard DK-simplification, I is simply deleted, but in RDK-simplification, we delete each vertex in I with probability 0.5, independently. This creates a convex polytope that is contained inside $\mathcal{P}$ (or shares some boundaries). This process can be thought of as removing disjoint convex cells of constant complexity from $\mathcal{P}$. See Figure 2 for an illustration.

We can repeatedly apply DK- or RDK-simplifications until the inner polytope is reduced to a simplex and the outer polytope to a halfspace. This results in a hierarchy which is denoted by $\mathcal{H}(\mathcal{P})$. An easy observation that we will be using is the following.

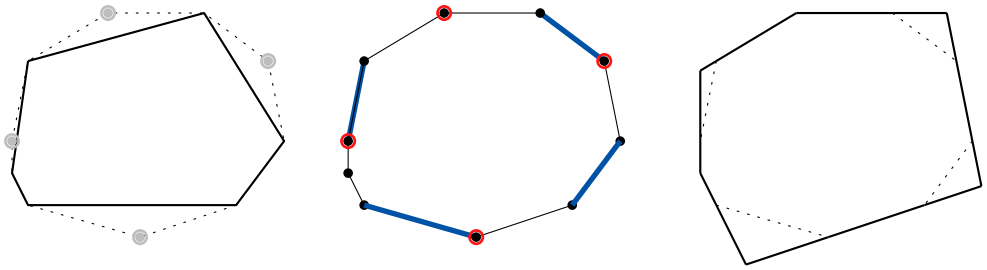


Figure 2 The inner and outer simplifications are shown in 2D, for clarity. The central polytope is being simplified here. To the left, we have its inner simplification after deleting the vertices marked with red and to the right, we have the outer simplification after deleting the edges marked with blue.

▶ **Observation 9.** *If every simplification reduces the complexity of $\mathcal{P}$ by a constant factor, then, $\mathcal{H}(\mathcal{P})$ consists of $O(\log n)$ convex polytopes, i.e., it is the overlay of $O(\log n)$ convex polytopes which decomposes $\mathbb{R}^3$ into $O(|\mathcal{P}|)$ convex cells of constant complexity.*

A simplification round. Let T be a parameter that will be defined later (T will actually be set to $\sqrt{\log n}$). Given a convex polytope $\mathcal{P}$, *one round of simplification of $\mathcal{P}$* is defined as follows. If the complexity of $\mathcal{P}$ is at most 2^T, then $\mathcal{P}$ is simplified via DK-simplification, reducing the complexity of $\mathcal{P}$ by a constant factor at each simplification to yield a hierarchy of $O(\log |\mathcal{P}|) = O(T)$ simplifications. Otherwise, starting from $\mathcal{P}$, we repeatedly apply inner (resp. outer) RDK-simplification T times to obtain T progressively smaller (resp. larger) polytopes. The smallest and the largest polytopes that are obtained are called the *boundary* polytopes. Based on this, we perform the following steps for every polytope $\mathcal{P}_i$ in the input.

1. Initialize a list $X = \{\mathcal{P}_i\}$ and the set $S = \{\mathcal{P}_i\}$ of simplifications of $\mathcal{P}_i$. Each element of S will maintain a *counter* that counts how many rounds it has been simplified for, and for $\mathcal{P}_i$ this counter is initialized to zero.
2. While X is not empty:
   a. Remove the first polytope $\mathcal{P}$ from X and let c be the counter of $\mathcal{P}$.
   b. Simplify $\mathcal{P}$ for one round, resulting in $O(T)$ simplifications which are added to S; all of these simplifications have their counter set to $c + 1$.
   c. If any boundary polytope is produced (i.e., if $\mathcal{P}$ had complexity at least 2^T), then they are added to X.

We also define a subdivision $\mathcal{A}_j$ is as the overlay of all the polytopes (over all sets S) whose counter is at least j. The following lemma captures some of the important properties of our construction.

▶ **Lemma 10.** *Assuming $T \geq c + \log\log n$ for a large enough constant c, the following properties hold with probability at least $1 - n^2$ (whp).*

(I) *Let x be the maximum value of the counters of the polytopes in $\mathcal{A}_0$. We have $x = O\left(\frac{\log n}{T}\right)$ (whp).*
(II) *Let M be the number of polytopes in $\mathcal{A}_0$. We have $M = O(m2^x T)$.*
(III) *The size of $\mathcal{A}_j$ is $O(nM^2)$, for all j.*
(IV) *$\mathcal{A}_j$ decomposes $\mathbb{R}^3$ into convex cells of complexity $O(M)$.*
(V) *Let Δ be a cell in $\mathcal{A}_j$ and let $\mathcal{P}$ be a polytope in $\mathcal{A}_{j-1}$. Then, the number of vertices of $\mathcal{P}$ that are contained in Δ as well as the number of edges of $\mathcal{P}$ that intersect Δ is bounded by $O(2^{O(T)}\log n) = O\left(2^{O(T)}\right)$. (whp).*
(VI) *The size of $\mathcal{A}_x$ is $O\left(\left(m2^x T 2^T\right)^{O(1)}\right)$.*

Proof. For claim (I), observe that if a polytope $\mathcal{P} \in X$ has size 2^T, then it is simplified with a standard DK-simplification and it adds no additional polytopes to X. Thus, it suffices to consider polytopes that have complexity at least $2^T \geq \log^c n$. Now the claim (I) follows from a standard Chernoff bound: Since in our RDK-simplification, we will select an independent set I of size $\Omega(|\mathcal{P}|) = \Omega(c \log n)$ and on expectation, we will delete half of elements but since $\mathbb{E}[|I|] \geq c \log n$, it follows that with with high probability at least $|I|/4$ of the elements of I are deleted. This reduces the complexity of polytope by a constant fraction. Consequently, the number of RDK-simplifications steps an input polytope can go through is bounded by $O(\log n)$ and thus the number of rounds is $O(\frac{\log n}{T})$ which implies claim (I).

Claim (II) follows from the observation that during each round, each polytope creates two additional polytopes (whose counters are incremented by one). In other words, the number of polytopes of counter $c+1$ is at most double the number of polytopes with counter c. Combined with (I) this implies that the total number of polytopes ever added to X is $O(m2^x)$. For each such a polytope (whose counter is at least j) the overlay $\mathcal{A}_j$ includes T additional polytopes in S.

Claim (III) is a consequence of Lemma 6 and (II).

To see (IV), consider one polytope $\mathcal{P}$ in $\mathcal{A}_j$. Observe that all the inner and outer simplifications of $\mathcal{P}$ are also included in $\mathcal{A}_j$ this is because our simplification process continues until the inner simplification of $\mathcal{P}$ is reduced to a simplex and the outer simplification of $\mathcal{P}$ is reduced to a halfspace. All of these simplifications have a counter value that is at least j and thus by definition of $\mathcal{A}_j$ they are included in $\mathcal{A}_j$. Thus, some DK-hierarchy, $\mathcal{H}(\mathcal{P})$, of $\mathcal{P}$ is included in $\mathcal{A}_j$. By Observation 9, $\mathcal{H}(\mathcal{P})$ decomposes $\mathbb{R}^3$ into cells of constant complexity. The number of such hierarchies that are included in $\mathcal{A}_j$ is upper bounded by the number of polytopes, M. Each cell Δ of $\mathcal{A}_j$ is thus the intersection of at most M convex cells of constant complexity and thus Δ will have complexity at most $O(M)$.

Now consider claim (V). Observe that if $\mathcal{P}$ is included in $\mathcal{A}_j$, then Δ cannot contain any vertex of $\mathcal{P}$ or intersect any of its edges. Thus, we can assume $\mathcal{P}$ is not included in $\mathcal{A}_j$. But this also implies that the counter of $\mathcal{P}$ must have value exactly $j-1$. At some point $\mathcal{P}$ will be considered during our simplification process. Assume that Δ is intersected by Y edges of $\mathcal{P}$ and $Y = \Omega(2^T \log n)$. This implies that $\mathcal{P}$ must have undergone our RDK-simplification process. Consider one step of our randomized DK-simplification: regardless of whether we are dealing with inner or outer simplification and regardless of which independent set is chosen, each of these Y edges are kept with probability at least $\frac{1}{4}$. By a standard Chernoff bound, it follows that with high probability, at least $\frac{1}{8}$ fraction of them are kept. $\mathcal{P}$ undergoes at most T such simplification steps until its counter is incremented by one and the resulting boundary polytopes are added to $\mathcal{A}_j$. This implies that with high probability, the resulting boundary polytopes have at least $\frac{Y}{8^T}$ edges intersecting Δ. If $Y \geq c8^T \log n$, this is a contradiction and thus the claim follows. A similar argument applies to the vertices of $\mathcal{P}$ contained in Δ.

Finally, the claim (VI) follows by the observation that at the last round, all polytopes must have complexity at most 2^T and there can be at most $m2^x T$ of them by claim (II) and now we plug these values in Lemma 6 to prove this claim. ◀

The next lemma enables us to give an upper bound on the number of cells of $\mathcal{A}_{j-1}$ that can intersect the cells of $\mathcal{A}_j$. This is a very crucial part of our point location data structure but the proof is relatively technical so it is postponed to the full version [5].

▶ **Lemma 11.** *Let A and B be the overlay of at most L convex polytopes each such that they form a decomposition of $\mathbb{R}^3$ into convex cells of complexity at most δ. In addition, assume that for every cell Δ in B, and every polytope $\mathcal{P}$ in A, the number of vertices of $\mathcal{P}$ that are contained in Δ and the number of edges of $\mathcal{P}$ that intersect Δ are bounded by X.*

Then, every cell Δ of B is intersected by at most $O(L^3X + L^3\delta + L^2\delta^2 + L\delta^3)$ cells of A.

3.1.2 Proof of Lemma 8 (Advanced Point Location)

Our data structure is as follows. Set $T = \sqrt{\log n}$. Observe that we have $2^T = 2^{\frac{\log n}{T}}$. We build the previously mentioned overlays $\mathcal{A}_i$ for $i = 1, \cdots, O(\log_T n)$. Then, for each cell Δ in $\mathcal{A}_j$, we find all the cells Δ' in $\mathcal{A}_{j-1}$ that intersect Δ and then store the hyperplanes that define them in what we call a *fast data structure*. Given a set of ℓ hyperplanes, such a fast data structure can store them using $O(\ell^3)$ space such that point location queries can be answered in $O(\log \ell)$ time [12]. This concludes our data structure. Our query algorithm is relatively simple: if we know the cell Δ in $\mathcal{A}_j$ that contains the query point, then by querying the fast data structure built for Δ, we can find the cell Δ' in $\mathcal{A}_{j-1}$ that contains the query point and then we can continue until we finally point locate the query. We now bound the space and the query time of the data structure.

The query time. By Lemma 10 (VI), we can locate the query point q in a cell of $\mathcal{A}_x$ in $O(\log m + x + \log T + T) = O(\sqrt{\log n})$ time. Thus, assume, we know q is in a cell Δ of $\mathcal{A}_j$. By Lemma 10, each $\mathcal{A}_j$ will have total complexity $O(n2^{O(T)})$ and is composed of $2^{O(T)}$ polytopes with cells of complexity $2^{O(T)}$. By Lemma 11 (using $\delta, L = M, X$ all set to $2^{O(T)}$) and Lemma 10(II,IV,V), Δ will intersect at most $2^{O(T)}$ cells Δ', whp. Note that we can assume this holds in the worst-case because if a cell intersects more than this amount, then we can simply rebuild the data structure and the expected number of such trials is $O(1)$. Thus, querying the fast data structure will take $O(T)$ time. Finally, observe that the depth of the recursion is $x = \frac{\log n}{T}$. So we get the query time of $O(\log n)$.

The space analysis. As mentioned each $\mathcal{A}_i$ will have complexity $O(n2^{O(T)})$. For each cell we store a fast data structure that uses $2^{O(T)}$ space, leading to $O(n2^{O(T)})$ total space usage.

3.2 Solving LEIS3D

In this section, we prove our main technical tool.

▶ **Theorem 5.** *The on-demand LEIS3D problem for an input of size n on catalog graph $\mathbf{G}$ can be solved with a data structure that uses $O(n)$ space and has the following query time: (i) $O(\log n \log\log n + |\mathbf{H}| \log\log n)$ if $\mathbf{G}$ is a path and (ii) $O(\log n + |\mathbf{H}|\sqrt{\log n})$ when $\mathbf{G}$ is a graph of constant degree.*

Proof. Note that when $\mathbf{G}$ is a general graph of maximum degree $d = O(1)$, we can assume that its maximum degree is 3 by replacing a vertex of degree $d' \leq d$ with a binary tree of depth $\log d'$ which only blows up the query time by a constant factor.

Recall that our data structure should store a k-shallow cutting of H_v for each vertex $v \in \mathbf{G}$, and let c be a large enough constant. When $\mathbf{G}$ is a path we set $\ell = \log n \cdot \log\log n$ and $x = k(2\ell)^c$ and when $\mathbf{G}$ is a general graph, we set $\ell = \sqrt{\log n}$ and $x = k2^{c\ell}$. We say a vertex $u \in \mathbf{G}$ is *light* if $|H_v| \leq x$, and *heavy* otherwise. For a vertex v, its *ℓ-neighborhood*, denoted by $\mathcal{N}_\ell(v)$, is the set of all the vertices of $\mathbf{G}$ that have distance at most ℓ to v.

The data structure. For every vertex v, we build a x-shallow cutting $\mathcal{C}_v$ for the set of hyperplanes H_v. If v is heavy, $\mathcal{C}_v$ is built via Lemma 3 (together with the all the corresponding apparatus in the lemma) and thus $\mathcal{C}_v$ is assumed to be convex (i.e., the lower envelope of some hyperplanes). However, if v is light, then $\mathcal{C}_v$ is assumed to consist of just one triangle τ, the hyperplane $Z = +\infty$ as an infinite triangle and the conflict list of τ in this case is H_v. In either case, for every triangle τ in the shallow cutting $\mathcal{C}_v$, we compute and store the

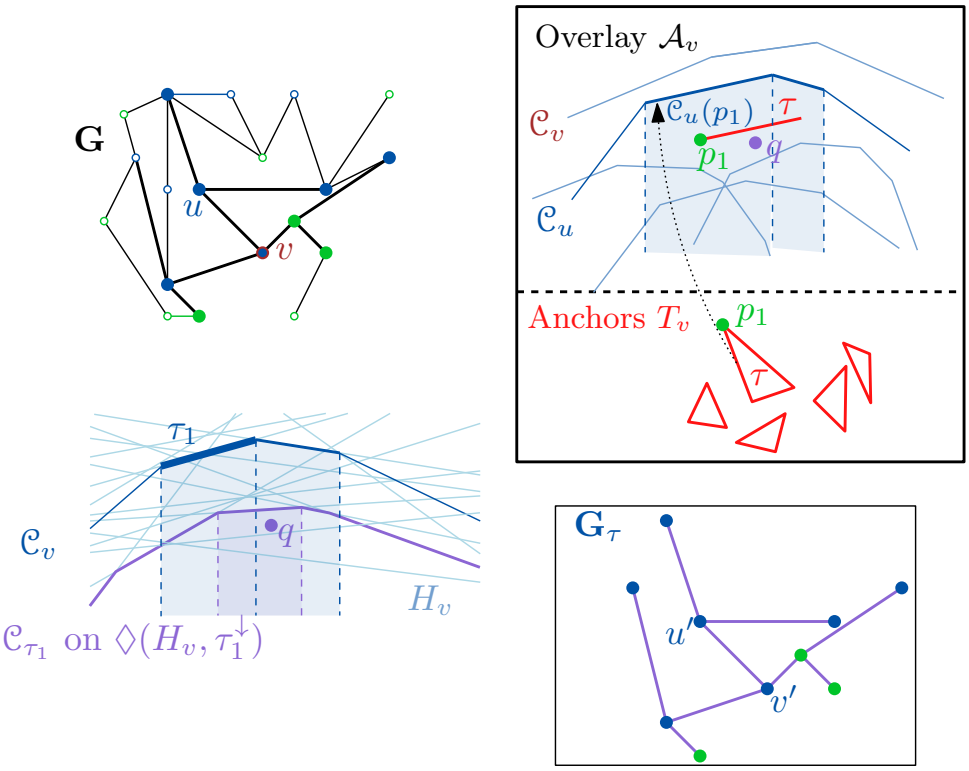


Figure 3 An overview of the data structure. Light nodes shown in green, heavy nodes in blue, and $\mathcal{N}_2(v)$ is marked with large disks. Each triangle τ in the point location data structure on $\mathbf{G}_u$ stores a pointer to a copy of this neighborhood. From there, we can jump to cells in $\mathcal{C}_v$.

conflict list, $\Diamond(H_v, \tau^{\downarrow})$, and then we build a k-shallow cutting, $\mathcal{C}_\tau$, on $\Diamond(H_v, \tau^{\downarrow})$; we call $\mathcal{C}_\tau$ *the secondary shallow cutting*. This is also built via Lemma 3. Note that as $x \geq k$, these secondary shallow cuttings cover the $\leq k$-level of H_v.

The overlays. Next, we consider each heavy vertex v: we consider all the vertices $u \in \mathcal{N}_\ell(v)$ and the shallow cuttings $\mathcal{C}_u$ as convex polytopes and create a 3D point location data structure on the overlay all the convex polytopes $\mathcal{C}_u$, for all u. Let $\mathcal{A}_v$ be this overlay. When $\mathbf{G}$ is a path, $\mathcal{N}_\ell(v)$ contains at most 2ℓ vertices and this overlay is built via Lemma 7. When $\mathbf{G}$ is a graph, it contains at most $2^{\ell+1}$ vertices and we use Lemma 8. In either case, the data structure will store a number of triangles as anchors, denoted by T_v. For each anchor $\tau \in T_v$ we consider its corners (vertices) p_1, p_2 and p_3. Then, consider every $u \in \mathcal{N}_\ell(v)$ and its shallow cutting $\mathcal{C}_u$; store a pointer from τ to the triangle, $\mathcal{C}_u(p_i)$, in $\mathcal{C}_u$ that lies above p_i, for $1 \leq i \leq 3$. Observe that for the anchor τ, we store at most $3\mathcal{N}_\ell(v)$ pointers (at most three per $u \in \mathcal{N}_\ell(v)$). These pointers are arranged in a particular way: we make a copy of $\mathcal{N}_\ell(v)$, denoted by $\mathbf{G}_\tau$, where each vertex $u \in \mathcal{N}_\ell(v)$ has a copy $u' \in \mathbf{G}_\tau$ and we store the three pointers from u' to $\mathcal{C}_u(p_i)$, for $1 \leq i \leq 3$.

The query. Consider a query, consisting of a point $q \in \mathbb{R}^3$ and an initial vertex u of the query subgraph $\mathbf{H}$.

1. If u is light, $\mathcal{C}_u$ contains one triangle τ which is the plane $Z = +\infty$ and one cell which is the entire $\mathbb{R}^3$. We look at the secondary shallow cutting $\mathcal{C}_\tau$ and locate $\mathcal{C}_\tau(q)$ if it exists and return conflict list $\Diamond(H_u, \mathcal{C}_\tau(q)^{\downarrow})$ which by Lemma 3 contains $\Diamond(H_u, q^{\downarrow})$.

2. This process can be repeated until we encounter the first heavy vertex u. We query the 3D point location data structure (Lemma 8 for general graphs and Lemma 7 for when $\mathbf{G}$ is a path) to get an anchor triangle τ with corners p_1, p_2 and p_3. We use the pointers associated with the copy u' of u in $\mathbf{G}_\tau$ to find the triangle $\tau_i = \mathcal{C}_u(p_i)$, $1 \leq i \leq 3$. The 3D point location data structure guarantees that the corners of τ all lie below $\mathcal{C}_u$, and thus all three of these triangles τ_i exist, if and only if q lies below $\mathcal{C}_u$.
   We query the secondary shallow cuttings $\mathcal{C}_{\tau_i}$, $1 \leq i \leq 3$, and for each, we find $\mathcal{C}_{\tau_i}(q)$, if it exists and then report the pointer to the corresponding secondary conflict list.
3. Now assume the query is given a sequence of vertices of $\mathbf{G}$, $u_0 = u, u_1, \cdots, u_m$, one by one, such that they all belong to $\mathcal{N}_\ell(u)$. Let τ be the triangle found in the previous step. The crucial observation is that τ does not intersect any triangle in the shallow cuttings $\mathcal{C}_w$, for any $w \in \mathcal{N}_\ell(u)$ since τ was a triangle in the overlay of all the shallow cuttings. Thus, in this case, the anchor does not change. In this case, for each revealed vertex u_j, we simply need to follow the pointer from u'_{j-1} to u'_j in $\mathbf{G}_{u,\tau}$ and from u'_j to $\mathcal{C}_{u_j}(p_i)$, for $1 \leq i \leq 3$. The rest of the algorithm is the same as Step 2.
4. Eventually, the algorithm will reveal a vertex u_{m+1} that is outside the $\mathcal{N}_\ell(u)$. In this case, we simply go to Step 1 or 2, i.e., we assume u_{m+1} is the initial vertex.

The query time. Step 1 takes $O(\log(\frac{x}{k}))$ time by Lemma 3 since each conflict in $\mathcal{C}_v$ has size $O(x)$ and we have a k-shallow cutting for them. In Step 2, we use $Q = O(\log n \log\log n)$ time for when $\mathbf{G}$ is a path by Lemma 7 and $Q = O(\log n)$ time for a general graph to identify the anchor τ. Obtaining the triangles $\mathcal{C}_u(p_i)$ for $1 \leq i \leq 3$ is a constant time operation and then we again query the secondary shallow cuttings which takes $O(\log(\frac{x}{k}))$ time. In Step 3, the query time is only $O(\log(\frac{x}{k}))$ because we do not need to find the anchor τ, and the cost is simply the cost of querying the secondary shallow cutting. Finally, observe that whenever we reach Step 2, at least ℓ other vertices of $\mathbf{H}$ must be revealed until we exceed $\mathcal{N}_\ell(u)$. In other words, Step 2 is only repeated once ℓ additional vertices of $\mathbf{H}$ have been revealed. Summing this up, we get that the query time is asymptotically bounded by

$$Q + \log\left(\frac{x}{k}\right)|\mathbf{H}| + \frac{|\mathbf{H}|}{\ell} \cdot Q.$$

By plugging in the corresponding parameters, when $\mathbf{G}$ is a path, the query time is

$$O(\log n \log\log n + |\mathbf{H}| \log\log n)$$

and for a general graph $\mathbf{G}$ it is

$$O(\log n + |\mathbf{H}|\sqrt{\log n}).$$

The space analysis. The total size of the shallow cuttings $\mathcal{C}_v$ and the secondary shallow cuttings is $O(n)$. Let M be the maximum size of $\mathcal{N}_\ell(v)$, i.e., for a path $\mathbf{G}$ we have $M = 2\ell$ and for a general graph of maximum degree three we have $M = 2^\ell$. Observe that the parameter x is at least M^c in both cases, for some constant c that we can choose. The number of heavy vertices is at most $\frac{n}{x}$ and each heavy vertex can appear in the ℓ-neighborhood of at most M other vertices. Thus, the total number of overlays that we will create is at most $\frac{nM}{x}$ and each overlay will contain at most M convex polytopes. By choosing c large enough and by plugging the bounds from Lemma 7 and Lemma 8, we get that the total space of the 3D point location can be bounded by $O\left(\frac{n}{M}\right)$ which is also an upper bound for the number of anchors. Finally, for every anchor, we store at most M pointers for a total of $O(n)$ space. ◀

4 Some Applications

Here, we briefly mention quick applications of Theorem 5. Note that by lifting map, similar problems for 2D disks can also be solved.

▶ **Theorem 12.** *Let P be a set of n points in 3D, each associated with a real-valued weight. We can store P in a data structure that uses $O(n \log n)$ space such that given a query halfspace h and two values w_1 and w_2, we can find all the points P that lie inside h and whose weight lies between w_1 and w_2 in $O(\log^{3/2} n + t)$ query time where t is the size of the output.*

Proof. Use duality to get a set of n hyperplanes and let q be point dual to the query halfspace h. Store the hyperplanes in a balanced binary tree $\mathbf{G}$, ordered by weight. Every node v of $\mathbf{G}$ defines a *canonical set* H_v which is the subset of hyperplanes stored in the subtree of v. Store H_v in a data structure of Afshani and Chan [2]. This uses $O(n \log n)$ space in total. Then, answering the query can be reduced to answering $O(\log n)$ 3D halfspace range reporting queries on two root to leaf paths in the tree. Simply querying the data structure of [2] gives $O(\log^2 n + t)$ query time which is already optimal if $t \geq \log^2 n$. To improve the query time, it thus suffices to assume $t \leq \log^2 n$. In which case, we use Theorem 5 with parameter $k = \log^2 n$ and store each conflict list stored by Theorem 5 in a data structure of [2]. This combination still uses linear space and when $t \leq \log^2 n$, it returns $O(1)$ conflict lists of size $O(\log^2 n)$ each. Querying the data structure of [2] on the conflist lists gives the claimed query time. ◀

Note that offline halfspace range reporting queries can also be solved with a very similar technique. We now consider 3D halfspace max queries.

▶ **Theorem 13.** *3D halfspace range max queries can be solved $O(n \log n)$ space and $O(\log^{3/2} n)$ query time.*

Proof. Use duality and let $H = \{h_1, .., h_n\}$ be the dual input hyperplanes ordered by increasing weights, and let w_i be the weight of h_i. We construct a balanced binary tree $\mathbf{G}$, whose leaves correspond to the hyperplanes $h_1, .., h_n$. For each node v, let H_v denote the subset of hyperplanes stored in the leaves below v. We now build the data structure of Theorem 5, with parameter $k = 0$. Total number of hyperplanes is $O(n \log n)$, the data structure uses $O(n \log n)$ space.

Consider the query point $q \in \mathbb{R}^3$ dual to the query halfspace, and let h_i be the (unknown) heaviest plane that passes below q. To find h_i, we construct a path $\mathbf{H}$ of length $O(\log n)$ in an on-demand manner. The main idea is as follows. At node v, with left child ℓ and right child r, we want to test if q lies below the lower envelope (the 0-level) of H_r. If so, then h_i must lie in the left subtree rooted at ℓ, hence we continue the search there. If q lies above the lower envelope of H_r, we continue the search in the subtree rooted at r. The process finishes after we find the leaf containing h_i in $O(\log n)$ rounds. This process generates a path $\mathbf{H}$ of length $O(\log n)$. Hence, the total query time is $O(\log^{3/2} n)$ as claimed. ◀

As discussed, this also improves the query time of the best range sampling data structures for weighted points. See [6]. It can also be combined with Theorem 12 to answer "top-k" reporting queries of halfspaces in 3D [19] as well as approximate counting queries [18].

▶ **Theorem 14.** *Let P be a set of points, each associated with a color from the set $[m]$. We can store P in a data structure, s.t., given a query halfspace h, we can report the t distinct colors of the points in h in (i) $O(\log n \log\log + m \log\log n)$ time using $O(n)$ space or (ii) $O(\log n + t \log(m/t)\sqrt{\log n})$ time using $O(n \log m)$ space.*

Proof. Once again use duality and let H be the of dual halfspaces. Let H_i be the hyperplanes of color i, $1 \leq i \leq m$. For the first claim (i), simply use a path of length m as graph $\mathbf{G}$ and invoke Theorem 5 where all query subgraphs $\mathbf{H}$ are equal to $\mathbf{G}$.

For the second claim (ii), add a balanced binary tree of height $\log m$ on the path and let $\mathbf{G}$ be the resulting graph. For every node $v \in \mathbf{G}$, define H_v to be the union of hyperplanes stored in the subtree of v. We now build the Theorem 5 data structure with $k = 0$ the graph $\mathbf{G}$. The total size of all sets H_v, and thus of our data structure is $O(n \log m)$. Now, consider the query point $q \in \mathbb{R}^3$ dual to h. Once again, we use the on-demand capability and at every node $v \in \mathbf{G}$ we decide whether either child of v has any output color, and if so, we recurse into that child. Let $\mathbf{H}$ denote the subtree of $\mathbf{G}$ visited by this procedure. By classical results, size of $\mathbf{H}$ is $O(t \log(n/t))$, when t is the number of leaves visited (and thus the number of distinct colors reported). Note that $\mathbf{H}$ can be assumed to be a walk by simpling fully finishing the recursion on each child of v, then backing up to v and then potentially recursing on the other child. The total query time is $O(\log n + t \log(m/t)\sqrt{\log n})$. ◀

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 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009. doi:10.1137/1.9781611973068.21.

3 Peyman Afshani and Pingan Cheng. 2D generalization of fractional cascading on axis-aligned planar subdivisions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 716–727, 2020. doi:10.1109/FOCS46700.2020.00072.

4 Peyman Afshani, Pingan Cheng, Aniket Basu Roy, and Zhewei Wei. On range summary queries. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 7–1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2023.

5 Peyman Afshani, Yakov Nekrich, and Frank Staals. Convexity Helps Iterated Search in 3D. *CoRR*, 2504.07545, 2025. arXiv:2504.07545.

6 Peyman Afshani and Jeff M. Phillips. Independent Range Sampling, Revisited Again. In *Symposium on Computational Geometry (SoCG)*, pages 4:1–4:13, 2019. doi:10.4230/LIPICS.SOCG.2019.4.

7 Peyman Afshani, Bryan Wilkinson, Yufei Tao, and Cheng Sheng. Concurrent range reporting in two-dimensional space. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 983–994, 2014. doi:10.1137/1.9781611973402.

8 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.

9 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of (≤ k)-levels in three dimensions. *SIAM Journal of Computing*, 30(2):561–575, 2000. doi:10.1137/S0097539798349188.

10 Timothy M. Chan and K. Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56, 2016.

11 Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022. doi:10.1137/1.9781611977073.10.

12 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.

13 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986. doi:10.1007/BF01840440.

14 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1:163–191, 1986. doi:10.1007/BF01840441.

15 Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *Journal of Computer and System Sciences*, 68(2):269–284, 2004. doi:10.1016/J.JCSS.2003.07.003.

16 David P Dobkin and David G Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *Journal of Algorithms*, 6(3):381–392, 1985. doi:10.1016/0196-6774(85)90007-0.

17 Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. doi:10.1007/S00454-010-9248-1.

18 Saladi Rahul. Approximate range counting revisited. In *Symposium on Computational Geometry (SoCG)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

19 Saladi Rahul and Yufei Tao. Generic techniques for building top-k structures. *ACM Transactions on Algorithms*, 18(4):38:1–38:23, 2022. doi:10.1145/3546074.

20 Edgar A. Ramos. On range reporting, ray shooting and k-level construction. In *Symposium on Computational Geometry (SoCG)*, pages 390–399, 1999. doi:10.1145/304893.304993.

21 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry: Algorithms and Applications*, 2(3):169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

22 V.K Vaishnavi and D Wood. Rectilinear line segment intersection, layered segment trees, and dynamization. *Journal of Algorithms*, 3(2):160–176, 1982. doi:10.1016/0196-6774(82)90016-5.

23 Dan E. Willard. New data structures for orthogonal range queries. *SIAM Journal of Computing*, 14:232–253, 1985. doi:10.1137/0214019.

A Subquadratic Algorithm for Computing the L_1-Distance Between Two Terrains

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

Boris Aronov
Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA

Olivier Devillers
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Christian Knauer
University of Bayreuth, Germany

Guillaume Moroz
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

We study the problem of computing the L_1-distance between two piecewise-linear bivariate functions f and g, defined over a bounded polygonal domain $\mathbb{M} \subset \mathbb{R}^2$, that is, computing the quantity $\|f-g\|_1 = \int_{\mathbb{M}} |f(x,y)-g(x,y)|\,dx\,dy$. If f and g are defined by linear interpolation over triangulations $\mathbf{T}_f$ and $\mathbf{T}_g$, respectively, of $\mathbb{M}$ with a total of n triangles, we show that $\|f-g\|_1$ can be computed in $\tilde{O}(n^\alpha)$ time, where $\alpha = \max\{(\omega+1)/2, 8/5\}$, ω is the matrix multiplication exponent, and $\tilde{O}$ notation hides factors of the form n^ε for any $\varepsilon > 0$. This bound holds for the currently best known value of ω, which is approximately 2.37. More generally, if the complexity of the overlay of $\mathbf{T}_f$ and $\mathbf{T}_g$ is κ, then the runtime of our algorithm is $\tilde{O}(\kappa^{\alpha-1}n^{2-\alpha})$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Terrain similarity, volume computation, polynomial interpolation, geometric cuttings, bivariate multipoint evaluation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.4

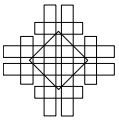
Funding G.M. acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-24-CE48-1899 (project StratMESH). Work by P.K.A. has been partially supported by NSF grants CCF-20-07556, CCF-22-23870, and IIS-24-02823, and by the Binational Science Foundation Grant 2022131. Work by B.A. has been partially supported by NSF grants CCF-15-40656 and CCF-20-08551.

Acknowledgements This work was initiated at the Workshop on Problems in Computational Geometry at Bellairs Research Institute, Holetown, Barbados in 2012, and the authors wish to thank Éric Colin de Verdière for useful discussions. The authors also wish to thank Bruno Salvy for useful remarks on recent multipoint evaluation results.

1 Introduction

Combining tools from computational geometry and computer algebra has led to new results that defy intuitive lower bounds. For example, using algebraic tools, it is possible to compute the centroid of the vertices in an arrangement of n lines in the plane in $O(n \log^2 n)$ operations, without computing the $\Theta(n^2)$ vertices [5]. One notable algebraic tool used for this problem is an algorithm to evaluate efficiently a univariate polynomial on multiple points [13]. This tool

© Pankaj K. Agarwal, Boris Aronov, Olivier Devillers, Christian Knauer, and Guillaume Moroz;
licensed under Creative Commons License CC-BY 4.0



was used to speed up, among others, the computation of Shapley values in the plane [10], the location of points among wireless transmitters [6], and the computation of the L_2-distance between two piecewise-linear bivariate functions f and g [16].

Bivariate functions over a planar domain are ubiquitous in geographical information systems, where they are used to represent a digital elevation model of a geospatial area, or, more generally, a function over a two-dimensional region such as precipitation, air pollution, population density, or quantities of wine production. One way to represent such a function is to triangulate the underlying domain and define a linear function over each triangle, resulting in a piecewise-linear function, often referred to as a *triangulated irregular network* (TIN). In this paper, we design an efficient algorithm for computing the L_1-distance between two piecewise-linear bivariate functions.

Problem statement. Let $\mathbb{M}$ be a bounded polygonal region of $\mathbb{R}^2$, and let $f, g : \mathbb{M} \to \mathbb{R}$ be two piecewise-linear functions defined over their common domain $\mathbb{M}$. Each of f, g is associated with a triangulation of $\mathbb{M}$, denoted by $\mathbf{T}_f, \mathbf{T}_g$, respectively, and f (resp., g) is defined to be linear over each triangle of $\mathbf{T}_f$ (resp., $\mathbf{T}_g$). Note that $\mathbf{T}_f$ and $\mathbf{T}_g$ are not assumed to be the same. Let n be sum of the numbers of vertices in $\mathbf{T}_f$ and $\mathbf{T}_g$. The graph of f (or g) is a xy-monotone polyhedral surface in $\mathbb{R}^3$, referred to as a *terrain*. For any integer $p > 0$, we define the L_p-distance between f and g to be

$$\|f - g\|_p = \left(\int_{\mathbb{M}} |f - g|^p \, dx \, dy \right)^{1/p}. \tag{1}$$

The goal in this paper is to compute $\|f - g\|_1$.

We will work in the Real RAM model [18], where inputs and intermediate values are real numbers and where algebraic operations and comparisons on real values can be performed in constant time, but modulus or rounding-to-an-integer operations are not allowed. We assume that the coefficients of the piecewise-linear functions f and g and the coordinates of the triangles in $\mathbf{T}_f$ and $\mathbf{T}_g$ are real numbers.

Related work. The problem of comparing two terrains, defined over the same polygonal domain, has been addressed in previous work [2, 16]. Their focus was on identifying a linear dependence between two functions or terrains. Agarwal et al. [2] studied the problem of computing the minimal distance

$$\sigma_p(f, g) = \min_{s,t} \|sf + t - g\|_p \tag{2}$$

for $p = 1, \infty$, minimizing over scaling factor s and translation t. They showed that $\sigma_\infty(f, g)$ can be computed in $\tilde{O}(n^{4/3})$ time (as stated in the abstract, $\tilde{O}$ notation hides polylogarithmic factors), and that an $(1 + \varepsilon)$-approximation of $\sigma_1(f, g)$ can be computed in $O(n/\varepsilon)$ time if $\mathbf{T}_f$ and $\mathbf{T}_g$ are identical. However if $\mathbf{T}_f$ and $\mathbf{T}_g$ are different, the runtime of their algorithm is $O(\kappa/\varepsilon)$, where κ is the complexity of the overlay of the two triangulations; $\kappa = \Theta(n^2)$ in the worst case.

Moroz and Aronov [16] showed that even for unaligned triangulations, $\|f - g\|_2$ and $\sigma_2(f, g)$ can be computed in $O(n \log^4 n \log \log n)$ time. This bound also applies to computing the L_{2p}-distance, where p is a constant positive integer. Unfortunately, these approaches do not extend to computing $\sigma_1(f, g)$ or $\|f - g\|_1$ in subquadratic time in the general case. One notable obstacle is that the L_1-distance involves the absolute value function, which is not polynomial. One key algebraic tool used in [16] to expedite the computation of $\|f - g\|_2$

is the evaluation of a *univariate* polynomial of degree n on n points in $\tilde{O}(n)$ operations. Fast *univariate multipoint evaluation* algorithms in the Real RAM model have been known since 1972 [13]. A subquadratic multipoint-evaluation algorithm for bivariate polynomials was not known until much later [17], and developing fast multipoint-evaluation algorithms for multivariate polynomials is still an active research area [19]. We note that, if the coefficients of polynomials as well as coordinates of points are integers or in finite fields, then algorithms for multipoint evaluation are known to be near-linear in the size of the output [9, 14].

Our results. The main result of the paper is a subquadratic algorithm for computing $\|f-g\|_1$. Its runtime depends on ω, the standard matrix multiplication exponent, which is known to be less than 2.371552 [20] and conjectured by some to be 2.

▶ **Theorem 1.** *Let f and g be two bivariate functions defined by linear interpolation over triangulations $\mathbf{T}_f$ and $\mathbf{T}_g$, respectively, over a common bounded domain $\mathbb{M}$ with a total of n triangles, then $\|f-g\|_1$ can be computed in $\tilde{O}(n^\alpha)$ time, where $\alpha = \max\{(\omega+1)/2, 8/5\}$ and $O(n^\omega)$ is the time required to multiply two $n\times n$ matrices; currently the best known value of ω is around 2.37. If the complexity of the overlay of $\mathbf{T}_f$ and $\mathbf{T}_g$ is κ, then $\|f-g\|_1$ can be computed in $\tilde{O}(\kappa^{\alpha-1}n^{2-\alpha})$ time.*

This result is based on three main ideas: we first show that the problem can be reduced to computing a double sum of rational functions, where the denominators depend only on two variables of the outer sum (Section 2). Those sums are defined over pairs of geometric objects that satisfy some specific algebraic inequalities. Next, we use geometric cuttings to obtain a sign-invariant biclique decomposition of those pairs of geometric objects (Section 3). Finally, we employ fast multipoint polynomial evaluation to evaluate those sums efficiently (Section 4).

Recently, we became aware of a preprint improving the bound on fast multipoint polynomial evaluation [19]. This seems to strengthen the bound of Theorem 1 to $O(n^{8/5+\varepsilon})$ for any $\varepsilon > 0$. We discuss this (possible) improvement at the end of the paper.

2 Reduction to Multi-point Evaluation

In this section, we first express the desired quantity $\|f-g\|_1$ as a sum of integrals associated with vertices of the overlay of $\mathbf{T}_f$ and $\mathbf{T}_g$ (in Section 2.1) and ultimately as a double sum of low-degree rational expressions of the parameters of f and g (in Section 2.2).

We assume hereafter, without loss of generality, that the (x,y)-coordinate system has been rotated so that none of the edges of $\mathbf{T}_f$ and $\mathbf{T}_g$ are parallel to the y-axis, and that $\mathbb{M}$ lies strictly to the right of the y-axis. Moreover, for any two triangles $T_1 \in \mathbf{T}_f$ and $T_2 \in \mathbf{T}_g$, if $f(T_1)$ and $g(T_2)$ are not parallel, with no loss of generality, we assume that the intersection line of their supporting planes crosses the vertical plane $x = 0$.

2.1 Reduction to overlay vertices

Let V_f (resp., V_g) be the set of vertices of $\mathbf{T}_f$ (resp., $\mathbf{T}_g$), and let E_f (resp. E_g) is the set of edges of $\mathbf{T}_f$ (resp. $\mathbf{T}_g$) that do not lie on the boundary of $\mathbb{M}$. Let Σ be the overlay of $\mathbf{T}_f$ and $\mathbf{T}_g$, a convex subdivision of $\mathbb{M}$ that is a refinement of both $\mathbf{T}_f$ and $\mathbf{T}_g$. Let V (resp., C) be the set of vertices (resp., cells) of Σ, and let $\mathsf{X} \subseteq \mathsf{E}_f \times \mathsf{E}_g$ be the family of pairs of segments that intersect. Following the same argument as in [16, Corollary 4], $\|f-g\|_1$ can be expressed as

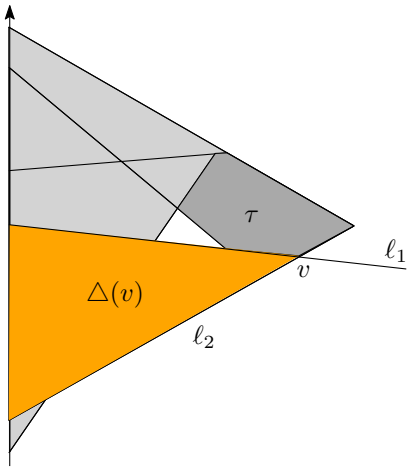


Figure 1 A typical triangle $\triangle(v)$ associated with vertex $v = \ell_1 \cap \ell_2$ in the xy-plane (adapted from [16]).

$$\|f - g\|_1 = \sum_{v \in \mathsf{V}} \sum_{\tau \in \mathsf{C}: \partial\tau \ni v} F(v, \tau, h_\tau), \tag{3}$$

where the function $F(v, \tau, h_\tau)$ is defined as follows. Let e_1, e_2 be the two edges of τ incident on v, let ℓ_i be the line supporting e_i for $i = 1, 2$, and let f_τ (resp. g_τ) be the linear function of f (resp. g) over τ. By our assumption, v lies to the right of the y-axis. Let $\triangle(v)$ be the triangle defined by the y-axis and the lines ℓ_1, ℓ_2 (lying to the right of the y-axis), let

$$h_\tau(x, y) = |f_\tau(x, y) - g_\tau(x, y)| \quad \text{and} \quad \delta(v, \tau) = \begin{cases} +1 & \tau \text{ lies between } \ell_1, \ell_2, \\ -1 & \text{otherwise.} \end{cases}$$

Then

$$F(v, \tau, h_\tau) = \delta(v, \tau) \int_{(x,y) \in \triangle(v)} h_\tau(x, y)\, dx\, dy. \tag{4}$$

A vertex in Σ is either an original vertex of $\mathbf{T}_f$ or $\mathbf{T}_g$ or an intersection point of an edge of E_f with an edge of E_g. We can rewrite (3) as:

$$\|f - g\|_1 = \sum_{v \in \mathsf{V}_f \cup \mathsf{V}_g} \sum_{\tau \in \mathsf{C}: \partial\tau \ni v} F(v, \tau, h_\tau) + \sum_{(e_1, e_2) \in \mathsf{X}} \sum_{\tau \in \mathsf{C}: \partial\tau \ni e_1 \cap e_2} F(e_1 \cap e_2, \tau, h_\tau). \tag{5}$$

By preprocessing $\mathbf{T}_f$ and $\mathbf{T}_g$ for planar point-location queries, the cells of C incident on the vertices of $\mathsf{V}_f \cup \mathsf{V}_g$ can be computed in a total time of $O(n \log n)$, and thus the first term in (5) can be computed in $O(n \log n)$ time; see [16] for details. We thus focus on computing the second term, which we denote by $\Phi(\mathsf{E}_f, \mathsf{E}_g)$.

For an intersecting pair of edges $(e_1, e_2) \in \mathsf{X}$, the vertex $e_1 \cap e_2 \in \mathsf{V}$ is incident on four cells of Σ, and thus it contributes to four terms in $\Phi(\mathsf{E}_f, \mathsf{E}_g)$. Each of these terms has the following form: Let f_{11}, f_{12} (resp., g_{21}, g_{22}) be the bivariate linear functions of f (resp., g) associated with the triangles of $\mathbf{T}_f$ (resp. $\mathbf{T}_g$) lying above and below e_1 (resp., e_2) respectively. For $i, j \in \{1, 2\}$, define

$$h_{ij}(x, y) = |f_{1i}(x, y) - g_{2j}(x, y)|, \qquad \mathbb{1}(i = j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$F_{ij}(e_1, e_2) = (-1)^{\mathbb{1}(i=j)} \int_{(x,y)\in\triangle(e_1\cap e_2)} h_{ij}(x, y)\, dx\, dy;$$

in words, $F_{ij}(e_1, e_2)$ is the signed volume between the graphs of f_{1i} and g_{2j} over $\triangle(e_1 \cap e_2)$. Using these definitions, for a pair $(e_1, e_2) \in \mathsf{X}$, we can write

$$\sum_{\tau\in\mathsf{C}:\partial\tau\ni e_1\cap e_2} F(e_1 \cap e_2, \tau, h_\tau) = \sum_{i,j\in\{1,2\}} F_{ij}(e_1, e_2) \tag{6}$$

and thus

$$\Phi(\mathsf{E}_f, \mathsf{E}_g) = \sum_{i,j\in\{1,2\}} \sum_{(e_1,e_2)\in\mathsf{X}} F_{ij}(e_1, e_2). \tag{7}$$

In the next subsection we show that each F_{ij} can be written as the sum of at most five rational functions of constant degree each.

2.2 Reduction to a sum of rational expressions

We now reexamine the integrals that appear in the previous section and express them as sums of low-degree rational functions, with appropriate signs. We first focus on a single function $F_{ij}(e_1, e_2)$. This function can be seen as the volume between the two halfplanes $h_1 \colon z = f_{1i}(x, y)$ and $h_2 \colon z = g_{2j}(x, y)$ above the triangle $\triangle(e_1 \cap e_2)$.

Let $f_{1i}(x, y) = a_1x + b_1y + c_1$ and $g_{2j}(x, y) = a_2x + b_2y + c_2$ be two linear functions in x, y. By abuse of notation, we let $e_1(x) = \alpha_1x + y_1$ and $e_2(x) = \alpha_2x + y_2$ be the two linear functions in x extending the edges e_1 and e_2. After the same preprocessing as in [16], we can assume that $y_1 > y_2$, $\alpha_1 < \alpha_2$, so that the projection of the edges intersect at $x = \frac{y_1-y_2}{\alpha_2-\alpha_1} > 0$. Let h_1 and h_2 be the two halfplanes in $\mathbb{R}^3$ defined by $z = f_{1i}(x, y), y \le e_1(x)$ and $z = g_{2j}(x, y), y \ge e_2(x)$ respectively. Below we consider the triangle $T = \triangle ABC$ defined as the set of points (x, y) with $x \ge 0$ and $e_2(x) \le y \le e_1(x)$; refer to Figure 2, where $h_1^\downarrow$ and $h_2^\downarrow$ are respectively the projections of h_1 and h_2 to the xy-plane, restricted to $x \ge 0$.

The goal of this section is to prove that the volume between h_1 and h_2 over T is a sum of rational expressions in the parameters of h_1 and h_2. Their exact form depends on the sign variables $\delta_A, \delta_B, \delta_C$ defined as $+1$ or -1, where $+1$ indicates that h_1 has a higher z-coordinate than h_2 over the corresponding vertex of T. Explicit formulas for those signs are given in Section 2.2.1.

More precisely, the halfplane h_1 is contained in the plane π_1 defined by $z = f_{1i}(x, y)$ and bounded by the edge $e_1^\uparrow$, while h_2 is contained in π_2 defined by $z = g_{2j}(x, y)$ and bounded by $e_2^\uparrow$ such that:

$$h_1 \left| \begin{array}{ll} \pi_1: & z = a_1x + b_1y + c_1 \\ e_1^\uparrow: & \begin{pmatrix} t \\ y_1 + \alpha_1 t \\ z_1 + \beta_1 t \end{pmatrix} \end{array} \right. \qquad h_2 \left| \begin{array}{ll} \pi_2: & z = a_2x + b_2y + c_2 \\ e_2^\uparrow: & \begin{pmatrix} t \\ y_2 + \alpha_2 t \\ z_2 + \beta_2 t \end{pmatrix} \end{array} \right. \tag{8}$$

Moreover, since $e_1^\uparrow \subset \pi_1$ and $e_2^\uparrow \subset \pi_2$, the following relations hold:

$$\begin{cases} z_1 = b_1y_1 + c_1 \\ \beta_1 = a_1 + b_1\alpha_1 \end{cases} \qquad \begin{cases} z_2 = b_2y_2 + c_2 \\ \beta_2 = a_2 + b_2\alpha_2 \end{cases} \tag{9}$$

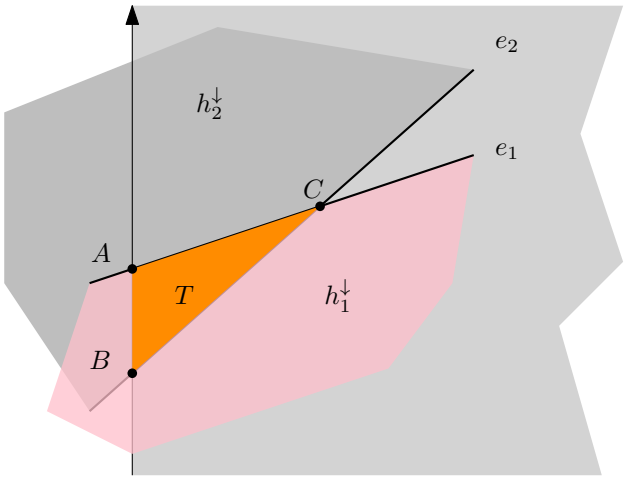


Figure 2 A typical triangle $T = \triangle ABC$ in the xy-plane.

Thus, on the xy-plane, the triangle T has three vertices A, B, C:

$$A \left| \begin{matrix} 0 \\ y_1 \end{matrix} \right. \qquad B \left| \begin{matrix} 0 \\ y_2 \end{matrix} \right. \qquad C \left| \begin{matrix} \frac{y_1 - y_2}{\alpha_2 - \alpha_1} \\ \frac{\alpha_2 y_1 - \alpha_1 y_2}{\alpha_2 - \alpha_1} \end{matrix} \right. \tag{10}$$

2.2.1 Explicit sign conditions

The explicit rational expressions for $\int_T |f_{1i} - g_{2j}|$ will depend on the value of $\delta_A, \delta_B, \delta_C$, namely the sign of the difference $f_{1i} - g_{2j}$ above A, B, and C respectively. Combining eqs. (8), (9), and (10) and using the assumption $\alpha_2 > \alpha_1$, we can explicitly expand the formula for δ_A, δ_B, and δ_C:

$$\begin{aligned} \delta_A &= \operatorname{sign}(z_1 - b_2 y_1 - c_2) \\ \delta_B &= \operatorname{sign}(b_1 y_2 + c_1 - z_2) \\ \delta_C &= \operatorname{sign}\left(\frac{(z_1 - z_2)(\alpha_2 - \alpha_1) + (\beta_1 - \beta_2)(y_1 - y_2)}{\alpha_2 - \alpha_1} \right) \\ &= \operatorname{sign}\left((z_1 - z_2)(\alpha_2 - \alpha_1) + (\beta_1 - \beta_2)(y_1 - y_2) \right) \\ &= \operatorname{sign}\left(\gamma_1 + \gamma_2 + z_1\alpha_2 + z_2\alpha_1 - \beta_1 y_2 - \beta_2 y_1 \right) \end{aligned} \tag{11}$$

where $\gamma_i = \beta_i y_i - z_i \alpha_i$, and $\operatorname{sign}(x) = +1$ if $x \geq 0$ and -1 otherwise. Note that, after the introduction of γ_i, each of $\delta_A, \delta_B, \delta_C$ is the sign of a *bilinear* expression in at most five variables. Each of those variables can also be expressed as a polynomial function of the five parameters $a_i, b_i, c_i, \alpha_i, y_i$ describing each of the two halfplanes.

2.2.2 Explicit rational expressions

The volume $\int_T |f_{1i} - g_{2j}|$ will be written as a weighted sum of three rational expressions. For that, we introduce the intersection point D of the line $e_1^{\uparrow}$ with the plane π_2, the intersection point E of $e_2^{\uparrow}$ and π_1, and the intersection point F of the line DE with the vertical plane $x = 0$. These points exist due to our general position assumptions, as soon as δ_A, δ_B and δ_C don't all have the same sign. Figures 3 and 4 illustrate those notations in different cases.

Let $D^{\downarrow}$, $E^{\downarrow}$, and $F^{\downarrow}$ be projections of D, E, and F respectively to the xy-plane. Finally, the points A_1, B_1, C_1 are the points above A, B, C respectively, with z-coordinates $f_{1i}(x_A, y_A)$, $f_{1i}(x_B, y_B)$ and $f_{1i}(x_C, y_C)$, and the points A_2, B_2, C_2 are the points above A, B, C respectively, with z-coordinates $g_{2j}(x_A, y_A)$, $g_{2j}(x_B, y_B)$ and $g_{2j}(x_C, y_C)$.

We will express the volume between h_1 and h_2 over T as a weighted sum of volumes of polytopes spawned by C, D, and E respectively, toward the vertical plane $x = 0$. The following expressions represent the signed volumes of the polytopes between the planes π_1 and π_2, above the triangles $T = \triangle ABC$, $\triangle AD^{\downarrow}F^{\downarrow}$, and $\triangle BE^{\downarrow}F^{\downarrow}$ respectively. More precisely, letting $R_C = \int_T (f_{1i} - g_{2j})$, $R_D = \int_{\triangle AD^{\downarrow}F^{\downarrow}} (f_{1i} - g_{2j})$ and $R_E = \int_{\triangle BE^{\downarrow}F^{\downarrow}} (f_{1i} - g_{2j})$, we have:

$$\begin{aligned} R_C &= \frac{1}{6} x_C (y_A - y_B) \sum_{X \in \{A,B,C\}} (f_{1i}(x_X, y_X) - g_{2j}(x_X, y_X)), \\ R_D &= \frac{1}{6} x_D (y_F - y_A)(f_{1i}(x_A, y_A) - g_{2j}(x_A, y_A)), \text{ and} \\ R_E &= \frac{1}{6} x_E (y_F - y_B)(f_{1i}(x_B, y_B) - g_{2j}(x_B, y_B)). \end{aligned} \tag{12}$$

We now proceed to express R_C, R_D, and R_E as fractions in the parameters of h_1 and h_2. In particular, we will show that the denominator of each fraction depends only on either two parameters of h_1, or two parameters of h_2.

Rational representation of R_C

Using the formula for the signed volume R_C in eq. (12), we expand the evaluation of f_{1i} and g_{2j} at A, B, C to obtain

$$\begin{aligned} R_C &= \frac{1}{6}(y_1 - y_2) \frac{y_1 - y_2}{\alpha_2 - \alpha_1} \left(\left\langle \begin{pmatrix} a_1 - a_2 \\ b_1 - b_2 \end{pmatrix}, A + B + C \right\rangle + 3(c_1 - c_2) \right) \\ &= \frac{(y_1 - y_2)^2 \left[\begin{array}{r} (\alpha_2 - \alpha_1)\big[(b_1 - b_2)(y_1 + y_2) + 3(c_1 - c_2)\big] \\ + (a_1 - a_2)(y_1 - y_2) + (b_1 - b_2)(\alpha_2 y_1 - \alpha_1 y_2) \end{array} \right]}{6\,(\alpha_2 - \alpha_1)^2}, \end{aligned} \tag{13}$$

where $\langle \cdot, \cdot \rangle$ is the dot product. This implies that R_C is a rational expression with denominator $(\alpha_2 - \alpha_1)^2$.

Rational representation of R_D and R_E

For quantities R_D and R_E, we need to compute the coordinates of D, E, and F. Using eq. (8), the abscissa t_D of D satisfies $z_1 + \beta_1 t_D = a_2 t_D + b_2 y_1 + b_2 \alpha_1 t_D + c_2$. A similar relation for the abscissa t_E of E leads to

$$\begin{aligned} t_D &= \frac{z_1 - b_2 y_1 - c_2}{b_2 \alpha_1 + a_2 - \beta_1} & t_E &= \frac{z_2 - b_1 y_2 - c_1}{b_1 \alpha_2 + a_1 - \beta_2} \\ &= \frac{(b_1 - b_2) y_1 + c_1 - c_2}{(b_2 - b_1)\alpha_1 + a_2 - a_1} & &= \frac{(b_1 - b_2) y_2 + c_1 - c_2}{(b_2 - b_1)\alpha_2 + a_2 - a_1} \end{aligned}$$

Moreover, F is at the intersection of h_1, h_2, and the vertical plane $x = 0$. In particular, its coordinates are the solution of $x = 0, z = b_1 y + c_1 = b_2 y + c_2$, so we have

$$D \left| \begin{array}{c} t_D \\ y_1 + \alpha_1 t_D \\ z_1 + \beta_1 t_D \end{array} \right. \qquad E \left| \begin{array}{c} t_E \\ y_2 + \alpha_2 t_E \\ z_2 + \beta_2 t_E \end{array} \right. \qquad F \left| \begin{array}{c} 0 \\ \frac{c_2 - c_1}{b_1 - b_2} \\ \frac{b_1 c_2 - b_2 c_1}{b_1 - b_2} \end{array} \right. \tag{14}$$

Also, the points A_1, A_2, B_1, B_2 have coordinates

$$\begin{array}{c|c|c|c|c|c|c|c} A_1 & \begin{matrix}0\\ y_1\\ b_1y_1+c_1\end{matrix} & A_2 & \begin{matrix}0\\ y_1\\ b_2y_1+c_2\end{matrix} & B_1 & \begin{matrix}0\\ y_2\\ b_1y_2+c_1\end{matrix} & B_2 & \begin{matrix}0\\ y_2\\ b_2y_2+c_2\end{matrix} \end{array} \tag{15}$$

Finally, replacing the coordinates in eq. (12), this leads to:

$$\begin{aligned} R_D &= \frac{1}{6}\frac{(b_1-b_2)y_1+c_1-c_2}{(b_2-b_1)\alpha_1+a_2-a_1}\left(\frac{c_2-c_1}{b_1-b_2}-y_1\right)(b_1y_1+c_1-b_2y_1-c_2) \\ &= \frac{1}{6}\frac{((b_1-b_2)y_1+c_1-c_2)^3}{(b_2-b_1)\left((b_2-b_1)\alpha_1+a_2-a_1\right)} \end{aligned} \tag{16}$$

and

$$\begin{aligned} R_E &= \frac{1}{6}\frac{(b_1-b_2)y_2+c_1-c_2}{(b_2-b_1)\alpha_2+a_2-a_1}\left(\frac{c_2-c_1}{b_1-b_2}-y_2\right)(b_1y_2+c_1-b_2y_2-c_2) \\ &= \frac{1}{6}\frac{((b_1-b_2)y_2+c_1-c_2)^3}{(b_2-b_1)\left((b_2-b_1)\alpha_2+a_2-a_1\right)}. \end{aligned} \tag{17}$$

2.2.3 Case distinctions for the volume expression

Our goal is to provide a formula for the volume of the polyhedron between π_1 and π_2 above the triangle ABC, as a linear combination of fractions where the denominators depend only on two variables from the triangulation $\mathbf{T}_f$ or on two variables from $\mathbf{T}_g$. This will allow us to use fast multipoint evaluation algorithms for bivariate polynomials (see Section 4).

We will see that for the different cases of signs of $\delta_A, \delta_B, \delta_C$, the volume above T can be expressed as a weighted signed sum of R_C, R_D, and R_E. We consider the cases where the vector of signs $(\delta_A, \delta_B, \delta_C)$ is $(+1,+1,+1)$, $(+1,+1,-1)$, or $(+1,-1,-1)$. The other cases are deduced by symmetry. In the following, we denote by $V_{\delta_A\delta_B\delta_C}$ the volume between π_1 and π_2 above ABC for the corresponding vector of signs.

When $\delta_A, \delta_B, \delta_C$ are positive: the halfplanes don't meet above the triangle

In this case, $V_{+++}(h_1, h_2) = R_C$, a rational expression with denominator $(\alpha_2 - \alpha_1)^2$. By symmetry, when $\delta_A, \delta_B, \delta_C$ are -1, we have $V_{---}(h_1, h_2) = -R_C$.

When δ_A, δ_B are positive and δ_C is negative

Using the notation of Section 2.2.2, the volume between the planes over T in this case is the sum of the volumes $\mathrm{vol}(A_1A_2B_2B_1DE)$ and $\mathrm{vol}(DEC_2C_1)$; refer to Figure 3. With δ_A, δ_B positive, and δ_C negative, the signed volume R_C is equal to $\mathrm{vol}(A_1A_2B_1DE) - \mathrm{vol}(DEC_2C_1)$. Thus, the volume V_{++-} can be expressed as

$$V_{++-}(h_1, h_2) = 2\,\mathrm{vol}(A_1A_2B_2B_1DE) - R_C\,.$$

The volume of the polytope $A_1A_2B_2B_1DE$ is computed as the difference of volumes of the two simplices B_1B_2FE and A_1A_2FD. When δ_A and δ_B are positive, the y-coordinate of the point F can be either (i) greater than y_A, or (ii) less than y_B.

In case (i), we have $\mathrm{vol}(A_1A_2B_2B_1DE) = \mathrm{vol}(B_1B_2FE) - \mathrm{vol}(A_1A_2FD)$. Moreover $\mathrm{vol}(A_1A_2FD) = R_D$ and $\mathrm{vol}(B_1B_2FE) = R_E$, so that $\mathrm{vol}(A_1A_2B_2B_1DE) = R_E - R_D$.

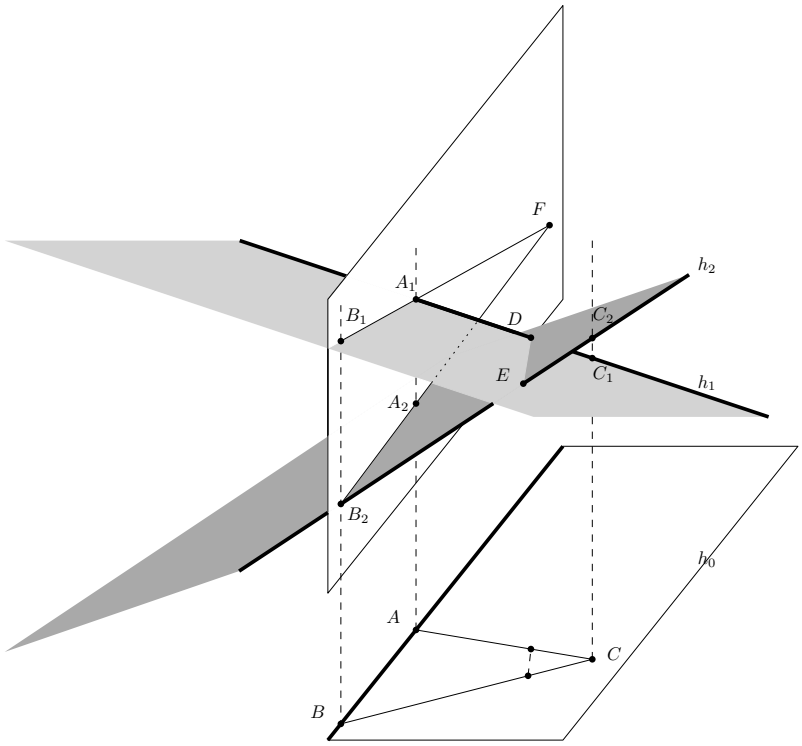


Figure 3 Case where δ_C has a sign different from δ_A and δ_B.

In case (ii), we have $\mathrm{vol}(A_1A_2B_2B_1DE) = \mathrm{vol}(A_1A_2FD) - \mathrm{vol}(B_1B_2FE)$, $\mathrm{vol}(A_1A_2FD) = -R_D$, and $\mathrm{vol}(B_1B_2FE) = -R_E$, so that once again $\mathrm{vol}(A_1A_2B_2B_1DE) = R_E - R_D$.

Thus, in both cases, we have $\mathrm{vol}(A_1A_2B_2B_1DE) = R_E - R_D$, and therefore

$$V_{++-}(h_1, h_2) = 2R_E - 2R_D - R_C\,.$$

Similarly, by symmetry, we have $V_{--+} = -2R_E + 2R_D + R_C$.

When δ_A is positive and δ_B, δ_C are negative

This case is illustrated in Figure 4. Using the notation of the previous section the volume between the two halfplanes is the sum of volumes of the polytopes A_1A_2FD and $B_1B_2C_2C_1DF$. First, with δ_A positive and δ_B, δ_C negative, the signed volume R_C is $\mathrm{vol}(A_1A_2FD) - \mathrm{vol}(B_1B_2C_2C_1DF)$. Moreover, since δ_A is positive and δ_B is negative, the y-coordinate of F is in the interval $[y_B, y_A]$ and we have $\mathrm{vol}(A_1A_2FD) = -R_D$. Thus, the volume V_{+--} can be expressed as

$$V_{+--}(h_1, h_2) = 2\,\mathrm{vol}(A_1A_2FD) - R_C = -2R_D - R_C\,. \tag{18}$$

By symmetry, we have $V_{-++} = 2R_D + R_C$. Moreover, the role of D and E are symmetric, and when δ_A, δ_C are negative, and δ_B is positive, we have similarly $V_{-+-} = 2\,\mathrm{vol}(B_2B_1FE) - R_C = 2R_E - R_C$ and $V_{+-+} = -2R_E + R_C$.

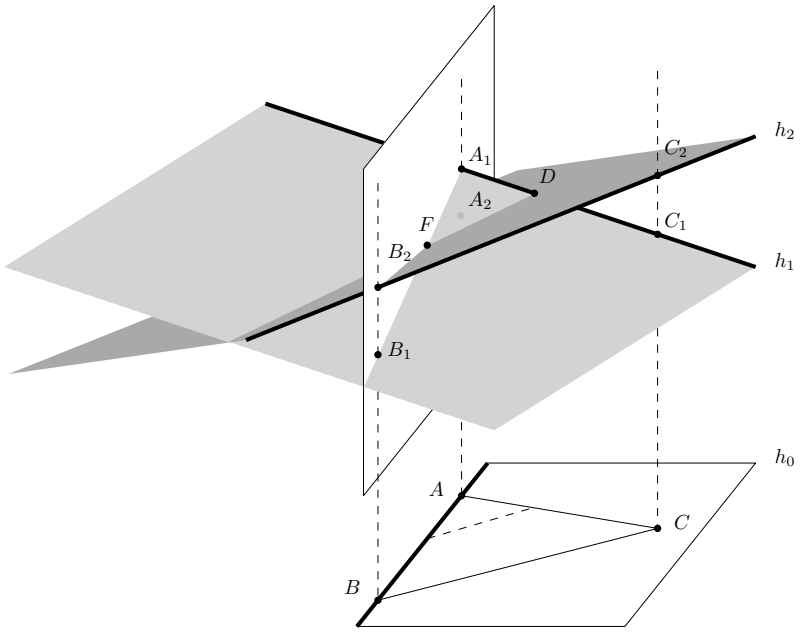


Figure 4 Case where δ_A has a sign different from δ_B and δ_C.

2.2.4 Gathering all the cases

Depending on the signs δ_A, δ_B, and δ_C, we have different formulas for the volume between the planes over T. We can gather them all within one expression that is bilinear in $\delta_A, \delta_B, \delta_C$ on one side, and R_A, R_B, R_C on the other side.

▶ **Lemma 2.** *Assume that $v_1 = (a_1, b_1, c_1, y_1, \alpha_1)$ and $v_2 = (a_2, b_2, c_2, y_2, \alpha_2)$ are vectors of real numbers such that $y_1 > y_2$ and $\alpha_1 < \alpha_2$. Let $T \subset \mathbb{R}^2$ be a triangular cell defined by the set of points (x, y) such that $x \geq 0$, $y \leq y_1 + \alpha_1 x$, and $y \geq y_2 + \alpha_2 x$. And let f_{1i} and g_{2j} be two linear functions defined by $f_{1i}(x, y) = a_1 x + b_1 y + c_1$ and $g_{2j}(x, y) = a_2 x + b_2 y + c_2$. Then the volume $\int_T |f_{1i} - g_{2j}|$ can be expressed as:*

$$\int_{(x,y)\in T} |f_{1i}(x, y) - g_{2j}(x, y)|\, dx\, dy = \delta_C(R_D - R_E + R_C) - \delta_A R_D + \delta_B R_E$$

where $\delta_A, \delta_B, \delta_C$, defined in eq. (11), depend on at most four entries of v_1 and four entries of v_2, and:

$$R_C(v_1; v_2) = \frac{p_C(v_1; v_2)}{q_C(\alpha_1; \alpha_2)} \qquad R_D(v_1; v_2) = \frac{p_A(v_1, v_2)}{q_A(v_1; (a_2, b_2))} \qquad R_E(v_1; v_2) = \frac{p_B(v_1, v_2)}{q_B((a_1, b_1); v_2)}$$

are rational functions of constant degrees defined in eqs. (13), (16) and (17).

▶ Remark. If instead of the assumptions $y_1 > y_2$ and $\alpha_1 < \alpha_2$, we have the assumptions $y_2 < y_1$ and $\alpha_2 < \alpha_1$, then the signed volume above the triangle ABC becomes $-R_C$, and δ_C becomes $-\delta_C$ since we factored out the sign of $\alpha_2 - \alpha_1$ in eq. (11). Without assumption on the order between α_1 and α_2, letting $\sigma = \text{sign}(\alpha_2 - \alpha_1)$, the formula for the volume becomes:

$$\int_{(x,y)\in T} |f_{1i}(x, y) - g_{2j}(x, y)|\, dx\, dy = \delta_C R_C + \delta_C \sigma(R_D - R_E) - \delta_A R_D + \delta_B R_E\,.$$

Returning to expressing (7) as a sum of rational functions, we introduce the following notation. For the edge $e_1 \in \mathsf{E}_f$, e_{11}^* (resp., e_{12}^*) is the point $(a_1, b_1, c_1, y_1, \alpha_1)$ in $\mathbb{R}^5$ corresponding to the linear function associated with the triangle of $\mathbf{T}_f$ lying below (resp., above) the line supporting e_i. The same notation holds for edges of E_g.

$$F_{ij}(e_1, e_2) = (-1)^{\mathbb{1}(i=j)} \Big[\delta_C(e_{1i}^*, e_{2j}^*) R_C(e_{1i}^*, e_{2j}^*) + \sigma(e_{1i}^*, e_{2j}^*) \delta_C(e_{1i}^*, e_{2j}^*)(R_D(e_{1i}^*, e_{2j}^*) - R_E(e_{1i}^*, e_{2j}^*)) - \delta_A(e_{1i}^*, e_{2j}^*) R_D(e_{1i}^*, e_{2j}^*) + \delta_B(e_{1i}^*, e_{2j}^*) R_E(e_{1i}^*, e_{2j}^*) \Big].$$

Since there are five terms in the above expression for F_{ij}, eq. (7) can be expressed as follows:

▶ **Corollary 3.** *Let $\mathsf{E}_f, \mathsf{E}_g$ be the set of non-boundary edges in $\mathbf{T}_f$ and $\mathbf{T}_g$ respectively, and $\mathsf{X} \subseteq \mathsf{E}_f \times \mathsf{E}_g$ be the intersecting pairs of segments. Then*

$$\Phi(\mathsf{E}_f, \mathsf{E}_g) = \sum_{k=1}^{5} \sum_{i,j \in \{1,2\}} \sum_{(e_1,e_2) \in \mathsf{X}} (-1)^{\mathbb{1}(i=j)} \sigma_{ijk}(e_{1i}^*, e_{2j}^*) \cdot \xi_{ijk}(e_{1i}^*, e_{2j}^*) \cdot F_{ijk}(e_{1i}^*, e_{2j}^*), \quad (19)$$

where σ_{ijk} is either σ or the constant function 1, $\xi_{ijk} \in \{\pm\delta_A, \pm\delta_B, \pm\delta_C\}$, and $F_{ijk} \in \{R_C, R_D, R_E\}$.

3 Computing a Biclique Partition

Using Corollary 3, $\Phi(\mathsf{E}_f, \mathsf{E}_g)$ can be computed in $O(\kappa)$ time, where $\kappa = |\mathsf{X}|$ is the number of intersecting pairs of edges in $\mathsf{E}_f \times \mathsf{E}_g$, in a straightforward manner if we have the overlay Σ of $\mathbf{T}_f$ and $\mathbf{T}_g$ at our disposal. In this section, we show how to group the terms in Equation (19) using biclique partitions of X, so that the total time spent in evaluating the sum is always subquadratic. We describe the algorithm for computing

$$\Phi_C(\mathsf{E}_f, \mathsf{E}_g) = \sum_{(e_1^*, e_2^*) \in \mathsf{X}} \sigma(e_1, e_2) \delta_C(e_{11}^*, e_{22}^*) R_C(e_{11}^*, e_{22}^*).$$

The other terms (those involving R_D and R_E) can be evaluated in an analogous manner. For simplicity, we use e_i^* to denote e_{i1}^* (resp. e_{i2}^*) if e_i is a segment of E_f (resp. E_g). We compute the desired biclique partition of X in two stages. The first stage is similar to that described in [16], but the second stage is more involved and relies on geometric cuttings [11, 3].

First stage. As in Chazelle et al. [12] (see also [16]), we compute, in time $O(n \log^2 n)$, a *biclique partition* $\mathcal{B} = \{(R_1, B_1), \ldots, (R_s, B_s)\}$ of X such that

- **(i)** for every $k \le s$, $R_k \subseteq \mathsf{E}_f$ and $B_k \subseteq \mathsf{E}_g$ and every segment in R_k intersects every segment in B_k;
- **(ii)** every segment in R_k has lower slope than that of every segment in B_k, or vice-versa;
- **(iii)** for every intersecting pair $(e_1, e_2) \in \mathsf{X}$, there exists exactly one $k \le s$ with $e_1 \in R_k$ and $e_2 \in B_k$; and
- **(iv)** $\sum_k (|R_k| + |B_k|) = O(n \log^2 n)$ and $\sum_k |R_k| \times |B_k| = \kappa$.

For any $1 \le k \le s$, $\sigma(e_1^*, e_2^*)$ is the same for all $(e_1, e_2) \in R_k \times B_k$, which we denote by $\sigma(R_k, B_k)$. Therefore, we express $\Phi_C(\mathsf{E}_f, \mathsf{E}_g)$ in terms of $\mathcal{B}$ as follows:

$$\Phi_C(\mathsf{E}_f, \mathsf{E}_g) = \sum_{k=1}^{s} \sigma(R_k, B_k) \sum_{(e_1,e_2) \in R_k \times B_k} \delta_C(e_1^*, e_1^*) \cdot R_C(e_1^*, e_2^*). \quad (20)$$

We fix a biclique $(R_k, B_k) \in \mathcal{B}$. Without loss of generality, assume that $\sigma(R_k, B_k) = 1$. We describe how to compute

$$\Phi_C(R_k, B_k) = \sum_{(e_1,e_2)\in R_k\times B_k} \delta_C(e_1^*, e_2^*) \cdot R_C(e_1^*, e_2^*).$$

Second stage. We thus have the following problem at hand. Let $\mathsf{R} \subseteq \mathsf{E}_f$ and $\mathsf{B} \subseteq \mathsf{E}_g$ be two sets of segments in $\mathbb{R}^2$ such that every pair of segments in $\mathsf{R} \times \mathsf{B}$ intersects and the slope of every segment in R is less than that of any segment in B; set $m = |\mathsf{R}|$ and $n = |\mathsf{B}|$. Without loss of generality assume that $m \le n$. We compute a *sign-invariant* (biclique) partition $\Pi = \{(\mathsf{R}_1, \mathsf{B}_1), \ldots, (\mathsf{R}_u, \mathsf{B}_u)\}$ of $\mathsf{R} \times \mathsf{B}$, as described below, such that for all $1 \le k \le u$, the value of $\delta_C(e_i^*, e_j^*)$ is the same for all pairs $(e_i, e_j) \in (\mathsf{R}_k, \mathsf{B}_k)$, which we denote by $\delta_C(\mathsf{R}_k, \mathsf{B}_k)$, so that

$$\Phi_C(\mathsf{R}, \mathsf{B}) = \sum_{k=1}^{u} \left(\delta_C(\mathsf{R}_k, \mathsf{B}_k) \cdot \sum_{(e_i,e_j)\in \mathsf{R}_k\times \mathsf{B}_k} R_C(e_i^*, e_j^*) \right). \tag{21}$$

We show below in Section 4 how $\Phi_C(\mathsf{R}_k, \mathsf{B}_k) = \sum_{(e_i,e_j)\in \mathsf{R}_k\times \mathsf{B}_k} \Phi_C(e_i^*, e_j^*)$ can be computed efficiently. It thus suffices to describe how we compute Π and to analyze the overall run time.

By eq. (11), δ_C is the sign function of a bilinear function $P(\boldsymbol{\varphi}, \boldsymbol{\psi})$, where $\boldsymbol{\varphi}$ (resp. $\boldsymbol{\psi}$) is a set of five variables representing the parameters $y_i, z_i, \alpha_i, \beta_i, \gamma_i$ corresponding to the edges e_i of R (resp., B) used in the definition of δ_C; note that four of these parameters are independent and $\gamma_i = z_i\alpha_i - \beta_i y_i$ depends on them. Let Γ denote the quadratic surface $\gamma_i = z_i\alpha_i - y_i\beta_i$. Abusing the notation slightly, we continue to represent an edge $e_i \in \mathsf{R}$ as a point e_i^* in $\mathbb{R}^5$ with the new set of parameters. We also map e_i to a hyperplane $\tilde{e}_i \colon P(e_i^*, \boldsymbol{\psi}) = 0$ in $\mathbb{R}^5$. Similarly, we map an edge $e_j \in \mathsf{B}$ to a point $e_j^* \in \mathbb{R}^5$ and a hyperplane $\tilde{e}_j \colon P(\boldsymbol{\varphi}, e_j^*) = 0$ in $\mathbb{R}^5$. For a subset A of R (or B), we define $\mathsf{A}^* = \{e_j^* \mid e_j \in \mathsf{A}\}$ and $\widetilde{\mathsf{A}} = \{\tilde{e}_j \mid e_j \in \mathsf{A}\}$. By definition, $\mathsf{A}^* \subset \Gamma$. Our goal is to compute a partition Π of $\mathsf{R} \times \mathsf{B}$ such that for any pair $(\mathsf{R}_k, \mathsf{B}_k) \in \Pi$, all points of B_k^* lie on the same side of all hyperplanes in $\widetilde{\mathsf{R}}_k$, or equivalently all points of R_k^* lie on the same side of all hyperplanes in $\widetilde{\mathsf{B}}_k$.

Let H be a set of m hyperplanes in $\mathbb{R}^d$ and Δ be a simplex. For a parameter $r > 1$, a partition of Δ into a family Ξ of simplices, referred to as *cells*, is called a $(1/r)$-*cutting* of H within Δ if every cell of Ξ is crossed by at most m/r hyperplanes of H. (For $r > m$, cells of Ξ are not crossed by any hyperplane of H, i.e., Ξ is a refinement of $\mathcal{A}(\mathsf{H})$.) The *conflict list* of a cell $\triangle \in \Xi$, denoted by $\mathsf{H}_\triangle$, is the subset of hyperplanes that cross $\triangle$. It is well known that a $(1/r)$-cutting of H of size $O(r^d)$ along with the conflict lists of all of its cells can be computed in $O(mr^{d-1})$ time [11, 15]. For any point set $P \subset \mathbb{R}^d$, one can compute $P \cap \triangle$ for every $\triangle \in \Xi$ in additional $O(|P| \log r)$ time.

In our context, we compute the desired sign-invariant partition Π of $\mathsf{R} \times \mathsf{B}$ using geometric cuttings in a round-robin manner as in [4, 15]. We choose r to be a sufficiently large constant, see below. We assume that the input is *balanced*, i.e., $n/2 \le m \le n$. Otherwise (i.e., $m < n/2$), we use the standard batching technique – we partition B into $\ell = \lfloor n/m \rfloor$ subsets $\mathsf{B}_1, \ldots, \mathsf{B}_\ell$, each of size between m and $2m$, and compute the desired partition for each pair $(\mathsf{R}, \mathsf{B}_j)$, $1 \le j \le \ell$, independently.

Each round of the algorithm works in two stages. In the first stage, we map R to the set $\widetilde{\mathsf{R}}$ of hyperplanes in $\mathbb{R}^5$ and compute a $(1/r)$-cutting Ξ of size $O(r^5)$. For each cell $\triangle \in \Xi$, let $\widetilde{\mathsf{R}}_\triangle \subseteq \widetilde{\mathsf{R}}$ be the conflict list of $\triangle$, let $\widetilde{\mathsf{R}}_\triangle^+$ (resp., $\widetilde{\mathsf{R}}_\triangle^-$) be the set of hyperplanes of $\widetilde{\mathsf{R}}$ that lie above (resp., below) $\triangle$. Let $\mathsf{R}_\triangle, \mathsf{R}_\triangle^+, \mathsf{R}_\triangle^- \subseteq \mathsf{R}$ be the sets of segments correspondig to $\widetilde{\mathsf{R}}_\triangle, \widetilde{\mathsf{R}}_\triangle^+, \widetilde{\mathsf{R}}_\triangle^-$, respectively. Let $\mathsf{B}_\triangle^* = \mathsf{B}^* \cap \triangle$ and $\mathsf{B}_\triangle = \{e_j \mid e_j^* \in \mathsf{B}_\triangle^*\}$. Set $m_\triangle = |\mathsf{R}_\triangle|$ and

$n_\triangle = |\mathsf{B}_\triangle|$. It follows from known results [8, 7, 1] that the surface Γ intersects a subset $\widetilde{\Xi}$ of $O(r^4 \log r)$ simplices of Ξ. Therefore $\mathsf{B}_\triangle \neq \emptyset$ implies that $\triangle \in \widetilde{\Xi}$, so we focus on $\widetilde{\Xi}$ and ignore the cells of $\Xi \setminus \widetilde{\Xi}$. If $n_\triangle \geq n/r^4$ for any cell $\triangle \in \widetilde{\Xi}$, we partition $\triangle$ further into subcells each of which contains at most n/r^4 points of B^*. With a slight abuse of notation, we will continue to use $\widetilde{\Xi}$ to denote the set of refined cells that contain a point of B^*. The number of cells even after this refinement is at most $c_0 r^4 \log r$ for some constant $c_0 > 0$. For every cell $\triangle \in \widetilde{\Xi}$, we add $(\mathsf{R}^+_\triangle, \mathsf{B}_\triangle)$ and $(\mathsf{R}^-_\triangle, \mathsf{B}_\triangle)$ as bicliques to the partition Π. This completes the description of the first stage. which takes $O(mr^4 + n \log r) = O(n)$ since $r = O(1)$ and $m \leq n$.

In the second stage, for each cell $\triangle \in \Xi$, we map $\mathsf{B}_\triangle$ to the set $\widetilde{\mathsf{B}}_\triangle$ of hyperplanes and compute a $(1/r)$ cutting $\Xi_\triangle$ of $\widetilde{\mathsf{B}}_\triangle$ of size $O(r^5)$. For a cell $\tau \in \Xi_\triangle$, we define $\mathsf{B}_\tau, \mathsf{B}^+_\tau, \mathsf{B}^-_\tau, \mathsf{R}_\triangle$ as in the first stage (but reversing the roles of R and B). Let $\widetilde{\Xi}_\triangle$ be the subset of cells of $\Xi_\triangle$ that contain a point of $\mathsf{R}^*_\triangle$. If $|\mathsf{R}_\tau| \geq m/r^5$, we split τ into subcells each of which contains at most m/r^5 points of R_τ. Let $m_\tau = |\mathsf{R}_\tau|$ (after the refinement) and $n_\tau = |\mathsf{B}_\tau|$. By construction, $m_\tau \leq m/r^5$ and $n_\tau \leq n_\triangle/r \leq n/r^5$ for all $\tau \in \widetilde{\Xi}_\triangle$ and for all $\triangle \in \widetilde{\Xi}$. For every $\triangle \in \widetilde{\Xi}$ and $\tau \in \widetilde{\Xi}_\triangle$, we add $(\mathsf{R}_\tau, \mathsf{B}^+_\tau)$ and $(\mathsf{R}_\tau, \mathsf{B}^-_\tau)$ as bicliques to the partition Π.

For every cell τ of a second-stage cutting, we recursively compute a partition of $(\mathsf{R}_\tau, \mathsf{B}_\tau)$. The recursion stops when $\min\{|\mathsf{R}_\tau|, |\mathsf{B}_\tau|\} < r$, in which case we add each pair $(e_i, e_j) \in \mathsf{R}_\tau \times \mathsf{B}_\tau$ as a singleton biclique to Π. This completes the description of computing Π. The correctness follows using a standard argument, see, e.g., [3].

Finally, for each biclique $(\mathsf{R}_k, \mathsf{B}_k) \in \Pi$, we compute $\Phi_C(\mathsf{R}_k, \mathsf{B}_k)$ using Corollary 7. This completes the description of the algorithm.

Analysis. We now analyze the overall runtime of the algorithm, including the time spent in computing $\Phi_C(\mathsf{R}_k, \mathsf{B}_k)$ for each clique $(\mathsf{R}_k, \mathsf{B}_k) \in \Pi$. First, we focus on the run time for the balanced case. For notational simplicity, assume that $|\mathsf{R}| = |\mathsf{B}| = n$. Let $T(n)$ be the total time spent in computing the partition Π plus the time spent in computing $\Phi_C(\cdot, \cdot)$ for each biclique in Π.

If $n < n_0$, where n_0 is a constant that depends on r, the algorithm takes $O(1)$ time. Otherwise, the algorithm spends $O(n)$ time in the two stages of the first round to compute the cuttings, the conflict lists, and the bicliques. Then it solves at most $c_1 r^8 \log^2 r$ recursive problems, where $c_1 > 0$ is a constant independent of r, each of size at most n/r^5. Furthermore, the first round generates at most $O(r^8 \log^2 r)$ bicliques, and by Corollary 7, computing Φ_C for each biclique takes $\tilde{O}(n^{(\omega+1)/2})$. Thus the total time spent in the first round in computing the bicliques and Φ_C for all these is at most $c_3 n^{(\omega+1)/2} \log^{c_2} n$, where $c_2 > 0$ is a constant independent of r and $c_3 > 0$ is a constant that depends on r. Hence, we obtain the following recurrence for $T(n)$:

$$T(n) \leq \begin{cases} c_1 r^8 \log^2 r \cdot T(n/r^5) + c_3 n^{(\omega+1)/2} \log^{c_2} n & n \geq n_0, \\ c_4 & n < n_0, \end{cases} \tag{22}$$

where $c_4 > 0$ and $n_0 > 0$ are constants that depend on r. Note that the recursive subproblems may not be balanced but the total time spent in a round with subproblem size $t \geq n_0$ is still at most $c_3 t^{(\omega+1)/2} \log^{c_2} t$, so the above recurrence holds.

Assuming r is chosen sufficiently large, the solution of the above recurrence is

$$T(n) \leq A n^\alpha \log^{c_2} n, \qquad \text{where } \alpha = \max\{8/5 + \varepsilon, (\omega+1)/2\} \tag{23}$$

for any constant $\varepsilon > 0$, where A is an appropriate constant that depends on r. Since the current best known value of ω is roughly 2.37, we simply write the solution of the recurrence as $T(n) = \tilde{O}(n^{(\omega+1)/2})$.

For general values of m and n with $m \leq n$, the batching technique implies that the total time spent in computing $\Phi_C(\mathsf{R}, \mathsf{B})$ is $(1 + n/m) \cdot \tilde{O}(m^\alpha) = \tilde{O}(nm^{\alpha-1})$. For $m > n$, we flip the roles of R and B, and the run time is $\tilde{O}(mn^{\alpha-1})$. Hence, we obtain the following:

▶ **Lemma 4.** *For a biclique (R_k, B_k) in the biclique partition computed in the first stage, with $|R_k| = m_k$ and $|B_k| = n_k$, $\Phi_C(R_k, B_k)$ can be computed in $\tilde{O}(m_k n_k^{\alpha-1} + n_k m_k^{\alpha-1})$ time.*

To bound the total time spent in computing $\Phi(\mathsf{E}_f, \mathsf{E}_g)$, we sum the bound in Lemma 4 over all bicliques of $\mathcal{B}$. Using Hölder's inequality and property (iv) of the biclique partition $\mathcal{B}$, we obtain the following:

$$\begin{aligned}\sum_{k=1}^{s} m_k n_k^{\alpha-1} &= \sum_{k=1}^{s} (m_k n_k)^{\alpha-1} m_k^{2-\alpha} \\ &\leq \left(\sum_{k=1}^{s} m_k n_k\right)^{\alpha-1} \cdot \left(\sum_{k=1}^{s} m_k\right)^{2-\alpha} \\ &= \tilde{O}(\kappa^{\alpha-1} n^{2-\alpha}),\end{aligned}$$

which is $\tilde{O}(n)$ for $\kappa = n$ and $\tilde{O}(n^\alpha)$ for $\kappa = n^2$. Similarly, we can argue that $\sum_k n_k m_k^{\alpha-1} = \tilde{O}(\kappa^{\alpha-1} n^{2-\alpha})$. Adding the $O(n \log n)$ time spent in evaluating (5) at the vertices of $\mathsf{V}_f \cup \mathsf{V}_g$, the total time spent in computing $\|f - g\|_1$ is $\tilde{O}(\kappa^{\alpha-1} n^{2-\alpha})$. This proves Theorem 1.

4 Multipoint Evaluation of Bivariate Rational Functions

In this section, we show how computing the quantities $\Phi_C(\mathsf{R}, \mathsf{B})$ associated to each $m \times n$ biclique $\mathsf{R} \times \mathsf{B}$ can be reduced to computing a double sum of m rational functions $\frac{1}{q_j(x,y)}$ over n points $p_i = (x_i, y_i)$. This sum can then be evaluated using fast multipoint evaluation techniques for bivariate polynomials ([17]), based on fast modular composition and linear algebra ([21, Chapter 12, §12.2]). Recall that we work in the Real RAM model of computation, where the fastest known algorithm for achieving this runs in subquadratic time [17]. As in the previous section, the other terms appearing in $\Phi(\mathsf{E}_f, \mathsf{E}_g)$ can be computed in an analogous manner. We start with a technical lemma on the fast multipoint evaluation of a batch of bivariate polynomials.

▶ **Lemma 5.** *Given n polynomials $f_j(x, y)$ of degree n and n^2 points $p_i \in \mathbb{R}^2$, it is possible to evaluate all the $f_j(p_i)$ in $\tilde{O}(n^{\omega+1})$ arithmetic operations.*

Proof. The main idea is to reduce this question to a problem of fast multipoint evaluation of univariate polynomials.

First, we interpolate two polynomials $g(x)$ and $h(x)$ of degree n^2, such that, for each root r of h, the point $(r, g(r))$ is a point p_i. More formally we require that

$$\{p_1, \ldots, p_{n^2}\} = \{(x_i, g(x_i)) \mid h(x_i) = 0\}.$$

(For clarity of presentation, we assume that all the coordinates x_i are distinct; the general case can be handled by introducing a generic shear transformation of the xy-plane at no asymptotic cost; see [17, §6].) Polynomials $g(x)$ and $h(x)$ can be computed in $\tilde{O}(n^2)$ operations using fast interpolation algorithms [21, Chapter 10.2].

Then, denoting by $\widetilde{f}_j(x)$ the univariate polynomial $f_j(x, g(x)) \bmod h(x)$, note that $f_j(x_i, y_i) = \widetilde{f}_j(x_i)$ for all i. Hence, if we know the coefficients of $\widetilde{f}_j$, we can evaluate each $\widetilde{f}_j$ at all x_i in $\tilde{O}(n^2)$ arithmetic operations, and all $\widetilde{f}_j$ at all x_i in $\tilde{O}(n^3)$ operations.

Finally, computing all the $\widetilde{f_j}$ can be done in $\tilde{O}(n^{\omega+1})$ arithmetic operations using Lemma 10(iii) of [17], which can be seen as a variation of the fast composition algorithm 12.3 in [21, Chapter 12, §12.2]. Thus, the final complexity is in $\tilde{O}(n^{\max(3,\omega+1)}) = \tilde{O}(n^{\omega+1})$. ◀

We can now deduce our corollaries for computing the sum of rational functions. First we start with the case where the numerators are all 1.

▶ **Corollary 6.** *Given n constant-degree polynomials $q_j(x,y)$ and n points $p_i \in \mathbb{R}^2$, the n sums $\sum_j \frac{1}{q_j(p_i)}$ for all $1 \leq i \leq n$ can be computed in $\tilde{O}(n^{\frac{\omega+1}{2}})$ arithmetic operations.*

Proof. First, partitioning the set of indices j into $O(\sqrt{n})$ subsets J_ℓ of size $O(\sqrt{n})$, for each ℓ, there exists a polynomial $f_\ell(x,y)$ of degree $O(\sqrt{n})$ such that $\sum_{j\in J_\ell} \frac{1}{q_j(x,y)} = \frac{f_\ell(x,y)}{\prod_{j\in J_\ell} q_j(x,y)}$. Each f_ℓ can be computed in $\tilde{O}(n)$ arithmetic operations, using fast bivariate multiplication [21, §8.4]. Then, using Lemma 5, we can evaluate all the $O(\sqrt{n})$ polynomials f_ℓ on all the n points p_i in $\tilde{O}(n^{\frac{\omega+1}{2}})$ operations. With the same approach, we can expand the denominators and evaluate them on the points p_j in $\tilde{O}(n^{\frac{\omega+1}{2}})$ operations. ◀

Then we generalize this result to sums of general rational functions.

▶ **Corollary 7.** *For positive integers k, n, j, with $j \leq n$, let f_j and g_j be polynomials of constant degree and in constant number of variables, k and 2 respectively. Let $A \subset \mathbb{R}^k$ contain at most n points. In the Real RAM model, the sum*

$$S = \sum_{(a_1,\ldots,a_k)\in A} \sum_{1\leq j\leq n} \frac{f_j(a_1,\ldots,a_k)}{g_j(a_1,a_2)}$$

can be computed in $\tilde{O}(n^{\frac{\omega+1}{2}})$ arithmetic operations.

Proof. Given k symbolic variables $X_1,\ldots,X_k$, all the polynomials $f_j(X_1,\ldots,X_k)$ can be written as a weighted sum of s monomials M_ℓ, where $M_\ell = X_1^{e_{\ell,1}} \cdots X_k^{e_{\ell,k}}$ with $(e_{\ell,1},\ldots,e_{\ell,k}) \in \mathbb{N}^k$. In particular, each polynomial $f_j(X_1,\ldots,X_k)$ can be expanded as

$$c_{j,1}M_1 + \cdots + c_{j,s}M_s\,.$$

According to Lemma 6, letting $q_j = \frac{g_j}{c_{j,1}}$, we can compute for all the points in $(a_1,\ldots,a_k) \in A$ the sums $\sum_{j=1}^n \frac{1}{q_j(a_1,a_2)}$ in $\tilde{O}(n^{\frac{\omega+1}{2}})$ operations. Since the number s of monomials is constant, we can compute with the same complexity for all $1 \leq j \leq s$ and all points $p = (a_1,\ldots,a_k) \in A$ the sums :

$$S_{p,\ell} = \sum_{j=1}^{n} \frac{c_{j,\ell}}{g_j(a_1,a_2)}\,.$$

On the other hand, since s and the degrees of the f_j are constant, we can directly compute all the monomials M_ℓ at all the points of A in $O(n)$ operations. Using an additional number of operations linear in n, this allows us to compute, for all the points in $p \in A$,

$$S_p = \sum_{j=1}^{n} \frac{f_j(a_1,\ldots,a_k)}{g_j(a_1,a_2)} = \sum_{\ell=1}^{s} S_{p,\ell}M_\ell(p)\,.$$

Finally, we get the desired result by summing the quantities S_p over all $p \in A$. ◀

This concludes the description of the multipoint evaluation tools needed in Section 3 to complete our algorithm.

5 Conclusion

A recent preprint improves the bound on multipoint evaluation for multivariate polynomials. It shows that we could evaluate $n^{\omega-1}$ bivariate polynomials of degree n^2 on n^2 points in $\tilde{O}(n^{\omega+1})$ operations [19, Theorem 3.8]. Compared to Lemma 5 derived from earlier work [17], this allows us to evaluate more polynomials with the same number of operations. In turn, that would improve Corollary 7, and the complexity of evaluating the sum of rational functions on a biclique of size $n \times n$ would become $\tilde{O}(n^{2-1/\omega})$ instead of $O(n^{(\omega+1)/2})$, by summing over batches of $n^{2/\omega}$ points. With the currently known bound on ω, $2-1/\omega$ is approximately 1.58, which is below 8/5. Thus the cost of computing the biclique decomposition would become dominant, and the total cost of computing $\|f-g\|_1$ could be further improved to $O(n^{8/5+\varepsilon})$, for any constant $\varepsilon > 0$.

References

1 Pankaj K. Agarwal. Simplex range searching and its variants: A review. In *Journey through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 1–30. Springer, Berlin, 2017. doi:10.1007/978-3-319-44479-6_1.

2 Pankaj K. Agarwal, Boris Aronov, Marc Van Kreveld, Maarten Löffler, and Rodrigo I. Silveira. Computing correlation between piecewise-linear functions. *SIAM J. Comput.*, 42(5):1867–1887, 2013. doi:10.1137/120900708.

3 Pankaj K. Agarwal, Esther Ezra, and Micha Sharir. Semi-algebraic off-line range searching and biclique partitions in the plane. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, pages 4:1–4:15, 2024. doi:10.4230/LIPIcs.SoCG.2024.4.

4 Pankaj K. Agarwal and Micha Sharir. The number of congruent simplices in a point set. *Discret. Comput. Geom.*, 28(2):123–150, 2002. doi:10.1007/s00454-002-0727-x.

5 Deepak Ajwani, Saurabh Ray, Raimund Seidel, and Hans Raj Tiwary. On computing the centroid of the vertices of an arrangement and related problems. In *Algorithms and data structures. 10th international workshop, WADS 2007, Halifax, Canada, August 15–17, 2007. Proceedings.*, pages 519–528. Berlin: Springer, 2007. doi:10.1007/978-3-540-73951-7_45.

6 Boris Aronov and Matthew J. Katz. Batched point location in SINR diagrams via algebraic tools. *ACM Trans. Algorithms*, 14(4), August 2018. doi:10.1145/3209678.

7 Boris Aronov, Marco Pellegrini, and Micha Sharir. On the zone of a surface in a hyperplane arrangement. *Discrete Comput. Geom.*, 9(2):177–186, 1993. doi:10.1007/BF02189317.

8 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the number of cells defined by a family of polynomials on a variety. *Mathematika*, 43(1):120–126, 1996. doi:10.1112/S0025579300011621.

9 Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. *J. ACM*, 71(3):32, 2024. Id/No 22. doi:10.1145/3652025.

10 Sergio Cabello and Timothy M. Chan. Computing Shapley values in the plane. *Discrete Comput. Geom.*, 67(3):843–881, 2022. doi:10.1007/s00454-021-00368-3.

11 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discret. Comput. Geom.*, 9:145–158, 1993. doi:10.1007/BF02189314.

12 Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica*, pages 116–132, 1994. doi:10.1007/BF01182771.

13 Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 88–93. ACM, 1972. doi:10.1145/800152.804900.

14 Sumanta Ghosh, Prahladh Harsha, Simao Herdade, Mrinal Kumar, and Ramprasad Saptharishi. Fast numerical multivariate multipoint evaluation. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1426–1439. IEEE, 2023. doi:10.1109/FOCS57990.2023.00088.

15 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10(2):157–182, 1993. doi:10.1007/BF02573972.

16 Guillaume Moroz and Boris Aronov. Computing the distance between piecewise-linear bivariate functions. *ACM Trans. Algorithms*, 12(1), 2016. doi:10.1145/2847257.

17 Michael Nüsken and Martin Ziegler. Fast multipoint evaluation of bivariate polynomials. In Susanne Albers and Tomasz Radzik, editors, *Algorithms – ESA 2004*, pages 544–555, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30140-0_49.

18 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer, Berlin, 1985. doi:10.1007/978-1-4612-1098-6.

19 Joris van der Hoeven and Grégoire Lecerf. Faster multi-point evaluation over any field. Working paper or preprint, November 2024. URL: https://hal.science/hal-04774026.

20 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. 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 3792–3835. SIAM, 2024. doi:10.1137/1.9781611977912.134.

21 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013. doi:10.1017/CBO9781139856065.

Optimal Motion Planning for Two Square Robots in a Rectilinear Environment

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

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Benjamin Holmgren
Department of Computer Science, Duke University, Durham, NC, USA

Alex Steiger
Department of Computer Science, Duke University, Durham, NC, USA

Martijn Struijs
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

Let $\mathcal{W} \subset \mathbb{R}^2$ be a rectilinear polygonal environment (that is, a rectilinear polygon potentially with holes) with a total of n vertices, and let A, B be two robots, each modeled as an axis-aligned unit square, that can move rectilinearly inside $\mathcal{W}$. The goal is to compute an optimal *collision-free motion plan* $\boldsymbol{\pi}$ for A and B between a given pair of *source* and *target* configurations. We study two variants of this problem and obtain the following results.

- Min-Sum: Here the goal is to compute a motion plan that minimizes the sum of the lengths of the paths of the robots. We present an $O(n^4 \log n)$-time algorithm for computing an optimal solution to the min-sum problem. This is the first polynomial-time algorithm to compute an optimal, collision-free motion of two robots amid obstacles in a planar polygonal environment.
- Min-Makespan: Here the robots can move with at most unit speed, and the goal is to compute a motion plan that minimizes the maximum time taken by a robot to reach its target location. We prove that the min-makespan variant is NP-hard.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Computational geometry, motion planning, multiple robots, rectilinear paths

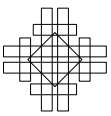
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.5

Funding *Pankaj K. Agarwal*: Supported by NSF grants CCF-20-07556, CCF-22-23870, and IIS-24-02823, and by the Binational Science Foundation Grant 2022131.
Mark de Berg: Supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.
Benjamin Holmgren: Supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0024386.

1 Introduction

A wide range of applications have led to extensive work on designing efficient algorithms for computing high-quality motion plans in a structured or an unstructured environment for a system of robots; see e.g. [7, 45] for recent surveys. Several criteria have been proposed to measure the quality of a motion plan, including the total length of the robot paths, the makespan (i.e., the maximum time taken by a robot to complete its motion), and the utility of the plan (i.e., how well it performs the underlying task). Already for two simple robots, such as unit squares or disks, translating in a planar polygonal environment, little is known

© Pankaj K. Agarwal, Mark de Berg, Benjamin Holmgren, Alex Steiger, and Martijn Struijs;
licensed under Creative Commons License CC-BY 4.0



about computing an optimal motion plan. Although polynomial-time algorithms are known for computing a collision-free motion plan of two simple robots [8,35], no polynomial-time algorithm is known for computing a plan such that the sum (or the maximum) of the path lengths of the two robots is minimized, nor is the problem known to be NP-hard. In this paper we study the problem of computing an optimal motion plan for two simple robots, each modeled as a unit square.

Problem statement. Let $\square := \{x \in \mathbb{R}^2 \mid ||x||_\infty \leqslant 1/2\}$ denote the axis-aligned square of unit side length – referred to as a *unit square* for short – centered at the origin. Let A and B be two robots, each modeled as a unit square, that can translate inside the same closed rectilinear polygonal environment $\mathcal{W} \subset \mathbb{R}^2$. In other words, the shared *workspace* $\mathcal{W}$ is a rectilinear polygon, possibly with holes. Let n be the number of vertices of $\mathcal{W}$. A placement of A (and, similarly, of B) is represented by the position of its center in the workspace $\mathcal{W}$. For such a placement to be free of collision with the boundary $\partial\mathcal{W}$ of $\mathcal{W}$, the representing point should be at ℓ_∞-distance at least $1/2$ from $\partial\mathcal{W}$. (Note that the robot is allowed to touch an obstacle, since we define $\mathcal{F}$ to be a closed set.) We let $\mathcal{F} \subset \mathcal{W}$ denote the *free space* of a single robot, which is the subset of $\mathcal{W}$ consisting of all collision-free placements. A (joint) *configuration* $\boldsymbol{p}$ of A and B is represented as a pair $\boldsymbol{p} = (p_A, p_B) \in \mathcal{W} \times \mathcal{W}$, where p_A and p_B are the placements of A and B, respectively. The *configuration space*, called *C-space* for short, is the set of all configurations, and is thus represented as $\mathcal{W} \times \mathcal{W} \subset \mathbb{R}^4$. A configuration $\boldsymbol{p} = (p_A, p_B) \in \mathbb{R}^4$ is called *free* if $p_A, p_B \in \mathcal{F}$ and $||p_A - p_B||_\infty \geqslant 1$. Let $\boldsymbol{F} \subseteq \mathcal{F} \times \mathcal{F}$ denote the *four-dimensional free space*, comprising the set of all free configurations.

Let $\boldsymbol{s} = (s_A, s_B)$ be a given *source configuration* and let $\boldsymbol{t} = (t_A, t_B)$ be a given *target configuration*. An $(\boldsymbol{s}, \boldsymbol{t})$*-plan* is a continuous function $\boldsymbol{\pi} : [0, T] \to \mathcal{W} \times \mathcal{W}$, for some $T \in \mathbb{R}_{\geqslant 0}$, with $\boldsymbol{\pi}(0) = \boldsymbol{s}$ and $\boldsymbol{\pi}(T) = \boldsymbol{t}$. The image of $\boldsymbol{\pi}$ is a continuous curve in the C-space, referred to as an $(\boldsymbol{s}, \boldsymbol{t})$*-path*. With a slight abuse of notation, we use $\boldsymbol{\pi}$ to denote its image as well. If $\boldsymbol{\pi} \subset \boldsymbol{F}$ we say that $\boldsymbol{\pi}$ is *feasible*, and if there exists a feasible $(\boldsymbol{s}, \boldsymbol{t})$-plan we say that the pair $(\boldsymbol{s}, \boldsymbol{t})$ is *reachable*. For a plan $\boldsymbol{\pi} : [0, T] \to \mathcal{W} \times \mathcal{W}$, let $\pi_A : [0, T] \to \mathcal{W}$ and $\pi_B : [0, T] \to \mathcal{W}$ be the projections of $\boldsymbol{\pi}$ onto the two-dimensional plane spanned by the first two coordinates and the last two coordinates, respectively. The functions π_A and π_B specify the motions of A and B that $\boldsymbol{\pi}$ induces, that is, $\boldsymbol{\pi}(\lambda) = (\pi_A(\lambda), \pi_B(\lambda))$ for all $\lambda \in [0, T]$. Again, with a slight abuse of notation, we also use π_A and π_B to denote the paths followed by A and B, respectively. We define two versions of optimal motion planning.

- Min-Sum. For a path γ in $\mathcal{W}$, let $\|\gamma\|$ denote its ℓ_1-length. We define $\|\boldsymbol{\pi}\|$, the *cost* of an $(\boldsymbol{s}, \boldsymbol{t})$-plan $\boldsymbol{\pi}$, by $\|\boldsymbol{\pi}\| := \|\pi_A\| + \|\pi_B\|$, that is, $\|\boldsymbol{\pi}\|$ is the sum of the ℓ_1-lengths of the paths of the two robots. The problem is to decide whether a given pair $\boldsymbol{s}, \boldsymbol{t} \in \boldsymbol{F}$ is reachable, and, if so, compute a minimum-cost feasible $(\boldsymbol{s}, \boldsymbol{t})$-plan. As explained later, we can restrict our attention to rectilinear paths because we work with the ℓ_1-metric inside a rectilinear environment.
- Min-Makespan. We define the *makespan* of an $(\boldsymbol{s}, \boldsymbol{t})$-plan $\boldsymbol{\pi} : [0, T] \to \mathcal{W} \times \mathcal{W}$, denoted by $¢(\boldsymbol{\pi})$, to be T. The problem is to decide for a given pair $\boldsymbol{s}, \boldsymbol{t} \in \boldsymbol{F}$ if $(\boldsymbol{s}, \boldsymbol{t})$ is reachable and, if so, compute a feasible $(\boldsymbol{s}, \boldsymbol{t})$-plan that minimizes the makespan under the condition that the maximum speed of each robot is at most 1. Note that we do not require the $(\boldsymbol{s}, \boldsymbol{t})$-plan to be C^1 continuous, so the speed of A or B may instantaneously change from 0 to 1, and the robots may take sharp turns.

Both variants of the optimal motion planning problem are interesting in their own right. The first minimizes the total work done by the robots, while the second minimizes the total time taken until both robots have reached their destination.

Related work. It is beyond the scope of this paper to review the known results on motion-planning algorithms; for an overview of key results, we refer the reader to recent books and surveys on the topic [7, 22, 23, 30, 31, 33]. We mention here only a small sample of results – ones that are most closely related to the problem at hand.

Computing an optimal motion plan for a single translating square robot, or more generally for a single convex polygonal translating robot with a constant number of vertices, is equivalent – through C-space formulation – to computing a shortest path for a point robot amid polygonal obstacles with $O(n)$ vertices, and it can be solved in $O(n \log n)$ time [16,25,44]. If the robot is allowed to rotate, the problem becomes much harder and it is NP-hard in some cases [1, 9, 10, 34]. Computing the shortest path for a point robot amid polyhedral obstacles in $\mathbb{R}^3$ is NP-hard [15], and fast $(1+\varepsilon)$-approximation algorithms are known [5, 17].

Computing a feasible (not necessarily optimal) plan for a team of translating unit square robots in a polygonal environment is PSPACE-hard [38]; see [13, 14, 24, 26, 41, 46] for related intractability results. Notwithstanding a rich literature [18, 33, 36, 42, 43] on multi-robot motion planning, little is known about algorithms for computing plans with provable quality guarantees. Even in the absence of obstacles, computing the min-sum motion plan for two unit squares/disks is non-trivial under the ℓ_2-metric [21, 29]. Variants with more than two robots in a setting without obstacles have been studied as well [19, 20, 27]. Recently, Agarwal et al. [4] presented the first polynomial-time approximation algorithm for the min-sum motion-planning problem for two squares that computes an $(1+\varepsilon)$-approximate solution in $\varepsilon^{-O(1)} n^2 \log n$ time (under both the ℓ_1- and ℓ_2-metrics). They leave it as an open problem whether an optimal plan in this case can be computed in polynomial time. Approximation algorithms have been proposed for min-sum multi-robot motion planning under the assumption that the source and target positions of robots are *separated* from each other and from the obstacles [3, 37, 40]; see also [2, 11]. There is also work on the *unlabeled* version of the problem, where each robot can end up at any of the (collective) target positions, as long as all of the target positions are occupied by robots at the end of the motion. Finally, we remark that sampling-based approaches have been proposed for optimal multi-robot motion planning [18, 28, 36, 39]; and that there is extensive work on the discrete version of the problem, where robots are moving on graphs; see e.g., [7, 42] for recent surveys.

Our contributions. Our main result is that Min-Sum is in P, while Min-Makespan is NP-hard. This is stated in the two following theorems.

▶ **Theorem 1.** *Let $\mathcal{W}$ be a closed rectilinear polygonal environment with n vertices, let A, B be two axis-parallel unit-square robots translating inside $\mathcal{W}$, and let $\boldsymbol{s}, \boldsymbol{t}$ be source and target configurations of A, B. We can compute an optimal min-sum motion plan from $\boldsymbol{s}$ to $\boldsymbol{t}$ under the ℓ_1-metric, or determine that no feasible motion exists, in $O(n^4 \log n)$ time.*

▶ **Theorem 2.** *Let $\mathcal{W}$ be a closed rectilinear polygonal environment with n vertices, let A, B be two axis-parallel unit-square robots translating inside $\mathcal{W}$, let $\boldsymbol{s}, \boldsymbol{t}$ be source and target configurations of A, B, and let $T_{\max}$ be a given maximum time. It is NP-hard to determine whether there is a feasible $(\boldsymbol{s}, \boldsymbol{t})$-plan $\boldsymbol{\pi}^*$ such that the maximum speed of each robot is at most 1 and the makespan of $\boldsymbol{\pi}^*$ is at most $T_{\max}$.*

We prove Theorem 2 by a reduction from the Partition problem, which is to decide whether a given set Y of integers admits a partition into subsets Y_A, Y_B such that $\sum_{y_i \in Y_A} p = \sum_{y_i \in Y_B} q$. The idea of the reduction is to build a workspace $\mathcal{W}$ that consists of gadgets $\mathcal{W}_1, \ldots, \mathcal{W}_m$,

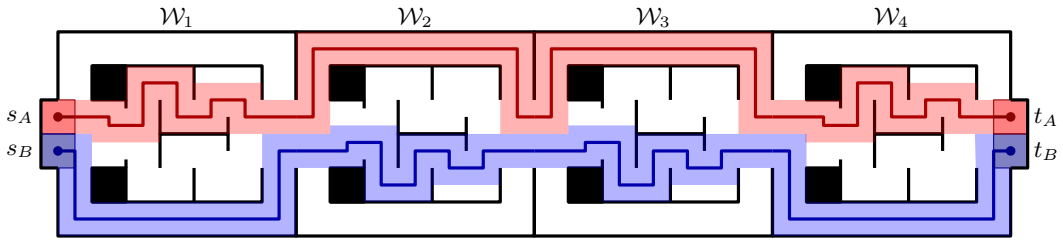


Figure 1 The workspace created for a set $Y = \{y_1, ..., y_4\}$. The paths shown for the two robots correspond to the partition $Y_A = \{y_1, y_4\}$ and $Y_B = \{y_2, y_3\}$. The black areas are obstacles, and the red/blue regions are the areas swept by the two robots. The detour made by a robot in a gadget $\mathcal{W}_i$ when it goes via the top (for A) or bottom (for B) of the gadget, is equal to y_i. Hence, a solution with certain makespan can only be achieved if there is an equal partition of Y.

each corresponding to an element of Y and to choose a parameter $\mu \geqslant 0$, so that both robots must pass through every gadget and there is a plan with makespan at most μ if and only if there is a valid partition of Y. See Figure 1.

Our main technical contribution is a polynomial-time exact algorithm for Min-Sum, as stated in Theorem 1. The crucial step toward designing this algorithm is to prove the existence of an optimal plan that is a *canonical-grid plan*, in which the path followed by each robot lies on an $O(n) \times O(n)$ non-uniform predefined grid; see Theorem 4.

For a single robot translating in a rectilinear environment in $\mathbb{R}^2$, it is well known that there is an optimal path, under the ℓ_1-metric, on the grid defined by the horizontal and vertical lines through the source and target positions and the lines containing the edges of the free space $\mathcal{F}$ [32]. For a pair of robots, the existence of an optimal plan on a suitably defined grid seems intuitive as well. Indeed, consider an optimal plan $\boldsymbol{\pi}^*(\boldsymbol{s}, \boldsymbol{t})$. As argued in Section 2, we can assume each robot follows a rectilinear path (in $\mathcal{F}$) under the plan $\boldsymbol{\pi}^*$. Imagine trying to push a horizontal (resp. vertical) segment of π_A^* to a horizontal (resp. vertical) line of the (nonuniform) grid formed by the lines supporting the edges of $\mathcal{F}$ or the lines at distance 1 from $\partial\mathcal{F}$. It may happen that we fail to push the segment onto the grid because robot B is blocking it. The hope is that we can continue to push the segment of π_A and push the appropriate segment of π_B in the same direction – essentially A is pushing B out of the way – until B hits $\partial\mathcal{W}$– without increasing the total cost. Thus, we have pushed a segment of π_B onto a line containing an edge of $\mathcal{F}$ and the segment of π_A onto a line at distance 1 from that line. Making this idea work is highly nontrivial, though. For instance, we must choose the direction into which we push a specific segment of π_A in such a way that $\|\pi_A\|$ does not increase, but this may force us to push π_B in such a way that $\|\pi_B\|$ increases. This is further complicated by the fact that B may be on different sides of a segment of π_A at different moments in time. Finally, a major complication is that pushing (a segment of) one robot onto a line at distance i from an edge of $\mathcal{F}$, so that its length does not increase, may cause the other robot to end up on a line at distance $i + 1$ from this edge, and this may cause a cascading effect so that the final grid will be too large.

To overcome these issues, we combine local pushing operations with global re-routing operations in such a way that the transformed paths lie on lines at distance 0, 1, or 2 from the edges of $\mathcal{F}$. These operations may increase the length of one path, but we prove that whenever this happens, the length of the other path decreases by at least the same amount. Thus we obtain an optimal plan that lies on a grid of size $O(n) \times O(n)$ and is still optimal.

After having established the existence of an optimal canonical-grid plan, we show that we can construct a 4-dimensional weighted grid graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{V}$, of size $O(n^4)$, in $\boldsymbol{F}$ such that a shortest $(\boldsymbol{s}, \boldsymbol{t})$-path in $\mathcal{G}$ corresponds to a min-sum $(\boldsymbol{s}, \boldsymbol{t})$-plan.

2 A polynomial-time algorithm for Min-Sum

This section describes a polynomial-time algorithm for Min-Sum. We first define a *canonical-grid plan* in Section 2.1 – an $(\boldsymbol{s},\boldsymbol{t})$-plan where both robots move on a predefined $O(n)\times O(n)$ nonuniform grid – and claim that for any reachable pair $\boldsymbol{s},\boldsymbol{t}\in\boldsymbol{F}$, there exists an optimal $(\boldsymbol{s},\boldsymbol{t})$-plan that is a canonical-grid plan. In Section 2.2 we give a high-level overview of the proof of this claim; a more detailed sketch of the proof is deferred to Section 3. Finally, we describe our algorithm in Section 2.3.

2.1 Canonical-grid plans

Rectilinear, decoupled plans. Since $\mathcal{W}$ is a rectilinear environment and A, B are squares, it is easily seen that $\boldsymbol{F}$ is a polyhedral region. Hence, if there is a feasible $(\boldsymbol{s},\boldsymbol{t})$-plan, then there is a piecewise-linear optimal $(\boldsymbol{s},\boldsymbol{t})$-plan, that is, an optimal plan whose image is a polygonal chain in $\boldsymbol{F}$ [4]. We thus focus on piecewise-linear plans. We refer to the vertices of such a plan as *breakpoints*. For a path $\boldsymbol{\pi}=(\pi_A,\pi_B)$, the breakpoints of π_A and π_B are the projections of the breakpoints of $\boldsymbol{\pi}$.

An $(\boldsymbol{s},\boldsymbol{t})$-plan $\boldsymbol{\pi}$ is *decoupled* if only one robot moves at any given time; the other robot is then *parked* somewhere in the free space. A decoupled plan can be represented as a sequence of *moves*, with each move specifying the parking location of one robot and the motion of the other robot. By definition of breakpoints, each parking spot of A (resp. B) is a breakpoint of π_A (resp. π_B). Agarwal et al. [4] proved that for any pair $\boldsymbol{s},\boldsymbol{t}$ of reachable configurations, there is an optimal $(\boldsymbol{s},\boldsymbol{t})$-plan that is decoupled. Since only one robot moves at a time, the parked robot can be considered as a rectilinear obstacle during the move, and (since we measure path length in the ℓ_1-metric) we can ensure that each robot follows a rectilinear path in $\mathcal{F}$ in each move. We refer to such a plan as a *rectilinear* plan. The breakpoints of π_A and π_B in a rectilinear plan $\boldsymbol{\pi}=(\pi_A,\pi_B)$ are the parking spots of A and B and the points at which they switch between horizontal and vertical segments. We conclude the following.

▶ **Proposition 3.** *For any reachable pair of configurations $\boldsymbol{s},\boldsymbol{t}\in\boldsymbol{F}$, there is an optimal $(\boldsymbol{s},\boldsymbol{t})$-plan that is rectilinear and decoupled.*

The canonical grid. For an integer $i\geqslant 0$, we define an *i-line* to be a horizontal or vertical line ℓ that lies at distance i from an edge of $\partial\mathcal{F}$ parallel to ℓ, or that lies at distance i from one of the points s_A, s_B, t_A, t_B. Note that 0-lines support an edge of $\partial\mathcal{F}$ or pass through one of the points s_A, s_B, t_A, t_B. Let L_i denote the set of all i-lines, and set $L_{\leqslant i}:=\cup_{j=0}^{i}L_j$. We will be mostly interested in $L_{\leqslant 2}$. The *arrangement* [6] of $L_{\leqslant 2}$, denoted by $\mathcal{A}(L_{\leqslant 2})$, is the subdivision of $\mathbb{R}^2$ induced by $L_{\leqslant 2}$. Since each edge of $\mathcal{F}$ is contained in a 0-line, each face of $\mathcal{A}(L_{\leqslant 2})$ is a rectangle that is either contained in $\mathcal{F}$ or disjoint from $\mathcal{F}$. The vertices and edges of $\mathcal{A}(L_{\leqslant 2})$ form a planar grid graph. Let $G=(V,E)$ denote the subgraph of this graph that lies in $\mathcal{F}$, that is, V and E are the sets of vertices and edges, respectively, of $\mathcal{A}(L_{\leqslant 2})$ that lie in $\mathcal{F}$. Note that $|V|=O(n^2)$ and $|E|=O(n^2)$. We refer to G as a *grid*, to a vertex of V as a *grid point*, to an edge of E as a *grid edge*, and to a line of $L_{\leqslant 2}$ as a *grid line*. We call a rectilinear, decoupled $(\boldsymbol{s},\boldsymbol{t})$-plan $\boldsymbol{\pi}=(\pi_A,\pi_B)$ a *canonical-grid plan* if each breakpoint of π_A and π_B is a grid point. Thus each segment of the path is the union of a number of consecutive collinear grid edges. By definition, each parking spot of A or B in a canonical-grid plan is a grid point. Our main technical result is the following theorem.

▶ **Theorem 4.** *Let $\mathcal{W}$ be a closed rectilinear polygonal environment with n vertices, let A, B be two axis-parallel unit-square robots translating inside $\mathcal{W}$. For a reachable pair of configurations $(\boldsymbol{s},\boldsymbol{t})$, there is an optimal $(\boldsymbol{s},\boldsymbol{t})$-plan that is a canonical-grid plan.*

2.2 Converting an optimal plan to a canonical-grid plan

We give an overview of how we convert an optimal $(\boldsymbol{s}, \boldsymbol{t})$-plan $\boldsymbol{\pi}$ to an optimal $(\boldsymbol{s}, \boldsymbol{t})$-plan that is a canonical-grid plan. By Proposition 3, we can assume $\boldsymbol{\pi}$ is rectilinear and decoupled.

Alternating plans. We assume that π_A alternates between vertical and horizontal segments; hence, each breakpoint of π_A is incident on a horizontal and a vertical segment and is thus a *vertex* of π_A. This *alternation property* for π_A can be ensured by adding zero-length segments at breakpoints where necessary. The alternation property, which we also assume for π_B, implies that all parking spots of A and B lie on the vertices of their respective paths.

We convert $\boldsymbol{\pi}$ into a canonical-grid plan in two phases: the first phase modifies the plan $\boldsymbol{\pi}$ such that all horizontal segments of π_A and π_B lie on grid lines, and the second phase does the same for the vertical segments. Here we describe the first phase; the second phase proceeds analogously, with the roles of the x and y-axis exchanged.

The first phase. We define a *bad horizontal segment* to be a horizontal segment on π_A or π_B that does not lie on a grid line. As long as $\boldsymbol{\pi}$ contains a bad horizontal segment e, we modify $\boldsymbol{\pi}$ into a new plan $\boldsymbol{\pi}^* = (\pi_A^*, \pi_B^*)$ so that the following conditions are satisfied:

- (P1) Feasibility: The new plan $\boldsymbol{\pi}^*$ is feasible.
- (P2) Optimality: The cost of the plan does not increase, that is, $\|\boldsymbol{\pi}^*\| \leqslant \|\boldsymbol{\pi}\|$.
- (P3) Progress: The plan $\boldsymbol{\pi}^*$ has at least one fewer bad horizontal segment than $\boldsymbol{\pi}$.
- (P4) Vertical alignment: Any vertical segment of π_A^* is collinear with a vertical segment of π_A or π_B, or it is contained in a vertical grid line; and the same holds for any vertical segment of π_B^*.
- (P5) Alternation: $\boldsymbol{\pi}^*$ is an alternating, rectilinear, decoupled plan.

The Progress property guarantees that by applying the procedure finitely many times, starting with the optimal plan $\boldsymbol{\pi}$, we obtain a plan without bad horizontal segments. The Feasibility and Optimality properties imply that $\boldsymbol{\pi}^*$ is an optimal $(\boldsymbol{s}, \boldsymbol{t})$-plan. Furthermore, the Alternation property implies that $\boldsymbol{\pi}^*$ is also a rectilinear, decoupled, alternating plan, so we can indeed apply the procedure iteratively.

After eliminating all bad horizontal segments in the first phase, we eliminate all bad vertical segments in the second phase. The Vertical Alignment property, which in the second phase applies to horizontal segments, guarantees that the horizontal segments are not moved off the grid in the second phase. So, all segments of $\boldsymbol{\pi}^*$ lie on grid lines, and since $\boldsymbol{\pi}^*$ is alternating, its breakpoints therefore lie on grid points. Hence, at the end of the second phase, we obtain an optimal, canonical-grid plan.

Note: Our transformation of the original plan $\boldsymbol{\pi}$ into a $\boldsymbol{\pi}^*$ satisfying (P1)–(P5) is done via several intermediate steps. When describing these steps, we typically denote, with a slight abuse of notation, the current plan by $\boldsymbol{\pi}$ and the next plan by $\boldsymbol{\pi}^*$.

Corridors. To be able to push bad horizontal segments onto horizontal grid lines without creating a cascading effect, we classify the horizontal grid lines into two classes: *primary* grid lines, which are the horizontal 0- and 1-lines, and *secondary* grid lines, which are the horizontal 2-lines. Let $\textsc{hor}(\mathcal{F})$ be the subdivision of $\mathcal{F}$ induced by the primary grid lines; see Figure 2. Each face of $\textsc{hor}(\mathcal{F})$ is a rectangle because every vertex of $\mathcal{F}$ is contained in a horizontal 0-line. We refer to the faces of $\textsc{hor}(\mathcal{F})$ as *corridors*. Note that the interior of a corridor may be intersected by secondary grid lines.

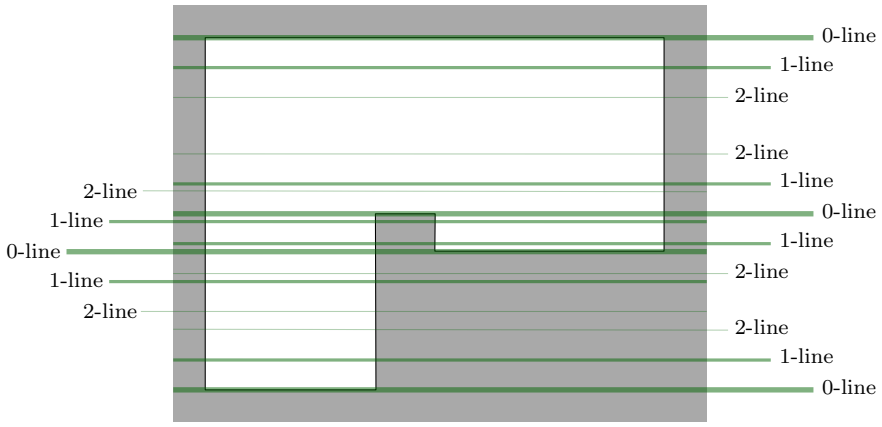


Figure 2 The 0-lines and 1-lines define $\textsc{hor}(\mathcal{F})$, the subdivision of $\mathcal{F}$ into corridors. To avoid cluttering the figure, the start and target configurations and the grid lines they define are omitted.

By definition, (the relative interior of) every bad horizontal segment lies in the interior of a corridor. At each step, we pick a bad segment e, say of π_A. We perform surgery on π_A and π_B so that e is *aligned* with one of the horizontal edges of the corridor R containing e, and so that the resulting plan $\boldsymbol{\pi}^*$ satisfies (P1)–(P5). The surgery and its proof of correctness are quite delicate because the topology of $\mathcal{F}$ in the neighborhood of R can be complex.

2.3 Computing an optimal canonical-grid plan

The configuration graph. Let $G = (V, E)$ be the canonical grid described in Section 2.1. We construct an edge-weighted *configuration graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with weight function $w : \mathcal{E} \to \mathbb{R}_{\geqslant 0}$ whose nodes correspond to free configurations $\boldsymbol{u} \in \boldsymbol{F}$. More precisely,

$$\mathcal{V} := \{\boldsymbol{u} = (u_A, u_B) \in V \times V : \|u_A - u_B\| \geqslant 1\}.$$

By construction, $\boldsymbol{s}, \boldsymbol{t} \in \mathcal{V}$. The set $\mathcal{E}$ of edges of the configuration graph is defined as

$$\mathcal{E} := \Big\{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{V} \times \mathcal{V} : (u_A = v_A \text{ and } (u_B, v_B) \in E) \text{ or } (u_B = v_B \text{ and } (u_A, v_A) \in E)\Big\}.$$

An edge $((p, u_B), (p, v_B)) \in \mathcal{E}$ corresponds a move in which robot A is parked at p while robot B moves from u_B to an adjacent grid point v_B. Similarly, an edge $((u_A, q), (v_A, q)) \in \mathcal{E}$ corresponds to a move in which A moves from a grid point u_A to an adjacent grid point v_A while B is parked at q. (We can prove that such moves are feasible.) The weight of an edge $(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{E}$ is defined to be $w(\boldsymbol{u}, \boldsymbol{v}) := \|\boldsymbol{uv}\|$, which is the ℓ_1-distance between $\boldsymbol{u}$ and $\boldsymbol{v}$ in $\mathbb{R}^4$.

The algorithm. We first compute $\mathcal{F}$ in $O(n \log n)$ time in a standard manner [12, Chapter 13]. Next, we compute $L_{\leqslant 2}$ in $O(n)$ time and then compute the grid graph G in $O(n^2 \log n)$ time using a sweep-line algorithm. After computing G, we compute the configuration graph $\mathcal{G}$ in $O(n^4)$ time. Finally, we compute a shortest path from $\boldsymbol{s} \in \mathcal{V}$ to $\boldsymbol{t} \in \mathcal{V}$ in $\mathcal{G}$ in $O(n^4 \log n)$ time using Dijkstra's algorithm. If we find that no path from $\boldsymbol{s}$ to $\boldsymbol{t}$ exists in $\mathcal{G}$, we report the instance to be infeasible, otherwise we return the $(\boldsymbol{s}, \boldsymbol{t})$-plan corresponding to the shortest path in $\mathcal{G}$. The total run time is $O(n^4 \log n)$. This completes the proof of Theorem 1.

3 Existence of an optimal canonical-grid plan

Let $\boldsymbol{\pi}$ be an optimal plan that contains a bad horizontal segment e, say of π_A. Let $R = I_x \times I_y$ where $I_x = [x_R^-, x_R^+]$ and $I_y = [y_R^-, y_R^+]$, be the corridor that contains e. This section describes how to modify $\boldsymbol{\pi}$ so that e becomes aligned with a grid line and the resulting plan satisfies (P1)–(P5). We begin by introducing some notation. For an axis-aligned rectangle $\rho = \delta_x \times \delta_y$, let $\mathsf{top}(\rho)$ and $\mathsf{bot}(\rho)$ be the top and bottom edge of ρ, respectively, and let $\mathsf{ht}(\rho) := |\delta_y|$ denote its height. For $0 \leqslant \mu_1 \leqslant \mu_2$, let $\boldsymbol{\pi}[\mu_1, \mu_2]$ denote the subplan of $\boldsymbol{\pi}$ (or its image) during the closed interval $[\mu_1, \mu_2]$, and let $\boldsymbol{\pi}(\mu_1, \mu_2)$ denote the subplan during the open interval (μ_1, μ_2). We define $\pi_A[\mu_1, \mu_2]$ and $\pi_A(\mu_1, \mu_2)$, and $\pi_B[\mu_1, \mu_2]$ and $\pi_B(\mu_1, \mu_2)$, in the same manner. For a horizontal segment g, let $y(g)$ be its y-coordinate, and for a point p let p_x and p_y denote its x- and y-coordinates.

Let $[\lambda', \lambda'']$ be the time interval associated with e; thus $e = \pi_A[\lambda', \lambda'']$. We now define two critical time values related to the bad segment e:

$$\begin{aligned} \lambda_1 &= \max\{\lambda < \lambda' \mid \pi_A(\lambda) \in \mathsf{top}(R) \cup \mathsf{bot}(R)\}, \text{ and} \\ \lambda_2 &= \min\{\lambda > \lambda'' \mid \pi_A(\lambda) \in \mathsf{top}(R) \cup \mathsf{bot}(R)\}. \end{aligned} \tag{1}$$

Thus, $\pi_A(\lambda_1)$ (resp. $\pi_A(\lambda_2)$) is the last (resp. first) point on π_A before (resp. after) e at which A lies on a horizontal edge of R. Note that $e \subset \pi_A(\lambda_1, \lambda_2) \subset R \setminus (\mathsf{top}(R) \cup \mathsf{bot}(R))$ and that $\pi_A(\lambda_1, \lambda_2)$ may touch a vertical edge of R. The goal of our surgery is to push $\pi_A[\lambda_1, \lambda_2]$ to $\mathsf{top}(R)$ or $\mathsf{bot}(R)$ to align it with a grid line. This may cause collisions with $\pi_B[\lambda_1, \lambda_2]$, so we have to proceed carefully. Sometimes we push part of $\pi_A[\lambda_1, \lambda_2]$ to $\mathsf{top}(R)$ and part to $\mathsf{bot}(R)$, and in one case we even modify the paths outside the interval $[\lambda_1, \lambda_2]$. In all cases, we resolve the collisions by pushing B out of the way, without increasing the overall length of the paths.

In the remainder of this section, we sketch the ideas underlying our pushing strategy. We first define a neighborhood of R, called the *influence region* of R and denoted by $\mathcal{I}(R)$, and prove a few key properties of $\mathcal{I}(R)$. Next, we define *unsafe* and *swap* time intervals, subintervals of $[\lambda_1, \lambda_2]$ during which the surgery on $\boldsymbol{\pi}$ requires more care, and we state the crucial structural properties of an optimal plan during these intervals. With these properties at our disposal, we describe the surgery and give some ideas about the correctness proof. Before proceeding, we state a simple but important property of $\mathcal{F}$ that we will use repeatedly.

▶ **Observation 5.** *For any axis-aligned line ℓ, the length of each connected component of $\ell \setminus \mathcal{F}$ is more than* 1*. Consequently, for any axis-aligned segment pq of length at most* 1 *such that $p, q \in \mathcal{F}$, we have $pq \subseteq \mathcal{F}$.*

The influence region. Let $\square$ be the unit square as defined in Section 1. Let $R_\square := R \oplus 2\square$. Thus, $R_\square = I_x^\oplus \times I_y^\oplus$ where $I_x^\oplus := [x_R^- - 1, x_R^+ + 1]$ and $I_y^\oplus := [y_R^- - 1, y_R^+ + 1]$. For any pair $p, q \in \mathbb{R}^2$, if $p \in R$ and $(p + \square) \cap (q + \square) \neq \emptyset$, then $q \in R_\square$. Next, we define two different extensions of R that play a role in our analysis; see Figure 3. Let $R_\leftrightarrow := I_x^\oplus \times I_y$, that is, $R_\leftrightarrow$ is obtained by extending R by unit length to the left and to the right. Similarly, $R_\updownarrow := I_x \times I_y^\oplus$ is the extension of R by unit length in the upward and downward directions. Observe that $R_\square \setminus (R_\updownarrow \cup R_\leftrightarrow)$ consists of four semi-open unit squares, which we refer to as the *corner squares* of $R_\square$. Finally, we define $Z = I_x^\oplus \times [y_R^+ - 1, y_R^- + 1]$, and $Z^- = I_x^\oplus \times [y_R^- - 1, y_R^+ - 1)$, and $Z^+ = I_x^\oplus \times (y_R^- + 1, y_R^+ + 1]$. Note that $\mathsf{ht}(Z) = 1 + \delta$ and $\mathsf{ht}(Z^-) = \mathsf{ht}(Z^+) = 1 - \delta$. The closures of Z^-, Z^+ are translates of $R_\leftrightarrow$ by distance 1 in the y-direction, and Z^-, Z, Z^+ is a

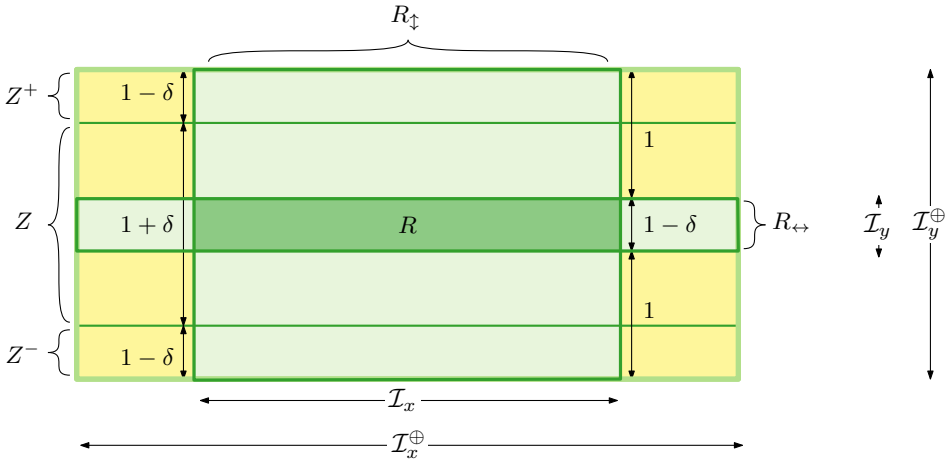


Figure 3 An overview of the various regions inside $R_{\square}$ (not to scale). The regions Z^-, Z, Z^+ partition $R_{\square}$. Corner squares are yellow.

partition of $R_{\square}$. Since no primary grid line of $L_{\leqslant 1}$ intersects $\mathrm{int}(R)$, the regions Z^+ and Z^- do not contain any vertices of $\mathcal{F}$. By Observation 5, the connected components of $Z^+ \cap \mathcal{F}$ and $Z^- \cap \mathcal{F}$ are therefore rectangles, and $R_{\updownarrow} \cap \mathcal{F}$ is an x-monotone rectilinear polygon.

We define the *influence region* of R to be $\mathcal{I}(R) := \mathcal{F} \cap R_{\square}$. For $\lambda \in [\lambda_1, \lambda_2]$, if $\pi_B(\lambda) \notin \mathcal{I}(R)$ then $\pi_A(\lambda)$ can be pushed to $\mathsf{top}(R)$ or $\mathsf{bot}(R)$ without colliding with B, so we have to worry about subintervals of $[\lambda_1, \lambda_2]$ during which π_B lies in $\mathcal{I}(R)$. As illustrated in Figure 4, the influence region $\mathcal{I}(R)$ may consist of several connected components: a *giant* component $\gamma(R)$ that contains R, and several *tiny* components, each of which lies inside a corner square of $R_{\square}$. Tiny components are xy-monotone by Observation 5. Note that the giant component may intersect the corner squares. In particular, let ρ be a corner square that lies, say, to the left of $R_{\updownarrow}$. By Observation 5, there is at most one connected component ϕ of $\rho \cap \mathcal{F}$ adjacent to the right edge of ρ. If such a component ϕ exists, then $\phi = \rho \cap \gamma(R)$ and (like the tiny components) ϕ is xy-monotone. Other components of $\rho \cap \mathcal{F}$ are tiny components of $\mathcal{I}(R)$. Let ρ, ρ' be two corner squares on the same side of $R_{\updownarrow}$. If both $\rho \cap \mathcal{F}$ and $\rho' \cap \mathcal{F}$ have non-empty connected components inside $\gamma(R)$, denoted by ϕ and ϕ' respectively, then we refer to pairs of points in $\phi \times \phi'$ as *blocked pairs*. The next lemma states the significance of blocked pairs.

▶ **Lemma 6.** *Let p, q be two points lying in the same connected component of $\mathcal{I}(R)$. Then the shortest paths from p to q in $\mathcal{F}$ are x-monotone if and only if p, q is not a blocked pair.*

Thus, if $\pi_B[\lambda_1, \lambda_2]$ contains a blocked pair, then the surgery of $\boldsymbol{\pi}$ and the proof of its correctness become significantly harder.

Unsafe and swap intervals. We next define critical subintervals of the time interval $[\lambda_1, \lambda_2]$ during which the surgery of $\boldsymbol{\pi}$ requires more care, and then prove some structural properties of $\boldsymbol{\pi}$ during these intervals. Again, we begin with a few definitions. A configuration $\boldsymbol{p} = (p_A, p_B)$ is called *x-separated* if $|(p_A)_x - (p_B)_x| \geqslant 1$ and *y-separated* if $|(p_A)_y - (p_B)_y| \geqslant 1$. We say that two x-separated configurations $\boldsymbol{p}, \boldsymbol{q}$ have the *same x-order* if $\mathrm{sign}((p_A)_x - (p_B)_x) = \mathrm{sign}((q_A)_x - (q_B)_x)$; otherwise the x-order is *swapped*. An interval $[\nu_1, \nu_2]$ is called *x-separated* if $\boldsymbol{\pi}(\nu_1)$ and $\boldsymbol{\pi}(\nu_2)$ are both x-separated. Note that this definition does not require that $\boldsymbol{\pi}(\nu)$ is x-separated for all $\nu \in [\nu_1, \nu_2]$.

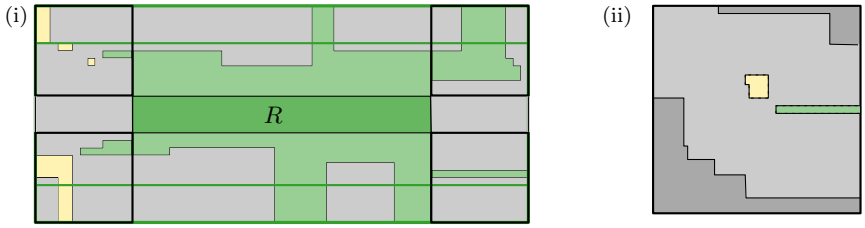


Figure 4 (i) An example influence region within $R_\square$ (not to scale), with the giant component $\gamma(R)$ drawn in green, and tiny components in the corner squares in yellow. (ii) Example of tiny components in a corner square, with obstacles in dark gray and the forbidden space in light gray.

▶ **Definition 7.** *An interval* $[\nu_1, \nu_2] \subseteq [\lambda_1, \lambda_2]$ *is* unsafe *if*

(i) $[\nu_1, \nu_2]$ *is a maximal interval such that* $\pi_B(\nu) \in \mathcal{I}(R)$ *and* $\boldsymbol{\pi}(\nu)$ *is* y*-separated for all* $\nu \in [\nu_1, \nu_2]$*; and*

(ii) *there is a time* $\nu \in [\nu_1, \nu_2]$ *such that* $\boldsymbol{\pi}(\nu)$ *is not* x*-separated.*

For any unsafe interval $[\nu_1, \nu_2]$, either robot B lies above A throughout the interval – that is, $\pi_B(\nu)_y \geqslant \pi_A(\nu)_y$ for all $\nu \in [\nu_1, \nu_2]$ – or B lies below A throughout the interval. Indeed, the robots cannot change their y-order during an unsafe interval, because they must remain y-separated. If an endpoint $\nu_i \in \{\nu_1, \nu_2\}$ is not x-separated then either $\nu_i = \lambda_i$ or $\pi_B(\nu_i) \in \partial\mathcal{I}(R)$. An important type of unsafe interval is the so-called swap interval, as defined next.

▶ **Definition 8.** *A* swap interval *is an* x*-separated unsafe interval such that the* x*-orders at* $\boldsymbol{\pi}(\nu_1)$ *and* $\boldsymbol{\pi}(\nu_2)$ *are swapped.*

Roughly speaking, the roles of unsafe and swap intervals are as follows. If a time $\lambda \in [\lambda_1, \lambda_2]$ does not lie in an unsafe interval, translating $\pi_A(\lambda)$ vertically within R does not cause a conflict with $\pi_B(\lambda)$ because $\boldsymbol{\pi}(\lambda)$ is x-separated or $\pi_B(\lambda) \notin \mathcal{I}(R)$. If an unsafe interval is not x-separated, we show that we can modify π_B to avoid conflicts after translating $\pi_A(\lambda)$ vertically without increasing the length of π_B. We prove the following crucial property of x-separated unsafe intervals using Lemma 6. Suppose $\lambda, \mu \in [\lambda_1, \lambda_2]$ are the endpoints of x-separated unsafe intervals, not necessarily endpoints of the same unsafe interval, such that $\lambda < \mu$ and $\boldsymbol{\pi}(\lambda), \boldsymbol{\pi}(\mu)$ have the same x-order. If $\pi_B(\lambda)$, $\pi_B(\mu)$ lie in the same component of $\mathcal{I}(R)$ and are not a blocked pair, then we can obtain another feasible plan $\boldsymbol{\pi}^*[\lambda, \mu]$ by re-parametrizing $\boldsymbol{\pi}[\lambda, \mu]$ such that the configuration $\boldsymbol{\pi}^*(\nu)$ is x-separated for all $\nu \in [\lambda, \mu]$. Thus, we can eliminate all such unsafe intervals during $[\lambda, \mu]$ and push $\pi_A[\lambda, \mu]$ vertically to $\mathsf{top}(R) \cup \mathsf{bot}(R)$ as above.

Hence, the challenging case is to perform surgery when $[\lambda_1, \lambda_2]$ contains a swap interval. We keep the situation under control using Lemma 9, which bounds the number of swap intervals and which relies on the above property of x-separated unsafe intervals. The proof of this lemma is involved and needs a sequence of intermediate lemmas; see the full version.

We say that a modification $\boldsymbol{\pi}^*$ of $\boldsymbol{\pi}$ is *compliant* if the Feasibility (P1), Optimality (P2), Vertical alignment (P4) and Alternation (P5) properties hold, and all bad segments of $\boldsymbol{\pi}^*$ are either present in $\boldsymbol{\pi}$ or are a segment of $\pi_A^*[\lambda_1, \lambda_2]$.

▶ **Lemma 9.** *Let* $\boldsymbol{\pi}$ *be an optimal* $(\boldsymbol{s}, \boldsymbol{t})$*-plan, let* e *be a bad horizontal segment on* π_A*, and let* $[\lambda_1, \lambda_2]$ *be as defined in (1). Then there is a compliant modification* $\boldsymbol{\pi}^*$ *of* $\boldsymbol{\pi}$ *such that any unsafe* x*-separated interval is a swap interval with the following properties.*

(i) *The plan* $\boldsymbol{\pi}^*[\lambda_1, \lambda_2]$ *has at most one swap interval with* B *above* A *and at most one swap interval with* B *below* A*.*

(ii) *If* $\boldsymbol{\pi}^*[\lambda_1, \lambda_2]$ *has two swap intervals* $[\mu_0, \mu_1]$ *and* $[\mu_2, \mu_3]$*, then* $\mathcal{I}(R)$ *is connected,* $\pi_B^*[\lambda_1, \lambda_2]$ *contains a blocked pair, and* $\boldsymbol{\pi}_B^*[\mu_0, \mu_3]$ *leaves* $\mathcal{I}(R)$*.*

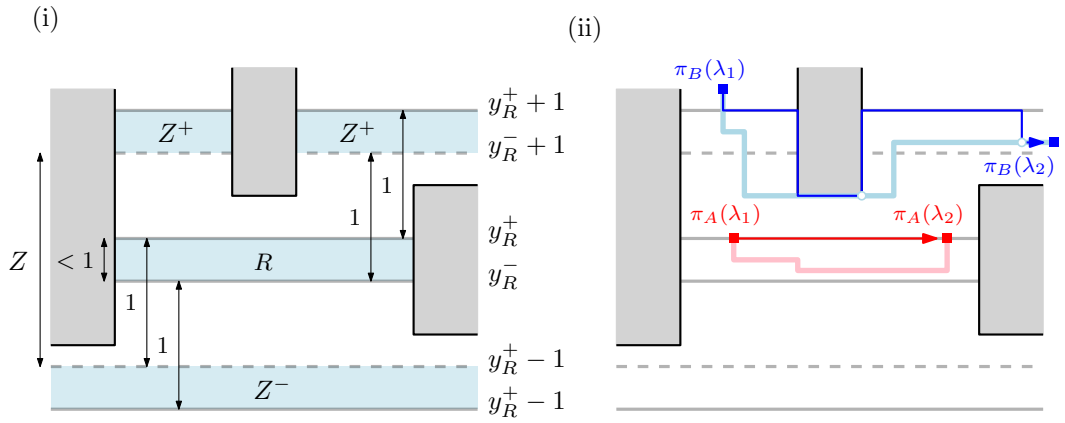


Figure 5 (i) Example of a corridor R, the surrounding lines and areas within $R + 2\square$. (ii) The modification in Case I. The original paths are shown in pink and light blue. The light-blue disks mark the endpoints of a swap interval on π_B.

We now describe the surgery we perform on π_A and π_B to align a given bad horizontal segment e of π_A with a grid line. The surgery consists of a sequence of one or more PUSH operations, described below, which push a part of π_A or π_B vertically onto a grid line. Let $\boldsymbol{\pi}^*$ denote the (modified) plan after the surgery.

The Push procedure. A PUSH operation takes three parameters: a robot $X \in \{A, B\}$, a time interval $I = [\mu_1, \mu_2]$, and a y-coordinate y^* of a horizontal grid line – a line from the set $L_{\leqslant 2}$ – and it pushes the subpath $\pi_X[I]$ onto the grid line $\ell : y = y^*$. More formally, the new plan $\boldsymbol{\pi}^*_X$ resulting from the operation $\textsc{Push}(X, I, y^*)$ is obtained by setting

$$\pi^*_X(\mu)_x := \pi_X(\mu)_x \text{ for all } \mu, \qquad \text{and} \qquad \pi^*_X(\mu)_y := \begin{cases} y^* & \text{if } \mu \in I, \\ \pi_X(\mu)_y & \text{otherwise.} \end{cases}$$

Thus a PUSH operation only changes the y-coordinates on the subpath $\pi_X[I]$. The exact surgery – the values of the parameters X, I, y^* used in the PUSH operations that we perform – depends on the location of $\pi_A(\lambda_1)$, the set of unsafe and swap intervals during $[\lambda_1, \lambda_2]$, and the position of B during the swap intervals, as explained later.

A PUSH operation may have two side effects that we need to address. First, $\textsc{Push}(X, I, y^*)$ may introduce a discontinuity at μ_1 and/or μ_2, the endpoints of I. In other words, for $\mu \in \{\mu_1, \mu_2\}$ we may have $p^-(\mu) \neq p^+(\mu)$, where $p^-(\mu) := \lim_{\lambda \uparrow \mu} \pi^*_X(\lambda)$ and $p^+(\mu) := \lim_{\lambda \downarrow \mu} \pi^*_X(\lambda)$. This happens when $\pi_X(\mu) \neq \pi^*_X(\mu)$. We can resolve the discontinuity by adding the vertical segment $p^-(\mu)p^+(\mu)$, called a *ghost segment*, to π^*_X to make the path continuous, and by re-parametrizing the motion along the ghost segments so that the resulting motion plan is feasible. The second side effect of pushing $\pi_X[I]$ to $\pi^*_X[I]$ is that X may collide with the other robot during the interval I. When this happens we perform a set of *secondary pushes*, as described later.

The overall surgery. Without loss of generality, we assume that $\pi_A(\lambda_2) \in \mathsf{top}(R)$. Our modification of $\boldsymbol{\pi}$ into $\boldsymbol{\pi}^*$ follows three main cases:

- *Case I:* $\pi_A(\lambda_1) \in \mathit{top}(R)$.
  We perform one push operation: $\textsc{Push}(A, [\lambda_1, \lambda_2], y(\mathsf{top}(R)))$; see Figure 5(ii).

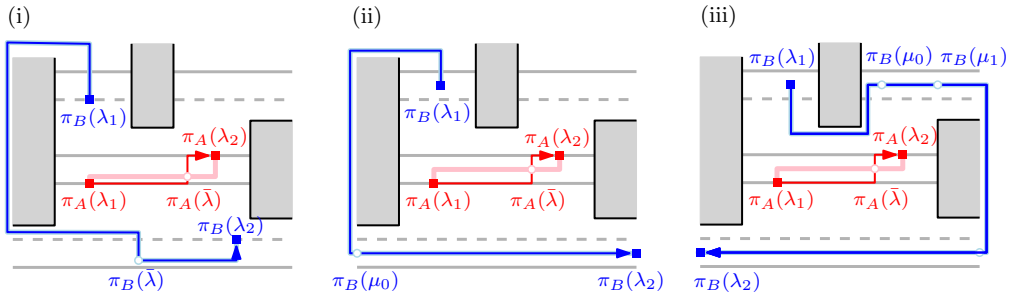


Figure 6 The modifications in the subcases of Case II. The original paths are shown in pink and light blue. In all cases, π_A^* jumps to $\mathsf{top}(R)$ at a suitable time $\bar{\lambda}$. (i) The modification in Case II(a). $\bar{\lambda}$ is the moment when $\boldsymbol{\pi}$ enters the last y-separated interval. (ii) The modification in Case II(b). $\bar{\lambda} = \mu_1$ is the moment when $\boldsymbol{\pi}$ enters the swap interval with B below A. (iii) The modification in Case II(c). $\bar{\lambda} = \mu_2$ is the moment when $\boldsymbol{\pi}$ leaves the swap interval with B above A.

- *Case II: $\pi_A(\lambda_1) \in \mathsf{bot}(R)$. Furthermore, either there is at most one swap interval, or there are two swap intervals and π_B lies above π_A during the first swap interval.*
  In this case, we pick $\bar{\lambda} \in [\lambda_1, \lambda_2]$ suitably and perform two push operations: $\textsc{Push}(A, [\lambda_1, \bar{\lambda}], y(\mathsf{bot}(R)))$ and $\textsc{Push}(A, [\bar{\lambda}, \lambda_2], y(\mathsf{top}(R)))$. Thus, robot A moves along $\mathsf{bot}(R)$ from time λ_1 to $\bar{\lambda}$, and then it switches to moving along $\mathsf{top}(R)$ until time λ_2. We have three subcases for the choice of $\bar{\lambda}$, illustrated in Figure 6.
  - *Case II(a): There are no swap intervals.* If every configuration in $\boldsymbol{\pi}[\lambda_1, \lambda_2]$ is y-separated, then either B lies below A, or above A, during the entire interval $[\lambda_1, \lambda_2]$. If B lies below (resp. above) A during $[\lambda_1, \lambda_2]$, set $\bar{\lambda} = \lambda_1$ (resp. $\bar{\lambda} = \lambda_2$). On the other hand, if there is a configuration in $\boldsymbol{\pi}[\lambda_1, \lambda_2]$ that is not y-separated, set $\bar{\lambda} := \sup\{\lambda \in [\lambda_1, \lambda_2] : \boldsymbol{\pi}(\lambda) \text{ is not } y\text{-separated}\}$.
  - *Case II(b): There is one swap interval and π_B is below π_A during this swap interval.* Let $[\mu_0, \mu_1]$ be this swap interval. Set $\bar{\lambda} := \mu_0$.
  - *Case II(c): There is at least one swap interval and π_B lies above π_A during the first one.* Let $[\mu_0, \mu_1]$ be this swap interval. Set $\bar{\lambda} := \mu_1$.

- *Case III: $\pi_A(\lambda_1) \in \mathsf{bot}(R)$ and there are exactly two swap intervals, and π_B lies below π_A the during the first swap interval.*
  Let $[\mu_0, \mu_1]$ and $[\mu_2, \mu_3]$ be the two swap intervals, with $\mu_1 < \mu_2$. Since all configurations in $\boldsymbol{\pi}[\mu_0, \mu_1]$ are y-separated and B lies below A during $[\mu_0, \mu_1]$, the path $\pi_B[\mu_0, \mu_1]$ lies in a connected component of $\mathcal{F} \cap Z^-$, which is a rectangle Q'. We prove that $\pi_B[\mu_0, \mu_1]$ is contained in a rectangle $Q \supseteq Q'$ of the subdivision of $\mathcal{F}$ induced by the lines of L_0. Let $[\nu_0, \nu_1]$ be the maximal interval containing $[\mu_0, \mu_1]$ such that $\pi_B(\nu_0, \nu_1) \subset \mathrm{int}(Q)$. The push operations depend on which of $\pi_B(\nu_0), \pi_B(\nu_1)$ lie on $\mathsf{top}(Q)$.
  - *Case III(a): $\pi_B(\nu_0) \in \mathsf{bot}(Q)$ and $\pi_B(\nu_1) \in \mathsf{top}(Q)$.* We perform two push operations, $\textsc{Push}(A, [\lambda_1, \lambda_2], y(\mathsf{bot}(R)))$ and $\textsc{Push}(B, [\nu_0, \mu_1], y(\mathsf{bot}(Q)))$.
  - *Case III(b): $\pi_B(\nu_0) \in \mathsf{top}(Q)$ and $\pi_B(\nu_1) \in \mathsf{bot}(Q)$.* Symmetric to the previous case.
  - *Case III(c): $\pi_B(\nu_0), \pi_B(\nu_1) \in \mathsf{top}(Q)$.* Again, we perform two push operations. The first is $\textsc{Push}(A, [\lambda_1, \lambda_2], y(\mathsf{top}(R)))$. (Note that, unlike in the previous two subcases, we pushed A to $\mathsf{top}(R)$ instead of $\mathsf{bot}(R)$.) Recall that $\pi_B[\mu_0, \mu_1] \subset \mathcal{F} \cap Z^-$, and let $[\rho_0, \rho_1] \supseteq [\mu_0, \mu_1]$ be the maximal interval such that $\pi_B[\rho_0, \rho_1]$ lies on or below the horizontal line $\ell^- : y = y(\mathsf{top}(R)) - 1$ that contains $\mathsf{top}(Z^-)$. The second push operation is $\textsc{Push}(B, [\rho_0, \rho_1], y(\mathsf{top}(R)) - 1)$.
  - *Case III(d): $\pi_B(\nu_0), \pi_B(\nu_1) \in \mathsf{bot}(Q)$.* We argue that this case does not occur.

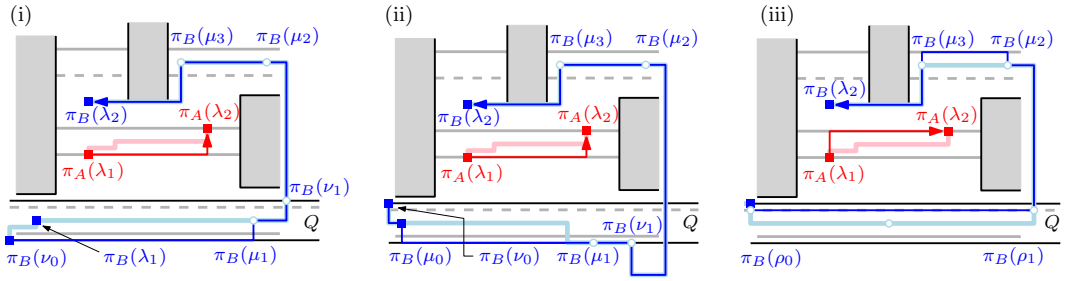


Figure 7 The modifications in the subcases of Case III. The original paths are shown in pink and light blue. (i) In Case III(a) we push $\pi_A[\lambda_1, \lambda_2]$ onto $\mathsf{bot}(R)$, and we push $\pi_B[\nu_0, \mu_1]$ onto $y(\mathsf{bot}(Q))$. (ii) In Case III(b) we also push $\pi_A[\lambda_1, \lambda_2]$ onto $\mathsf{bot}(R)$, and we push $\pi_B[\lambda_1, \nu_1]$ onto $y(\mathsf{bot}(Q))$. In this example, we have $\mu_0 = \lambda_1$. (iii) In Case III(c) we push $\pi_A[\lambda_1, \lambda_2]$ onto $\mathsf{top}(R)$, and we push $\pi_B[\rho_0, \rho_1]$ onto $\mathsf{top}(R) - 1$. We also have a secondary push, namely $\pi_B[\mu_2, \mu_3]$ onto $\mathsf{top}(R) + 1$.

Secondary pushes. After applying the above surgery, we possibly apply secondary pushes to π_B, as follows. For every unsafe interval $[\nu_1, \nu_2]$, if there is a time $\nu \in [\nu_1, \nu_2]$ such that $\pi_A^*(\nu)$ conflicts with $\pi_B^*(\nu)$, then we execute $\textsc{Push}(B, (\nu_1, \nu_2), y^*)$, where

$$y^* := \begin{cases} y(\mathsf{top}(R)) + 1 & \text{if } \pi_B(\nu)_y > \pi_A(\nu)_y, \\ y(\mathsf{bot}(R)) - 1 & \text{if } \pi_B(\nu)_y < \pi_A(\nu)_y. \end{cases} \tag{2}$$

Correctness proof (sketch). To prove the correctness of the surgery, we need to prove that the new plan has properties (P1)–P(5). The proofs of (P3)–(P5) are not difficult – they are essentially satisfied by construction. The proof of Feasibility (P1) for the surgery performed in Cases (I) and (II) is not hard either, but it is highly nontrivial for Case (III) because π_B might get modified beyond the interval $[\lambda_1, \lambda_2]$, e.g. during the interval $[\nu_0, \mu_1]$ with $\nu_0 < \lambda_1$ in Case III(a) or during the interval $[\rho_0, \mu_1]$ in Case III(c). Since A might not lie in R during $[\nu_0, \mu_1]$ (or $[\rho_0, \mu_1]$), we have to prove that the modified plan $\boldsymbol{\pi}^*$ during the interval $[\nu_0, \lambda_1]$ (or $[\rho_0, \lambda_1]$) remains feasible. We can prove the feasibility of $\boldsymbol{\pi}^*[\nu_0, \lambda_1]$ in Case III(a) by showing that if $\boldsymbol{\pi}^*[\nu_0, \lambda_1]$ were not feasible then $\boldsymbol{\pi}$ would not have been an optimal plan. We prove the feasibility of $\boldsymbol{\pi}^*[\rho_0, \mu_1]$ in Case III(c) by showing the existence of an optimal plan with additional structural properties, performing a compliant modification on $\boldsymbol{\pi}$, if necessary, to ensure these properties, and arguing that these properties ensure that $\boldsymbol{\pi}^*[\rho_0, \mu_1]$ is feasible. Finally, we need to prove Optimality (P2) of the modified path, which is also nontrivial because the surgery may increase the length of one of the paths. To illustrate this, we show that the surgery in Case I does not increase the total length of the plan, even though it may increase the length of the plan of robot A.

▶ **Lemma 10.** *The modification in Case I yields a plan $\boldsymbol{\pi}^*$ that is optimal.*

Proof. By construction, $\|\pi_A^*[\lambda_1, \lambda_2]\| \leqslant \|\pi_A[\lambda_1, \lambda_2]\|$. If π_B remains unchanged then trivially $\|\boldsymbol{\pi}^*\| \leqslant \|\boldsymbol{\pi}\|$, so assume there is an unsafe interval $[\nu_1, \nu_2]$ in which π_B lies in Z^+ and is pushed to $\mathsf{top}(R_\square)$. If $\boldsymbol{\pi}(\nu_1)$ is not x-separated then either $\nu_1 = \lambda_1$ or π_B enters $\mathcal{I}(R)$ through $\mathsf{top}(R_\square)$. (It cannot enter through the right or left edge of $R_\square$, because then $\boldsymbol{\pi}(\nu_1)$ would be x-separated.) Since $\pi_A(\lambda_1) \in \mathsf{top}(R)$, we have $\pi_B(\nu_1) \in \mathsf{top}(R_\square)$ if $\nu_1 = \lambda_1$; this is true since $\boldsymbol{\pi}(\nu)$ is y-separated for all $\nu \in [\nu_1, \nu_2]$. Hence, if $\boldsymbol{\pi}(\nu_1)$ is not x-separated then $\pi_B(\nu_1)_y = y_R^+ + 1$. Since π_B is pushed to $\mathsf{top}(R_\square)$ during $[\nu_1, \nu_2]$, we prove that this push procedure does not increase the length of the path of B. The same holds if $\boldsymbol{\pi}(\nu_2)$ is not x-separated. Hence, $\boldsymbol{\pi}^*$ is still optimal in these cases.

Now assume that both $\boldsymbol{\pi}(\nu_1)$ and $\boldsymbol{\pi}(\nu_2)$ are x-separated. By Lemma 9, we know that $[\nu_1, \nu_2]$ is a swap interval. Since there is at most one swap interval with B above A, by Lemma 9, there is at most one swap interval $[\nu_1, \nu_2]$ during which π_B is modified. Pushing B to $\mathsf{top}(R_\square)$ during $[\nu_1, \nu_2]$ increases $\|\pi_B^*\|$ by at most the total length of the ghost segments $g(\nu_1)$ and $g(\nu_2)$, which is $2y_R^+ + 2 - \pi_B(\nu_1)_y - \pi_B(\nu_2)_y$. Pushing A to $\mathsf{top}(R)$ during $[\nu_1, \nu_2]$ decreases the length of its path by at least $2y_R^+ - \pi_A(\nu_1)_y - \pi_A(\nu_2)_y$. By definition, the configurations $\boldsymbol{\pi}(\nu_i)$ are y-separated for $i = 1, 2$, so $\pi_B(\nu_i)_y - \pi_A(\nu_i)_y \geqslant 1$. Hence,

$$\begin{aligned}\|\boldsymbol{\pi}\| - \|\boldsymbol{\pi}^*\| &\geqslant \Big(2y_R^+ - \pi_A(\nu_1)_y - \pi_A(\nu_2)_y\Big) - \Big(2y_R^+ + 2 - \pi_B(\nu_1)_y - \pi_B(\nu_2)_y\Big) \\ &= \Big(\pi_B(\nu_1)_y - \pi_A(\nu_1)_y - 1\Big) + \Big(\pi_B(\nu_2)_y - \pi_A(\nu_2)_y - 1\Big) \geqslant 0.\end{aligned}$$

We thus conclude that $\boldsymbol{\pi}^*$ is optimal. ◀

Omitting further details of the correctness, we conclude that Theorem 4 holds, which is the crucial ingredient for Theorem 1. This concludes our sketch of the proof of our main result.

4 Conclusion

We presented an $O(n^4 \log n)$ algorithm to compute a plan of minimum total length for two unit squares in a rectilinear environment; this is the first polynomial algorithm for 2-robot optimal motion planning in a polygonal environment. In contrast, we showed that minimizing the makespan is weakly NP-hard. We conclude with some open problems:

- Can the runtime of our algorithm for the Min-Sum problem be improved to $O(n^2 \log n)$? Can our algorithm be extended to k unit squares with $n^{O(k)}$ runtime?
- Is the Min-Sum problem for two squares in a general polygonal environment in P?
- Is there a pseudo-polynomial-time algorithm for the Min-Makespan problem for two unit squares translating in a rectilinear environment?

References

1. Mohammad Ali Abam and Mohammad Ghodsi. An approximation algorithm for d_1-optimal motion of a rod robot with fixed rotations. *Int. J. Comput. Math.*, 83(3):357–370, 2006. doi:10.1080/00207160600847787.
2. Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Trans. Autom. Sci. Eng.*, 12(4):1309–1317, 2015. doi:10.1109/TASE.2015.2470096.
3. Pankaj K. Agarwal, Tzvika Geft, Dan Halperin, and Erin Taylor. Multi-robot motion planning for unit discs with revolving areas. *Comput. Geom.*, 114:102019, 2023. doi:10.1016/j.comgeo.2023.102019.
4. 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 *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4942–4962, 2024. doi:10.1137/1.9781611977912.176.
5. Pankaj K. Agarwal, R. Sharathkumar, and Hai Yu. Approximate Euclidean shortest paths amid convex obstacles. In *Proceedings of the 2009 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 283–292, 2009. doi:10.1137/1.9781611973068.32.
6. Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. North-Holland, 2000. doi:10.1016/B978-044482537-7/50003-6.

7 Luke Antonyshyn, Jefferson Silveira, Sidney Givigi, and Joshua A. Marshall. Multiple mobile robot task and motion planning: A survey. *ACM Comput. Surv.*, 55(10):213:1–213:35, 2023. doi:10.1145/3564696.

8 Boris Aronov, Mark de Berg, A. Frank van der Stappen, Petr Svestka, and Jules Vleugels. Motion planning for multiple robots. *Discret. Comput. Geom.*, 22(4):505–525, 1999. doi:10.1007/PL00009476.

9 Tetsuo Asano, David Kirkpatrick, and Chee K. Yap. d_1-Optimal motion for a rod (extended abstract). In *Proceedings of the 12th Annual Symposium on Computational Geometry*, pages 252–263, 1996. doi:10.1145/237218.237394.

10 Tetsuo Asano, David G. Kirkpatrick, and Chee-Keng Yap. Minimizing the endpoint trace length of rod motions amidst polygonal obstacles is NP-hard. In *Proc. 15th Canadian Conf. Comput. Geom.*, pages 10–13, 2003. URL: https://api.semanticscholar.org/CorpusID:123390076.

11 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 *Proceedings of the 38th International Symposium on Computational Geometry (SoCG)*, pages 12:1–12:16, 2022. doi:10.4230/LIPICS.SOCG.2022.12.

12 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications (3rd edition)*. Springer-Verlag, 2008.

13 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 *Proceedings of the 10th International Conference on Fun with Algorithms (FUN)*, volume 157 of *LIPIcs*, pages 15:1–15:16, 2021. doi:10.4230/LIPICS.FUN.2021.15.

14 Josh Brunner, Lily Chung, Erik D. Demaine, Dylan H. Hendrickson, Adam Hesterberg, Adam Suhl, and Avi Zeff. 1×1 rush hour with fixed blocks is PSPACE-complete. In *Proceedings of the 10th International Conference on Fun with Algorithms (FUN)*, volume 157 of LIPIcs, pages 7:1–7:14, 2021. doi:10.4230/LIPICS.FUN.2021.7.

15 John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 49–60, 1987. doi:10.1109/SFCS.1987.42.

16 Danny Z Chen and Haitao Wang. Computing shortest paths among curved obstacles in the plane. *ACM Trans. Alg.*, 11(4):1–46, 2015. doi:10.1145/2660771.

17 Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, pages 56–65, 1987. doi:10.1145/28395.28402.

18 Dror Dayan, Kiril Solovey, Marco Pavone, and Dan Halperin. Near-optimal multi-robot motion planning with finite sampling. *IEEE Trans. Robotics*, 39(5):3422–3436, 2023. doi:10.1109/TRO.2023.3281152.

19 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Parameterized algorithms for coordinated motion planning: Minimizing energy. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297 of *LIPIcs*, pages 53:1–53:18, 2024. doi:10.4230/LIPICS.ICALP.2024.53.

20 Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 28:1–28:16, 2023. doi:10.4230/LIPICS.SOCG.2023.28.

21 Guillermo Esteban, Dan Halperin, Víctor Ruíz, Vera Sacristán, and Rodrigo I. Silveira. Shortest coordinated motions for square robots. In *Proceedings of the 18th International Symposium on Algorithms and Data Structures (WADS)*, volume 14079 of *Lecture Notes in Computer Science*, pages 430–443, 2023. doi:10.1007/978-3-031-38906-1_28.

22 Dan Halperin, Lydia Kavraki, and Kiril Solovey. Robotics. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 51, pages 1343–1376. Chapman & Hall/CRC, 3rd edition, 2018.

23 Dan Halperin, Micha Sharir, and Oren Salzman. Algorithmic motion planning. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50, pages 1311–1342. Chapman & Hall/CRC, 3rd edition, 2018.

24 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005. doi:10.1016/J.TCS.2005.05.008.

25 John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999. doi:10.1137/S0097539795289604.

26 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". *Int. J. Robotics Res.*, 3(4):76–88, 1984.

27 Iyad Kanj and Salman Parsa. On the parameterized complexity of motion planning for rectangular robots. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 65:1–65:15, 2024. doi:10.4230/LIPICS.SOCG.2024.65.

28 Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Res.*, 30(7):846–894, 2011. doi:10.1177/0278364911406761.

29 David G. Kirkpatrick and Paul Liu. Characterizing minimum-length coordinated motions for two discs. In *Proc. 28th Canadian Conf. Comput. Geom.*, pages 252–259, 2016.

30 Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

31 Joseph S.B. Mitchell. Shortest paths and networks. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 51, pages 811–848. Chapman & Hall/CRC, 3rd edition, 2018.

32 Pedro J. Rezende, Der-Tsai Lee, and Ying-Fung Wu. Rectilinear shortest paths in the presence of rectangular barriers. *Discrete Comput. Geom.*, 4(1):41–53, 1989. doi:10.1007/BF02187714.

33 Oren Salzman. Sampling-based robot motion planning. *Commun. ACM*, 62(10):54–63, 2019. doi:10.1145/3318164.

34 Micha Sharir. A note on the papadimitriou-silverberg algorithm for planning optimal piecewise-linear motion of a ladder. *Inf. Process. Lett.*, 32(4):187–190, 1989. doi:10.1016/0020-0190(89)90042-2.

35 Micha Sharir and Shmuel Sifrony. Coordinated motion planning for two independent robots. *Ann. Math. Artif. Intell.*, 3(1):107–130, 1991. doi:10.1007/BF01530889.

36 Rahul Shome, Kiril Solovey, Andrew Dobson, Dan Halperin, and Kostas E. Bekris. dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Auton. Robots*, 44(3-4):443–467, 2020. doi:10.1007/S10514-019-09832-9.

37 Israela Solomon and Dan Halperin. Motion planning for multiple unit-ball robots in $\mathbb{R}^d$. In *Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 799–816, 2018. doi:10.1007/978-3-030-44051-0_46.

38 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *Int. J. Robotics Res.*, 35(14):1750–1759, 2016. doi:10.1177/0278364916672311.

39 Kiril Solovey, Lucas Janson, Edward Schmerling, Emilio Frazzoli, and Marco Pavone. Revisiting the asymptotic optimality of RRT*. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2189–2195, 2020. doi:10.1109/ICRA40945.2020.9196553.

40 Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems*, 2015. doi:10.15607/RSS.2015.XI.011.

41 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.

42 Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proc. 12th Int. Sympos. Combinatorial Search*, pages 151–159, 2019. doi:10.1609/SOCS.V10I1.18510.

43 Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Auton. Robots*, 37(4):401–415, 2014. doi:10.1007/S10514-014-9412-1.

44 Haitao Wang. A new algorithm for Euclidean shortest paths in the plane. In *Proceedings of the 53rd Symposium on the Theory of Computing (STOC)*, pages 975–988, 2021. doi:10.1145/3406325.3451037.

45 Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

46 Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proc. 27th Conf. Artificial Intelligence*, 2013. doi:10.1609/AAAI.V27I1.8541.

A Sparse Multicover Bifiltration of Linear Size

Ángel Javier Alonso
Graz University of Technology, Austria

Abstract

The k-cover of a point cloud X of $\mathbb{R}^d$ at radius r is the set of all those points within distance r of at least k points of X. By varying r and k we obtain a two-parameter filtration known as the multicover bifiltration. This bifiltration has received attention recently due to being choice-free and robust to outliers. However, it is hard to compute: the smallest known equivalent simplicial bifiltration has $O(|X|^{d+1})$ simplices. In this paper we introduce a $(1+\varepsilon)$-approximation of the multicover bifiltration of linear size $O(|X|)$, for fixed d and ε. The methods also apply to the subdivision Rips bifiltration on metric spaces of bounded doubling dimension yielding analogous results.

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

Keywords and phrases Multicover, Approximation, Sparsification, Multiparameter persistence

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.6

Related Version *Full Version*: https://arxiv.org/abs/2411.06986

Funding This research was funded in whole, or in part, by the Austrian Science Fund (FWF) 10.55776/P33765.

Acknowledgements The author would like to thank his advisor Michael Kerber for helpful comments. He would also like to deeply thank an anonymous reviewer for many comments that helped to substantially improve the exposition. Part of the writing was carried out while the author was visiting Yasuaki Hiraoka's group at the Kyoto University Institute of Advanced Study.

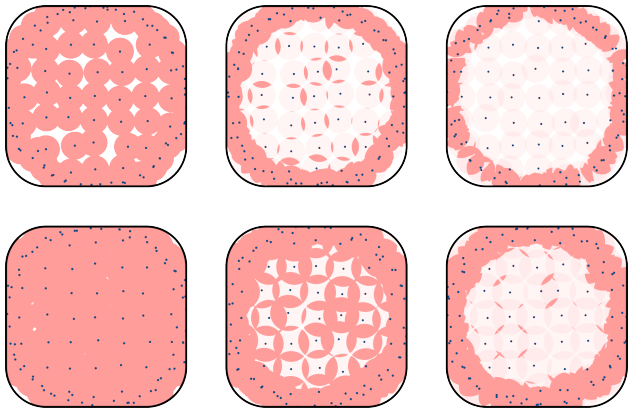


Figure 1 Illustration of the multicover bifiltration $\mathrm{Cov}_{r,k}$ for various values of r and k. The left column has $k=1$ and is thus the union of balls around the points, the middle column show those points covered by at least two balls, $k=2$, and the right column has $k=3$: those points covered by at least three balls. The top row has a smaller scale parameter r than the bottom row.

© Ángel Javier Alonso;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

This paper aims to approximate the multicover bifiltration of a finite subset X of $\mathbb{R}^d$, depicted in Figure 1. The *k-cover* $\mathrm{Cov}_{r,k}$ for a scale $r \geq 0$ is given by all those points covered by at least k balls of radius r around the points of X:

$$\mathrm{Cov}_{r,k} \coloneqq \{p \in \mathbb{R}^d \mid \|x - p\| \leq r \text{ for at least } k \text{ points } x \in X\}.$$

For any $r \leq r'$ and $k \geq k'$, we have that $\mathrm{Cov}_{r,k} \subset \mathrm{Cov}_{r',k'}$ and as such the $\mathrm{Cov}_{r,k}$ assemble into a *bifiltration* known as the *multicover bifiltration*.

The multicover bifiltration offers a view into the topology of the point cloud X across multiple scales, in a way that is robust to outliers [7]. However, computing the multicover bifiltration remains a challenge. Current exact combinatorial (simplicial or polyhedral) methods [28, 34] are expensive, of size $O(|X|^{d+1})$, and previous approximate methods [9, 12] are in general exponential and only linear on $|X|$ when taking k to be at most a constant.

For general metric spaces, instead of subsets of $\mathbb{R}^d$, an analogue of the multicover is the subdivision Rips bifiltration [53]. As the multicover, it has received attention recently, both in its theoretical [7] and computational aspects [41, 42, 43], where it faces similar challenges as the multicover. Previous approximation schemes [7, 41, 42, 43, 44] are of size polynomial (but not linear) in $|X|$, see the related work section below.

In this paper, we define a simplicial bifiltration, the *sparse subdivision bifiltration* (Definition 13), that $(1+\varepsilon)$-approximates the multicover and that is of linear size $O(|X|)$, independently of k and for fixed ε and d. It can be computed in time $O(|X| \log \Delta)$, where Δ is the *spread* of X, the ratio between the longest and shortest distances between points in X. In addition, the methods extend to obtain analogous results for the subdivision Rips bifiltration for metric spaces of bounded doubling dimension, as we discuss in Section 6.

The sparse subdivision bifiltration is (homotopically) equivalent to the *sparse multicover bifiltration* SCov (Definition 6), a bifiltration of subsets of $\mathbb{R}^d$. It is via SCov that the sparse subdivision bifiltration $(1+\varepsilon)$-approximates the multicover, by which we mean that

$$\mathrm{Cov}_{r,k} \subset \mathrm{SCov}_{(1+3\varepsilon)r,k} \text{ and } \mathrm{SCov}_{r,k} \subset \mathrm{Cov}_{(1+\varepsilon)r,k}$$

for any r and k, as shown in Theorem 8. In other words, the sparse subdivision bifiltration and the multicover are *(multiplicatively)* $(1+\varepsilon)$*-homotopy-interleaved* [6].

This approximation is analogous to the one obtained by Cavanna, Jahanseir and Sheehy [16] for the case of fixed $k = 1$, and our techniques can be traced back to their work; see the related work section. A key principle is that instead of balls of radius r, the sparse multicover bifiltration is based on *sparse balls* (Definition 1), understood as balls with a lifetime of three phases: on the first phase they grow normally (at scale r they have radius r), and then they are slowed down (they keep growing but at scale r they have radius less than r), before eventually disappearing. Cavanna, Jahanseir and Sheehy [16] (there is a video [15]) do the same, with technical differences that include stopping balls from growing, rather than growing slow. The size bounds here are also an adaptation of their methods, as well as the technique of lifting the construction one dimension higher, as in the *cones* of Section 3.1.

We guarantee that when a sparse ball disappears it is approximately covered by another present sparse ball, and we keep track of which present sparse ball covers which disappeared sparse ball via a *covering map* (Definition 4). A sparse ball is weighted by as many sparse balls it approximately covers according to the covering map – the *sparse k-cover* $\mathrm{SCov}_{r,k}$ consists of those points covered by sparse balls whose total weight is at least k.

1.1 Motivation and related work

For $k = 1$, the 1-cover is the union of balls of radius r around points of X. The union of balls is commonly used in reconstructing submanifolds from samples [3, 4, 46] and is a cornerstone of *persistent homology* methods in topological data analysis, being homotopy equivalent to the Čech complex and the alpha complex [31, 32], and related to the (Vietoris-)Rips complex. The union of balls assembles into a 1-parameter filtration: by increasing the scale parameter r we obtain inclusions $\mathrm{Cov}_{r,1} \hookrightarrow \mathrm{Cov}_{r',1}$, for $r \leq r'$. Taking the homology of each $\mathrm{Cov}_{r,1}$, we obtain a *persistence module*, an algebraic descriptor of the topology of the filtration across the different scales. Such a persistence module is stable to perturbations of the points [25, 26], but is not robust to outliers (already appreciated in the first column of Figure 1) and insensitive to differences in density in the point cloud.

There are multiple methods that address the lack of robustness to outliers and changes in density within the 1-parameter framework. These include density estimation [8, 20, 47], distance to a measure [1, 2, 10, 11, 18, 19, 38, 39], and subsampling [5]. However, they all depend on choosing a parameter. The problem is that it is not clear how to choose such a parameter for all cases, and such a choice might focus on a specific range of scales or densities. We refer to [7, Section 1.7] for a complete overview of the methods and their limitations.

It is then natural to consider constructions over two parameters, scale and density. Examples include the density bifiltration [13], degree bifiltration [44], and the multicover, closely related to both the distance to a measure [18] and k-nearest neighbors [53]. The advantage of the multicover is that it does not depend on any further choices (like choosing a density estimation function) and that it is robust to outliers, as a consequence of its strong stability properties [7].

However, one currently problematic aspect of the multicover is its computation. Sheehy [53] introduced an influential simplicial model of the multicover called the *subdivision Čech bifiltration*, based on the barycentric subdivision of the Čech filtration. It has exponentially many vertices on the number of input points, making its computation infeasible. A crucial ingredient in the theory is the *multicover nerve theorem* [7, 14, 53] that establishes the topological equivalence (see Section 3.1 for a precise definition) of the subdivision Čech and multicover bifiltrations. Such a multicover nerve theorem has its analogue for the sparse multicover, the *sparse multicover nerve theorem* (Theorem 14).

Dually to an hyperplane arrangement in $\mathbb{R}^{d+1}$, Edelsbrunner and Osang [33, 34] define the *rhomboid tiling* and use it to compute, up to homotopy, slices (fixing k and varying r or viceversa) of the multicover bifiltration. Corbet, Kerber, Lesnick and Osang [27, 28] build on it to define two bifiltrations topologically equivalent to the multicover, of size $O(|X|^{d+1})$.

Recently, Buchet, Dornelas and Kerber [12] introduced an elegant $(1+\varepsilon)$-approximation of the multicover whose m-skeleton has size that is linear on $|X|$ but that incurs an exponential dependency on the maximum order k we want to compute. A crucial difference between our work and the Buchet-Dornelas-Kerber (BDK) sparsification is that BDK work at the level of intersection of balls, while we work directly at the level of balls. This starting point is what allows us to ultimately obtain a construction of linear size, independently of k.

In addition, BDK work by freezing the lenses: at a certain scale they stop growing. It is not clear (or is technically challenging, in the words of BDK) how to compute exactly the first scale at which freezing lenses first intersect – a problem they sidestep by discretizing the scale parameter (at no complexity cost). In contrast, by letting sparse balls grow slowly at a certain rate, rather than freezing them completely, we can compute their first intersection time exactly, as we explain in Section 5.3.

Sparsification. Our methods are part of a line that can be traced to the seminal work of Sheehy [52, 54] to obtain a linear size approximation of the Vietoris-Rips filtration. Subsequently, Cavanna, Jahanseir and Sheehy [16] simplified and generalized Sheehy's methods to obtain a linear size $(1+\varepsilon)$-approximation of the union-of-balls filtration $\mathrm{Cov}_{r,1}$. They construct a filtration S such that $S_r \subset \mathrm{Cov}_{r,1} \subset S_{(1+\varepsilon)r}$, resembling Theorem 8 here. Moreover, the *sparse balls* we use here are directly inspired by their methods: they use balls that grow normally, stop growing, and eventually disappear.

These methods, broadly referred to as *sparsification*, have also been applied to the Delaunay triangulation [55] and, as already mentioned, the multicover itself [12]. A fundamental ingredient in many of these constructions is the greedy permutation [30, 37, 49], or variants of it, that we also use in the form of *persistent nets* (Section 2.1).

General metric spaces. In general metric spaces, an analogue of the subdivision Čech bifiltration of Sheehy is the *subdivision Rips bifiltration*. Its computation faces similar challenges, and, in fact, no subexponential size simplicial model of it exists [42]. Thus, there has been recent interest in approximations of subdivision Rips. Indeed, the *degree Rips bifiltration* [44], which can be shown to be a $\sqrt{3}$-approximation [7] whose m-skeleton has size $O(|X|^{m+2})$, and has been implemented [44, 48, 50]. Furthermore, it was recently shown that subdivision Rips admits $\sqrt{2}$-approximations whose k-skeleta have the same size as degree Rips [41, 42]. Most recently, Lesnick and McCabe [42, 43], in the more particular case of metric spaces of bounded doubling dimension (which include Euclidean spaces), give a $(1+\varepsilon)$-approximation of subdivision Rips whose m-skeleton has $O(|X|^{m+2})$ simplices, for fixed ε and dimension. Our methods also apply to the subdivision Rips setting, yielding analogous results, as discussed in Section 6.

Miniball. In Section 5.3, we compute the first scale at which a set of sparse balls have non-empty intersection, for Euclidean space. If instead of sparse balls we would be using usual balls, such a scale would be the radius of the minimum enclosing ball of the centers of the balls – the *miniball problem*, first stated by Sylvester in 1857 [56]. Our solution to the analogous problem for sparse balls has its origin in Welzl algorithm [57], which takes randomized linear time on the number of centers. The miniball problem is of *LP-type* as introduced later by Matoušek, Sharir and Welzl [45], and Welzl algorithm for the miniball can be generalized to the Matoušek-Sharir-Welzl (MSW) algorithm for LP-type problems. Fischer and Gärtner [35] solve the problem of computing the minimum enclosing ball of balls – generalizing the miniball – via the MSW-algorithm in a way that is practical and efficient, as in the implementation in the CGAL library [36]. Here, we frame the problem of computing the first scale of intersection of sparse balls as an LP-type algorithm and show how to solve it as an extension of Fischer and Gärtner's methods. The miniball and similar problems have also been stated and solved through the geometric optimization point of view, see [17, 29] and references therein.

2 A sparse multicover bifiltration

In this section we introduce the *sparse multicover bifiltration*. As already noted in the introduction, instead of using balls of radius r, the sparse multicover involves balls with a lifetime of three phases: they start as usual balls of radius r at scale r, at some point they start to *grow slowly*, that is, they have radii smaller than r at scale r, and eventually they disappear. We call this version of balls *sparse balls*. For their definition and the subsequent proofs we use the notion of *persistent nets*, which we define first.

2.1 Persistent nets

Persistent nets, defined below, are a reinterpretation of farthest point sampling, or greedy permutations, as known in the sparsification literature [12,16,55], and as such are guaranteed to exist by Gonzalez algorithm [37]. We will revisit these concepts in Section 5.

Let (X, ∂) be a finite metric space. We say that a subset $S \subset X$ is a *r-net* for a radius $r \geq 0$, if it is

1. a r-covering: for every $x \in X \setminus S$ there is an $a \in S$ with $\partial(x, a) < r$, and
2. a r-packing: for every two $a, b \in S$ we have $\partial(a, b) \geq r$.

In the spirit of persistence, a *persistent net* $\mathcal{S}$ of X is a collection of r-nets $\mathcal{S}(r) \subset X$, one for each $r \geq 0$, such that for any two $r' \geq r$ we have $\mathcal{S}(r') \subset \mathcal{S}(r)$.

The *insertion radius* $\mathrm{ins}(x) \in \mathbb{R} \cup \{\infty\}$ of a point $x \in X$, with respect to a persistent net $\mathcal{S}$, is the supremum of those r such that $x \in \mathcal{S}(r)$. There is a large enough r so that $\mathcal{S}(r)$ is a single point (it has to be a r-packing), and so there is only one point x with $\mathrm{ins}(x) = \infty$.

2.2 Sparse balls

In what follows, we work over a fixed finite subset $X \subset \mathbb{R}^d$, with $\mathbb{R}^d$ equipped with any norm. We also fix an error parameter $\varepsilon > 0$ and a persistent net $\mathcal{S}$ of X, that we often drop from the notation. We define the *slowing time* $\mathrm{slow}(x)$ of a point $x \in X$ by

$$\mathrm{slow}(x) \coloneqq \frac{1+\varepsilon}{\varepsilon}\,\mathrm{ins}(x).$$

▶ **Definition 1.** *We define the* sparse ball $\mathrm{SB}(x, r)$ *of $x \in X$ at scale r as:*

$$\mathrm{SB}(x, r) \coloneqq \begin{cases} B(x, \rho_x(r)) & r \leq (1+3\varepsilon)\,\mathrm{slow}(x), \\ \varnothing & r > (1+3\varepsilon)\,\mathrm{slow}(x), \end{cases}$$

where $B(x, s)$ denotes the closed ball around x of radius s, and $\rho_x \colon \mathbb{R} \to \mathbb{R}$ is a radius function*: any strictly increasing function such that*

- *on the interval $[0, \mathrm{slow}(x)]$, is $\rho_x(r) = r$, and*
- *on the interval $(\mathrm{slow}(x), (1+3\varepsilon)\,\mathrm{slow}(x)]$, satisfies*

$$L_x(r) \leq \rho_x(r) \leq U_x(r), \text{ where}$$
$$L_x(r) \coloneqq \frac{1}{1+3\varepsilon} r + \frac{\varepsilon}{1+\varepsilon}\,\mathrm{slow}(x) \text{ and } U_x(r) \coloneqq \frac{1}{3(1+\varepsilon)} r + \frac{2+3\varepsilon}{3(1+\varepsilon)}\,\mathrm{slow}(x).$$

For example, the radius function of a point $x \in X$ can be

$$\rho_x(r) = \begin{cases} r, & r < \mathrm{slow}(x), \\ U_x(r), & r \geq \mathrm{slow}(x), \end{cases}$$

but other options are useful for computation (Section 5.3). In any case, the sparse ball of x has radius r up to scale $\mathrm{slow}(x)$, it then starts to grow slowly with radius at most $U_x(r) < r$, until eventually disappearing at scale $(1+3\varepsilon)\,\mathrm{slow}(x)$. We say that the sparse ball around $x \in X$ is *slowed* at scale r if $r \geq \mathrm{slow}(x)$.

The functions $L_x(r)$ and $U_x(r)$ are defined in this way for technical reasons related to properties of the sparse multicover that we will prove shortly. For now, let us say that whenever a sparse ball disappears (that is, at scale $(1+3\varepsilon)\,\mathrm{slow}(x)$), it is covered by another non-slowed sparse ball:

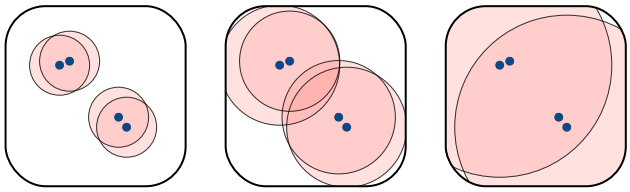


Figure 2 Sparse balls around four points in the plane. On the left square, they are not slowed yet. On the middle square, two of them are growing slow, and are starting to get covered by the other two. On the right square, two of the sparse balls have disappeared.

▶ **Lemma 2.** *Let $x \in X$ be a point and write $\gamma \coloneqq (1+3\varepsilon)\operatorname{slow}(x)$. There exists a point $y \in X$ such that $\operatorname{slow}(y) \geq \gamma$ and $\operatorname{SB}(x,\gamma) \subset \operatorname{SB}(y,\gamma)$.*

Proof. Note that the sparse ball of x has radius at most $L_x(\gamma) = U_x(\gamma) = \frac{1+2\varepsilon}{1+\varepsilon}\operatorname{slow}(x)$. At scale γ, the centers of the non-slowed balls are precisely the $\frac{\varepsilon}{1+\varepsilon}\gamma$-net $\mathcal{S}(\frac{\varepsilon}{1+\varepsilon}\gamma)$. Thus, since $\frac{\varepsilon}{1+\varepsilon}\gamma + \frac{1+2\varepsilon}{1+\varepsilon}\operatorname{slow}(x) < \gamma$, there is a y in $\mathcal{S}(\frac{\varepsilon}{1+\varepsilon}\gamma)$ such that the sparse ball around y covers the ball around x of radius $\frac{1+2\varepsilon}{1+\varepsilon}\operatorname{slow}(x)$,

$$\operatorname{SB}(x,\gamma) \subset B\left(x, \frac{1+2\varepsilon}{1+\varepsilon}\operatorname{slow}(x)\right) \subset B(y,\gamma) = \operatorname{SB}(y,\gamma). \qquad \blacktriangleleft$$

2.3 Covering map

One of the intuitive principles behind the sparse multicover is that, at every scale $r \geq 0$, the non-empty sparse balls *approximately cover* the balls of radius r around all the points of X. Below, we formally define a map that takes each point x to the non-empty sparse ball that approximately covers the ball of x.

▶ **Definition 3.** *We define the* covering sequence $(x_0, \ldots, x_m)$ *of a point $x \in X$ as follows: we set $x_0 = x$ and, recursively, given x_{i-1} and if $\operatorname{slow}(x_{i-1}) < \infty$, we set x_i to be the nearest neighbor (breaking ties arbitrarily) of x among those points y with $\operatorname{slow}(y) \geq (1+3\varepsilon)\operatorname{slow}(x_{i-1})$. In other words, the nearest neighbor among those points whose sparse balls are non-slowed when the sparse ball of x_{i-1} disappears.*

▶ **Definition 4.** *For each scale $r \geq 0$, the* covering map $\mathcal{C}_r \colon X \to X$ *takes x to the first x_i in its covering sequence whose sparse ball is non-empty at scale r.*

For a point $x \in X$ and $r \geq 0$, we say that $c_r(x) \coloneqq |\mathcal{C}_r^{-1}(x)|$, the cardinality of its preimage under the covering map, is its covering weight.

We will use the following lemma multiple times. It describes how the covering map inherits covering and packing properties from the persistent net.

▶ **Lemma 5.** *The covering map has the following properties, for each $r \geq 0$:*

1. *The image $\mathcal{C}_r(X)$ is the set of points whose sparse balls at radius r are non-empty,*
2. *the points in $\mathcal{C}_r(X)$ are $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}r$-packed, and*
3. *for each $x \in X$ and writing $y \coloneqq \mathcal{C}_r(x)$, $\|x-y\| \leq \frac{\varepsilon}{1+\varepsilon}\min(r, \operatorname{slow}(y))$.*

Proof. The first property is by definition.

For the second, by the previous property the sparse ball of a $x \in \mathcal{C}_r(X)$ has not disappeared at scale r, and thus $r \leq (1+3\varepsilon)\operatorname{slow}(x) = (1+3\varepsilon)\frac{1+\varepsilon}{\varepsilon}\operatorname{ins}(x)$. This gives that $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}r \leq \operatorname{ins}(x)$, which implies what we want by the definition of $\operatorname{ins}(x)$ and persistent net.

For the third, if $x = y$ there is nothing to prove. Otherwise, let z be the point directly before y in the covering sequence of x, and write $\gamma \coloneqq (1+3\varepsilon)\operatorname{slow}(z)$. By construction, $\gamma < r$ and $\gamma \leq \operatorname{slow}(y) = \frac{1+\varepsilon}{\varepsilon}\operatorname{ins}(y)$. Thus, y is in $\mathcal{S}(\frac{\varepsilon}{1+\varepsilon}\gamma)$, and y is the nearest neighbor of x in such subset. We conclude that $\|x-y\| \leq \frac{\varepsilon}{1+\varepsilon}\gamma \leq \frac{\varepsilon}{1+\varepsilon}\min(r, \operatorname{slow}(y))$. ◀

2.4 The sparse multicover

We can write the k-cover $\operatorname{Cov}_{r,k}$ for a scale $r \geq 0$ as

$$\operatorname{Cov}_{r,k} \coloneqq \left\{ p \in \mathbb{R}^d \mid \text{exists } S \subset X \text{ with } p \in \bigcap_{x \in S} B_r(x) \text{ and } |S| \geq k \right\}.$$

In parallel, the sparse multicover uses sparse balls instead of balls, and the covering weight instead of the number of balls:

▶ **Definition 6.** *The* sparse k-cover $\operatorname{SCov}_{r,k}$*, for $r \geq 0$ and $k \in \mathbb{N}$, is the set of points $p \in \mathbb{R}^d$ covered by sparse balls at scale r whose sum of covering weights at scale r is at least k,*

$$\operatorname{SCov}_{r,k} \coloneqq \left\{ p \in \mathbb{R}^d \mid \textit{exists } S \subset X \textit{ with } p \in \bigcap_{x \in S} \operatorname{SB}(x,r) \textit{ and } \sum_{x \in S} c_r(x) \geq k \right\}.$$

The *sparse multicover* is the bifiltration given by the sparse covers:

▶ **Proposition 7.** *For $r \leq r'$ and $k \geq k'$, we have $\operatorname{SCov}_{r,k} \subset \operatorname{SCov}_{r',k'}$.*

Proof. Let $p \in \operatorname{SCov}_{r,k}$ and let S be a subset whose sparse balls of radius r cover p with covering weight at least k. Take $A \coloneqq \mathcal{C}_r^{-1}(S)$ and note that there are at least k points in A. For each $a \in A$, we claim that the sparse ball of $\mathcal{C}_{r'}(a)$ (the point that "covers" a at scale r') also contains p. Proving the claim finishes the proof, as the sum of covering weights of $\mathcal{C}_{r'}(A)$ is at least $k \geq k'$.

We write $x \coloneqq \mathcal{C}_r(a)$ and $y \coloneqq \mathcal{C}_{r'}(a)$. The claim is immediate if $x = y$. Otherwise, note that necessarily $\operatorname{ins}(x) < \infty$, because the point with infinite insertion radius is unique. Also note that $x \notin \mathcal{C}_{r'}(X)$ and, thus, $(1+3\varepsilon)\operatorname{slow}(x) \leq \min(r', \operatorname{slow}(y))$. This fact, together with the triangle inequality and Lemma 5, gives

$$\begin{aligned}
\|y-p\| &\leq \|y-a\| + \|a-x\| + \|x-p\| \\
&\leq \frac{\varepsilon}{1+\varepsilon}\min(r', \operatorname{slow}(y)) + \frac{\varepsilon}{1+\varepsilon}\min(r, \operatorname{slow}(x)) + \rho_x(r). \\
&\leq \frac{\varepsilon}{1+\varepsilon}\min(r', \operatorname{slow}(y)) + \frac{\varepsilon}{1+\varepsilon}\operatorname{slow}(x) + \frac{1+2\varepsilon}{1+\varepsilon}\operatorname{slow}(x) \\
&\leq \frac{\varepsilon}{1+\varepsilon}\min(r', \operatorname{slow}(y)) + \frac{1+3\varepsilon}{1+\varepsilon}\operatorname{slow}(x) \\
&\leq \left(\frac{\varepsilon}{1+\varepsilon} + \frac{1}{1+\varepsilon}\right)\min(r', \operatorname{slow}(y)) \\
&= \min(r', \operatorname{slow}(y)) \leq \rho_y(r').
\end{aligned}$$ ◀

The sparse multicover approximates the multicover:

▶ **Theorem 8.** *Writing $\delta \coloneqq \frac{1+2\varepsilon}{1+\varepsilon} < 1+\varepsilon$, we have*

$$\operatorname{Cov}_{r,k} \subset \operatorname{SCov}_{(1+3\varepsilon)r,k} \textit{ and } \operatorname{SCov}_{r,k} \subset \operatorname{Cov}_{\delta r,k}.$$

Proof. We start with the first inclusion. Let $p \in \mathrm{Cov}_{r,k}$, which implies that there is a subset $S \subset X$ of k points such that $\|a - p\| \leq r$ for all $a \in S$.

Let $S' = \mathcal{C}_{(1+3\varepsilon)r}(S)$ and note that the sum of covering weights $\sum_{x \in S'} c_{(1+3\varepsilon)r}(x)$ is at least k. It is left to show that every sparse ball around points of S' at scale $(1+3\varepsilon)r$ contains p. Consider a point $a \in S$ and write $x \coloneqq \mathcal{C}_{(1+3\varepsilon)r}(a) \in S'$. Note that necessarily $r \leq \mathrm{slow}(x)$, because otherwise the sparse ball of x has disappeared by scale $(1+3\varepsilon)r$. We have

$$\|x - p\| \leq \|x - a\| + \|a - p\| \leq \frac{\varepsilon}{1+\varepsilon} \min((1+3\varepsilon)r, \mathrm{slow}(x)) + r.$$

Now we do a case distinction. If $(1+3\varepsilon)r < \mathrm{slow}(x)$, we have

$$\|x - p\| \leq \frac{\varepsilon}{1+\varepsilon}(1+3\varepsilon)r + r \leq (1+3\varepsilon)r = \rho_x((1+3\varepsilon)r),$$

and otherwise $(1+3\varepsilon)r \geq \mathrm{slow}(x)$, so we have

$$\|x - p\| \leq \frac{\varepsilon}{1+\varepsilon} \mathrm{slow}(x) + r = L_x((1+3\varepsilon)r) \leq \rho_x((1+3\varepsilon)r).$$

We now prove the second inclusion. Let $p \in \mathrm{SCov}_{r,k}(X)$. This means that there exists a subset $S \subset \mathcal{C}_r(X)$ whose sparse balls cover p at radius r with covering weight at least k. Consider $A \coloneqq \mathcal{C}_r^{-1}(S)$. Then, by definition, $|A| \geq k$ and it is left to show that for each $a \in A$ we have $\|a - p\| \leq \delta r$. Indeed, writing $x \coloneqq \mathcal{C}_r(a)$,

$$\|a - p\| \leq \|a - x\| + \|x - p\| \leq \frac{\varepsilon}{1+\varepsilon} \min(r, \mathrm{slow}(x)) + \rho_x(r).$$

We do a case distinction again. If $r \leq \mathrm{slow}(x)$, then

$$\|a - p\| \leq \frac{\varepsilon}{1+\varepsilon} r + r = \frac{1+2\varepsilon}{1+\varepsilon} r.$$

Otherwise, if $r > \mathrm{slow}(x)$,

$$\begin{aligned} \|a - p\| &\leq \frac{\varepsilon}{1+\varepsilon} \mathrm{slow}(x) + U_x(r) \\ &= \frac{\varepsilon}{1+\varepsilon} \mathrm{slow}(x) + \frac{1}{3(1+\varepsilon)} r + \frac{2+3\varepsilon}{3(1+\varepsilon)} \mathrm{slow}(x) \leq \frac{1+2\varepsilon}{1+\varepsilon} r. \end{aligned}$$

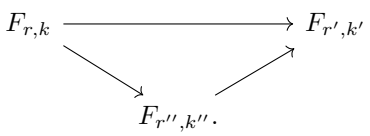
◀

3 An equivalent simplicial bifiltration

We look for a bifiltration of simplicial complexes that is homotopically equivalent, as defined shortly, to the sparse multicover bifiltration. This will be the *sparse subdivision bifiltration*, which is an extension of the subdivision Čech bifiltration of Sheehy [53].

3.1 Homotopically equivalent bifiltrations

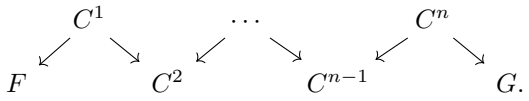
We now recall the fundamental definitions of the homotopy theory of filtrations, where filtrations are viewed as diagrams, as is standard; see, e.g., [7]. Let F be a bifiltration of topological spaces indexed, in our case, by radii $r \in [0, \infty)$ and order $k \in \mathbb{N}$; that is, a collection of topological spaces $F_{r,k}$, one for each $r \geq 0$ and $k \in \mathbb{N}$, together with a continuous map $F_{(r,k)\to(r',k')} \colon F_{r,k} \to F_{r',k'}$ for every $r \leq r'$ and $k \geq k'$, such that $F_{(r,k)\to(r,k)}$ is the identity and the following diagram commutes



A *pointwise weak equivalence* $F \xrightarrow{\cong} G$ between two bifiltrations F and G, is a collection of weak homotopy equivalences $\alpha_{r,k} \colon F_{r,k} \to G_{r,k}$ such that the following diagram commutes for every $r \leq r'$ and $k \geq k'$:

$$\begin{array}{ccc} F_{r,k} & \longrightarrow & F_{r',k'} \\ {\scriptstyle \alpha_{r,k}}\downarrow & & \downarrow{\scriptstyle \alpha_{r',k'}} \\ G_{r,k} & \longrightarrow & G_{r',k'} \,. \end{array}$$

Pointwise weak equivalence is not a equivalence relation, but the following is. Two bifiltrations F and G are *weakly equivalent* if there exists a zigzag of pointwise homotopy equivalences that connect them:



3.2 The sparse subdivision bifiltration

We now define the *sparse subdivision bifiltration* and prove that is weakly equivalent to the multicover bifiltration. As already mentioned, the sparse subdivision bifiltration is an extension of Sheehy's *subdivision Čech bifiltration* [53], which we review first.

Subdivision Čech bifiltration. We start with the following extension of the Čech complex:

▶ **Definition 9.** *For a given scale $r \in [0, \infty)$ and order $k \in \mathbb{N}$, the* k-Čech poset $\check{\mathrm{C}}_{r,k}$ *is*

$$\check{\mathrm{C}}_{r,k} \coloneqq \left\{ \sigma \subset X \mid \bigcap_{x \in \sigma} B(x, r) \neq \varnothing \text{ and } |\sigma| \geq k \right\}$$

with the order given by inclusion.

The 1-Čech poset is (the face poset of) the usual Čech simplicial complex. The collection of the Čech posets $\check{\mathrm{C}}_{r,k}$ assemble into a bifiltration that we call the *multi-Čech bifiltration* $\check{\mathrm{C}}$: for any $r \leq r'$ and $k \geq k'$, we have $\check{\mathrm{C}}_{r,k} \subset \check{\mathrm{C}}_{r',k'}$.

To obtain a bifiltration of simplicial complexes we take the order complex pointwise, as below. By *order complex* $\Delta(P)$ of a poset P we mean the simplicial complex whose set of vertices is P and whose m-simplices are the chains of P of length $m + 1$.

▶ **Definition 10.** *The* subdivision Čech bifiltration Sub *is given by* $\mathrm{Sub}_{r,k} \coloneqq \Delta(\check{\mathrm{C}}_{r,k})$ *with the internal maps being the inclusions.*

The subdivision Čech bifiltration is usually defined directly at the level of the subdivision of the Čech complex [53], rather than through the multi-Čech bifiltration. Now we can state the multicover nerve theorem [7, 14, 53]:

▶ **Theorem 11** (Multicover nerve theorem)**.** *The multicover bifiltration* Cov *is weakly equivalent to the subdivision Čech bifiltration* Sub*.*

Sparsification. We now extend the subdivision Čech bifiltration and the multicover nerve theorem to the sparse multicover bifiltration. As in the definition of the sparse multicover, we replace the balls with sparse balls and the cardinality with the covering weight, resulting in the following definition; to be compared with the definition of k-Čech poset (Definition 9.)

▶ **Definition 12.** *For a given scale $r \in [0,\infty)$ and order $k \in \mathbb{N}$, we define the* sparse Čech poset $\mathrm{S\check{C}}_{r,k}$ *to be*

$$\mathrm{S\check{C}}_{r,k} \coloneqq \bigcup_{s \le r} \left\{ \sigma \subset X \mid \bigcap_{x \in \sigma} \mathrm{SB}(x,s) \neq \varnothing \text{ and } \sum_{x \in \sigma} c_s(x) \ge k \right\}$$

with the order given by inclusion.

In the definition, the union is there to guarantee that the $\mathrm{S\check{C}}_{r,k}$ assemble into a bifiltration, because now sparse balls disappear, unlike before; this is the combinatorial analogue of taking *cones* as in Cavanna, Jahanseir and Sheehy's work [16]. Again taking the order complex:

▶ **Definition 13.** *The* sparse subdivision bifiltration SSub *is given by taking* $\mathrm{SSub}_{r,k} \coloneqq \Delta(\mathrm{S\check{C}}_{r,k})$*, the order complex of the sparse intersection poset* $\mathrm{S\check{C}}_{r,k}$*.*

In other words, the m-simplices of $\mathrm{SSub}_{r,k}$ are sequences $\sigma_0 \subset \cdots \subset \sigma_m$ of $m+1$ subsets of points, all in $\mathrm{S\check{C}}_{r,k}$ and where each containment is strict.

▶ **Theorem 14** (Sparse multicover nerve theorem)**.** *The sparse subdivision bifiltration* SSub *is weakly equivalent to the sparse multicover* SCov*.*

The proof is in the full version. It reduces to the multicover nerve theorem, incorporating the covering weight and using the *cones* strategy of Cavanna, Jahanseir and Sheehy [16].

4 Size

We are now ready to bound the size of the sparse subdivision bifiltration. Specifically, we bound the number of simplices in the largest complex in the bifiltration, and show that each simplex is born at a constant number of *critical grades*. We say that the grade $(r,k) \in [0,\infty) \times \mathbb{N}$ is *critical* for a simplex σ, if $\sigma \in \mathrm{SSub}_{r',k'}$ for all $r < r'$ and $k \ge k'$, but $\sigma \notin \mathrm{SSub}_{s,l}$ for any $s < r$ and $l > k$.

4.1 Number of simplices

The rest of this section is dedicated to proving the following theorem. We write SSub to refer both to the bifiltration and to the largest complex in the bifiltration, $\mathrm{SSub} = \bigcup_{r,k} \mathrm{SSub}_{r,k}$, when no confusion is possible.

▶ **Theorem 15.** *The bifiltration* SSub *has $O(|X|)$ simplices for fixed ε and dimension d.*

The following argument is analogous to the one used in previous sparse filtrations [16, 54]. Consider the map $\min\colon \mathrm{SSub} \to X$ that takes each simplex $\sigma = \sigma_0 \subset \cdots \subset \sigma_m$ to the minimum point x in σ_m with respect to the order given by their insertion radius $\mathrm{ins}(x)$, breaking ties arbitrarily. The points in each preimage of this map $\min\colon \mathrm{SSub} \to X$ are not too far from each other:

▶ **Lemma 16.** *Let $\sigma = \sigma_0 \subset \cdots \subset \sigma_m \in \mathrm{SSub}$ be a simplex and let $x \coloneqq \min(\sigma)$. The points σ_m are contained in a ball of radius $2(1+3\varepsilon)\,\mathrm{slow}(x)$ around x.*

Proof. Since the sparse ball around x is empty at scales greater than $\alpha \coloneqq (1+3\varepsilon)\,\mathrm{slow}(x)$, any intersection between the sparse balls of σ_m must happen at a scale $r \le \alpha$. Let p be any point in this intersection. It follows that for any other $y \in \sigma_m$ we have $\|x-y\| \le \|x-p\| + \|p-y\| \le 2\alpha$. ◀

▶ **Lemma 17.** *For each $x \in X$, the number of simplices mapped to x under* min *is $O(1)$, for fixed ε and dimension d.*

Proof. Let $A \coloneqq \bigcup_{\sigma_0 \subset \cdots \subset \sigma_m \in \min^{-1}(x)} \sigma_m$ be the set of all points in simplices σ such that $\min \sigma = x$. It suffices to show that A has $O(1)$ points, for fixed ε and d. Let $\alpha \coloneqq (1+3\varepsilon)\operatorname{slow}(x)$. We claim that $A \subset \mathcal{C}_\alpha(X)$. Indeed, for every $a \in A$ we have $\operatorname{ins}(a) \geq \operatorname{ins}(x)$, by the definition of min, and thus the sparse ball of a is non-empty at scale α.

By Lemma 16, the points in A are contained in a ball of radius 2α, and, by Lemma 5, they are $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}\alpha$-packed, because $A \subset \mathcal{C}_\alpha(X)$ as shown. By comparing the volume of balls of radius 2α and $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}\alpha$, we conclude that A consists of $O\left(\left(\frac{(1+\varepsilon)(1+3\varepsilon)}{\varepsilon}\right)^d\right)$ points. ◀

All in all, there can be at most $O(|X|)$ simplices, since at most a constant number, for a fixed ε and d, of simplices is mapped to each point, finishing the proof of the theorem.

4.2 Critical grades

The sparse subdivision bifiltration, unlike the subdivision Čech bifiltration, is *multicritical*, which means that each simplex may have multiple critical grades. Let us briefly describe the critical grades of a simplex $\sigma = \sigma_0 \subset \cdots \subset \sigma_m \in \mathrm{SSub}$. Ordering the critical grades by scale r, the first critical grade has scale r equal to the first scale at which the sparse balls around the points σ_m intersect, and order k equal to the sum of the covering weights of the points of σ_0 at such an scale. Further critical grades of σ correspond to scales r' at which the covering weight of the points of σ_0 increase, until one of the associated sparse balls disappears. Still, there are not many critical grades, as another packing argument shows:

▶ **Theorem 18.** *Each simplex in* SSub *has $O(1)$ critical grades, for fixed d and ε.*

Proof. Let $\sigma = \sigma_0 \subset \cdots \subset \sigma_m \in \mathrm{SSub}$ and let $x \coloneqq \min \sigma$. The simplex σ has no critical grade with scale greater than $\alpha \coloneqq (1+3\varepsilon)\operatorname{slow}(x)$, because the sparse ball of x disappears then. Thus, we bound the number of scales $s \leq \alpha$ at which the sum of covering weights $\sum_{x \in \sigma_0} c_s(x)$ of the points of σ_0 increases, which bounds the number of critical grades of σ.

By the proof of the Lemma 17, the points in σ_0 are a 2β-packing, where $\beta \coloneqq \frac{\varepsilon}{2(1+\varepsilon)(1+3\varepsilon)}\alpha$. Thus, any intersection between balls around points in σ_0 must happen at scales at least β, and thus we are interested in the range of scales $[\beta, \alpha]$.

Let $A \subset X$ be the subset of points $a \in X \setminus \sigma_0$ whose sparse ball disappears in the interval $[\beta, \alpha)$, meaning $(1+3\varepsilon)\operatorname{slow}(a) \in [\beta, \alpha)$ and, as a result, the covering weight of a point in σ_0 increases, meaning there is a point $y \in X$ such that a and a point $w \in \sigma_0$ are one after the other in the covering sequence of y, that is, $y_i = a$ and $y_{i+1} = w$. To finish the argument, it suffices to show that $|A| = O(1)$. First, note that $A \subset \mathcal{C}_\beta(X)$. It follows that the points in A are $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}\beta$-packed, by Lemma 5.

Let $a \in A$, and let $y \in X$ and $w \in \sigma_0$ as in the definition of A. By Lemma 5 and Lemma 16,

$$\begin{aligned}\|x - a\| &\leq \|x - w\| + \|w - y\| + \|y - a\| \\ &\leq 2\alpha + (1+3\varepsilon)\operatorname{slow}(a) + \frac{\varepsilon}{1+\varepsilon}\operatorname{slow}(a) \leq 4\alpha.\end{aligned}$$

All in all, the points of A are contained in a ball of radius 4α and are all $\frac{\varepsilon}{(1+\varepsilon)(1+3\varepsilon)}\beta = \frac{\varepsilon^2}{2(1+\varepsilon)^2(1+3\varepsilon)^2}\alpha$ apart. The result follows by comparing the volume of disjoint balls around points in A and the volume of a ball of radius 4α. ◀

5 Computation

We show how to compute the sparse subdivision bifiltration. The first step is computing a persistent net and the covering map. As already mentioned in Section 2.1 this reduces to computing a *greedy permutation*. The next step is to compute all potential simplices that appear in the bifiltration, as those consisting of points around balls like those of Lemma 16; for which we use a suitable proximity data structure. Finally, to obtain the critical grades of the simplices we can use the covering map and the minimum scale at which a subset of the sparse balls intersect. The problem of computing this minimum scale, in the Euclidean case, can be written as a LP-type problem [45], and we show how to solve it, much like computing the smallest enclosing ball of balls [35]. All these steps can be done in $O(|X| \log \Delta)$ time, where Δ is the *spread* of X, the ratio between the longest and shortest distances in X.

5.1 Greedy permutations

Let $(x_1, \dots, x_n)$ be an ordering of the n points in X. Such a sequence is called *greedy* if for every x_{i+1} and prefix $X_i \coloneqq \{x_1, \dots, x_i\}$, one has

$$d(x_{i+1}, X_i) = \max_{x \in X} d(x, X_i),$$

where $d(x, X_i) = \min_{x_j \in X_i} \|x - x_j\|$. The distance $r_{i+1} \coloneqq d(x_{i+1}, X_i)$ is the *insertion radius* r_{i+1} of x_{i+1}, where we take $r_1 = \infty$ by convention. A greedy permutation gives a persistent net $\mathcal{S}$ by taking $\mathcal{S}_r \coloneqq \{x_i \mid r \leq r_i\}$.

Gonzalez [37] gives a method to compute a greedy permutation. It consists of n phases, divided in an initialization phase and $|X| - 1$ update phases. At the end of phase i, it maintains a subset $S_i = \{x_1, \dots, x_i\} \subset X$ and for each $x_j \in S_i$ its *Voronoi set*: the subset of points in X that are closer to x_j than to any other point in S_i. We break ties arbitrarily but consistently (say, by minimum index in S_i), so that the Voronoi sets form a partition of X. In the initialization phase we pick a random point x_1 and we set its Voronoi set to be every other point. In the update phase i, we obtain the new set of leaders S_i by adding to S_{i-1} the point x_i whose distance to S_{i-1} is maximal, and updating the Voronoi sets. This algorithm can be implemented in $O(|X|^2)$-time by going over the whole set of points during each phase. An algorithm of Clarkson [23, 24], which also maintains the Voronoi sets, can be shown to take $O(|X| \log \Delta)$-time [40], and has been implemented [24, 51].

As mentioned, at every phase of the algorithm, each point $x \in X$ is assigned a Voronoi set of a point $x_j \in S_i$: its nearest neighbor among those in S_i. We call the sequence of such points x_j, as the algorithm progresses, the *sequence of leaders* of a point x. Such a sequence necessarily starts with x_1 and ends with the point itself.

From the sequence of leaders we can obtain the covering map. The covering sequence of a point is a subsequence of its sequence of leaders, in reverse order. To see this, note that the covering sequence of x starts with x itself, as in the reversed sequence of leaders. Then, when the sparse ball of x disappears the next point in its covering sequence is its nearest neighbor among those points with non-slowed sparse ball. This set of points with non-slowed sparse ball is necessarily equal to one of the subsets S_i we have encountered through the execution of the algorithm. In addition, balls are slowed in the same order as the reversed sequence of leaders. All in all, we conclude the following:

▶ **Lemma 19.** *A persistent net, the insertion radius of all the points, and the covering map can be computed in time $O(|X| \log \Delta)$.*

5.2 Neighborhoods

We also need to find those subsets of points whose sparse balls have a non-empty intersection, at any scale. By Lemma 16, any such subset A is contained in a ball of radius $2(1+3\varepsilon)\operatorname{slow}(x)$ around x, where x is the point of minimal insertion point in A. Making use of this observation, and following [12], given a point $x \in X$, we call the points of higher insertion radius than the one of x and within distance $2(1+3\varepsilon)\operatorname{slow}(x)$ its *friends*. Note that the number of friends of a point is $O(1)$, because we are only considering points of higher insertion radius, as in Lemma 17. Once we have computed the friends of x, we can go over every possible subset and check whether the associated sparse balls intersect at some scale, which is done in the following section. This procedure is similar to other sparsification schemes [12, 16].

To compute the friends of every point, we can use a proximity search data structure as follows. First, we initialize it empty. In decreasing order of insertion radius (that is, the order given by the greedy permutation), we add the points one by one. After adding a point x, we query all the points within $2(1+3\varepsilon)\operatorname{slow}(x)$ distance from x.

Cavanna, Jahanseir and Sheehy [16] gives such a data structure to compute all friends in such a way in time $O(|X|)$, again for fixed ε and d. In practice one can also use the related greedy trees [22], which have been implemented [51].

▶ **Lemma 20.** *Computing the friends of all points can be done in $O(|X|)$ time.*

5.3 Intersection of sparse balls

We now show how to compute the scale at which a subset of sparse balls first intersect, if they do so. We do it only for the Euclidean case; the l^∞-norm, which is also of interest, is an easier case, since it is enough to compute the intersection times pairwise. We show that this problem is of LP-type [45] and can be solved efficiently following the blueprint of the LP-algorithm for computing the smallest enclosing ball of balls [35], which is efficient in practice and implemented in CGAL [36].

A specific radius function. Let us choose a specific radius function first, with the added restriction that $0 < \varepsilon \leq 1$. For a point $x \in X$, we take

$$\rho_x(r) = \begin{cases} r & r \leq \operatorname{slow}(x), \\ \sqrt{K_\varepsilon r^2 + (1-K_\varepsilon)\operatorname{slow}(x)^2} & \operatorname{slow}(x) \leq r, \end{cases} \tag{1}$$

where, $K_\varepsilon = \frac{1}{3(1+\varepsilon)^2}$. See Figure 3. It can be checked that $\rho_x(r)$ is indeed a radius function.

Solving a problem. Equipped with the radius function above and given a subset of points $A \subset X$, we consider the problem

$$\begin{aligned} \min_{z\in\mathbb{R}^d, r\geq 0} \quad & r^2, & \\ \text{subject to} \quad & \|a-z\|^2 \leq r^2, & a \in A, \\ & \|a-z\|^2 \leq K_\varepsilon r^2 + (1-K_\varepsilon)\operatorname{slow}(x)^2, & a \in A. \end{aligned} \tag{2}$$

Now it is apparent why we chose the specific radius function of Equation (1): the right side of the inequalities above is linear on r^2. We claim that the solution of this problem gives the minimum scale at which the sparse balls of A have non-empty intersection, if such a scale exists. Let $z_{\text{opt}} \in \mathbb{R}^d$ and $r_{\text{opt}} \geq 0$ be the solution of the problem above (a

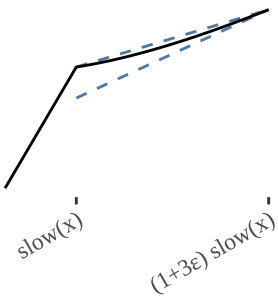


Figure 3 Graph of $\rho_x(r)$ of (1), for some values $\text{slow}(x)$ and $0 < \varepsilon < 1$. The dashed lines are the functions $L_x(r)$ and $U_x(r)$, as in Definition 1.

unique solution always exists, see the full version). One can check that r_{opt} is the minimum scale at which the balls $B(a, \rho_a(r))$ around points $a \in X$ first intersect, by noting that $r^2 \leq K_\varepsilon r^2 + (1 - K_\varepsilon)\,\text{slow}(x)^2$ whenever $r \leq \text{slow}(x)$. Thus, r_{opt} is the minimum scale at which the sparse balls of A intersect, unless a sparse ball disappears before then, that is, unless $r_{\text{opt}} > (1 + 3\varepsilon)\,\text{slow}(a)$ for any $a \in A$.

We solve the problem of Equation (2) via a more general problem. Given a set H of m constraints, each being a triple consisting of a point $p_i \in \mathbb{R}^d$, a positive real constant $\alpha_i \in \mathbb{R}$ and a non-negative real constant $\beta_i \in \mathbb{R}$, we look for

$$\begin{aligned} &\min_{z \in \mathbb{R}^d, s \geq 0} \quad s, \\ &\text{subject to} \quad \|p_i - z\|^2 \leq \alpha_i s + \beta_i, \quad i = 1, \ldots, m. \end{aligned} \tag{M}$$

The problem of Equation (2) is an instance of (M) with $2|A|$ constraints.

Note that by taking a set of points $p_1, \ldots, p_m$ and setting $\alpha_i = 1$ and $\beta_i = 0$, the solution to (M) above is the smallest enclosing ball of the points (with radius $\sqrt{s}$). As already hinted, the smallest enclosing ball can be computed as a LP-type problem [45]. In the full version, we show how to solve (M) as a LP-type problem following the strategy of the smallest enclosing ball of balls [35, 58].

6 Discussion

We have introduced a $(1 + \varepsilon)$-approximation of the multicover that is of linear size, which is optimal, for fixed dimension d and error ε. As it is usual in sparsification and similar approximations, such a bound on the size (and computation) hides constants that have an exponential dependency on d. The computation time is dominated by the computation of the greedy permutation, which is $O(|X| \log \Delta)$, where Δ is the spread. We remark that one can compute a $(1 + \frac{1}{\text{poly}(|X|)})$-approximate greedy permutation in $O(|X| \log |X|)$ time [21, 40].

Although so far we have focused on the multicover (and subdivision) bifiltration in $\mathbb{R}^d$, the sparsification scheme works in general metric spaces, where one takes the subdivision Rips bifiltration. Indeed, and as expanded upon in the full version, the construction presented here, for the same reasons as the construction of Cavanna, Jahanseir and Sheehy [16] that inspired it, does not depend on the chosen norm, and the size bounds can be seen to extend to spaces with bounded doubling dimension. Thus, we can isometrically embed any finite metric space X into $(\mathbb{R}^{|X|}, l^\infty)$, where the Rips complex is equal to the Čech complex, and take the sparse multicover there.

References

1 Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-Based Filtrations. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry, SoCG 2019*, volume 129 of *LIPIcs*, pages 58:1–58:15, 2019. doi:10.4230/LIPIcs.SoCG.2019.58.

2 Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-Based Filtrations. In Nils A. Baas, Gunnar E. Carlsson, Gereon Quick, Markus Szymik, and Marius Thaule, editors, *Topological Data Analysis*, pages 33–66. Springer, 2020. doi:10.1007/978-3-030-43408-3_2.

3 Dominique Attali, André Lieutier, and David Salinas. Vietoris-Rips complexes also provide topologically correct reconstructions of sampled shapes. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, SoCG 2011*, pages 491–500, June 2011. doi:10.1145/1998196.1998276.

4 Dominique Attali, André Lieutier, and David Salinas. Vietoris–Rips complexes also provide topologically correct reconstructions of sampled shapes. *Computational Geometry*, 46(4):448–465, May 2013. doi:10.1016/j.comgeo.2012.02.009.

5 Andrew J. Blumberg, Itamar Gal, Michael A. Mandell, and Matthew Pancia. Robust Statistics, Hypothesis Testing, and Confidence Intervals for Persistent Homology on Metric Measure Spaces. *Foundations of Computational Mathematics*, 14:745–789, August 2014. doi:10.1007/s10208-014-9201-4.

6 Andrew J. Blumberg and Michael Lesnick. Universality of the homotopy interleaving distance. *Transactions of the American Mathematical Society*, 376:8269–8307, 2023. doi:10.1090/tran/8738.

7 Andrew J. Blumberg and Michael Lesnick. Stability of 2-Parameter Persistent Homology. *Foundations of Computational Mathematics*, 24:385–427, 2024. doi:10.1007/s10208-022-09576-6.

8 Omer Bobrowski, Sayan Mukherjee, and Jonathan E. Taylor. Topological consistency via kernel estimation. *Bernoulli*, 23(1):288–328, February 2017. doi:10.3150/15-BEJ744.

9 Mickaël Buchet, Bianca B. Dornelas, and Michael Kerber. Sparse Higher Order Čech Filtrations. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023*, volume 258 of *LIPIcs*, pages 20:1–20:17, 2023. doi:10.4230/LIPIcs.SoCG.2023.20.

10 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 168–180, January 2015. doi:10.1137/1.9781611973730.13.

11 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. *Computational Geometry*, 58:70–96, October 2016. doi:10.1016/j.comgeo.2016.07.001.

12 Mickaël Buchet, Bianca B. Dornelas, and Michael Kerber. Sparse Higher Order Čech Filtrations. *Journal of the ACM*, 71(4):1–23, August 2024. doi:10.1145/3666085.

13 Gunnar Carlsson and Afra Zomorodian. The Theory of Multidimensional Persistence. *Discrete & Computational Geometry*, 42(1):71–93, July 2009. doi:10.1007/s00454-009-9176-0.

14 Nicholas J. Cavanna, Kirk P. Gardner, and Donald R. Sheehy. When and Why the Topological Coverage Criterion Works. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2679–2690, January 2017. doi:10.1137/1.9781611974782.177.

15 Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. Visualizing Sparse Filtrations. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015*, volume 34, pages 23–25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICS.SOCG.2015.23.

16 Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. A Geometric Perspective on Sparse Filtrations. In *Proceedings of the 27th Canadian Conference on Computational Geometry*, pages 116–121, 2015. arXiv:1506.03797.

17 Mark E. Cawood and P. M. Dearing. The weighted Euclidean one-center problem in R^n. *Computational Optimization and Applications*, August 2024. doi:10.1007/s10589-024-00599-z.

18 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric Inference for Probability Measures. *Foundations of Computational Mathematics*, 11(6):733–751, December 2011. doi:10.1007/s10208-011-9098-0.

19 Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Robust topological inference: Distance to a measure and kernel distance. *J. Mach. Learn. Res.*, 18(1):5845–5884, January 2017.

20 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-Based Clustering in Riemannian Manifolds. *Journal of the ACM*, 60(6):1–38, November 2013. doi:10.1145/2535927.

21 Oliver Chubet, Don Sheehy, and Siddharth Sheth. Simple Construction of Greedy Trees and Greedy Permutations, December 2024. doi:10.48550/arXiv.2412.02554.

22 Oliver A. Chubet, Parth Parikh, Donald R. Sheehy, and Siddharth Sheth. Proximity Search in the Greedy Tree. In *2023 Symposium on Simplicity in Algorithms (SOSA)*, January 2023. doi:10.1137/1.9781611977585.

23 Kenneth L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *24th Annual Symposium on Foundations of Computer Science*, pages 226–232, November 1983. doi:10.1109/SFCS.1983.16.

24 Kenneth L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for sb(S), 2003. URL: https://kenclarkson.org/Msb/white_paper.pdf.

25 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Proceedings of the Twenty-First Annual Symposium on Computational Geometry, SoCG 2005*, SoCG '05, pages 263–271, June 2005. doi:10.1145/1064092.1064133.

26 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of Persistence Diagrams. *Discrete & Computational Geometry*, 37(1):103–120, January 2007. doi:10.1007/s00454-006-1276-5.

27 René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. Computing the multicover bifiltration. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 27:1–27:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.27.

28 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.

29 P. M. Dearing and Mark E. Cawood. The minimum covering Euclidean ball of a set of Euclidean balls in R^n. *Annals of Operations Research*, 322(2):631–659, March 2023. doi:10.1007/s10479-022-05138-9.

30 M.E Dyer and A.M Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, February 1985. doi:10.1016/0167-6377(85)90002-1.

31 Herbert Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13:415–440, 1995. doi:10.1007/BF02574053.

32 Herbert Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, Providence, R.I, 2010.

33 Herbert Edelsbrunner and Georg Osang. The Multi-cover Persistence of Euclidean Balls. In *34th International Symposium on Computational Geometry, SoCG 2018*, LIPIcs, page 14 pages, 2018. doi:10.4230/LIPICS.SOCG.2018.34.

34 Herbert Edelsbrunner and Georg Osang. The Multi-Cover Persistence of Euclidean Balls. *Discrete & Computational Geometry*, 65(4):1296–1313, June 2021. doi:10.1007/s00454-021-00281-9.

35 Kaspar Fischer and Bernd Gärtner. The smallest enclosing ball of balls: Combinatorial structure and algorithms. *International Journal of Computational Geometry & Applications*, 14(04n05):341–378, October 2004. doi:10.1142/S0218195904001500.

36 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`.

37 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`.

38 Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-Distance. *Discrete & Computational Geometry*, 49(1):22–45, January 2013. `doi:10.1007/s00454-012-9465-x`.

39 Leonidas J. Guibas, Quentin Mérigot, and Dmitriy Morozov. Witnessed k-distance. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, SoCG 2011*, pages 57–64, June 2011. `doi:10.1145/1998196.1998205`.

40 Sariel Har-Peled and Manor Mendel. Fast Construction of Nets in Low Dimensional Metrics, and Their Applications. *SIAM Journal on Computing*, 35(5):1148–1184, January 2006. `doi:10.1137/S0097539704446281`.

41 Niklas Hellmer and Jan Spaliński. Density Sensitive Bifiltered Dowker Complexes via Total Weight, September 2024. `arXiv:2405.15592`.

42 Michael Lesnick and Ken McCabe. Nerve Models of Subdivision Bifiltrations, June 2024. `doi:10.48550/arXiv.2406.07679`.

43 Michael Lesnick and Kenneth McCabe. Sparse Approximation of the Subdivision-Rips Bifiltration for Doubling Metrics, August 2024. `doi:10.48550/arXiv.2408.16716`.

44 Michael Lesnick and Matthew Wright. Interactive Visualization of 2-D Persistence Modules, December 2015. `arXiv:1512.00180`.

45 Jiří Matoušek, Micha Sharir, and Emo Welzl. A Subexponential Bound for Linear Programming. *Algorithmica*, 16:498–516, 1996. `doi:10.1007/BF01940877`.

46 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`.

47 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geometric Inference on Kernel Density Estimates. In *31st International Symposium on Computational Geometry, SoCG 2015*, volume 34, pages 857–871. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. `doi:10.4230/LIPICS.SOCG.2015.857`.

48 Alexander Rolle and Luis Scoccola. Stable and Consistent Density-Based Clustering via Multiparameter Persistence. *Journal of Machine Learning Research*, 25(258):1–74, 2024. URL: `http://jmlr.org/papers/v25/21-1185.html`.

49 Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. `doi:10.1137/0206041`.

50 Luis Scoccola and Alexander Rolle. Persistable: Persistent and stable clustering. *Journal of Open Source Software*, 8(83):5022, March 2023. `doi:10.21105/joss.05022`.

51 Donald R. Sheehy. Greedypermutations. URL: `https://github.com/donsheehy/greedypermutation`.

52 Donald R. Sheehy. Linear-size approximations to the Vietoris-Rips filtration. In *Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry, SoCG 2012*, pages 239–248, June 2012. `doi:10.1145/2261250.2261286`.

53 Donald R. Sheehy. A Multicover Nerve for Geometric Inference. In *Proceedings of the 24th Canadian Conference on Computational Geometry*, page 5, 2012. URL: `https://donsheehy.net/research/sheehy12multicover.pdf`.

54 Donald R. Sheehy. Linear-Size Approximations to the Vietoris–Rips Filtration. *Discrete & Computational Geometry*, 49(4):778–796, June 2013. `doi:10.1007/s00454-013-9513-1`.

55 Donald R. Sheehy. A Sparse Delaunay Filtration. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189, pages 58:1–58:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.SOCG.2021.58`.

56 J. J. Sylvester. A question on the geometry of situation. *Quarterly Journal of Pure and Applied Mathematics*, 1:79, 1857. URL: http://resolver.sub.uni-goettingen.de/purl?PPN600494829_0001.

57 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.

58 Samuel Zürcher. Smallest Enclosing Ball for a Point Set with Strictly Convex Level Sets. Master's thesis, ETH Zurich, Institute of Theoretical Computer Science, 2007. URL: https://samzurcher.ch/assets/pdf/master-thesis.pdf.

Single-Source Shortest Path Problem in Weighted Disk Graphs

Shinwoo An
Pohang University of Science and Technology, South Korea

Eunjin Oh
Pohang University of Science and Technology, South Korea

Jie Xue
New York University Shanghai, China

Abstract

In this paper, we present efficient algorithms for the single-source shortest path problem in weighted disk graphs. A disk graph is the intersection graph of a family of disks in the plane. Here, the weight of an edge is defined as the Euclidean distance between the centers of the disks corresponding to the endpoints of the edge. Given a family of n disks in the plane whose radii lie in $[1, \Psi]$ and a source disk, we can compute a shortest path tree from a source vertex in the weighted disk graph in $O(n \log^2 n \log \Psi)$ time. Moreover, in the case that the radii of disks are arbitrarily large, we can compute a shortest path tree from a source vertex in the weighted disk graph in $O(n \log^4 n)$ time. This improves the best-known algorithm running in $O(n \log^6 n)$ time presented in ESA'23.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Disk graphs, shortest path problem, compressed quadtrees

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.7

Related Version *Full Version*: https://arxiv.org/abs/2504.06534

Funding The work by Ah and Oh was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00440239, Sublinear Scalable Algorithms for Large-Scale Data Analysis) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00358505).

1 Introduction

Geometric intersection graphs are a fundamental class of graphs representing spatial relationships among geometric objects. In this paper, we focus on the intersection graph of disks in the plane. More formally, for a set P of n points in the plane, where each point $v \in P$ has an associated radius r_v, the *disk graph* G of P is defined as the graph where each vertex corresponds to a point P, and two vertices u and v of P are connected by an edge if and only if the disks with center u, v and radii r_u, r_v intersect. For an *edge-weighted* disk graph, the weight of an edge uv is defined as $|uv|$. In the special case that all radii are the same, the disk graph is also called a *unit disk graph*. Disk graphs can be used as a model for broadcast networks: The disks of P represent transmitter-receiver stations with transmission power. One can view a broadcast range of a transmitter as a disk.

In this paper, we consider the *single-source shortest path* (SSSP) problem for edge-weighted disk graphs: Given a set P of points associated with radii and a specified point $s \in P$, compute a shortest path tree of the edge-weighted disk graph of P rooted at s. One straightforward way to deal with a disk graph is to construct the disk graph explicitly, and then run algorithms designed for general graphs. However, a disk graph might have complexity $\Theta(n^2)$ in the worst case even though it can be (implicitly) represented as n disks.

© Shinwoo An, Eunjin Oh, and Jie Xue;
licensed under Creative Commons License CC-BY 4.0

Therefore, it is natural to seek faster algorithms for a disk graph implicitly represented as its underlying set of disks. Besides the SSSP problem, many graph-theoretic problems have much more efficient solutions in disk graphs than in general graphs [1, 2, 4, 7, 10, 13, 15, 17].

Related work. As the single-source shortest path problem is fundamental in computer science, there are numerous work on this problem and its variant for unit disk graphs [5, 6, 8, 9, 11, 14, 16, 20]. In the case of unweighted unit disk graphs where all edge weights are one, the SSSP problem can be solved in $O(n \log n)$ time, and this is optimal [6, 8]. For edge-weighted unit disk graphs, the best known exact algorithm for the SSSP problem takes $O(n\frac{\log^2 n}{\log\log n})$ time [5], and the best known $(1+\varepsilon)$-approximation algorithm takes $O(n \log n + n \log^2(\frac{1}{\varepsilon}))$ time [20]. For general disk graphs, the best known exact algorithms for the unweighted and weighted SSSP problem takes $O(n \log^2 n)$ and $O(n \log^6 n)$ time, respectively [13, 15].

Another variant of the problem is the reverse shortest path problem, where the input is an edge-weighted graph G, a start vertex, a target vertex, and a threshold value. The problem is to find a minimum value r such that the length of the shortest path from the start vertex to the target vertex in G_r is at most the threshold. Here, G_r is the subgraph of G consisting of edges whose weights are at most r. The problem was considered in various metrics in the weighted and unweighted cases. The best-known algorithms for both weighted and unweighted unit disk graph metric take $O^*(n^{6/5})$ randomized time [13, 21]. In addition, the best-known algorithm for weighted disk graph metric takes $O^*(n^{5/4})$ randomized time [13].

Our results. In this paper, we present two algorithms for the SSSP problem for edge-weighted disk graphs with n vertices. The first algorithm runs in $O(n \log^2 n \log \Psi)$ time, where Ψ is the maximum radius ratio of P. The second algorithm runs in $O(n \log^4 n)$ time. This improves the best-known algorithm for this problem running in $O(n \log^6 n)$ time [13].

Model of computation. In Section 3, our algorithm uses a compressed quadtree. To implement this efficiently, we need to extend the real RAM model [18] by an *floor function* that rounds a real number down to the next integer [12]. While the floor function is generally considered too powerful, it is widely regarded as reasonable for tasks such as finding the cell of a given level of the grid that contains a given point in constant time [12]. On the other hand, our algorithm in Section 2 operates within the standard real RAM model.

Preliminaries. Throughout this paper, we let P be a set of n points in the plane, where each point $v \in P$ has an associated radius r_v, and G be the edge-weighted disk graph of P. We interchangeably denote $v \in P$ as a point or as a vertex of G if it is clear from the context. Also, we let s be a source vertex of P. We assume that $1 \leq r_v$ for all $v \in P$, and thus in the case of *bounded radius ratio*, all radii must come from the range $[1, \Psi]$ for a constant Ψ. The *length* of a path of G is defined as the sum of the weights of the edges contained in the path. The *distance* between u and v is defined as the minimum length among all u-v paths of G. For a vertex $v \in G$, we simply let $d(v)$ denote the distance between s and v.

2 SSSP on Disk Graphs of Bounded Radius Ratio Ψ

In this section, we describe our algorithm for the SSSP problem on disk graphs. Given a set P of n points with associated radii in $[1, \Psi]$, our goal is to compute a shortest path tree from a specified source vertex s in the edge-weighted disk graph G of P in $O(n \log^2 n \log \Psi)$ time. More precisely, our goal is to compute $\mathrm{dist}(v)$ and $\mathrm{prev}(v)$ for all vertices $v \in P$ such that $\mathrm{dist}(v) = d(v)$, and $\mathrm{prev}(v)$ is the predecessor of t in the shortest s-t path.

2.1 Sketch of Our Algorithm

We review the well-known Dijkstra's algorithm that computes a shortest path tree from a source vertex. Initially, the algorithm sets all dist-values to infinity except a source vertex, and sets $Q = P$. The algorithm sequentially applies the following steps until Q is empty.

- **(D1)** Pick the vertex u with smallest dist-value.
- **(D2)** For all neighbors $v \in Q$ of u, update $\mathrm{dist}(v) \leftarrow \min\{\mathrm{dist}(v), \mathrm{dist}(u) + |uv|\}$.
- **(D3)** Remove u from Q.

In the worst case, Dijkstra's algorithm takes quadratic time since a disk graph can have $\Theta(n^2)$ edges. In our algorithm, we simultaneously update the dist-values of several vertices by the UPDATE subroutine that was introduced in [20]. For any two vertex set U and V of G, UPDATE(U, V) does the following: For all $v \in V$,

- **(U1)** Compute $u := \arg\min\{\mathrm{dist}(u) + |uv|\}$ among all $u \in U$ s.t. uv is an edge of G.
- **(U2)** Update $\mathrm{dist}(v) \leftarrow \min\{\mathrm{dist}(v), \mathrm{dist}(u) + |uv|\}$.

That is, the subroutine updates the dist-values of all vertices of V using the dist-values of the vertices of U. The subroutine gets input U and V from a *hierarchical grid*. For each integer $0 \le i \le \lceil \log \Psi \rceil$, we use Γ_i to denote a *grid* of level i, which consists of axis-parallel square cells of diameter 2^i. Then we use Γ to denote the union of grid cells of Γ_i's. For a grid cell $c \in \Gamma$, we use $p(c)$ to denote the center of c, use $|c|$ to denote the diameter of c, and use $\square_c$ to denote the axis-parallel square of diameter $69|c|$ centered at $p(c)$. We use $\boxplus_c$ to denote the set of grid cells of Γ_{i-1}, Γ_i and Γ_{i+1} which are contained in $\square_c$. We say $c' \in \Gamma_{i-1}$ a *child cell* of c if c' is nested in c. Throughout the paper, we assume that no points of P lie on the boundary of any grid cell. We define two sets of vertices contained in c as follows.

$$P_{\mathrm{mid}}(c) := \{v \in P \mid v \in c, r_v \in [8|c|, 16|c|)\};\ P_{\mathrm{small}}(c) := \{v \in P \mid v \in c, r_v \in [1, 8|c|)\}.$$

Intuitively, $P_{\mathrm{mid}}(c)$ consists of vertices contained in c with radii *similar* to $|c|$ while $P_{\mathrm{small}}(c)$ consists of vertices with radii *smaller* than that of $P_{\mathrm{mid}}(c)$. Note that $P_{\mathrm{mid}}(c)$ forms a clique in G. For a vertex v in P, we let c_v be the (unique) cell such that $P_{\mathrm{mid}}(c_v)$ contains v.

Regular edge and Irregular edge. Let uv be an edge of G with $r_u \le r_v$. We call uv a *regular edge* if $\frac{r_v}{r_u} < 2$, and an *irregular edge* if $\frac{r_v}{r_u} \ge 2$. We can identify all edges of G efficiently by using $\boxplus_c$, $P_{\mathrm{mid}}(c)$ and $P_{\mathrm{small}}(c)$.

▶ **Lemma 1.** *Let uv be a regular edge. Then $c_u \in \boxplus_{c_v}$. Symmetrically, $c_v \in \boxplus_{c_u}$.*

▶ **Lemma 2.** *Let uv be an edge and $2r_u \le r_v$. There is a cell $c \in \boxplus_{c_v}$ such that $u \in P_{\mathrm{small}}(c)$.*

Basic strategy of [20]. We utilize the strategic adaptation of Dijkstra's algorithm in a cell-by-cell manner with $P_{\mathrm{small}}(\cdot)$ and $P_{\mathrm{mid}}(\cdot)$. This strategy was used in [20] to design an efficient algorithm for the weighted SSSP problem on unit disk graphs. For concreteness, we describe the details of the strategy below. Let v be the vertex with the smallest dist-value among all vertices not processed so far. As an invariant, we shall guarantee that all vertices of G which have been processed have correct dist-values, and v has a correct dist-value if its predecessor in the shortest s-v path has been processed. We want to process not only v but also all vertices in $P_{\mathrm{mid}}(c_v)$ simultaneously. However, notice that, at this moment, the vertices in $P_{\mathrm{mid}}(c_v)$ do not necessarily have correct dist-values.

Thus, as a first step, we compute the correct dist-values for all vertices in $P_{\mathrm{mid}}(c_v)$. If the predecessor w of a vertex $v' \in P_{\mathrm{mid}}(c_v)$ in the shortest s-v' path already has the correct dist-value, we can simply update the dist-values of $P_{\mathrm{mid}}(c_v)$ by considering all edges whose one endpoint is in $P_{\mathrm{mid}}(c_v)$. However, it is possible that w has not been processed yet so far and does not have the correct dist-value. Fortunately, even in this case, we can show that w has a correct dist-value. To see this, let u be the predecessor of w in the shortest s-w path. Then uv' cannot be an edge of the graph since otherwise u is the predecessor of v' in the shortest s-v' path. Therefore, $|uv'| > r_{v'}$. As $v, v' \in P_{\mathrm{mid}}(c_v)$, $|uv'| > r_{v'} > |vv'|$. Hence, $d(u) = d(v') - (|v'w| + |wu|) \leq d(v') - |v'u| < d(v') - |vv'| \leq d(v)$. The last inequality follows since the concatenation of vv' and the shortest s-v path is longer than the shortest s-v' path. Now u must have been processed since v is the vertex with the smallest dist-value among all vertices not processed so far. Due to the invariant, w must have the correct dist-value. Consequently, we can update the dist-values of all vertices of $P_{\mathrm{mid}}(c_v)$ by applying $\textsc{Update}(N(c_v), P_{\mathrm{mid}}(c_v))$ where $N(c_v)$ denotes the set of neighbors of $P_{\mathrm{mid}}(c_v)$ in G.

The second step is to transmit the correct dist-values of $P_{\mathrm{mid}}(c_v)$ into their neighbors in order to satisfy the second invariant. Lastly, we remove $P_{\mathrm{mid}}(c_v)$ from the graph.

Main obstacles and lazy update scheme. The complexity of this strategy primarily depends on the cost of searching all neighbors of $P_{\mathrm{mid}}(c_v)$. This was not a big deal in [20], as in a unit disk graph, each cell interacts with only a constant number of other cells. In the case of disk graphs, however, a vertex with radius $\Theta(\Psi)$ can be adjacent to vertices in $P_{\mathrm{mid}}(c)$ for $O(\Psi^2)$ distinct grid cells c. To avoid polynomial dependency on Ψ, we handle regular edges and irregular edges separately. More specifically, we can search all regular edges by considering $P_{\mathrm{mid}}(c)$ for all $c \in \boxplus_{c_v}$ by Lemma 1. Since $\boxplus_{c_v}$ consists of $O(1)$ cells, we can avoid the polynomial dependency on Ψ through the appropriate use of the Update subroutine.

However, the same approach cannot be used to search all irregular edges as up to $\Theta(\Psi^2)$ sets of $P_{\mathrm{mid}}(\cdot)$ may interact with $P_{\mathrm{mid}}(c_v)$ in the worst case. We address this issue with a novel approach, which we call *lazy update scheme*. We say w is a *small neighbor* of a vertex u if uw is an irregular edge and $r_u > r_w$. Furthermore, we say w is a *small neighbor* of a cell c if there is a vertex $u \in P_{\mathrm{mid}}(c)$ forming an irregular edge with w and $r_u > r_w$. In the basic strategy, we transmit the correct dist-value of v to *all* neighbors whenever we process v. In the lazy update scheme, we postpone the transmission to $P_{\mathrm{mid}}(c)$ for all cells c such that v is a *small* neighbor of c. We handle several postponed update requests at once at some point. More specifically, let V_x be the set of small neighbors of c whose dist-values are in the range $[x, x + 2|c|]$. We transmit the dist-values of V_x into $P_{\mathrm{mid}}(c)$ right after all vertices of V_x have been processed. Due to the following lemma, we can bound the number of lazy updates into $O(1)$ for each cell c, and this enables us to avoid the polynomial dependency on Ψ.

▶ **Lemma 3.** *Let u and u' be small neighbors of c. Then $|d(u) - d(u')| \leq 65|c|$.*

However, this may cause an *inconsistency issue* during the first step. When we process the vertices of $P_{\mathrm{mid}}(c_v)$, now the first update does not transmit the dist-values of the small neighbors of c_v. If a predecessor w of $v' \in P_{\mathrm{mid}}(c_v)$ in the shortest s-v' path is a small neighbor of v', it is possible that the lazy update has not occurred even though w has already been processed. We show that this cannot happen using the following geometric lemma.

▶ **Lemma 4.** *Let $u \neq s$ be the predecessor of v in the shortest s-v path. Then $|uv| \geq |r_v - r_u|$ unless $r_v < r_u$ and v is a leaf in the shortest path tree.*

By this lemma, $d(w) + 2|c_{v'}| < d(w) + \frac{1}{4}r_{v'} < d(w) + |r_w - r_{v'}| < d(w) + |wv'| = d(v')$. Subsequently, roughly speaking, the lazy update that transmits dist-values of w into v' occurs in advance when we process the vertices of $P_{\mathrm{mid}}(c_{v'})$.

2.2 Algorithm

We present our algorithm for the SSSP problem on disk graphs with radius ratio Ψ. The goal is to compute $d(v)$ for all $v \in P$. Initially, we set $\mathrm{dist}(v)$, the *dist-value* of v, as infinity for all vertices other than a source vertex s, and set $\mathrm{dist}(s) = 0$. Eventually, the algorithm will modify $\mathrm{dist}(v)$ into $d(v)$ for all $v \in P$. In addition, for each nonempty $P_{\mathrm{mid}}(c)$ for $c \in \Gamma$, we maintain a value $\mathrm{alarm}(c)$ initialized to ∞. These alarm values will take care of the moment of the lazy update. We set R by the set of points of P that has not been processed yet.

Pre-processing. We compute $d(v)$ for all $v \in P$ adjacent to s in G and set $\mathrm{dist}(v)$ to $|sv|$. Furthermore, for each grid cell c, let $\bar{L}(c)$ be the set of grid cells c' where $P_{\mathrm{mid}}(c)$ contains a small neighbor of c'. For technical reasons, we compute a superset $L(c)$ of $\bar{L}(c)$ which contains all cells c' with $|c| < |c'|$ such that there is an edge between a vertex of $P_{\mathrm{mid}}(c)$ and a vertex of $P_{\mathrm{mid}}(c')$. Furthermore, we set $L(c_s)$ as an empty set where c_s is the grid cell such that the source s is contained in $P_{\mathrm{mid}}(c_s)$. We will use the information of $L(c)$ to deal with the lazy update. Then the algorithm consists of several rounds. In each round, we check $\mathrm{dist}(v)$ for all $v \in R$ and $\mathrm{alarm}(c)$ for all $c \in \Gamma$. Subsequently, we find the minimum value k among these values, and proceed depending on the type of k. The algorithm terminates when R becomes empty. We utilize $\textsc{Update}(U, V)$ subroutine for both cases.

Algorithm 1 $\textsc{Update}(U, V)$.

1. **for** $v \in V, u \in U$ **do**
2. **if** uv is an edge of G **then**
3. **if** $\mathrm{dist}(v) > \mathrm{dist}(u) + |uv|$ **then**
4. $\mathrm{dist}(v) \leftarrow \mathrm{dist}(u) + |uv|$; $\mathrm{prev}(v) \leftarrow u$

Intuitively, if $u \in U$ has correct dist-values and u is the predecessor of $v \in V$ in the shortest s-t path, then all such v gets the correct dist-values after the subroutine $\textsc{Update}(U, V)$. The rest of this section is devoted to giving detailed instructions on case studies of k.

Algorithm 2 $\mathrm{SSSP}(P)$.

1. $R \leftarrow P$
2. **while** $R \neq \emptyset$ **do**
3. $k \leftarrow \min(\{\mathrm{dist}(v) : v \in R\} \cup \{\mathrm{alarm}(c) : c \in \Gamma\})$
4. **if** $k = \mathrm{dist}(v)$ for $v \in R$ **then**
5. $\textsc{Update}(\bigcup_{c \in \boxplus_{c_v}} P_{\mathrm{mid}}(c), P_{\mathrm{mid}}(c_v))$
6. $\textsc{Update}(P_{\mathrm{mid}}(c_v), \bigcup_{c \in \boxplus_{c_v}} P_{\mathrm{mid}}(c) \cup P_{\mathrm{small}}(c))$
7. **for** $c \in L(c_v)$ **do**
8. **if** $\mathrm{alarm}(c) = \infty$ **then**
9. $\mathrm{alarm}(c) \leftarrow \mathrm{dist}(v) + 2|c|$
10. $R \leftarrow R \setminus P_{\mathrm{mid}}(c_v)$
11. **if** $k = \mathrm{alarm}(c)$ for $c \in \Gamma$ **then**
12. $\textsc{Update}(\bigcup_{c' \in \boxplus_c} P_{\mathrm{small}}(c'), P_{\mathrm{mid}}(c))$
13. $\mathrm{alarm}(c) \leftarrow \infty$

Case 1: $k = \mathrm{dist}(v)$ for a vertex $v \in R$. In this case, we first apply the subroutine $\textsc{Update}(\bigcup_{c \in \boxplus_{c_v}} P_{\mathrm{mid}}(c), P_{\mathrm{mid}}(c_v))$. Interestingly, we will see later that after this, all vertices in $P_{\mathrm{mid}}(c_v)$ have the correct dist-values. Then using the correct dist-values of $P_{\mathrm{mid}}(c_v)$, we update the dist-values of the neighbors of the vertices in $P_{\mathrm{mid}}(c_v)$. This is done by applying $\textsc{Update}(P_{\mathrm{mid}}(c_v), \bigcup_{c \in \boxplus_{c_v}} P_{\mathrm{mid}}(c) \cup P_{\mathrm{small}}(c))$ and setting $\mathrm{alarm}(c) = \mathrm{dist}(v) + 2|c|$ for all $c \in L(c_v)$ with $\mathrm{alarm}(c) = \infty$, where the former takes care of the neighbors of the vertices in $P_{\mathrm{mid}}(c_v)$ connected by regular edges and the small neighbors of c_v, and the latter takes care of the other neighbors. Finally, we remove $P_{\mathrm{mid}}(c_v)$ from R.

Case 2: $k = \mathrm{alarm}(c)$ for a grid cell $c \in \Gamma$. In this case, we shall correct the dist-values of all $v \in P_{\mathrm{mid}}(c)$ such that the predecessor of v is a small neighbor of v. This is done by applying $\textsc{Update}(\bigcup_{c' \in \boxplus_c} P_{\mathrm{small}}(c'), P_{\mathrm{mid}}(c))$. After this, we reset $\mathrm{alarm}(c)$ to ∞.

2.3 Correctness

In this section, we show that the algorithm correctly computes a shortest path tree. For a vertex v, we use $\mathrm{prev}(v)$ to denote the predecessor of v in the shortest path tree. We shall maintain a simple invariant during the algorithm: $d(v) \leq \mathrm{dist}(v)$ for every $v \in P$. The following lemma is the simple corollary of Lemma 4.

▶ **Corollary 5.** *Suppose the shortest s-v path contains an irregular edge uv and $u \neq s$. Then $|uv| \geq \frac{1}{2} r_v$ unless $r_u > r_v$ and v is a leaf on the shortest path tree.*

Then we prove the key invariant of our algorithm.

▶ **Lemma 6.** *The following statements hold during the execution of Algorithm 2.*
(1) *At the onset of line 5,* $\mathrm{dist}(v) = d(v)$.
(2) *After line 5,* $\mathrm{dist}(u) = d(u)$ *if* $u \in P_{\mathrm{mid}}(c_v)$ *is not a leaf on the shortest path tree.*
(3) *After line 12,* $\mathrm{dist}(u) = d(u)$ *if* $u \in P_{\mathrm{mid}}(c)$, $\mathrm{prev}(u) \in \bigcup_{c' \in \boxplus_c} P_{\mathrm{small}}(c')$, *and* $\mathrm{prev}(u) \notin R$.

Proof sketch. We apply induction on the index i of the round. For the base case, the first round of the algorithm is a round of **Case 1** of $k = \mathrm{dist}(s)$, where s is a source vertex. The shortest s-u path with $u \in P_{\mathrm{mid}}(c_s)$ is su and $\mathrm{dist}(u) = d(u)$ due to the pre-processing.

Now we assume that Lemma 6 holds up to the $(i-1)$-th round. Suppose the i-th round performs line 5. First we show **(1)**. Let x be the closest vertex to v along the shortest s-v path such that $d(x) = \mathrm{dist}(x)$. Note that $x \neq s$ because all children of s have correct-dist values due to the pre-processing. If $x = v$, we are done. Otherwise, $x \notin R$ since $d(x) < d(v)$. Then by induction hypothesis on the round which removes x from R, a child y of x satisfies $d(y) = \mathrm{dist}(y)$ unless x is a small neighbor of y. If x is a small neighbor of y, $\mathrm{alarm}(c_y)$ was at most $d(x) + 2|c_y|$ after that round. Since $y \in P_{\mathrm{mid}}(c_y)$, $8|c_y| \leq r_y$. Hence, $\mathrm{alarm}(c_y) \leq d(x) + \frac{1}{4} r_y < d(x) + |xy| = d(y)$ due to Corollary 5. Thus, $d(y) = \mathrm{dist}(y)$ by induction hypothesis **(3)** on the round of $k = \mathrm{alarm}(c_y)$. This contradicts the choice of x.

Next we show **(2)**. We may assume u is not a source vertex s. Let $w = \mathrm{prev}(u)$. We show that $d(w) = \mathrm{dist}(w)$. If $w = s$ or ws is an edge of G, we are done by the preprocessing. Hence, we may assume that $x := \mathrm{prev}(w)$ exists and x is not a source vertex s. Note that the proof of **(1)** implicitly says that for any vertex v' with $d(v') \leq k$, $d(v') = \mathrm{dist}(v')$ at the onset of line 5. Hence, we are enough to consider the case that $d(w) > d(v)$. Notice that xu is not an edge of G since otherwise, x is the predecessor of u. Then $d(x) < d(x) + |xu| - r_u - r_x < d(x) + |xw| + |wu| - r_u = d(u) - r_u < d(v)$, where the

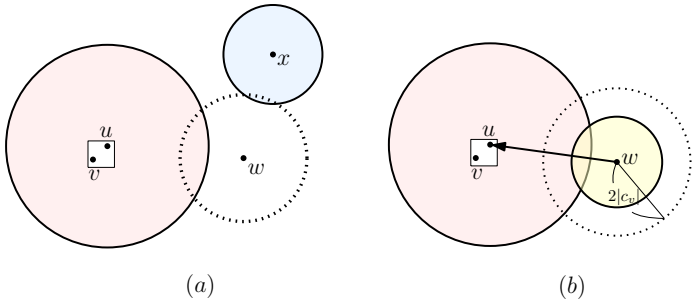


Figure 1 Illustrating some points in the proof of Lemma 6-(2). (a) An edge xu is not an edge of G. (b) If w is a small neighbor of u. alarm(c_v) rings before the round of $k = \text{dist}(v)$.

last inequality comes from $u, v \in P_{\text{mid}}(c_v)$. Hence, $d(x) = \text{dist}(x)$ and therefore $x \notin R$ due to the choice of v. Thus, by induction hypothesis on the round which removes x from R, $d(w) = \text{dist}(w)$ unless x is a small neighbor of w and $k < d(x) + 2|c_w|$. If x is a small neighbor of w, alarm(c_w) is set to at most $d(x) + 2|c_w|$ after that round. After a similar computation as we did in **(1)**, we can derive $d(x) + 2|c_w| < d(v)$ using the facts that $d(u) - \frac{1}{4}r_v < d(v)$ and Corollary 5. Now $d(w) = \text{dist}(w)$ due to induction hypothesis **(3)**.

Hence, for all cases, $d(w) = \text{dist}(w)$. Since u is not a leaf on the shortest path tree, $|wu| \geq \frac{1}{2}r_u$ due to Corollary 5. The only problematic case occurs when w is a small neighbor of u and the lazy update from w to $P_{\text{mid}}(c_u)$ has not occurred. In this case, $d(w)+2|c_u| > d(v)$ due to induction hypothesis **(3)**. Then $d(v) < d(w)+2|c_u| \leq d(w)+\frac{1}{4}r_u \leq d(w)+|wu|-\frac{1}{4}r_u = d(u) - \frac{1}{4}r_u < d(v)$ leads to a contradiction. Finally, **(3)** is followed by **(2)** and the fact that $\text{prev}(u) \notin R$ due to the choice of k. ◀

▶ **Lemma 7.** *Algorithm 2 returns a shortest path tree rooted at s.*

2.4 Time Complexity

In this section, we show that Algorithm 2 can be implemented in $O(n \log^2 n \log \Psi)$ time. For convenience, we assume that the subroutine Update(U, V) takes $O((|U| + |V|) \log^2 n)$ time (see the full version), and analyze the overall running time based on this assumption.

▶ **Lemma 8.** *The subroutine Update(U, V) takes $O((|U| + |V|) \log^2 n)$ time.*

Pre-processing. In the pre-processing step, we compute $\text{dist}(v)$ for all neighbors v of s, and $L(c)$ for all grid cells c. Here, $L(c)$ denotes the set of cells c' such that $P_{\text{mid}}(c)$ contains a small neighbor of c'. The former part takes $O(n)$ time by checking $|sv| - r_s - r_v$ for all $v \in P$. We *do not* use the floor function on the real RAM model to implement the hierarchical grid.

▶ **Lemma 9.** *One can compute a hierarchical grid Γ, and $P_{\text{mid}}(c)$ and $P_{\text{small}}(c)$ for all cells $c \in \Gamma$ in $O(n(\log n + \log \Psi))$ time on the real RAM model.*

▶ **Lemma 10.** *If $P_{mid}(c)$ contains a small neighbor of c', $c' \in L(c)$. Furthermore, one can compute $L(c)$ for all cells c in $O(n \log n \log \Psi)$ time.*

Time complexity of all rounds of Case 1. Let n_i be the number of vertices involved in $P_{\text{mid}}(c_v) \cup \bigcup_{c \in \boxplus c_v} (P_{\text{mid}}(c) \cup P_{\text{small}}(c))$ in the i-th round of line 5-6. If i-th round is a round of **Case 2**, we set n_i to 0. The time complexity of this part is then $\Sigma_i O(n_i \log^2 n)$. We show that

each $P_{\mathrm{mid}}(c)$ and $P_{\mathrm{small}}(c)$ appears in $O(1)$ rounds. First, since we remove $P_{\mathrm{mid}}(c_v)$ from R after the round $k = \mathrm{dist}(v)$, each $P_{\mathrm{mid}}(c)$ appears exactly once in the term $P_{\mathrm{mid}}(c_v)$. Suppose $c \in \boxplus_{c'}$. Then c is contained in axis-parallel square of diameter $64|c'|$ centered at $p(c)$ and $|c| \leq 2|c'|$. Conversely, c' is contained in the axis-parallel square of diameter $65|c|$ centered at $p(c)$. Therefore, the number of different c' with $c \in \boxplus_{c'}$ is $O(1)$ for a fixed c. Hence, each $P_{\mathrm{mid}}(c)$ and $P_{\mathrm{small}}(c)$ appears $O(1)$ rounds through the term $\bigcup_{c \in \boxplus_{c_v}} (P_{\mathrm{mid}}(c) \cup P_{\mathrm{small}}(c))$. Note that each vertex v is contained in one $P_{\mathrm{mid}}(c)$ and $O(\log \Psi)$ $P_{\mathrm{small}}(c)$ for $c \in \Gamma$. Thus,

$$\Sigma_i O(n_i \log^2 n) = \Sigma_c O(|P_{\mathrm{mid}}(c)| + \log \Psi |P_{\mathrm{small}}(c)|) \times O(\log^2 n) = O(n \log^2 n \log \Psi). \quad (1)$$

Time complexity of all rounds of Case 2. Let n'_i be the number of vertices involved in $P_{\mathrm{mid}}(c) \cup \bigcup_{c' \in \boxplus_c} (P_{\mathrm{small}}(c'))$ in the i-th round of line 12. If i-th round runs a round of **Case 1**, we set n'_i to 0. The time complexity of this part is then $\Sigma_i O(n'_i \log^2 n)$. Since alarm$(c)$ always set to $d(v) + 2|c|$ for some $v \in P$ and by Lemma 3, each cell c is referenced by line 11 $O(1)$ times. Again, there are $O(1)$ grid cells c' such that $\boxplus_{c'}$ contains a fixed cell c, and each v is contained in one $P_{\mathrm{mid}}(c)$ and $O(\log \Psi)$ $P_{\mathrm{small}}(c)$ sets. Thus, the total time complexity is

$$\Sigma_i O(n'_i \log^2 n) = \Sigma_c O(|P_{\mathrm{mid}}(c)| + \log \Psi |P_{\mathrm{small}}(c)|) \times O(\log^2 n) = O(n \log^2 n \log \Psi). \quad (2)$$

Priority queue. We maintain two priority queues, one of which stores vertex $v \in R$ with priority dist(v), and the other queue stores cell $c \in \Gamma$ with priority alarm(c). Once the algorithm performs line 3, we peek an element with minimum priority for each queue, and then choose k as the minimum value among them. The total cost of the queue operations is dominated by the total cost caused by UPDATE subroutines.

▶ **Theorem 11.** *There is an $O(n \log^2 n \log \Psi)$-time algorithm that solves the single-source shortest path problem on disk graphs with n vertices and radius ratio Ψ.*

3 SSSP on Disk Graphs of Arbitrary Radius Ratio

In this section, we extend the approach of Section 2 to devise an $O(n \log^4 n)$-time algorithm for the SSSP problem on disk graphs with arbitrary radius ratio. Basically, the $O(n \log^2 n \log \Psi)$ term in Theorem 11 came from the total size of $P_{\mathrm{mid}}(\cdot)$ and $P_{\mathrm{small}}(\cdot)$, which depends on the height ($=\Theta(\log \Psi)$) of the hierarchical grid. In the case of an arbitrary radius ratio, we cannot bind the height of the grid. To resolve this, we follow the approach presented in [2]. More specifically, we use a *compressed quadtree* (See Section 3.1) instead of a hierarchical grid. The height of the compressed quadtree might be $\Theta(n)$, which is still large. Hence, we use a *heavy path decomposition* (See Section 3.1) to compute $O(n)$ vertical paths in the compressed quadtree, which we call *canonical paths*, such that every root-node path of the compressed quadtree can be uniquely represented by the concatenation of $O(\log n)$ canonical paths. In this way, we can encode the edge information using near-linear time and space.

This approach has been applied to design efficient algorithms on disk graphs of arbitrary radius ratios, such as the dynamic connectivity problem [2] and the unweighted SSSP problem [15]. By integrating this approach with our lazy update scheme in a more sophisticated way, we can design an $O(n \log^4 n)$-time algorithm for the SSSP problem on weighted disk graphs with arbitrary radius ratios.

Throughout this section, we alternatively define the notions of $P_{\mathrm{mid}}(c), \square_c, \boxplus_c$, and regular/irregular edges. Let $h = 2^{10}$ and $\alpha > 2\pi / \arcsin(\frac{1}{100})$ be the integer constants.

▶ **Definition 12.** *For a grid cell c in $\mathcal{Q}$,*
- $P_{mid}(c) := \{v \in c : r_v \in [h|c|, 2h|c|)\}$.
- $\square_c$:= *Axis-parallel square of diameter* $8h^2|c|$ *centered at* $p(c)$.
- $\boxplus_c := \{c' \in \mathcal{Q} : c' \subset \square_c, |c'| \in [\frac{1}{h}|c|, h|c|]\}$.

▶ **Definition 13.** *For an edge uv of G with $r_u \le r_v$, uv is a* regular edge *if $\frac{r_v}{r_u} < h$, and an irregular edge otherwise.*

▶ **Lemma 14.** *Let uv be a regular edge. Then $c_u \in \boxplus_{c_v}$.*

3.1 Preliminaries: technical tools

In this section, we first briefly introduce technical items. Klost [15] implemented an efficient breadth-first search using these components. In contrast, our goal is to implement Dijkstra's algorithm over weighted graphs, which requires more sophisticated update strategies. We conclude this section by outlining the modified lazy update scheme that plays the central role in our algorithm for arbitrary radius ratios in Section 3.2.

Compressed quadtree. For an integer $i \ge 0$, let Γ_i be a grid of level i, which consists of axis-parallel square cells of diameter 2^i. Among all grid cells of $\bigcup_i \Gamma_i$, let c be the grid cell of the smallest level that contains P. A *compressed quadtree* $\mathcal{Q}$ is a rooted tree defined as follows. We start from c as the root of the tree and iteratively expand the tree. More specifically, whenever we process $c \in \Gamma_i$, we stop the expansion if c contains exactly one point of P. Otherwise, we compute at most four cells of Γ_{i-1} that are nested in c, and contain at least one point of P. If there is exactly one such child c', we remove c and connect c' to the parent node of c. Then we move on to children of c.

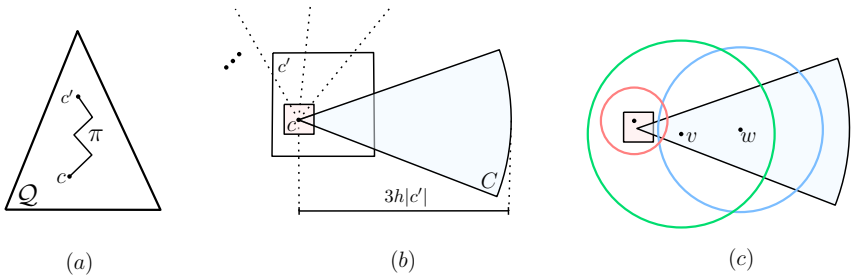
▶ **Lemma 15** ([12])**.** *In $O(n \log n)$ time, one can compute a compressed quadtree $\mathcal{Q}$ having $O(n)$ nodes and $O(n)$ height.*

For our purpose, the compressed quadtree is extended to contain all grid cells of $\boxplus_{c_v}$ for all $v \in P$. It also takes an $O(n \log n)$ time to compute the extended tree [2].

Heavy path decomposition and canonical paths. As the height of $\mathcal{Q}$ is $O(n)$, the direct use of Algorithm 2 is expensive: each vertex may be contained in $\Theta(n)$ grid cells, which was $O(\log \Psi)$ in Section 2. Our goal is to compute a path system in $\mathcal{Q}$ such that every root-node path of $\mathcal{Q}$ can be represented by the concatenation of $O(\log n)$ paths in the path system. We say an edge $cc' \in \mathcal{Q}$ is *heavy* if c' is the first child of c in the order of children, where we give an order by the total number of nodes in the subtree rooted at c' among all children of c. We say cc' *light* otherwise. A *heavy path* of $\mathcal{Q}$ is the maximal path on $\mathcal{Q}$ that consists of only heavy edges. The *heavy path decomposition* is the collection of all heavy paths in $\mathcal{Q}$.

▶ **Lemma 16** ([19])**.** *Let $\mathcal{Q}$ be the tree of n nodes. Then,*
1. *every root-leaf path of $\mathcal{Q}$ contains $O(\log n)$ light edges,*
2. *every node of $\mathcal{Q}$ lies on exactly one heavy path, and*
3. *the heavy path decomposition can be computed in $O(n)$ time.*

Then we use the approach of [2] as a black box. That is, we define a set Π of $O(n)$ *canonical paths*, such that every root-node path of $\mathcal{Q}$ can be uniquely represented by the concatenation of $O(\log n)$ disjoint canonical paths. Also, each canonical path is a subpath of a heavy path, and each node of $\mathcal{Q}$ is contained in $O(\log n)$ canonical paths. We can compute Π in $O(n \log n)$ time using standard data structures. See [2] for details.



(a) (b) (c)

Figure 2 (a) Canonical path π. (b) Illustration of $\lambda = (c, C)$. (c) A disk of $P_{\text{post}}(\lambda)$ (green) contains the disks of $P_{\text{small}}(\lambda)$ (red), while a disk of $P_{\text{large}}(\lambda)$ (blue) may not.

Handling irregular edges. We primarily follow the notion of a proxy graph as presented in [2, 15], with slight modifications to the notation for our purpose. For an irregular edge uv with $r_u < r_v$, we call u a *small neighbor* of v, and v a *large-neighbor* of u. For a canonical path π, we use c_π to denote the lowest cell of π. Also, we use $|\pi_\ell|(= |c_\pi|)$ and $|\pi_t|$ to denote the diameter of the lowest cell and the topmost cell of π, respectively. We then define a set $\mathcal{C}_\pi$ of α congruent cones of radius $3h|\pi_t|$, all sharing the same apex at $p(c_\pi)$. Let Λ be the set of all pairs (c_π, C) for all canonical paths π and all cones $C \in \mathcal{C}_\pi$. See Figure 2(a-b).

We say an irregular edge uv is *redundant* if $|uv| < |r_u - r_v|$. Due to Lemma 4, if a redundant edge uv appears on the shortest path tree, then one endpoint, say u, is a leaf on the tree. Therefore, we can postpone the correction of $\text{dist}(u)$ until the last, as no shortest s-$w(\neq u)$ path intersects u. In fact, our algorithm handles these edges during post-processing.

For a pair $\lambda = (c, C) \in \Lambda$, let $r(C)$ be the radius of C. For a vertex $v \in P$, recall that c_v is a grid cell such that $v \in P_{\text{mid}}(c_v)$. Let $\bar{c}_v$ be the smallest grid cell on the root-c_v path in $\mathcal{Q}$ whose diameter is at least $h|c_v|$. We let Π_v be the set of $O(\log n)$ consecutive canonical paths representing the root-$\bar{c}_v$ path. We define three subsets of P with respect to λ. Here, $p(c)$ is the center point of c.

$$P_{\text{small}}(\lambda) := \{v \in c \mid \exists \pi \in \Pi_v \text{ s.t. the lowest cell of } \pi \text{ is } c\}, \tag{3}$$

$$P_{\text{large}}(\lambda) := \{v \in C \mid r_v \in [h|c|, \frac{2}{3}r(C)] \text{ and } |r_v - |vp(c)|| < 5|c|\}, \text{ and} \tag{4}$$

$$P_{\text{post}}(\lambda) := \{v \in C \mid r_v \in [h|c|, \frac{2}{3}r(C)] \text{ and } 5|c| \leq r_v - |vp(c)|\}. \tag{5}$$

Intuitively, for any irregular edge uv, some λ encodes uv as $u \in P_{\text{small}}(\lambda)$ and $v \in P_{\text{large}}(\lambda) \cup P_{\text{post}}(\lambda)$. Moreover, $v \in P_{\text{large}}(\lambda)$ if uv is a non-redundant. Later, our algorithm runs Dijkstra's algorithm in a cell-by-cell manner w.r.t. $P_{\text{mid}}(\cdot), P_{\text{small}}(\cdot)$, and $P_{\text{large}}(\cdot)$.

▶ **Lemma 17.** *Suppose v is a small neighbor of u. There exists a pair $\lambda \in \Lambda$ such that $v \in P_{\text{small}}(\lambda)$ and $u \in P_{\text{large}}(\lambda) \cup P_{\text{post}}(\lambda)$. Moreover, if uv is non-redundant, $u \in P_{\text{large}}(\lambda)$.*

▶ **Corollary 18** (Corollary of Lemma 4)**.** *Suppose the shortest s-v path contains an irregular edge uv and v is not a leaf. Then $|uv| \geq (1 - \frac{1}{h})r_v$.*

▶ **Lemma 19.** *$\sum_{\lambda \in \Lambda} |P_{\text{small}}(\lambda)| = O(n \log n)$ and $\sum_{\lambda \in \Lambda} |P_{\text{large}}(\lambda)| + |P_{\text{post}}(\lambda)| = O(n \log n)$.*

Two-way lazy update scheme. Recall that Algorithm 2 updates small neighbors of v once it corrects the dist-values of $P_{\text{mid}}(c_v)$, whereas it postpones the update to (informal) large neighbors of v. In this section, we apply the lazy update scheme for both large and small

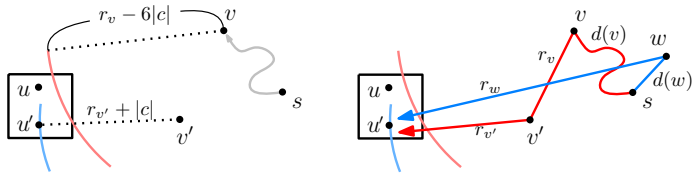


Figure 3 Illustration of some points in the proof of Lemma 20. (a) If $r_{v'} \leq r_v$, v' is contained in a disk of radius $r_v - 6|c|$ centered at v. (b) If $r_w >> r_v$, $d(w) << d(v)$ since the asymptotic lengths of the (shortest) blue path and the red path are $d(w) + r_w$ and $d(v) + r_v + r_{v'}$, respectively.

neighbors of v. The main reason is as follows. Due to Lemma 6-(2), all vertices of $P_{\text{mid}}(c_v)$ simultaneously get correct dist-values after the call of line 5. This property heavily relies on the geometric property of $P_{\text{mid}}(c_v)$ such that for any $u, w \in P_{\text{mid}}(c_v)$, $|uw| \leq \frac{1}{8} r_u$. However, this property no longer holds for $P_{\text{large}}(\lambda)$. More specifically, once we handle $v \in P_{\text{large}}(\lambda)$, there might be a vertex $u \in P_{\text{large}}(\lambda)$ whose predecessor in the shortest s-u has not been processed, and therefore, we cannot get correct dist-values of $P_{\text{large}}(\lambda)$ at this point.

Hence, one might consider a lazy update scheme to resolve this issue: delay the transmission of dist-values of $P_{\text{large}}(\lambda)$ in the range $[x, x + f(\lambda)]$ for some function f of λ until all dist-values smaller than $x + f(\lambda)$ have been processed. Unfortunately, this naive approach itself is not useful. Recall that for $v \in P_{\text{large}}(\lambda)$ with $\lambda = (c, C)$, r_v is in range $[h|c|, \frac{2}{3} r(C)]$ and $r(C)/|c|$ can be very large. Hence, we cannot partition the dist-values of $P_{\text{large}}(\lambda)$ into a constant number of ranges $[x, x + f(\lambda)]$, and the number of updates from $P_{\text{large}}(\lambda)$ to $P_{\text{small}}(\lambda)$ might be $\Theta(n)$, which leads to quadratic running time in total. We reduce the running time using the following geometric observation. See also Figure 3. Let $\lambda = (c, C)$.

▶ **Lemma 20.** *Let $v, v' \in P_{large}(\lambda)$, $u, u' \in P_{small}(\lambda)$, $r_v > r_{v'}$, the shortest s-u path contains vu, and $v'u'$ is an edge of G. Then $d(w) < d(v) + r_v - 6|c|$ where $w \in P_{large}(\lambda)$ is a predecessor of u' in the shortest s-u' path.*

Suppose we know the subset $P(\lambda)$ of $P_{\text{large}}(\lambda)$ whose dist-values have been corrected, and our algorithm will transmit dist-values of $P(\lambda)$ to $P_{\text{small}}(\lambda)$ right after all dist-values smaller than $d(v) + r_v - 6|c|$ have been processed for some $v \in P(\lambda)$. Then, Lemma 20 guarantees that $P(\lambda)$ contains all vertices of $P_{\text{large}}(\lambda)$ whose radii are smaller than r_v. Furthermore, after the lazy update, all vertices of $P_{\text{small}}(\lambda)$ adjacent to $u \in P_{\text{large}}(\lambda)$ with $r_u \leq r_v$ will have correct dist-values. Hence, the following procedure works.

- **Step 1.** Maintain a priority queue that stores $v \in P(\lambda)$ with priority $d(v) + r_v - 6|c|$.
- **Step 2.** Once we handle the lazy update caused by $d(v) + r_v - 6|c|$, we update dist-values of the vertices of $P_{\text{small}}(\lambda)$ which are adjacent to a vertex $v' \in P(\lambda)$ with $r_{v'} \leq r_v$.
- **Step 3.** After the update, we remove the updated vertices from $P_{\text{small}}(\lambda)$.

Recall that the running time of $\textsc{Update}(U, V)$ is $O((|U| + |V|) \log^2 n)$ time. Although there might $\Theta(n)$ calls of $\textsc{Update}$ for $(P_{\text{large}}(\lambda), P_{\text{small}}(\lambda))$, the total size of $|V|$ can be bounded by $\tilde{O}(n)$ due to **Step 3** and Lemma 19. Notice that U always corresponds to a subset $P(\lambda)$ of $P_{\text{large}}(\lambda)$, which starts as an empty set and grows as the algorithm progresses. Hence, we implement incremental data structures with near-linear construction time by carefully dynamizing $\textsc{Update}$ subroutine, inspired by the spirit of [3].

▶ **Definition 21.** *Let* Update-Inc(W) *be the incremental data structure with respect to* W *that supports the following operations:*

- *Initialize: set* $U = \emptyset$*,*
- *Insert(*v*): add a vertex* $v \in W$ *into* U *together with a dist-value* $d(v)$*, and*
- *Query(*V*): given a set* V *of vertices, perform* Update(U, V)*.*

▶ **Lemma 22.** *There is a data structure* Update-Inc$(P_{large}(\lambda))$ *that supports insert operation* $O(n \log^4 n)$ *time in total, and supports query operation on* V *in* $|V| \cdot O(\log^3 n)$ *time.*

Remark. We introduced an incremental data structure to handle the $\Theta(n)$ updates from $P_{\text{large}}(\lambda)$ to $P_{\text{small}}(\lambda)$. However, due to a similar reason, the number of lazy updates from $P_{\text{small}}(\lambda)$ to $P_{\text{large}}(\lambda)$ might be $\Theta(n)$. Fortunately, we can implement this direction of lazy update using solely Update subroutine carefully.

3.2 Algorithm

In this section, we present an $O(n \log^4 n)$-time algorithm for the SSSP problem on disk graphs of arbitrary radius ratio. Due to the lack of space, we mainly focus on the differences from Algorithm 2 in Section 2. During the pre-processing, we maintain two alarms alarm-up(λ), alarm-down(λ) together with a priority queue $Q(\lambda)$ for each pair $\lambda \in \Lambda$. Furthermore, we compute the set $L_1(v)$ (and $L_2(v)$) of pairs $\lambda \in \Lambda$ where $v \in P_{\text{small}}(\lambda)$ (and $v \in P_{\text{large}}(\lambda)$) and v has a neighbor in $P_{\text{large}}(\lambda)$ (and $P_{\text{small}}(\lambda)$) for each $v \in P$. In each round, we check dist(v) for all $v \in R$ and alarm-up(λ), alarm-down(λ) for all $\lambda \in \Lambda$. Then we find the minimum value k among them and proceed depending on the type of k. The algorithm moves to the post-processing step when R becomes empty. The algorithm utilizes the Update subroutine from Section 2.

Case 1: $k =$ dist(v) for a vertex $v \in R$. After correcting the dist-values of $P_{\text{mid}}(c_v)$ except those are leaves on the shortest path tree, we update the dist-values of the neighbors of $P_{\text{mid}}(c_v)$ through one direct update and two-way lazy updates. This is done by executing subroutine Update$(P_{\text{mid}}(c_v), \bigcup_{c \in \boxplus_{c_v}} P_{\text{mid}}(c))$, setting alarm-up$(\lambda) =$ dist$(v) + \frac{1}{4}v(\lambda)$ for every $\lambda \in L_1(v)$ with alarm-up$(\lambda) = \infty$, and inserting u with the priority dist$(u) + r_u - 6|c|$ into $Q(\lambda)$ and inserting u into Update-Inc$(P_{\text{large}}(\lambda))$ for every $u \in P_{\text{mid}}(c_v)$ and every $\lambda \in L_2(u)$. Here, $v(\lambda)$ is the smallest radius of the vertex of $P_{\text{large}}(\lambda)$ forming an edge with $P_{\text{mid}}(c_v)$. Intuitively, the subroutine takes care of the neighbors connected by regular edges, and alarm-up$(\cdot)$ and alarm-down$(\cdot)$ take care of large neighbors and small neighbors of the vertices of $P_{\text{mid}}(c_v)$, respectively.

Case 2: $k =$ alarm-up(λ) for a pair $\lambda \in \Lambda$. In this case, we shall correct the dist-values of all $v \in P_{\text{large}}(\lambda)$ whose predecessor u is in $P_{\text{small}}(\lambda)$ with $d(u) < k$. This is done by applying Update$(U_{\text{small}}(\lambda, k), U_{\text{large}}(\lambda, k))$ where $U_{\text{small}}(\lambda, k)$ and $U_{\text{large}}(\lambda, k)$ are carefully computed subsets of $P_{\text{small}}(\lambda)$ and $P_{\text{large}}(\lambda)$, respectively. See the exact computation of $U_{\text{small}}(\lambda, k), U_{\text{large}}(\lambda, k)$ and its correctness in the full version.

Case 3: $k =$ alarm-down(λ) for a pair $\lambda \in \Lambda$. We shall correct the dist-values of all $v \in P_{\text{small}}(\lambda)$ which are adjacent to $u \in P_{\text{large}}(\lambda)$ with $r_u \leq r_{u_k}$, where u_k is the vertex stored in $Q(\lambda)$ with minimum priority. This is done by the query operation on $D_{\text{small}}(\lambda, k)$ to Update-Inc$(P_{\text{large}}(\lambda, k))$. Here, $D_{\text{small}}(\lambda, k)$ denotes the set of vertices in $P_{\text{small}}(\lambda)$ such that (i) form an edge with $u \in P_{\text{large}}(\lambda)$ with $r_u \leq r_{u_k}$, and (ii) have not been considered in a preceding round of $k =$ alarm-down(λ). After the update, we modify $Q(\lambda)$ appropriately.

Algorithm 3 SSSP2(P).

1. $R \leftarrow P$ and initialize Update-Inc($P_{\text{large}}(\lambda)$) for all $\lambda \in \Lambda$
2. **while** $R \neq \emptyset$ **do**
3. $k \leftarrow \min(\{\text{dist}(v) : v \in R\} \cup \{\text{alarm-up}(\lambda) : \lambda \in \Lambda\} \cup \{\text{alarm-down}(\lambda) : \lambda \in \Lambda\})$
4. **if** $k = \text{dist}(v)$ for $v \in R$ **then**
5. Update($\bigcup_{c \in \boxplus_{c_v}} P_{\text{mid}}(c), P_{\text{mid}}(c_v)$)
6. Update($P_{\text{mid}}(c_v), \bigcup_{c \in \boxplus_{c_v}} P_{\text{mid}}(c)$)
7. **for** $\lambda \in L_1(v)$ **do**
8. **if** alarm-up$(\lambda) = \infty$ **then**
9. alarm-up$(\lambda) \leftarrow \text{dist}(v) + \frac{1}{4}v(\lambda)$
10. **for** $u \in P_{\text{mid}}(c_v)$ **do**
11. **for** $\lambda = (c, C) \in L_2(u)$ **do**
12. $Q(\lambda) \leftarrow \text{insert}(u, \text{dist}(u) + r_u - 6|c|)$
13. Update-Inc($P_{\text{large}}(\lambda)$).insert($u$)
14. alarm-down$(\lambda) \leftarrow$ min-priority($Q(\lambda)$)
15. $R \leftarrow R \setminus P_{\text{mid}}(c_v)$
16. **if** $k =$ alarm-up(λ) for $\lambda \in \Lambda$ **then**
17. Update($U_{\text{large}}(\lambda, k), U_{\text{small}}(\lambda, k)$)
18. alarm-up$(\lambda) \leftarrow \infty$
19. **if** $k =$ alarm-down(λ) for $\lambda \in \Lambda$ **then**
20. Update-Inc($P_{\text{large}}(\lambda)$).query($D_{\text{small}}(\lambda, k)$)
21. alarm-down$(\lambda) \leftarrow$ min-priority(delete($Q(\lambda)$))

Post-processing. In the post-processing step, we correct the dist-values for the remaining vertices. We execute Update($P_{\text{post}}(\lambda), P_{\text{small}}(\lambda)$) for every pair λ in Λ.

3.3 Correctness and Time Complexity

▶ **Lemma 23.** *The correctness of Algorithm 3 mainly follows from the following lemma, which is a counterpart to Lemma 7 from Section 2. The following statements hold during the execution of Algorithm 3.*

(1) *At the onset of line 5,* $\text{dist}(v) = d(v)$ *unless* v *is a leaf.*

(2) *After line 5,* $\text{dist}(u) = d(u)$ *if* $u \in P_{\text{mid}}(c_v)$ *and* u *is not a leaf.*

(3) *After line 17,* $\text{dist}(u) = d(u)$ *if* $u \in P_{\text{large}}(\lambda)$, $\text{prev}(u) \in P_{\text{small}}(\lambda)$, *and* $d(\text{prev}(u)) < k$.

(4) *After line 20, let* v *be the vertex in* $Q(\lambda)$ *with minimum priority. Then* $\text{dist}(u) = d(u)$ *if* $u \in P_{\text{small}}(\lambda)$ *and* u *is adjacent to* $v' \in P_{\text{large}}(\lambda)$ *with* $r_{v'} \leq r_v$.

▶ **Lemma 24.** *Algorithm 3 returns the shortest path tree rooted at* s.

Time Complexity. The main bottleneck of Algorithm 3 comes from the incremental data structure Update-Inc($P_{\text{large}}(\lambda)$). Due to Lemmas 19 and 22, we conclude the following.

▶ **Theorem 25.** *There is an algorithm to solve the single-source shortest path problem on disk graphs with* n *vertices in* $O(n \log^4 n)$ *time.*

References

1 Shinwoo An, Kyungjin Cho, and Eunjin Oh. Faster algorithms for cycle hitting problems on disk graphs. In *Algorithms and Data Structures Symposium: 18th International Symposium (WADS 2023)*, pages 29–42, 2023. doi:10.1007/978-3-031-38906-1_3.

2 Alexander Baumann, Haim Kaplan, Katharina Klost, Kristin Knorr, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Dynamic connectivity in disk graphs. *Discrete & Computational Geometry*, 71(1):214–277, 2024. doi:10.1007/S00454-023-00621-X.

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 Édouard Bonnet, Panos Giannopoulos, Eun Jung Kim, Pawel Rzazewski, and Florian Sikora. QPTAS and subexponential algorithm for maximum clique on disk graphs. In *34th International Symposium on Computational Geometry (SoCG 2018)*, pages 11–14, 2018.

5 Bruce W Brewer and Haitao Wang. An improved algorithm for shortest paths in weighted unit-disk graphs. *arXiv preprint arXiv:2407.03176*, 2024. doi:10.48550/arXiv.2407.03176.

6 Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015. doi:10.1016/J.COMGEO.2014.12.003.

7 Sergio Cabello and Wolfgang Mulzer. Minimum cuts in geometric intersection graphs. *Computational Geometry*, 94:101720, 2021. doi:10.1016/J.COMGEO.2020.101720.

8 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 2016)*, pages 24:1–24:13, 2016. doi:10.4230/LIPICS.ISAAC.2016.24.

9 Timothy M Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry*, 10(2):3–20, 2019. doi:10.20382/JOCG.V10I2A2.

10 Jared Espenant, J Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258, page 30, 2023.

11 Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC 2003)*, pages 483–492, 2003. doi:10.1145/780542.780613.

12 Sariel Har-Peled. *Geometric approximation algorithms*, volume 173 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2011.

13 Haim Kaplan, Matthew J Katz, Rachel Saban, and Micha Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. In *the proceedings of the 31st Annual European Symposium on Algorithms (ESA 2023)*, pages 67:1–67:14, 2023. doi:10.4230/LIPICS.ESA.2023.67.

14 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2495–2504, 2017. doi:10.1137/1.9781611974782.165.

15 Katharina 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.

16 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51(3):723–765, 2022.

17 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A framework for approximation schemes on disk graphs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 2228–2241, 2023. doi:10.1137/1.9781611977554.CH84.

18 Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

19 Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing (STOC 1981)*, pages 114–122, 1981.

20 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete & Computational Geometry*, 64(4):1141–1166, 2020. doi:10.1007/S00454-020-00219-7.

21 Haitao Wang and Yiming Zhao. Improved algorithms for distance selection and related problems. In *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274, page 101, 2023. doi:10.4230/LIPIcs.ESA.2023.101.

Rapid Mixing of the Flip Chain over Non-Crossing Spanning Trees

Konrad Anand
School of Informatics, Informatics Forum, University of Edinburgh, UK

Weiming Feng
School of Computing and Data Science, The University of Hong Kong, China

Graham Freifeld
School of Informatics, Informatics Forum, University of Edinburgh, UK

Heng Guo
School of Informatics, Informatics Forum, University of Edinburgh, UK

Mark Jerrum
School of Mathematical Sciences, Queen Mary University of London, UK

Jiaheng Wang
Faculty of Informatics and Data Science, University of Regensburg, Germany

Abstract

We show that the flip chain for non-crossing spanning trees of $n+1$ points in convex position mixes in time $O(n^8 \log n)$. We use connections between Fuss-Catalan structures to construct a comparison argument with a chain similar to Wilson's lattice path chain (Wilson 2004).

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Theory of computation → Computational geometry

Keywords and phrases non-crossing spanning trees, Markov chain, mixing time

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.8

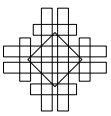
Related Version *Full Version*: https://arxiv.org/abs/2409.07892 [4]

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 947778). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.
Weiming Feng: Weiming Feng acknowledges the support of Dr. Max Rössler, the Walter Haefner Foundation, and the ETH Zürich Foundation during his affiliation at ETH Zürich.
Jiaheng Wang: Jiaheng Wang also acknowledges support from ERC (grant agreement No. 101077083, CountHom).

1 Introduction

The reconfiguration of planar objects is an intensively studied topic in computational geometry. Given a set of points, such a planar structure can be a triangulation, a non-crossing perfect matching, or a *non-crossing spanning trees* (NCST), etc. In this paper we focus on the last one. Given $n+1$ points in a plane, an NCST is a spanning tree such that no two edges intersect or overlap (except on the joint vertex) when all edges are drawn as straight line segments on the plane. The reconfiguration of NCSTs is done by *flips*, which remove an edge of the current tree, and then add an edge back so that the resulting graph is still a valid NCST. See Figure 1 for an example. Given two NCSTs over the same point set in general position, Avid and Fukuda [8] first proved that it takes at most $2n-2$ flips to move from

© Konrad Anand, Weiming Feng, Graham Freifeld, Heng Guo, Mark Jerrum, and Jiaheng Wang;
licensed under Creative Commons License CC-BY 4.0



one to the other. On the other hand, Hernando, Hurtado, Márquez, Mora and Noy [28] constructed an example where it takes at least $1.5n - O(1)$ flips, even when the points are *in convex position*. These bounds have not been improved until recently [1, 12, 11]. Very recently, Bjerkevik, Kleist, Ueckerdt, and Vogtenhuber [10] improved the upper bound to $5/3n - 3$ and the lower bound to $14/9n - O(1)$ when all points are in convex position.

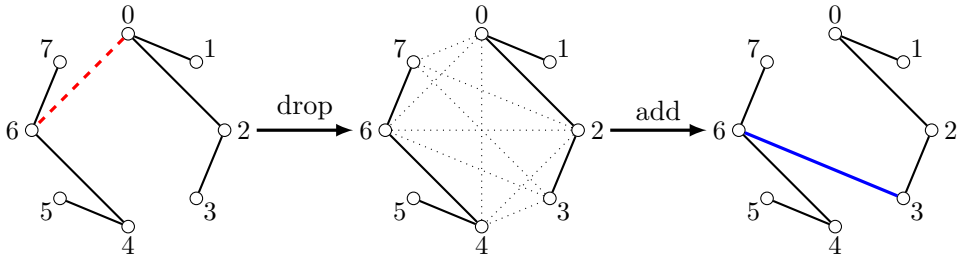


Figure 1 Illustration of a flip move. The red, dashed edge in the leftmost figure is dropped. This gives 9 possible edges (dotted in the middle figure) that can be added back to form a valid NCST. The blue, thick edge in the rightmost figure is picked.

Another classic topic in computational geometry is to count these planar structures. Traditional combinatorial studies focus on asymptotic upper and lower bounds of these structures (see e.g. [33, 43, 44]). Motivations for counting these structures have appeared in other topics, such as quantum gravity [32, 35], plane curves [47], algebraic geometry [20, 22], and the theory of discriminants [27]. From the computational perspective, the complexity of the counting problem seems to remain elusive in most cases, with an exception of counting triangulations of (not necessarily simple) polygons known to be #**P**-hard [25]. On the other hand, counting the number of triangulations of a given set of points remains unresolved (but it is conjectured to be #**P**-hard, see e.g. [21] or [17, §6.7]). Similarly, counting NCSTs over a set of points in general position apparently is also not known to be #**P**-hard nor polynomial-time solvable, although we conjecture the former to be the case. To cope with the apparent intractability, sub-exponential time and fixed-parameterised approximation algorithms have been studied [2, 3, 34], but the complexity of approximate counting NCSTs does not appear to be known either.

On the other hand, when these $n + 1$ points are in convex position, the corresponding counting problem becomes easy to solve. One can show that the number of NCSTs, denoted by $C_{2,n}$, is subject to the following recursion: (see e.g. [42])

$$C_{2,0} = 1,\ C_{2,n} = \sum_{\substack{i,j,k\in\mathbb{Z}_{\geq 0},\\ i+j+k=n-1}} C_{2,i}C_{2,j}C_{2,k}, \qquad \Longrightarrow \qquad C_{2,n} = \frac{1}{2n+1}\binom{3n}{n}. \tag{1}$$

Sharing a similar form as the famous *Catalan numbers*, this kind of generalisation is called the 2-*Fuss-Catalan numbers*. The (k-Fuss-)Catalan numbers capture more than 200 natural combinatorial structures. For example, the Catalan numbers count the number of triangulations, non-crossing perfect matchings and non-crossing partitions when the points are in convex position. The k-Fuss-Catalan numbers count the number of k-rooted plane trees, $(k + 1)$-ary trees, k-Dyck paths, and many more. We refer interested readers to Stanley's monograph [46] for an exhaustive survey about these Fuss-Catalan structures.

In this paper, we are interested in where reconfiguration and approximate counting meet. We consider a Markov chain whose steps are random flip moves over NCSTs, denoted the *flip chain*. In each step, we drop an edge chosen uniformly at random, and then add back

an edge uniformly at random among all valid choices. (This chain is formally defined in (7).) Reconfiguration results [8, 28, 10] thus imply that the flip chain is *irreducible* (namely, the state space of all NCSTs is connected via flip moves), and when the points are in convex position, the *diameter* of the chain is bounded between $14/9n - O(1)$ and $5/3n - 3$. Markov chains are also the most common approach to approximately count the number of combinatorial structures [30, 24, 31]. While the size of the state space (namely the total number of structures to count) can be exponentially large, the hope is that the Markov chain converges to the uniform distribution within polynomially many steps. The time to converge is called the *mixing time* (defined in Section 2). When the mixing time is bounded by a polynomial, the chain is said to be rapid mixing, and rapid mixing Markov chains usually lead to efficient approximate counting algorithms.

The mixing times of these geometrically defined Markov chains have received considerable attention from the Markov chain community in the past few decades, even when restricted to points in convex position, such as the chains over convex polygon triangulations [39, 38, 26], over triangulations on lattices and spheres [15, 14], or over lattice paths and (1-)Dyck paths [37, 48, 16]. See Cohen's thesis [18] for a more detailed explanation and comprehensive overview about random walks over 1-Fuss-Catalan structures. Beyond the fields of computational geometry and Markov chain analysis, these chains also find numerous applications, often thanks to the wide connections of Fuss-Catalan numbers, such as for quantum spin systems [13, 41, 40] or even in algebraic geometry [9].

In this paper we give the first polynomial upper bound on the mixing time of the flip chain over NCSTs for points in convex position.

▶ **Theorem 1.** *For $n+1$ points on the plane in convex position, the mixing time of the flip chain over their non-crossing spanning trees is $O(n^8 \log n)$.*

The main feature that sets Theorem 1 apart from the results mentioned above is that NCSTs for points in convex position are 2-Fuss-Catalan structures. This renders some tools unavailable and others, such as the path comparison method [23], a lot more intricate to use. We also note that while the very similarly defined bases-exchange chain over spanning trees of a graph is known to have an optimal mixing time [6, 19], NCSTs do not form the bases of a matroid and these results apparently do not apply. We give an overview of our proofs next.

1.1 Technical overview

Our main strategy is to compare the flip chain with another chain that is easier to analyse. Naturally we would choose a chain over structures that have a bijection with NCSTs. There are a few candidates (see e.g. [45]), and we choose 2-Dyck paths. The bijection is defined through the same recursive structure, such as (1), for both 2-Dyck paths and NCSTs. Note that while these structures share the same count and have bijections among them, a local move in one structure may cause drastic changes in another through the bijection. Thus we need to pick a starting point chain so that the distortion caused by the bijection is manageable and the overhead of the comparison can be bounded by a polynomial.

We represent a 2-Dyck path by a string consisting of $2n$ ups ($\nearrow$) and n downs ($\searrow$) such that any prefix of the string contains no fewer ups than twice the number of downs. The formal definition is given in Section 3.1. One can verify that the positions of the down arrows form a basis of a matroid. This generalises the so-called Catalan matroid introduced by Ardila [7]. Since the bases-exchange walk for these matroids has an $O(n \log n)$ mixing time [19], a natural choice would be to compare this chain with the flip chain over NCSTs. This chain randomly chooses a down arrow and moves it to a random valid location. However, as mentioned before, one move in the bases-exchange walk may change NCSTs drastically. See Example 2.

▶ **Example 2.** Consider two 2-Dyck paths of length 39:

$W_1 =$,

$W_2 =$.

They differ at position 20 (coloured by orange) and 38 (coloured by cyan). As W_2 can be obtained from W_1 through moving the position of one down arrow, they are connected by one move of the bases-exchange chain. However, their corresponding NCSTs induced by the bijection of the Fuss-Catalan recurrence (see Section 3) do not share any edge in common, and there is no obvious pattern to note. The corresponding trees are illustrated as W_1 and W_2 in Figure 2.

Since the bases-exchange chain appears to be difficult to compare with, we choose a different chain to control the changes in the corresponding NCSTs, and that is the adjacent move chain. Instead of moving a down arrow to a random location, we restrict the movement to be either left or right one position. More formally, each transition of this chain chooses two adjacent coordinates uniformly at random. If swapping them results in a valid path, we do so with probability 1/2, and otherwise make no change. For example, suppose the current state is ↗↗↘↗↗↘. If, with probability 1/5, we choose the second and third positions, then the path will not change as swapping them leads to an invalid 2-Dyck path. On the other hand, if (with probability 1/5) we choose the third and fourth positions, then the chain will move to ↗↗↗↘↗↘ with probability 1/2 and remain unchanged otherwise.

While this chain is not directly studied before, we verify that the coupling method of Wilson [48] still applies here. Wilson's original argument is applied to Lattice paths or the Bernoulli-Laplacian model (the uniform distribution over *all* strings with a fixed number of ups and downs). Without too much change, the same argument implies that the adjacent move chain over 2-Dyck paths mixes in $O(n^3 \log n)$ steps.

Our main technical contribution is then to compare the adjacent move chain over 2-Dyck paths with the flip chain over NCSTs. We need to characterise how one adjacent move changes the NCSTs. Consider again W_2 in Example 2, and another 2-Dyck path W_3 that can be reached from W_2 by a single adjacent move:

$W_2 =$,

$W_3 =$.

The adjacent move swaps the position 20 and 21. The corresponding NCST of W_3 is shown in Figure 2. Note that there are still a lot of altered edges, and in fact, one can also construct examples where $\Omega(n)$ edges get changed after an adjacent move. This time, however, the reader may have observed a pattern of the changes: the blue part in the NCST of W_3 is the same as the red part in that of W_2, but shifted clockwise once. As formally proved in Section 5.1, each adjacent move corresponds to at most two edge flips, plus one shifting in a particular form (see (9)).

To apply the path method for Markov chain comparison [23], we need to simulate an adjacent move with a sequence of flip moves, and to be able to recover the initial and final states of the adjacent move, given the two states of the flip move and at most polynomial amount of additional information. The main task is to simulate the shift. We do so by identifying a hierarchical structure of the shift, and only flip edges following a particular pattern. This design allows us to uniquely recover the initial and final states of the shift, as long as we know the current transition, the recursion depth, and constant amount of extra information. This is the crux of our whole argument and is given in Section 5.

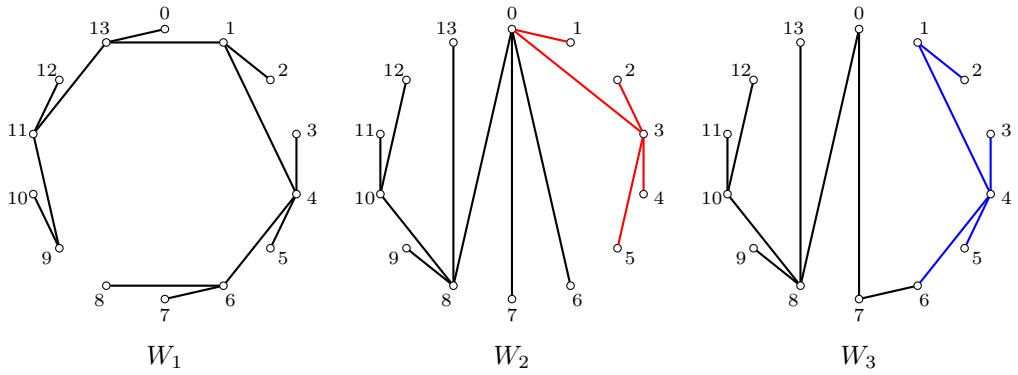


Figure 2 NCSTs corresponding to the 2-Dyck paths in Example 2.

Theorem 1 is shown by combining the comparison argument with the $O(n^3 \log n)$ mixing time of the adjacent move chain. The comparison has an overhead of $O(n^4)$, which, roughly speaking, consists of $O(n)$ for the encoding, $O(n)$ for the path length, and $O(n^2)$ to account for the transition probability difference. Finally, together with the $O(n)$ overhead resulting from spectral gaps, we obtain our mixing time bound of $O(n^8 \log n)$.

1.2 Related work

McShine and Tetali's mixing time argument for convex triangulations [38] is similar to our result in spirit, which is also built on a comparison argument with Wilson's lattice path chain [48]. However, in their case the starting point is 1-Dyck path and the argument is more straightforward. The idiosyncrasies of non-crossing spanning trees warrant a more complicated path construction.

We have mentioned that the down arrows of 2-Dyck paths form bases of a matroid. However it appears difficult to directly relate that matroid and related dynamics with the flip chain. Another very similar structure is the *graphic matroid*, whose bases are spanning trees of a graph. The bases-exchange walk for matroids is known to mix very rapidly [6, 19]. However, with the non-crossing restriction, even when the points are in convex position, NCSTs do *not* form the bases of a matroid if we take all possible edges as the ground set. To see this, we give an example where the basis exchange property of matroids fails. Consider 4 points $\{0, 1, 2, 3\}$ on a circle, a tree T_1 consisting of edges $(0, 1)$, $(1, 3)$ and $(2, 3)$, and another tree $T_2 = \{(0, 2), (1, 2), (2, 3)\}$. Remove the edge $(0, 1)$ from T_1. The possible choices of the new edge to add are $T_2 \setminus T_1 = \{(0, 2), (1, 2)\}$, but neither of them results in a valid non-crossing spanning tree, contradicting to the basis exchange property. Nonetheless, it is an interesting open problem to utilise techniques for the rapid mixing of graphic spanning trees to obtain better bounds in the non-crossing spanning trees context.

There has been a flurry of new tools building upon [6, 19] for analysing Markov chains, most notable among which is the spectral independence technique [5]. However, the results utilising spectral independence or its variants mostly focus on spin systems, and non-crossing trees are not spin systems. The trees all have the same size, which is a constraint usually not present in spin systems. There are some exceptions, such as the optimal mixing time of the down-up walk of independent sets of the same size [29]. These analyses rely heavily on the relationship between the fixed size distribution with the one without size restrictions. There is no similar connection to be utilised for non-crossing trees.

1.3 Open problems

The most straightforward open problem is to establish the correct order of the mixing time for the NCST flip chain. The $O(n^8 \log n)$ bound in Theorem 1 is unlikely to be tight. Our starting point, the adjacent move chain for 2-Dyck paths, has diameter $\Omega(n^2)$ by a simple potential argument,[1] and thus does not mix very fast. On the other hand, reconfiguration results [8, 10] imply that the flip chain has diameter $O(n)$. Therefore, we expect the flip chain to mix at least no slower than the adjacent move chain, instead of the polynomial slow-down in our current bound. On the other hand, there is no known lower bound on the mixing time other than the trivial $\Omega(n)$ obtained via the diameter bound. It would be interesting to close or shrink this gap.

Another interesting problem is to study the flip chain when points are *not necessarily* in convex position. Recall that the complexity of counting NCSTs with input points in general position, either exactly or approximately, is not known. Studying the flip chain would be helpful in resolving the approximate counting complexity. Constructing a set of points such that the flip chain mixes torpidly (i.e., not mixing in polynomial time) would rule out direct MCMC approaches, and often torpid-mixing instances help people construct gadgets for hardness reductions. On the other hand, if this chain always mixes in polynomial time, then efficient approximate counting would be possible.

2 Markov chain comparison

In this section we review some notions of discrete-time Markov chains over discrete state spaces. For detailed backgrounds we refer the reader to the book [36]. Let Ω be a finite discrete state space and π a distribution on Ω. Let $P : \Omega \times \Omega \to \mathbb{R}_{\geq 0}$ be the transition matrix of a Markov chain with stationary distribution π. We say P is reversible if $\pi(x)P(x, y) = \pi(y)P(y, x)$ for any $x, y \in \Omega$.

Given two distributions π and π' over Ω, the total variation distance between them is defined as $d_{\mathrm{TV}}(\pi, \pi') := \frac{1}{2}\sum_{x\in\Omega} |\pi(x) - \pi'(x)|$. For a Markov chain P with stationary distribution π, let $d(t) = \max_{x\in\Omega} \{d_{\mathrm{TV}}(P^t(x, \cdot), \pi)\}$. Then the mixing time of P is defined as $t_{\mathrm{mix}}(P) := \min\{t : d(t) \leq \frac{1}{2e}\}$. The constant $1/(2e)$ here is not usually important, as the error decays exponentially as t increases.

Let f, g be two real valued functions on Ω. The Dirichlet form of P with the stationary distribution π is

$$\mathcal{E}_P(f, g) := \frac{1}{2} \sum_{x,y\in\Omega} [f(x) - f(y)][g(x) - g(y)]\pi(x)P(x, y).$$

Define the variance of a function $f : \Omega \to \mathbb{R}$ by $\mathrm{Var}_\pi(f) := \mathbb{E}[f^2] - (\mathbb{E}[f])^2$. Then the *spectral gap* of the Markov chain P is

$$\lambda(P) := \inf \left\{ \frac{\mathcal{E}_P(f, f)}{\mathrm{Var}_\pi(f)} \mid f : \Omega \to \mathbb{R}, \mathrm{Var}_\pi(f) \neq 0 \right\}. \tag{2}$$

Equivalently, the spectral gap is the difference between the largest and second largest eigenvalues of the transition matrix of P, but the definition in (2) will be more convenient for us later.

[1] One can also consider how the area below the path evolves and show that the mixing time is at least $\Omega(n^3/\log n)$, so Wilson's bound is almost tight.

The mixing time of any reversible Markov chain P can be bounded using the inverse spectral gap (also called the *relaxation time*) (see e.g. [36, Theorem 12.4, 12.5]):

$$\frac{1}{\lambda(P)} - 1 \leq t_{\text{mix}}(P) \leq \frac{1}{\lambda(P)} \left(1 + \frac{1}{2} \log \pi_{\min}\right), \tag{3}$$

where $\pi_{\min} := \min_{x \in \Omega} \pi(x)$. This $\pi_{\min}$ term is usually a single inverse exponential, which means the mixing time bound obtained this way typically differs from the relaxation time by a linear factor.

2.1 Path method

The path method [23] is a way to compare two Markov chains P and $\widetilde{P}$, with stationary distributions π and $\widetilde{\pi}$, by simulating any transition in $\widetilde{P}$ using a series of transitions in P. This is a generalisation of the canonical path method by Jerrum and Sinclair [30].

For every pair of $x, y \in \Omega$, let $\Gamma_{xy} = (x = x_0, x_1, ..., x_{\ell-1}, x_\ell = y)$ be a path of length $\ell = |\Gamma_{xy}|$ in the state space where $P(x_{i-1}, x_i) > 0$ for all $i \in [\ell]$. The congestion ratio B for a collection of paths $\Gamma = \{\Gamma_{xy}\}_{x,y \in \Omega}$ is defined as

$$B := \max_{(x',y'):P(x',y')>0} \left(\frac{1}{\pi(x')P(x',y')} \sum_{\substack{x,y \\ \Gamma_{xy} \ni (x',y')}} \widetilde{\pi}(x)\widetilde{P}(x,y)|\Gamma_{xy}| \right). \tag{4}$$

▶ **Theorem 3** (Theorem 2.1 of [23]). *Let P and $\widetilde{P}$ be two reversible Markov chains. Let B be the congestion ratio for paths $\Gamma = \{\Gamma_{xy}\}$ using transitions in P. Then for all $f : \Omega \to \mathbb{R}$,*

$$\mathcal{E}_{\widetilde{P}}(f, f) \leq B\mathcal{E}_P(f, f).$$

Roughly speaking, the congestion bounds how much of a slowdown the path method can cause. It gives an upper bound on how much of a bottleneck can result from our simulated transitions.

3 Fuss-Catalan numbers and structures

This section introduces the combinatorial structures that we frequently utilise. These structures are closely related to the famous *Fuss-Catalan number*.

▶ Remark 4. Different authors appear to index Fuss-Catalan numbers differently in the literature. We adopt the definition used in Stanley's monograph [46].

3.1 Dyck paths

▶ **Definition 5** (k-Dyck path). *Let k be a positive integer. A sequence $(a_1, a_2, \cdots, a_{(k+1)n}) \in \{+1, -k\}^{(k+1)n}$ forms a* k-Dyck path *of length $(k+1)n$, if*

$$\forall j \in [(k+1)n], \qquad \sum_{i=1}^{j} a_i \geq 0, \qquad \text{and} \qquad \sum_{i=1}^{(k+1)n} a_i = 0.$$

Imagine that there is a hiking frog starting from the sea level. Each time it takes a step forward, it either lifts itself up by 1 meter, or drops by k meters. The two conditions are interpreted as (1) at any point it cannot go below the sea level, and (2) after $(k+1)n$ steps

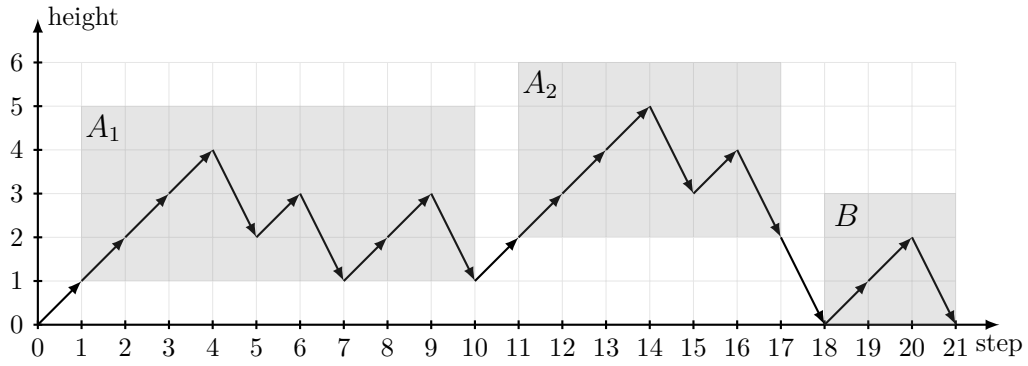


Figure 3 Decomposing Example 6 by Fact 7.

it must land on the sea level. This view leads to the following terminology for k-Dyck paths. We replace $+1$ with an *up-step* $\nearrow$, and $-k$ with a *down-step* $\searrow$. The partial sum

$$\mathrm{ht}(j) := \sum_{i=1}^{j} a_i$$

indicates the *height* of the frog after j steps. Below is an example of a valid 2-Dyck path of length 21.

▶ **Example 6.** $\nearrow\nearrow\nearrow\nearrow\searrow\nearrow\searrow\nearrow\nearrow\searrow\nearrow\nearrow\nearrow\nearrow\searrow\nearrow\searrow\searrow\nearrow\nearrow\searrow$ is a 2-Dyck path.

The following simple fact is useful later.

▶ **Fact 7.** *Any non-empty k-Dyck path W can be uniquely written in the form*

$$W = \nearrow A_1 \nearrow A_2 \cdots \nearrow A_k \searrow B,$$

where A_i's and B are (possibly empty) k-Dyck paths.

The decomposition can be obtained as follows. The B part begins at the position where the height *first* reaches 0 (except the starting point), namely $\mathsf{Start}(B) = \min\{j : \mathrm{ht}_W(j) = 0, j \geq 1\}$. Each A_i part begins at the position one step after where the height reaches $i-1$ for the *last* time before the B part, namely $\mathsf{Start}(A_i) = 1 + \max\{j : \mathrm{ht}_W(j) = i-1, 0 \leq j < \mathsf{Start}(B)\}$. See Figure 3 for an example of such a decomposition for a 2-Dyck path.

Fact 7 immediately implies the recurrence on the number of k-Dyck paths of length $(k+1)n$, denoted by $C_{k,n}$:

$$C_{k,0} = 1,\ C_{k,n} = \sum_{\substack{i_1,\cdots,i_{k+1}\in\mathbb{Z}_{\geq 0},\\ \sum_{j=1}^{k+1} i_j = n-1}} C_{k,i_1} C_{k,i_2} \cdots C_{k,i_{k+1}},\ \Longrightarrow\ C_{k,n} = \frac{1}{kn+1}\binom{(k+1)n}{n}. \quad (5)$$

This is called the *k-Fuss-Catalan number* [46, Chapter 4, A14], a generalisation of the Catalan number. It is well-known that the number of 1-Dyck paths equals the Catalan number.

3.2 Non-crossing spanning trees

Non-crossing spanning trees for points in convex position have a one-to-one correspondence with 2-Dyck paths. It will be more convenient to consider the following alternative way to draw the trees (see [45, Exercise 5.46]). Instead of putting points in convex position, we

arrange them on a line. Each edge is represented by a curve, drawn above the line segment. A tree is *non-crossing* if and only if it can be drawn this way without any two edges intersecting (except at the endpoints). See Figure 4(a)(b) for the two drawings of the same tree. In other words, each edge is represented by a pair of integers (a, b) where $0 \le a < b \le n$, and there does not exist any two edges (a, b) and (c, d) that $a < c < b < d$. As a result, any subtree must consist of points labeled by consecutive numbers. For this reason, if a subtree is spanned by $s, s+1, \cdots, t$, we use a shorthand $[s, t]$ for it. A subtree is empty (consists of no edges) if $s = t$.

▶ **Fact 8.** *Any n-edge non-empty non-crossing spanning tree T on points labeled $0, 1, \cdots, n$ can be uniquely decomposed into a tuple of three non-crossing trees (T_A, T_B, T_C), with the edge counts summing up to $n-1$, in the following way:*

- *Let t be the maximum index that there is an edge $(0, t)$.*
- *Let s be the index such that, after removing the edge $(0, t)$, the subtree $[0, s]$ is connected, but $[0, s+1]$ is not. In other words, there is a gap between s and $s+1$.*
- *The first tree T_A is the subtree $[0, s]$. It is empty if $s = 0$.*
- *The second tree T_B is the subtree $[s+1, t]$. It is empty if $t = s+1$.*
- *The last tree T_C is the subtree $[t, n]$. It is empty if $n = t$.*

Resembling Fact 7 for 2-Dyck paths, the above decomposition not only suggests exactly the same recurrence (1) for the number of n-edge non-crossing spanning trees, but also a *one-to-one correspondence* between 2-Dyck paths and non-crossing spanning trees. Compare Figure 4(c) and Figure 3 for instance.

▶ **Corollary 9.** *The following ruleset, when applied inductively, gives a one-to-one correspondence between 2-Dyck paths of length $3n$ and non-crossing spanning trees containing n edges.*

- *An empty path corresponds to an empty tree.*
- *If a tree T is decomposed into (T_A, T_B, T_C) by Fact 8, with each part corresponding to paths A, B and C respectively, then T corresponds to the path $\nearrow A \nearrow B \searrow C$.*

The decomposition has the following useful property.

▶ **Proposition 10.** *Let U and V be two 2-Dyck paths of length $3n_U$ and $3n_V$ respectively, and let T_U and T_V be their corresponding non-crossing spanning trees. Then, the non-crossing spanning tree T_{UV} corresponding to the concatenated Dyck path UV is a concatenation of T_U and T_V, meaning that the subtree $[0, n_U]$ of T_{UV} is T_U, and the subtree $[n_U, n_U + n_V]$ of T_{UV} is T_V (up to a shift in indexing).*

The proof of Proposition 10 can be found in the full version [4]. Inspired by the decomposition, we introduce the following terminology that will be useful later.

▶ **Definition 11.** *In the context of Fact 8:*

(pivot edge)	*the edge $(0, t)$ is called the* pivot edge *of T;*
(gap)	*the vertices between s and t are called the* gap *beneath the pivot edge $(0, t)$; the pivot edge is called the* overarching edge *of the gap;*
(overarching edge)	*an edge (a, b) is the* overarching edge *of another edge (c, d), if $a \le c < d \le b$, and there does not exist any other edge (a', b') that $a \le a' \le c < d \le b' \le b$;*
(minimal segment)	*a subtree $[c, d]$ is a* minimal segment *under an edge (a, b), if the edge (c, d) exists, and (a, b) is the overarching of (c, d); a subtree $[c, d]$ is an outmost minimal segment, if the edge (c, d) exists, and there is no overarching edge of (c, d).*

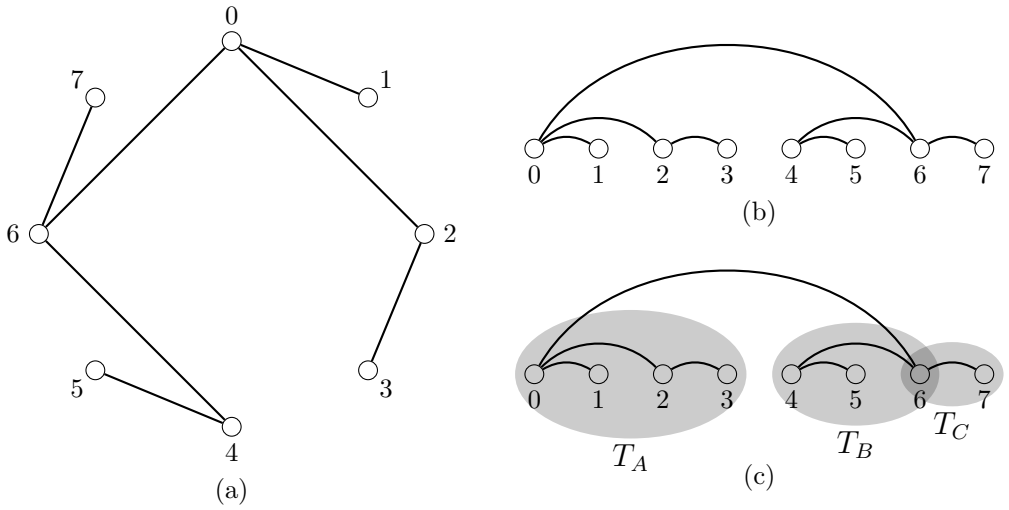


Figure 4 (a)(b) Two ways of representing a non-crossing spanning tree. The tree is corresponded to the 2-Dyck path in Example 6. (c) Decomposing the tree by Fact 8.

▶ **Example 12.** Consider the tree in Figure 4.

- The pivot edge of the whole tree is $(0, 6)$. The pivot edge of the subtree $[0, 3]$ is $(0, 2)$.
- The gap beneath the edge $(0, 6)$ is after 3. The gap beneath the edge $(4, 6)$ is after 5.
- The overarching edge of the edge $(0, 1)$ is $(0, 2)$, whose overarching edge is $(0, 6)$, which does not have any overarching edge.
- The minimal segments under the edge $(0, 6)$ are $[0, 2]$, $[2, 3]$ and $[4, 6]$. The outmost minimal segments are $[0, 6]$ and $[6, 7]$.

4 Spectral gap of the adjacent move chain

Over the state space of 2-Dyck paths of length $3n$, define the following *adjacent move chain*:

(1) starting from the current path X, choose a position $i \in [3n - 1]$;
(2) let X' be the same as X except that the steps at positions i and $i + 1$ are swapped;
(3) if X' is invalid, then the chain stays at X. Otherwise, it moves to X' with probability $1/2$ and stays at X with probability $1/2$.

Formally, let $B(X)$ be the set of 2-Dyck paths that can be obtained from X by swapping two adjacent coordinates. Trivially, $|B(X)| < n$. The transition probability of this chain is

$$P_{\mathrm{AM}}(X, X') = \begin{cases} 1 - \dfrac{B(X)}{6n-2} & \text{if } X = X'; \\ \dfrac{1}{6n-2} & \text{if } X' \in B(X); \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Note that $P_{\mathrm{AM}}(X, X) \geq 4/5$. It is easy to see that P_{AM} is irreducible and verify that $P(X, X') = P(X', X)$, so P_{AM} is reversible.

We have the following bound on the mixing time of this chain.

▶ **Theorem 13.** $t_{\mathrm{mix}}(P_{\mathrm{AM}}) = O(n^3 \log n)$.

The proof essentially follows Wilson's coupling argument [48]. We use the same potential function and verify that the key properties for the mixing time upper bound still hold. (The properties in [48] for the mixing time lower bound no longer hold.) More details are given in the full version [4]. Combining Theorem 13 and (3) has an immediate corollary.

▶ **Corollary 14.** $1/\lambda(P_{\mathrm{AM}}) = O(n^3 \log n)$.

5 Adjacent move versus flip move

In this section we analyse the mixing time of the flip chain by comparing it with the adjacent move chain and using Corollary 14. The flip chain P_{FM} is a random walk on non-crossing spanning trees (or non-crossing trees), where in each step we remove an edge uniformly at random, and add back a non-crossing edge between the two components. Such a move is called an (edge) flip. Formally, given two non-crossing spanning trees S and T, the probability of moving from S to T is

$$P_{\mathrm{FM}}(S,T) = \begin{cases} \displaystyle\sum_{T':\ |S\cap T'|=n-1} \frac{1}{n\delta(S,T')} & \text{if } S = T; \\ \dfrac{1}{n\delta(S,T)} & \text{if } |S\cap T| = n-1; \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $\delta(S,T)$ is the number of edges that one can add to $S \cap T$ to make it a non-crossing spanning tree. Note that

$$P_{\mathrm{FM}}(S,S) \geq 1/n, \quad \text{and } P_{\mathrm{FM}}(S,T) \geq \Omega(n^{-3}) \text{ if it is non-zero.} \tag{8}$$

It is easy to see that P_{FM} is irreducible. Note when $|X\cap Y| = n-1$, $P(X,Y) = \frac{1}{n\delta(S,T)} = \frac{1}{n\delta(T,S)} = P(Y,X)$, and it is trivial to see that $P(X,Y) = P(Y,X)$ in the other two cases, so P_{FM} is symmetric, thus is reversible.

5.1 Characterise the adjacent move via flip moves

A single adjacent move on non-crossing spanning trees still requires altering an $\Omega(n)$ number of edges. Yet, we will show that it consists of a constant number of flip moves, plus one subtree *shifting*, taking the form of

moving from to or vice versa, (9)

where the bubble does not change. Note that the overarching edge of the subtree remains unchanged. This is formally defined as follows, using the non-crossing tree decomposition.

▶ **Definition 15** (shifting). *Let T be a non-crossing spanning tree of size m decomposing into (T_A, T_B, T_C) by Fact 8, where T_B and T_C are empty. Let T' be another non-crossing spanning tree of size m decomposing into (T'_A, T'_B, T'_C), where both T'_A and T'_C are empty, and $T'_B = T_A$. A* shifting *is the transformation from T to T' (right shifting) or from T' to T (left shifting).*

This particular form of shifting may look strange at first sight. A more intuitive version would be just moving a subtree left or right, without caring about the overarching edge or the other side of the gap. However, as it will become clear later when we bound the congestion of the paths, it is this form of shifting that allows an efficient encoding.

Next is the characterisation of an adjacent move via flip moves and shifting.

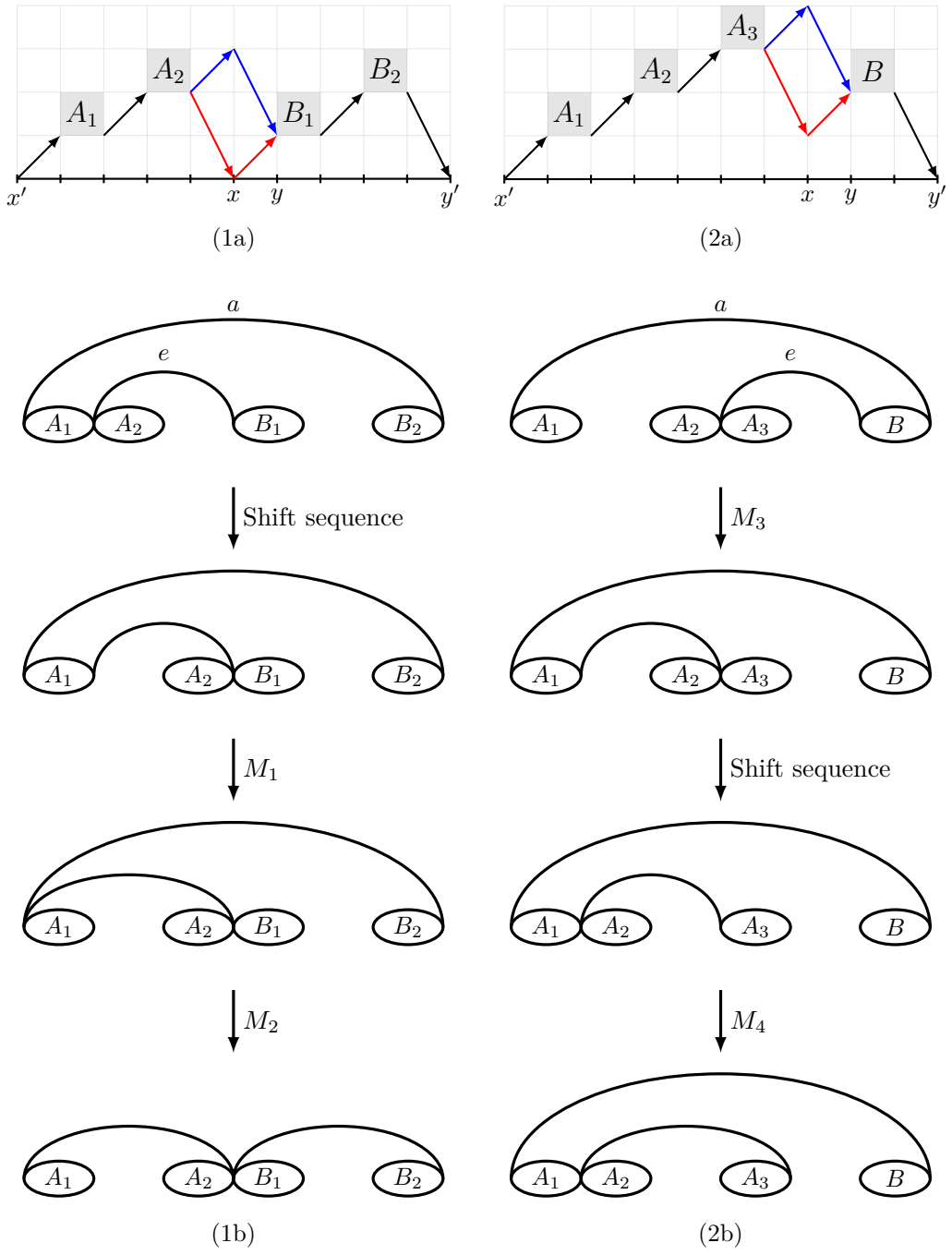


Figure 5 (∗a) The 2-Dyck path representations of $\tilde{I}$. In the adjacent move, the blue part is changed into red. (∗b) The corresponding flip move sequences. (1∗) Corresponding to **Type 1**. (2∗) Corresponding to **Type 2**.

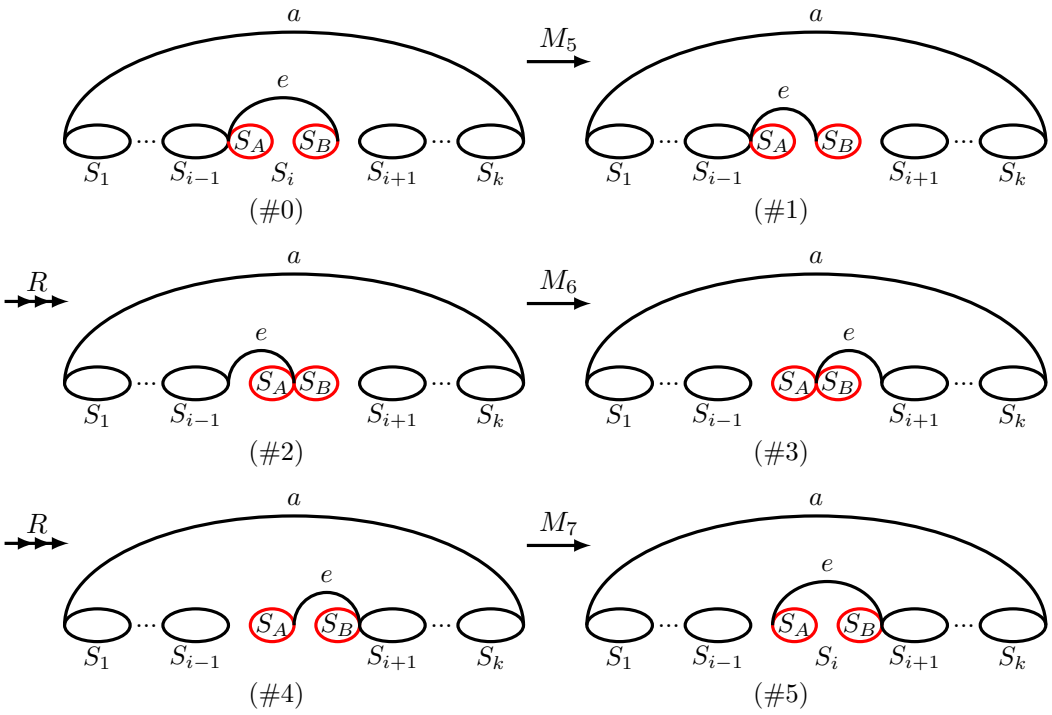


Figure 6 An example of a shift sequence. The black longer bubbles represent minimal segments, and there may be 0 or more of them. The red shorter bubbles represent valid sub-trees and are not necessarily minimal. The minimal segments $S_{i+1}, \ldots, S_k$ have finished moving, and $S_1, \ldots, S_{i-1}$ are yet to be moved, with S_i being the current one moving. M_5, M_6, M_7 are edge flips, and R represents a recursive move.

▶ **Lemma 16.** *Let $I \to F$ be a transition in the adjacent move P_{AM} such that $I \neq F$. This transition can be simulated from T_I to T_F by at most two edge flippings in P_{FM}, plus at most one shifting.*

The proof of Lemma 16 can be found in the full version [4]. There are two cases in this lemma, depending on whether one or two steps the first time the 2-Dyck path goes below the end of the adjacent move, which we call Type-1 and Type-2 moves. See Figure 5 (1a) and (2a) for illustrations of the two types of transitions. The flip moves to simulate the adjacent move are also intuitive. See the illustrations in Figure 5 (1b) and (2b).

The next step is to resolve the shifting, by designing a sequence of flippings that simulates it. We call this sequence of flip moves a *shift sequence*. The design of the shift sequence is the trickiest part of the whole comparison argument, because we have to be able to recover the initial and final states by just looking at a transition with some succinct extra encoding. We only need to deal with right shifting, as for left shifting we just reverse the sequence.

▶ **Lemma 17.** *Let T be a non-crossing spanning tree of m edges decomposing into (T_A, T_B, T_C) by Fact 8, where T_B and T_C are empty, and T_A is not empty. Then we can use $\ell \leq 3m$ flip moves to simulate a right shifting, i.e., reach a non-crossing spanning tree T' decomposing into (T'_A, T'_B, T'_C) where both T'_A and T'_C are empty, and $T'_B = T_A$.*

The proof of Lemma 17 can be found in the full version [4]. Essentially, we design a recursive way of simulating shifts. We decompose the subtree to shift into minimal segments, and then move them one at a time. To move a minimal segment, we perform edge flips so that subtrees of the minimal segment are in the form of a shift, and finish moving it using recursion. See Figure 6 for an illustration of an example.

Algorithm 1 Constructing the canonical path.

Input: two 2-Dyck paths $I \neq F$ of length $3n$ that $P_{\mathrm{AM}}(I, F) > 0$
Output: a canonical path $I = Z_0, Z_1, \cdots, Z_\ell$

Decide if I to F is a Type-1 or Type-2 move as in Lemma 16;
if *it is a Type-1 move* **then return** Transition1(I, F) **else return** Transition2(I, F);
procedure Transition1(I, F): `// Requiring` $I \to F$ `to be a Type-1 move.`
  Let $Z_0 \leftarrow I$;
  Find $a = (a_1, a_2)$, $e = (p, g_e + 1)$ and g_a as in Lemma 16;
  if $p \leq g_e - 1$ **then** let $(\ell - 2, Z_1, \cdots, Z_{\ell-2}) \leftarrow$ShiftRight$(p, g_e + 1, 1, 1)$;
  Perform an M_1 move of e to obtain $Z_{\ell-1}$;
  Let $Z_\ell \leftarrow F$, which is obtained by an M_2 move of e from $Z_{\ell-1}$;
  return $Z_0, \cdots, Z_\ell$;
procedure ShiftRight$(s, t, \mathsf{depth}, \mathsf{step})$:
  Let $k \geq 1$ be the number of minimal segments $S_1, \cdots, S_k$ beneath the edge (s, t);
  $\mathsf{last} \leftarrow \mathsf{step}$;
  for i *from* k *down to* 1 **do**
    Let $e = (p, q)$ be the pivot edge of S_i and g_e, g_{e+1} be the gap beneath e; `// (#0)`
    if $q \geq g_e + 2$ **then**
      Perform an M_5 move of e to obtain Z_{last}, and $\mathsf{last} \leftarrow \mathsf{last} + 1$; `// (#1)`
    if $p \leq g_e - 1$ **then**
      $(\ell', Z_{\mathsf{last}}, \cdots, Z_{\mathsf{last}+\ell'-1}) \leftarrow$ShiftRight$(p, g_e + 1, \mathsf{depth} + 1, \mathsf{last})$;
      Update $\mathsf{last} \leftarrow \mathsf{last} + \ell'$; `// (#2)`
    Perform an M_6 move of e to obtain Z_{last}, and update $\mathsf{last} \leftarrow \mathsf{last} + 1$; `// (#3)`
    if $q \geq g_e + 2$ **then**
      $(\ell'', Z_{\mathsf{last}}, \cdots, Z_{\mathsf{last}+\ell''-1}) \leftarrow$ShiftRight$(g_e + 1, q + 1, \mathsf{depth} + 1, \mathsf{last})$;
      Update $\mathsf{last} \leftarrow \mathsf{last} + \ell''$; `// (#4)`
    if $p \leq g_e - 1$ **then**
      Perform an M_7 move of e to obtain Z_{last}, and $\mathsf{last} \leftarrow \mathsf{last} + 1$; `// (#5)`
  return $(\mathsf{last} - \mathsf{step}, Z_{\mathsf{step}}, \cdots, Z_{\mathsf{last}})$;

5.2 Path construction

We are going to apply the path method, namely Theorem 3, to compare P_{AM} and P_{FM}. Because of the bijection between 2-Dyck paths and NCSTs, we simply treat P_{AM} and P_{FM} as having the same stationary distribution. To compare the two chains, we will need to construct a path (namely, a sequence of transitions) for any two 2-Dyck paths I and F where $P_{\mathrm{AM}}(I, F) > 0$ using only transitions in P_{FM}. We summarise the construction in Algorithm 1, with half of the procedure omitted due to redundancy. This construction follows closely the simulation of an adjacent move via two flip moves and one shifting in Lemma 16, and then simulating the shifting via flip moves in Lemma 17. The edges $a = (a_1, a_2)$, $e = (p, g_e + 1)$ and g_a in Transition1(I, F) are defined in the proof of Lemma 16. Intuitively, a and e are the distinguished edges marked in Figure 5, and $g_a, g_a + 1$ is the unique gap beneath a. The markers `(#0)` to `(#5)` are used in the proof of Lemma 19. We expand out the Type-1 move

and omit the details of the Type-2 move to avoid redundancy. (The flip moves for Type-2 moves are described in the proof of Lemma 16 and Figure 5(2∗), and the ShiftLeft sequence is the reverse of ShiftRight.)

Let T_I and T_F be the non-crossing spanning trees corresponding to I and F. The construction yields a path $T_I = Z_0 \to Z_1 \to \cdots \to Z_\ell = T_F$ where $P_{\mathrm{FM}}(Z_i, Z_{i+1}) > 0$ for any $i \le \ell - 1$. Its validity, namely that each transition is valid, is implied by Lemma 16 and Lemma 17. These two lemmas also imply an upper bound on the length of the path $\ell \le 3n + 2$ ($3n$ for the shift sequence and 2 for the two simple flips).

5.3 Bound the congestion

The goal of the Markov chain comparison argument is to show the following lemma:

▶ **Lemma 18.** *Let $\mathcal{E}_{\mathrm{AM}}$ be the Dirichlet form of the adjacent move chain P_{AM} over the uniform distribution of 2-Dyck paths of length $3n$, and $\mathcal{E}_{\mathrm{FM}}$ be the Dirichlet form of the flip chain P_{FM} over the uniform distribution of non-crossing spanning trees containing n edges. Then for all functions f, it holds that $\mathcal{E}_{\mathrm{FM}}(f, f) \ge \Omega(n^{-4}) \cdot \mathcal{E}_{\mathrm{AM}}(f, f)$.*

Lemma 18 is a straightforward application of Theorem 3 where we bound the congestion using Equation (8) and Corollary 20. The proof can be found in the full version [4].

The next lemma shows that our canonical path construction ensures that each transition in the flip move P_{FM} is used by a limited number of (I, F) pairs in the adjacent move P_{AM}. This is proved by designing an encoding such that we can uniquely recover the (I, F) pair from the current transition and the encoding. The number of (I, F) pairs is then bounded by the number of possible encodings, which we ensure to be linear in n.

▶ **Lemma 19** (encoding). *Let $Z \to Z'$ be a transition of P_{FM} over non-crossing spanning trees containing n edges. Let $I \ne F$ be two 2-Dyck paths such that $P_{\mathrm{AM}}(I, F) > 0$, and that the path constructed according to Algorithm 1 contains the transition $Z \to Z'$. Then there exists an injective mapping*

$$\varphi_{Z \to Z'} : \mathcal{T}_n \times \mathcal{T}_n \to \{\mathsf{Left}, \mathsf{Right}\} \times \{M_1, M_2, M_3, M_4, S_1, S_2\} \times [n]$$

where $\mathcal{T}_n$ is the set of all non-crossing spanning trees of n edges. In other words, given $Z \to Z'$, and (dir, M, d), we can uniquely determine I, F such that $\varphi_{Z \to Z'}(T_I, T_F) = (\mathsf{dir}, M, d)$, where $\mathsf{dir} \in \{\mathsf{Left}, \mathsf{Right}\}$, $M \in \{M_1, M_2, M_3, M_4, S_1, S_2\}$, and $d \in [n]$.

Proof sketch. We provide a proof sketch here, and the full proof can be found in the full version [4]. The mapping is defined as $\varphi_{Z \to Z'}(T_I, T_F) = (\mathsf{dir}, M, d)$, where

- dir indicates the direction of adjacent move from I to F,
- M indicates where the current transition $Z \to Z'$ is located in the construction of the canonical path, as in Algorithm 1, and
- d indicates the depth of the recursion of the shift sequence construction. It is defined only when $M \in \{S_1, S_2\}$, and otherwise arbitrary.

In particular, the six possibilities of M correspond to the six types of moves in Figure 5 (1b) and (2b), where $M_1, \cdots, M_4$ denote edge flips, and S_1 and S_2 denote shifting sequences.

To show the mapping is injective, we just need to be able to recover T_I and T_F given (dir, M, d). The transition $Z \to Z'$ gives the edge which is being moved in this step, and dir indicates if it is a left shifting or right shifting. First, we use M to determine where the current move is in the construction of the canonical path. If it is neither S_1 nor S_2, then the initial and final states can be reconstructed easily. Otherwise, the current step is in the

recursive construction of the canonical path. Starting from this edge being moved, together with the number d in the mapping above, we can find the sequence of overarching edges which the initial shifting involves. Furthermore, by looking at the current edge and the gap beneath its overarching edge, we can determine which step in the subroutine ShiftRight (or ShiftLeft) the current transition is in. This information allows us to rewind all the edge flips that have been performed and recover the initial state T_I. The final state T_F is recovered similarly. ◀

Let Γ_{IF} be the path from I to F. We write $(Z, Z') \in \Gamma_{IF}$ if the (Z, Z') transition is in the path Γ_{IF}. Lemma 19 immediately implies the following.

▶ **Corollary 20.** *For each (Z, Z') with $P_{\mathrm{FM}}(Z, Z') > 0$, there are at most $12n$ pairs (I, F) such that $(Z, Z') \in \Gamma_{IF}$.*

Proof of Theorem 1. We have

$$
\begin{aligned}
t_{\mathrm{mix}}(P_{\mathrm{FM}}) &\le \left(1 + \frac{1}{2}\log \pi_{\min}\right) \frac{1}{\lambda(P_{\mathrm{FM}})} = \left(1 + \frac{1}{2}\log \pi_{\min}\right) \frac{1}{\inf\left\{\frac{\mathcal{E}_{\mathrm{FM}}(f,f)}{\mathrm{Var}_\pi(f)}\right\}} && \text{(by (2))} \\
&\le \left(1 + \frac{1}{2}\log \pi_{\min}\right) \frac{O(n^4)}{\inf\left\{\frac{\mathcal{E}_{\mathrm{AM}}(f,f)}{\mathrm{Var}_\pi(f)}\right\}} && \text{(by Lemma 18)} \\
&= \left(1 + \frac{1}{2}\log \pi_{\min}\right) \frac{O(n^4)}{\lambda(P_{\mathrm{AM}})} && \text{(by (2))} \\
&= \left(1 + \frac{1}{2}\log \pi_{\min}\right) O(n^7 \log n) && \text{(by Corollary 14)} \\
&= O(n^8 \log n). && \blacktriangleleft
\end{aligned}
$$

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. *arXiv*, abs/2206.03879, 2022. doi:10.48550/arXiv.2206.03879.
2. Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting triangulations and other crossing-free structures via onion layers. *Discret. Comput. Geom.*, 53(4):675–690, 2015. doi:10.1007/S00454-015-9672-3.
3. Victor Alvarez, Karl Bringmann, Saurabh Ray, and Raimund Seidel. Counting triangulations and other crossing-free structures approximately. *Comput. Geom.*, 48(5):386–397, 2015. doi:10.1016/J.COMGEO.2014.12.006.
4. Konrad Anand, Weiming Feng, Graham Freifeld, Heng Guo, Mark Jerrum, and Jiaheng Wang. Rapid mixing of the flip chain over non-crossing spanning trees. *arXiv*, abs/2409.07892, 2024. doi:10.48550/arXiv.2409.07892.
5. Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *FOCS*, pages 1319–1330. IEEE, 2020. doi:10.1109/FOCS46700.2020.00125.
6. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *STOC*, pages 1–12. ACM, 2019. doi:10.1145/3313276.3316385.
7. Federico Ardila. The Catalan matroid. *J. Comb. Theory, Ser. A*, 104(1):49–62, 2003. doi:10.1016/S0097-3165(03)00121-3.
8. David Avis and Komei Fukuda. Reverse search for enumeration. *Discret. Appl. Math.*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

9 Elia Bisi, Piotr Dyszewski, Nina Gantert, Samuel G. G. Johnston, Joscha Prochno, and Dominik Schmid. Random planar trees and the Jacobian conjecture. *arXiv*, abs/2301.08221, 2023. arXiv:2301.08221.

10 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flipping non-crossing spanning trees. In *SODA*, pages 2313–2325. SIAM, 2025. doi:10.1137/1.9781611978322.77.

11 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of plane trees in convex geometric graphs. In *SoCG*, volume 293 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.22.

12 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Comput. Geom. Topol.*, 2(1):8:1–8:7, 2023. URL: https://www.cgt-journal.org/index.php/cgt/article/view/36.

13 Sergey Bravyi, Libor Caha, Ramis Movassagh, Daniel Nagaj, and Peter W. Shor. Criticality without frustration for quantum spin-1 chains. *Physical review letters*, 109(20):207202, 2012.

14 Thomas Budzinski. On the mixing time of the flip walk on triangulations of the sphere. *C. R. Math. Acad. Sci. Paris*, 355(4):464–471, 2017. doi:10.1016/j.crma.2017.02.011.

15 Pietro Caputo, Fabio Martinelli, Alistair Sinclair, and Alexandre Stauffer. Random lattice triangulations: structure and algorithms. *Ann. Appl. Probab.*, 25(3):1650–1685, 2015. doi:10.1214/14-AAP1033.

16 Alessandra Caraceni and Alexandre Stauffer. Polynomial mixing time of edge flips on quadrangulations. *Probab. Theory Related Fields*, 176(1-2):35–76, 2020.

17 Swee Hong Chan and Igor Pak. Computational complexity of counting coincidences. *Theor. Comput. Sci.*, 1015:114776, 2024. doi:10.1016/J.TCS.2024.114776.

18 Emma Cohen. *Problems in Catalan mixing and matchings in regular hypergraphs*. PhD thesis, Georgia Institute of Technology, 2016.

19 Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-Sobolev inequalities for strongly log-concave distributions. *Ann. Probab.*, 49(1):506–525, 2021.

20 Dimitrios I. Dais. Resolving 3-dimensional toric singularities. In *Geometry of toric varieties*, volume 6 of *Sémin. Congr.*, pages 155–186. Soc. Math. France, Paris, 2002.

21 Jesús A. De Loera, Jörg Rambau, and Francisco Santos. *Triangulations*, volume 25 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2010. Structures for algorithms and applications.

22 Mehmet Demirtas, Liam McAllister, and Andres Rios-Tascon. Bounding the Kreuzer-Skarke landscape. *Fortschr. Phys.*, 68(11-12):2000086, 13, 2020. doi:10.1002/prop.202000086.

23 Persi Diaconis and Laurent Saloff-Coste. Comparison theorems for reversible Markov chains. *Ann. Appl. Probab.*, 3(3):696–730, 1993.

24 Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991. doi:10.1145/102782.102783.

25 David Eppstein. Counting polygon triangulations is hard. *Discret. Comput. Geom.*, 64(4):1210–1234, 2020. doi:10.1007/S00454-020-00251-7.

26 David Eppstein and Daniel Frishberg. Improved mixing for the convex polygon triangulation flip walk. In *ICALP*, volume 261 of *LIPIcs*, pages 56:1–56:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICALP.2023.56.

27 I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants and multidimensional determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, Inc., Boston, MA, 2008. Reprint of the 1994 edition.

28 M. Carmen Hernando, Ferran Hurtado, Alberto Márquez, Mercè Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discret. Appl. Math.*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

29 Vishesh Jain, Marcus Michelen, Huy Tuan Pham, and Thuy-Duong Vuong. Optimal mixing of the down-up walk on independent sets of a given size. In *FOCS*, pages 1665–1681. IEEE, 2023. doi:10.1109/FOCS57990.2023.00101.

30 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989. doi:10.1137/0218077.

31 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993. doi:10.1137/0222066.

32 J. Jurkiewicz, A. Krzywicki, and B. Petersson. A numerical study of discrete Euclidean Polyakov surfaces. *Physics Letters B*, 168(3):273–278, 1986. doi:10.1016/0370-2693(86)90978-0.

33 Volker Kaibel and Günter M. Ziegler. Counting lattice triangulations. In *Surveys in combinatorics, 2003 (Bangor)*, volume 307 of *London Math. Soc. Lecture Note Ser.*, pages 277–307. Cambridge Univ. Press, Cambridge, 2003.

34 Marek Karpinski, Andrzej Lingas, and Dzmitry Sledneu. A QPTAS for the base of the number of crossing-free structures on a planar point set. *Theor. Comput. Sci.*, 711:56–65, 2018. doi:10.1016/J.TCS.2017.11.003.

35 V.A. Kazakov, I.K. Kostov, and A.A. Migdal. Critical properties of randomly triangulated planar random surfaces. *Physics Letters B*, 157(4):295–300, 1985. doi:10.1016/0370-2693(85)90669-0.

36 David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, 2017.

37 Russell A. Martin and Dana Randall. Sampling adsorbing staircase walks using a new Markov chain decomposition method. In *FOCS*, pages 492–502. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892137.

38 Lisa McShine and Prasad Tetali. On the mixing time of the triangulation walk and other Catalan structures. In *Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 147–160. DIMACS/AMS, 1997. doi:10.1090/DIMACS/043/09.

39 Michael Molloy, Bruce Reed, and William Steiger. On the mixing rate of the triangulation walk. In *Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 179–190. DIMACS/AMS, 1997. doi:10.1090/DIMACS/043/11.

40 Ramis Movassagh. The gap of Fredkin quantum spin chain is polynomially small. *Ann. Math. Sci. Appl.*, 3(2):531–562, 2018. doi:10.4310/AMSA.2018.v3.n2.a5.

41 Ramis Movassagh and Peter W. Shor. Supercritical entanglement in local systems: Counterexample to the area law for quantum matter. *Proceedings of the National Academy of Sciences*, 113(47):13278–13282, 2016. doi:10.1073/pnas.1605716113.

42 Marc Noy. Enumeration of noncrossing trees on a circle. *Discret. Math.*, 180(1-3):301–313, 1998. doi:10.1016/S0012-365X(97)00121-0.

43 Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electron. J. Comb.*, 18(1), 2011. doi:10.37236/557.

44 Micha Sharir, Adam Sheffer, and Emo Welzl. On degrees in random triangulations of point sets. *J. Comb. Theory, Ser. A*, 118(7):1979–1999, 2011. doi:10.1016/J.JCTA.2011.04.002.

45 Richard P. Stanley. *Enumerative combinatorics. Vol. 2*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1999.

46 Richard P. Stanley. *Catalan numbers*. Cambridge University Press, New York, 2015.

47 O. Ya. Viro. Real plane algebraic curves: constructions with controlled topology. *Algebra i Analiz*, 1(5):1–73, 1989.

48 David Bruce Wilson. Mixing times of Lozenge tiling and card shuffling Markov chains. *Ann. Appl. Probab.*, 14(1):274–325, 2004. doi:10.1214/aoap/1075828054.

Geometric Realizations of Dichotomous Ordinal Graphs

Patrizio Angelini
John Cabot University, Rome, Italy

Sabine Cornelsen
University of Konstanz, Germany

Carolina Haase
Trier University, Germany

Michael Hoffmann
Department of Computer Science,
ETH Zürich, Switzerland

Eleni Katsanou
National Technical University of Athens, Greece

Fabrizio Montecchiani
University of Perugia, Italy

Raphael Steiner
ETH Zürich, Switzerland

Antonios Symvonis
National Technical University of Athens, Greece

Abstract

A *dichotomous ordinal graph* consists of an undirected graph with a partition of the edges into *short* and *long* edges. A *geometric realization* of a dichotomous ordinal graph G in a metric space X is a drawing of G in X in which every long edge is strictly longer than every short edge. We call a graph G *pandichotomous* in X if G admits a geometric realization in X for every partition of its edge set into short and long edges.

We exhibit a very close relationship between the degeneracy of a graph G and its *pandichotomic Euclidean or spherical dimension*, that is, the smallest dimension k such that G is pandichotomous in $\mathbb{R}^k$ or the sphere $\mathcal{S}^k$, respectively. First, every d-degenerate graph is pandichotomous in $\mathbb{R}^d$ and $\mathcal{S}^{d-1}$ and these bounds are tight for the sphere and for $\mathbb{R}^2$ and almost tight for $\mathbb{R}^d$, for $d \geq 3$. Second, *every* n-vertex graph that is pandichotomous in $\mathbb{R}^k$ has at most μkn edges, for some absolute constant $\mu < 7.23$. This shows that the pandichotomic Euclidean dimension of any graph is linearly tied to its degeneracy and in the special case $k \in \{1, 2\}$ resolves open problems posed by Alam, Kobourov, Pupyrev, and Toeniskoetter.

Further, we characterize which complete bipartite graphs are pandichotomous in $\mathbb{R}^2$: These are exactly the $K_{m,n}$ with $m \leq 3$ or $m = 4$ and $n \leq 6$. For general bipartite graphs, we can guarantee realizations in $\mathbb{R}^2$ if the short or the long subgraph is constrained: namely if the short subgraph is outerplanar or a subgraph of a rectangular grid, or if the long subgraph forms a caterpillar.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Mathematics of computing → Graph theory; Human-centered computing → Graph drawings

Keywords and phrases Ordinal embeddings, geometric graphs, graph representations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.9

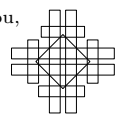
Related Version *Full Version*: arxiv.org/abs/2503.07361

Acknowledgements This work was initiated at the Annual Workshop on Graph and Network Visualization (GNV 2023), Chania, Greece, June 2023.

1 Introduction

For an *ordinal embedding*, we are given a set of objects $x_1, \ldots, x_n$ in an abstract space, together with a set of *ordinal constraints* of the form $\text{dist}(x_i, x_j) < \text{dist}(x_k, x_\ell)$. The objective is to compute a set of points $p_1, \ldots, p_n$ in a metric space while preserving as many ordinal constraints as possible. Ordinal embeddings were first studied in the 1960's by Shepard [20, 21] and Kruskal [13, 14] in the context of psychometric data analysis. Recently,

© Patrizio Angelini, Sabine Cornelsen, Carolina Haase, Michael Hoffmann, Eleni Katsanou, Fabrizio Montecchiani, Raphael Steiner, and Antonios Symvonis;
licensed under Creative Commons License CC-BY 4.0



there have been applications in the field of Machine Learning [23]. The computation of ordinal embeddings is also known in the literature as *non-metric multi-dimensional scaling*. For an extensive literature review on ordinal embeddings refer to [24].

Of particular interest in relation to our work is the application of ordinal embeddings to the problem of recognizing multidimensional preferences [5, 7, 19] in the field of Computational Social Science. The objects are either *voters* or *options*, which, together with the ordinal constraints (i.e., the voters' preferences), naturally define a bipartite graph. However, the goal is to find an embedding in $\mathbb{R}^d$ where all constraints are satisfied rather than to seek for an approximation. Efficient algorithms exist when $d = 1$ [6, 7], while for any $d \geq 2$ the problem is as hard as the existential theory of the reals [19]. In the simplest case, the preferences form a *dichotomy*, that is, a voter may either support or reject an option [9, 19].

This setting is naturally modeled as a *dichotomous ordinal graph*, which consists of an undirected graph $G = (V, E_s \cup E_\ell)$ together with a partition of the edges into a set E_s of *short* edges and a set E_ℓ of *long* edges. A *geometric realization* of a dichotomous ordinal graph G is a drawing Γ of G in some metric space along with a threshold $\delta > 0$ such that the short edges of G are exactly those that have length at most δ in Γ. In this work we consider two natural classes of drawings in which edges are drawn as geometrically shortest paths: (1) straight-line drawings in the Euclidean space $\mathbb{R}^d$ and (2) geodesic drawings on the sphere $\mathcal{S}^d$. Figure 1 shows two straight-line drawings of the same dichotomous ordinal graph in $\mathbb{R}^2$. The drawing in (a) is a valid geometric realization, whereas the drawing in (b) is not.

A dichotomous ordinal graph may or may not be realizable in any particular space. For instance, it is easy to see that K_4 where the short edges induce a star cannot be realized in $\mathbb{R}^1$. In general, it is NP-hard to decide whether a dichotomous ordinal graph admits a geometric realization in $\mathbb{R}^2$, even if the underlying graph is a complete graph and the short edges induce a planar graph [3, Lemma 1]. According to Peters [19, Theorem 4] the problem is $\exists\mathbb{R}$-complete for bipartite graphs.[1]

(a) valid drawing of $\triangle uvw$. **(b)** invalid drawing of $\triangle uvw$.

Figure 1 A dichotomous ordinal triangle. Short edges are shown blue/solid, long edges red/dashed.

In light of the motivation, it is desirable that a geometric realization always exists, no matter the preferences of the voters. We call a graph $G = (V, E)$ *pandichotomous* in a metric space X if G admits a geometric realization in X for every partition of E into short and long edges. Graphs that are pandichotomous in $\mathbb{R}^1$ are also called total-threshold-colorable [1, 2] or total weak unit interval graphs [3]. Angelini, Bekos, Gronemann, and Symvonis [4] exhibited some graph classes that are pandichotomous in $\mathbb{R}^2$, such as *double-wheels* (a cycle and two

[1] But even the full version of the paper [18] does not contain an explicit proof but only refers to the work of Kang and Müller [12, Theorem 1]. The graph constructed in this proof is not bipartite. While it is plausible that the construction could be adapted, it does not seem immediately obvious to us.

additional vertices connected to all vertices of the cycle), *2-degenerate* graphs (can be reduced to the empty graph by repeatedly removing a vertex of degree at most two), and *subcubic* graphs (vertex degree at most three). On the negative side, there exists a dichotomous ordinal graph G whose underlying graph can be obtained from a double-wheel by adding a single edge such that G does not admit a geometric realization in $\mathbb{R}^2$ [4]. Clearly, being pandichotomous is a *monotone* graph property, that is, if a graph G is pandichotomous in X, then so is every subgraph of G. For $X = \mathbb{R}^d$, being pandichotomous is also closed under taking disjoint unions, but for $X = \mathcal{S}^d$ this is not immediate.

A related question is the existence of a realization of a graph as a *unit disk graph*, where vertices are represented by unit disks, and they are connected by an edge if and only if the corresponding disks intersect. More generally, the *sphericity* [10, 15] of a graph G is the smallest d such that G can be realized as an intersection graph of unit balls in $\mathbb{R}^d$. The main difference between geometric realizations of dichotomous ordinal graphs and unit ball realizations lies in the different types of edges. In a unit disk realization there are only two types: edge and non-edge, and all of them have to be faithfully represented. In a geometric realization of dichotomous ordinal graphs there are three types of edges: long, short, and non-edges, and we have no constraints concerning the last type. Therefore, only upper bounds on the sphericity carry over to the dichotomous setting. Since every graph on n vertices has sphericity at most n [15], every graph on n vertices is also pandichotomous in $\mathbb{R}^n$. So for every finite graph G we can define its *pandichotomic Euclidean dimension* $\mathrm{ped}(G)$ to be the smallest dimension d such that G is pandichotomous in $\mathbb{R}^d$. Analogously, we define the *pandichotomic spherical dimension* $\mathrm{psd}(G)$. Trivially, $\mathrm{ped}(G) \leq \mathrm{psd}(G) + 1$.

Results. In this paper, we initiate the study of the pandichotomic dimension of graphs, with a focus on the bipartite case. In Section 3, we characterize the complete bipartite graphs that are pandichotomous in $\mathbb{R}^2$. Specifically, we show that $K_{m,n}$ is pandichotomous in $\mathbb{R}^2$ if and only if either (1) $m \leq 3$ or (2) $m = 4$ and $n \leq 6$.

A dichotomous ordinal graph has two natural induced subgraphs: The *short* subgraph, induced by the short edges, and the *long* subgraph, induced by the long edges. In Section 4, we show that if either of these subgraphs is sufficiently constrained, then a realization in $\mathbb{R}^2$ always exists. Specifically, this is the case if (1) the graph is bipartite and the short subgraph is outerplanar (Theorem 8); (2) the short subgraph forms a subgraph of a rectangular grid (Theorem 9); or (3) the long subgraph forms a caterpillar (Theorem 10). In (1) and (2), we can also ensure that the short subgraph is realized without edge crossings. However, there are bipartite dichotomous ordinal graphs that do not admit a geometric realization in $\mathbb{R}^2$, even though the short subgraph is planar (see, e.g., Lemma 6).

In Section 5, we study the pandichotomic Euclidean dimension and show that it is very closely related to the degeneracy. A graph is *k-degenerate* if it can be constructed starting from the empty graph by iteratively applying the following operation: Add a new vertex and make it adjacent to at most k existing vertices. The *degeneracy* $\mathrm{d}(G)$ of a graph G is the smallest k such that G is k-degenerate.

We show that all d-degenerate graphs are pandichotomous on $\mathcal{S}^{d-1}$ and in $\mathbb{R}^d$, for $d \geq 2$ (Theorem 11). In particular, it follows that all bipartite planar graphs are pandichotomous on $\mathcal{S}^2$ and in $\mathbb{R}^3$ (Corollary 12). Our bounds imply $\mathrm{ped}(G) \leq \mathrm{d}(G)$ and $\mathrm{psd}(G) \leq \mathrm{d}(G) - 1$, for every graph G with $\mathrm{d}(G) \geq 2$. We also show that these bounds are tight for the sphere (Corollary 14) and for $\mathbb{R}^2$ (Theorem 15) and almost tight for $\mathbb{R}^d$, for $d \geq 3$ (Theorem 13).

We also show that *every* n-vertex graph that is pandichotomous in $\mathbb{R}^d$ has at most μdn edges, for some constant $\mu < 7.23$ (Theorem 16), and this bound is optimal up to the value of μ. In the special cases $d \in \{1, 2\}$, this affirmatively answers two open problems posed explicitly by

Alam, Kobourov, Pupyrev, and Toeniskoetter [3]. Consequently, $\mathrm{d}(G)/(2\mu) \leq \mathrm{ped}(G) \leq \mathrm{d}(G)$ and $\mathrm{d}(G)/(2\mu) - 1 \leq \mathrm{psd}(G) \leq \mathrm{d}(G) - 1$, for every graph G (Corollary 19). In other words, the pandichotomic Euclidean and spherical dimensions are linearly tied to the degeneracy. As a direct consequence of these bounds (Corollary 20), we determine up to a constant factor the smallest dimension d for which every n-vertex (bipartite) dichotomous ordinal graph is realizable in $\mathbb{R}^d$ (or $\mathcal{S}^{d-1}$).

2 Preliminaries

For two points $p, q \in \mathbb{R}^d$ we denote by $\|p - q\|$ the Euclidean distance between p and q. For a point $c \in \mathbb{R}^d$ and a positive real number r, the *ball* $\mathrm{B}(c, r)$ of radius r around c is the set $\{p \in \mathbb{R}^d \colon \|p - c\| \leq r\}$ of all points that have Euclidean distance at most r to c. For a set $A \subset \mathbb{R}^d$ we denote by ∂A the *boundary* of A. The boundary of $\mathrm{B}(c, r)$ is formed by the *sphere* $\mathrm{S}(c, r) = \{p \in \mathbb{R}^d \colon \|p - c\| = r\}$. A *unit* ball or sphere has a radius of $r = 1$. The *geodesic distance* $\mathrm{d}_\gamma(p, q) \in [0, 2\pi]$ between two points $p, q \in \mathcal{S}_d$ is determined by the central angle of a shortest great circle arc between p and q.

Given a finite set X of geometric objects, such as hyperplanes or spheres, in $\mathbb{R}^d$, the *arrangement* $\mathcal{A}(X)$ of X is the partition of $\mathbb{R}^d$ induced by X into so-called cells of various dimension. The maximal connected open subsets of $\mathbb{R}^d \setminus \bigcup_{R \in X} R$ are the *d-cells* of $\mathcal{A}(X)$. And every relatively open k-dimensional intersection of two or more d-cells forms a k-cell of $\mathcal{A}(X)$. The d-cells are also called *full-dimensional cells* or even just *cells* of $\mathcal{A}(X)$. The 0-cells and 1-cells are also called *vertices* and *edges* of $\mathcal{A}(X)$, respectively.

For geometric realizations of dichotomous ordinal graphs in $\mathbb{R}^d$ we may fix a global scale and assume without loss of generality a threshold of $\delta = 1$. Furthermore, as we consider finite graphs only, we may assume that no two vertices have distance exactly one.

▶ **Observation 1.** *If a dichotomous ordinal graph admits a geometric realization in $\mathbb{R}^d$, then it admits a realization in $\mathbb{R}^d$ where no two vertices have distance exactly one.*

In contrast, such a global rescaling does not work in general for geometric realizations on $\mathcal{S}^d$. If the short subgraph of a dichotomous ordinal graph has several connected components, then we can draw these components mutually far apart. This yields the following.

▶ **Observation 2.** *A dichotomous ordinal graph admits a geometric realization in $\mathbb{R}^d$ if and only if each subgraph induced by a connected component of its short subgraph does.*

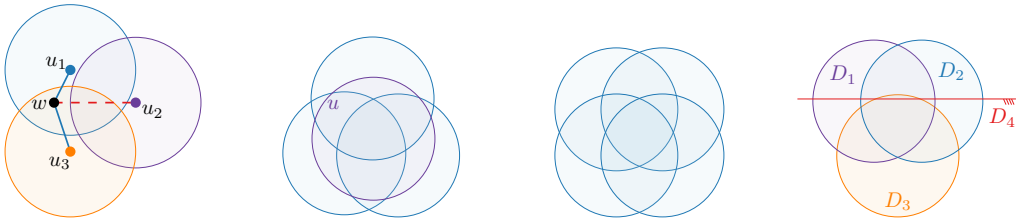
3 Complete Bipartite Graphs

In this section, we prove the following theorem. The proof is split into Lemmas 4, 6, and 7, in combination with the fact that being pandichotomous is a monotone graph property.

▶ **Theorem 3.** *The complete bipartite graph $K_{m,n}$ is pandichotomous in $\mathbb{R}^2$ if and only if either (1) $m \leq 3$ or (2) $m = 4$ and $n \leq 6$.*

A convenient way to reason about geometric realizations for bipartite graphs is in terms of arrangements of spheres. Consider a bipartite dichotomous ordinal graph $G = (U \cup W, E)$ and suppose that the vertices of U are already drawn as points in $\mathbb{R}^d$. Then, to obtain a geometric realization for G the task is to place each $w \in W$ such that for each $u \in U$ with $uw \in E$ we have $w \in \mathrm{B}(u, 1)$ if and only if the edge uw is short; see Figure 2a.

Let $U = \{u_1, \ldots, u_n\}$, let $D_i = \mathrm{B}(u_i, 1)$, let $C_i = \partial D_i$, and let $\mathcal{C}$ denote the arrangement of $C_1, \ldots, C_n$. To every vertex $w \in W$ we associate a set $V(w) \subseteq U$ such that $u \in V(w)$ if and only if uw is a short edge in G. We refer to $V(w)$ as a *singleton*, a *pair*, or a *triple*



(a) $K_{3,m}$ **(b)** $K_{4,m}$ all pairs **(c)** $K_{4,m}$ all triples **(d)** arbitrary radii

Figure 2 Arrangements of circles to represent one color class of a bipartite graph in $\mathbb{R}^2$.

if $|V(w)| = 1$, 2, or 3, respectively. A subset $X \subseteq U$ is *realized* by a drawing of U if there is a cell r in $\mathcal{C}$ such that $r \subseteq D_i$ if and only if $u_i \in X$. Then G admits a geometric realization if and only if there exists a drawing of U where $V(w)$ is realized, for all $w \in W$.

▶ **Lemma 4.** *The complete bipartite graph $K_{3,m}$ is pandichotomous in $\mathbb{R}^2$, for all $m \in \mathbb{N}$. The complete bipartite graph $K_{4,m}$ is pandichotomous in $\mathbb{R}^2$, for all $1 \leq m \leq 6$.*

Proof. For $K_{3,m}$ we can draw $U = \{u_1, u_2, u_3\}$ so that all eight subsets of U are realized; see Figure 2a. Therefore, any dichotomous ordinal $K_{3,m}$ admits a geometric realization. For $|U| \geq 4$ such a universal placement is not possible because an arrangement of n circles has at most $n(n-1)+2$ cells [22]. So an arrangement of four circles has at most 14 cells, whereas a four-element set has 16 subsets. However, for $|U| = 4$ and $|W| \leq 6$ we can always obtain a geometric realization as follows. Let $V(W) = \{V(w) \colon w \in W\} \subset 2^U$.

If there are at least three pairs in $V(W)$, then, given that $|V(W)| \leq |W| \leq 6$, the number of triples plus the number of singletons in $V(W)$ together is at most three. Thus, as $|U| = 4$, there exists a vertex $u \in U$ such that $\{u\} \notin V(W)$ and $U \setminus \{u\} \notin V(W)$. So we can use the drawing depicted in Figure 2b, where we assign u to the central disk. As all subsets of U other than $\{u\}$ and $U \setminus \{u\}$ are realized, this is a valid geometric realization of G.

Otherwise, there are at most two pairs in $V(W)$. We use the drawing depicted in Figure 2c, where we assign the vertices of U to the disks so that both pairs in $V(W)$ appear consecutively in the circular order of disks. (This works regardless of whether or not these pairs share a vertex.) As all subsets of U other than the two pairs that correspond to opposite circles in the drawing are realized, this is a valid geometric realization of G. ◀

The following elementary lemma about unit disks turns out helpful.

▶ **Lemma 5.** *Let D, E be unit disks in $\mathbb{R}^2$, let c be a chord of D, and let A be a closed part of D bounded by c and ∂D whose interior A° does not contain the center of D. Then $c \subset E \iff A \subset E$.*

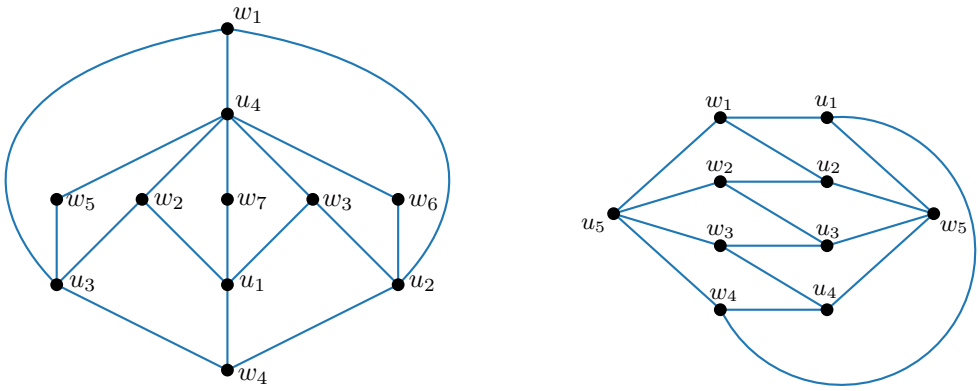
▶ **Lemma 6.** *There exists a dichotomous ordinal $K_{4,7}$ that is not realizable in $\mathbb{R}^2$.*

Proof Sketch. Let $U = \{u_1, u_2, u_3, u_4\}$ and $W = \{w_1, \ldots, w_7\}$ denote the vertex partition. For each w_i, we can specify an associated set $U_i \subseteq U$ (such that exactly the edges between w_i and U_i are short; see Figure 3a). We choose all four subsets of size three and the three subsets of size two that contain u_4, and distribute them among the vertices of W arbitrarily. In any geometric representation, each set U_i corresponds to a cell in the induced arrangement $\mathcal{C}$ of unit circles. Two more cells are required implicitly: The outer cell, which corresponds to $\emptyset \subset U$, and a cell that corresponds to the whole set U and is required by Helly's Theorem [11] because disks are convex and we specified all triples to be among the sets U_i.

Using these properties of $\mathcal{C}$ we can show that it cannot be realized using unit circles. It is crucial that the circles must have the same radius: If arbitrary radii are allowed, then a geometric realization exists (Figure 2d). ◀

▶ **Lemma 7.** *There exists a dichotomous ordinal $K_{5,5}$ that is not realizable in $\mathbb{R}^2$.*

Proof. First we specify a dichotomous ordinal $K_{5,5}$, see Figure 3 for an illustration.



(a) short edges of $K_{4,7}$ **(b)** short edges of $K_{5,5}$

Figure 3 A dichotomous ordinal $K_{4,7}$ and $K_{5,5}$, respectively, that does not admit a geometric realization. The drawn edges are the short edges. Edges between vertices labeled u on one hand and w on the other hand, that are not drawn, are long.

Let $U = \{u_1, u_2, u_3, u_4, u_5\}$ and $W = \{w_1, w_2, w_3, w_4, w_5\}$ denote the partition of the vertex set. To each $w_i \in W$, for $1 \leq i \leq 5$, we associate a set $\alpha(w_i) \subseteq U$ as follows:

$$\alpha(w_i) = \{u_i, u_{i\oplus 1}, u_5\}, \text{for } 1 \leq i \leq 4, \text{ and } \alpha(w_5) = U \setminus \{u_5\},$$

where $i \oplus 1 = (i \bmod 4) + 1$. Now consider an arbitrary but fixed geometric realization of this dichotomous ordinal $K_{5,5}$. In a slight abuse of notation we identify the vertices with the corresponding points in the geometric realization. Denote by D_i the unit disk centered at u_i, which represents the region of points that are in short distance to u_i. Let $W^- = \{w_1, w_2, w_3, w_4\}$. As $W^- \subset D_5$, whereas $w_5 \notin D_5$, by convexity of D_5 all points of W^- lie in an open halfplane through w_5. Let $\phi_1, \phi_2, \phi_3, \phi_4$ denote the counterclockwise order of the points from W^- around w_5 within this halfplane. By symmetry of W^- we may assume that $\phi_1 = w_1$ without loss of generality. We consider two cases.

Case 1: The set $\alpha(\phi_4)$ contains one of u_1 or u_2. Then by symmetry between u_1 and u_2 we may assume without loss of generality that $\phi_4 = w_4$. See Figure 4 (left) for illustration. By convexity of D_1, the triangle $\triangle = w_1w_4w_5 = \phi_1\phi_4w_5$ is contained in D_1. As $w_2, w_3 \notin D_1$, we have $\phi_2, \phi_3 \notin \triangle$. As $w_2, w_3, w_5 \in D_3$ but $w_1, w_4 \notin D_3$, the circle ∂D_3 crosses the line segment w_1w_4 twice in D_5, and it also crosses both of the line segments $\phi_1\phi_2$ and $\phi_3\phi_4$. Similarly, as $\triangle \subset D_1$ but $w_2, w_3 \notin D_1$, the circle ∂D_1 crosses both line segments w_5w_2 and w_5w_3, and it also crosses both of the line segments $\phi_1\phi_2$ and $\phi_3\phi_4$. Let c denote the chord of D_3 induced by $\phi_1\phi_4$. As $D_1 \supset c$, by Lemma 5 we know that D_1 contains the part A of D_3 on the side of c that does not contain the center of D_3. This is the part that contains w_5 because the other part contains $\phi_2, \phi_3 \notin D_1$. Similarly, as $D_5 \supset c$, by Lemma 5 it follows that also $D_5 \supset A$, which is a contradiction to $w_5 \notin D_5$. Thus, this case is impossible.

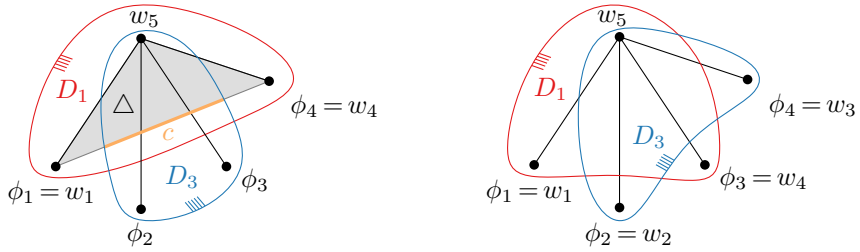


Figure 4 The two cases in the proof of Lemma 7.

Case 2: The set $\alpha(\phi_4)$ does not contain any of u_1 or u_2. Then we have $\phi_4 = w_3$ and both $|\alpha(\phi_1) \cap \alpha(\phi_3)| = 1$ and $|\alpha(\phi_2) \cap \alpha(\phi_4)| = 1$. By symmetry we may assume without loss of generality that $\phi_2 = w_2$ and $\phi_3 = w_4$. See Figure 4 (right) for illustration. Consider the two disks D_1 and D_3. They both contain w_5, and so the corresponding circles ∂D_1 and ∂D_3 intersect all of the rays $w_5\phi_i$, for $1 \leq i \leq 4$. As $\phi_1, \phi_3 \in D_1 \setminus D_3$ and $\phi_2, \phi_4 \in D_3 \setminus D_1$, the rays $w_5\phi_1$ and $w_5\phi_3$ intersect ∂D_1 before ∂D_3, whereas the rays $w_5\phi_2$ and $w_5\phi_4$ intersect ∂D_3 before ∂D_1. But then ∂D_1 and ∂D_3 cross in each of the cones $\phi_i w_5 \phi_{i+1}$, for $1 \leq i \leq 3$, which is impossible, since any two distinct circles intersect in at most two points.

As we arrived at a contradiction in both cases, we conclude that no geometric realization of this dichotomous ordinal $K_{5,5}$ exists. If, however, we allow disks with arbitrary radii, then a geometric realization exists, as depicted in Figure 5. ◀

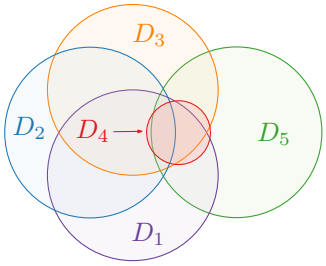


Figure 5 A realization of the dichotomous ordinal $K_{5,5}$ from Lemma 7 by disks of arbitrary radii.

4 Graphs with Constrained Short or Long Subgraphs

We show that every bipartite dichotomous ordinal graph admits a geometric realization if the short subgraph G_s is outerplanar or a subgraph of the rectangular grid or if the long subgraph G_ℓ is a caterpillar. In the first case, we construct a plane drawing of G_s in which the BFS-layers are drawn on horizontal lines (Figure 7b). In the second case, we suitably perturb the grid (Figure 6). And in the third case, we suitably place points on $\mathcal{S}^2$.

▶ **Theorem 8.** *A bipartite dichotomous ordinal graph admits a geometric realization if the subgraph induced by the short edges is outerplanar.*

Proof Sketch. Let $G = (V, E_s \cup E_\ell)$ be a bipartite dichotomous ordinal graph such that $G_s = (V, E_s)$ is outerplanar. By Observation 2, we may assume that G_s is connected. We root G_s at an arbitrary vertex r. Let V_k, $k = 0, \ldots$, be the BFS layers of G_s rooted at r, i.e., $V_0 = \{r\}$, V_1 is the set of neighbors of r, and V_{k+1}, $k \geq 1$ is the set of neighbors of the vertices in V_k that are not already in V_{k-1}. We say that w is a *child* of v and v is a *parent* of w if vw is an edge of G_s, $v \in V_k$ and $w \in V_{k+1}$ for some k. By outerplanarity, each vertex has at most two parents. We construct a planar drawing of G_s with the following properties.

- The root r is drawn with y-coordinate $y_0 = 0$. All vertices in layer V_k, $k > 0$ are on a horizontal line ℓ_k with y-coordinate y_k strictly between $k-1$ and k.
- The distance between a vertex and its children is at most 1 while the distance between two vertices on consecutive layers is greater than 1 if they are not adjacent in G_s.
- For each vertex v there is a vertical strip S_v such that
  - v is in S_v,
  - S_w is contained in the union of the strips of w's parents,
  - S_u and S_v are internally disjoint if u and v are on the same layer.

Special care has to be taken if a vertex $w \in V_{k+1}$ has two parents u and v, i.e., if w closes an internal face. In that case, we want to draw w on line ℓ_{k+1}, on the common boundary of S_u and S_v, and with distance exactly one to both u and v. ◀

▶ **Theorem 9.** *A dichotomous ordinal graph $G = (V, E_s \cup E_\ell)$ admits a geometric realization if the set of short edges induces a subgraph of the grid.*

Proof Sketch. Extend $G_s = (V, E_s)$ by the remaining grid edges and require the new edges to be long. Use the construction in Figure 6 to place the vertices. Then the short edges are shorter than $n^2 + 1/2$, while the long edges have length at least $n^2 + 1$. Scale the drawing. ◀

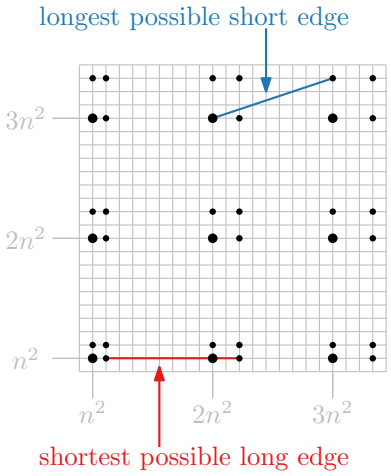
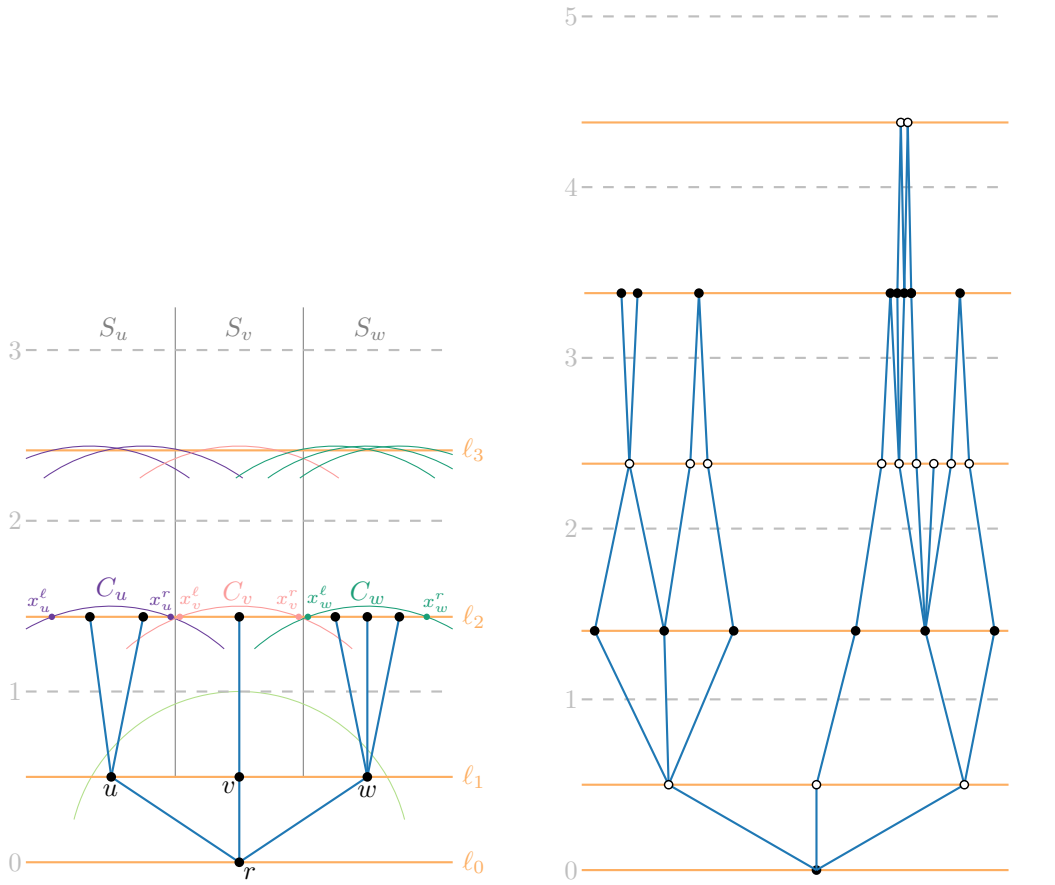


Figure 6 For each grid point (i, j), $1 \le i \le n$, $1 \le j \le n$ there are four possible points. If $i > 1$, the x-coordinate is in^2 if the edge between $(i-1, j)$ and (i, j) is short and $in^2 + i$ otherwise. If $j > 1$, the y-coordinate is jn^2 if the edge between $(i, j-1)$ and (i, j) is short and $jn^2 + j$ otherwise.

▶ **Theorem 10.** *A dichotomous ordinal graph admits a geometric realization in $\mathbb{R}^2$ and on $\mathcal{S}^2$ if each component of the long subgraph forms a caterpillar.*

Proof Sketch. We draw the vertices on a circle C of radius $\frac{1}{2} + \epsilon$, for some small $\epsilon > 0$. For each point $p \in C$ there is a region $R(p) \subset C$ around its antipodal point $\overline{p}$ such that for a point $q \in C$ we have $||p - q|| > 1 \iff q \in R(p)$. We draw the vertices $v_1, \ldots, v_k$ on the spine such that for $i = 1, \ldots, k-1$ the vertex v_{i+1} lies at the counterclockwise boundary of $R(v_i)$. Then we place the leaves (if any) attached to v_i within a very short arc around the point v_i' close to $\overline{v_i}$. More precisely, we obtain v_i' by shifting $\overline{v_i}$ counterclockwise along C by a small angle that monotonically increases with i. In this way, we ensure that the leaves of v_i are far from v_i but close to the leaves of the neighbors of v_i along the spine. ◀



(a) construction for a tree

(b) geometric realization when G_s is the graph in (c)

(c) bipartite outerplanar graph

Figure 7 How to construct a geometric realization of a bipartite dichotomous ordinal graph if the short edges induce an outerplanar graph.

5 Pandichotomic Dimension and Degeneracy

In this section, we exhibit a strong connection between the degeneracy of a graph and its pandichotomic dimension. Among others, we show that the pandichotomic dimension is bounded both from above and from below by a linear function of the degeneracy, and we give bounds on the coefficients in these linear functions.

▶ **Theorem 11.** *Every d-degenerate graph, for $d \geq 2$, is pandichotomous on $\mathcal{S}^{d-1}$ and in $\mathbb{R}^d$.*

Proof. Let G be a d-degenerate dichotomous ordinal graph, for $d \geq 2$, and let $u_1, \ldots, u_n$ be a vertex ordering such that u_i has at most d neighbors in $\{u_1, \ldots, u_{i-1}\}$, for $1 \leq i \leq n$. To obtain a geometric realization of G we place the vertices on the sphere $S = \{p \in \mathbb{R}^d \colon \|p\| = \sqrt{2}/2\}$ such that for every d-tuple of vertices the corresponding vectors are linearly independent. For a vertex $p \in S$ denote by $\vec{v}_p$ the corresponding vector; the points in distance less than one to p on S form a hemisphere, which consists of all points $q \in S$ such that $\vec{v}_p \vec{v}_q > 0$, where $\vec{v}_p \vec{v}_q$ denotes the scalar product between $\vec{v}_p$ and $\vec{v}_q$.

We place the vertices $u_1, \ldots, u_n$ in this order. The first vertex u_1 is placed arbitrarily on S. Then for each u_i, for $2 \leq i \leq n$, we have a set $Q = \{q_1, \ldots, q_t\} \subset S$ of points, which correspond to the $t \leq d$ neighbors of u_i in $u_1, \ldots, u_{i-1}$. Further, for each point in Q we know whether it should be close to or far from u_i. Now we need to find a suitable point on S that satisfies these constraints. We obtain the set $Q' = \{q'_1, \ldots, q'_t\}$ from Q by setting $q'_j = q_j$, if q_j should be close to u_i and setting q'_j to the point antipodal to q_j on S if q_j should be far from u_i. As Q is linearly independent, so is Q'. Furthermore, the linear system $\vec{v}_{q'_j} \vec{x} = \vec{1}$, for $1 \leq j \leq t$, has a (unique) solution $\vec{x}_0$. Normalizing $\vec{x}_0$ we obtain a point $r \in S$, such that $\vec{v}_r \vec{v}_{q'_j} > 0$, for all $1 \leq j \leq t$. Thus, we can place u_i at r such that all constraints with respect to Q are satisfied. The points on S that lie in a k-dimensional subspace spanned by k vertices, for $k < d$, can easily be avoided when selecting r. In this way we ensure that for all d-tuples of vertices on S the corresponding vectors are linearly independent. This realization also works for geodesic distances on $\mathcal{S}^{d-1}$ if we take an arc with angle $\pi/2$ as a unit. ◀

▶ **Corollary 12.** *Every bipartite planar graph is pandichotomous on $\mathcal{S}^2$ and in $\mathbb{R}^3$.*

Proof. Let $G = (V, E)$ be a planar bipartite graph on n vertices. As a consequence of Euler's formula, we have $|E| \leq 2n - 4$. By the handshaking lemma $\sum_{v \in V} \deg(v) = 2|E| \leq 4n - 8$ and, thus, the average degree in G is strictly less than 4. Consequently, every planar bipartite graph is 3-degenerate, and the statement follows by Theorem 11. ◀

We contrast the upper bound on the pandichotomic dimension in terms of the degeneracy provided by Theorem 11 by an almost matching lower bound in the Euclidean (Theorem 13) and a matching lower bound in the spherical case (Corollary 14), even for bipartite graphs. Note that every lower bound example is not just sporadic but spans an infinite family of examples, given that being pandichotomous is a monotone graph property.

▶ **Theorem 13.** *For every $d \geq 2$, there exists a $(d+2)$-degenerate bipartite graph that is not pandichotomous in $\mathbb{R}^d$.*

Proof. We build a graph $G = (V, E)$ starting with a set V_0 of $d + 2$ isolated vertices. Then for each of the 2^{d+2} many choices of assigning short or long to the vertices of V_0 we add a new vertex and connect it by edges to V_0 accordingly. Observe that G is bipartite and $(d + 2)$-degenerate by construction. Consider a geometric realization of G and the corresponding arrangement $\mathcal{A}$ of the $d + 2$ unit spheres centered at the vertices of V_0. In order to bound the number of full-dimensional cells in $\mathcal{A}$, we consider the standard parabolic lifting map λ, which orthogonally projects a point in $\mathbb{R}^d$ to the unit paraboloid in $\mathbb{R}^{d+1}$. This map has a number of useful properties, see, for instance, Edelsbrunner's book [8, Section 1.4]. Specifically, the image $\lambda(S)$ of a sphere $S \subset \mathbb{R}^d$ lies on a hyperplane $h_S \subset \mathbb{R}^{d+1}$ such that a point $p \in \mathbb{R}^d$ lies inside S if and only if its image $\lambda(p)$ lies below h_S. It follows that we may regard $\mathcal{A}$ as an arrangement of $d + 2$ hyperplanes in $\mathbb{R}^{d+1}$. As such an arrangement has at most $\sum_{i=0}^{d+1} \binom{d+2}{i} = 2^{d+2} - 1 < 2^{d+2}$ many full-dimensional cells [16, Proposition 6.1.1], for at least one of the vertices in $V \setminus V_0$ its distance requirements with respect to V_0 are violated. Thus, there is no geometric realization of G in $\mathbb{R}^d$. ◀

▶ **Corollary 14.** *For every $d \geq 2$, there exists a $(d+1)$-degenerate bipartite dichotomous ordinal graph that is not realizable on $\mathcal{S}^{d-1}$.*

Proof. We use the same graph G as in the proof of Theorem 13, except that we start with a set V_0 of $d+1$ rather than $d+2$ isolated vertices. We consider $\mathcal{S}^{d-1}$ as a unit sphere $S \subset \mathbb{R}^d$. For a point $p \in S$, the set of points on S in distance at most some fixed unit from p can be expressed as an intersection $S \cap u_p$ where u_p is a halfspace.

Suppose there is some geometric realization of G, and let $P \subset S$ denote the set of $d+1$ points that represent the vertices of V_0 in this realization. Consider the set H of the $d+1$ bounding hyperplanes of the halfspaces u_p, for $p \in P$. The arrangement $\mathcal{A}$ of H has at most $\sum_{i=0}^{d} \binom{d+1}{i} = 2^{d+1} - 1 < 2^{d+1}$ many full-dimensional cells [16, Proposition 6.1.1]. Thus, for at least one of the vertices in $V \setminus V_0$ there is no cell that satisfies its distance requirements with respect to V_0. It follows that there is no geometric realization of G in $\mathcal{S}^{d-1}$. ◀

In the Euclidean case, we give a tight lower bound in $\mathbb{R}^2$, albeit not for bipartite graphs.

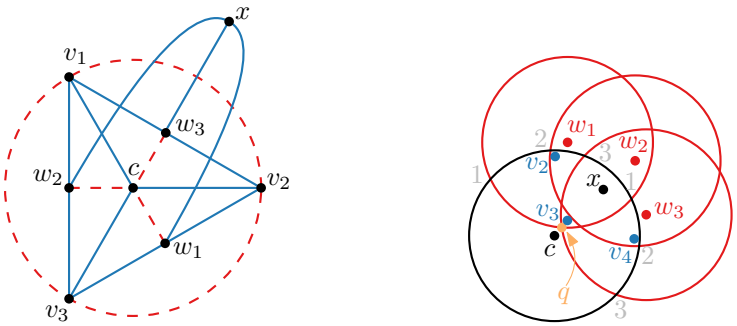


Figure 8 A 3-degenerate graph that cannot be realized in the plane along with a sketch of the proof. Long edges are dashed. The three points v_1, v_2, v_3 would have to lie in the intersection of the disk $B(c,1)$ and a unit disk within $B(w_1,1) \cup B(w_2,1) \cup B(w_3,1)$. But there is not enough space for three points with pairwise distance at least one.

▶ **Theorem 15.** *There exists a 3-degenerate graph that is not pandichotomous in $\mathbb{R}^2$.*

Proof. We build a dichotomous ordinal graph G on eight vertices as follows (Figure 8). Start with a set $V = \{v_1, v_2, v_3\}$ and a complete graph on V, all edges long. Then insert a vertex c and connect it to all vertices in V by a short edge. Next, insert a set $W = \{w_1, w_2, w_3\}$ of vertices as follows: the vertex w_i, for $i \in \{1,2,3\}$, is connected to c by a long edge and to each vertex in $V_i^- := V \setminus \{v_i\}$ by a short edge. Finally, we add a vertex x that is connected to all vertices in W by a short edge. Observe that G is 3-degenerate as constructed.

Consider any geometric realization Γ of G. In a slight abuse of notation we identify the vertices of G with the corresponding points that represent them in Γ. For a set $P \subset \mathbb{R}^2$ denote $I(P) = \bigcap_{p \in P} \mathrm{B}(p,1)$ and $U(P) = \bigcup_{p \in P} \mathrm{B}(p,1)$. As $x \in I(W)$, we have $I(W) \neq \emptyset$ and $U(W)$ is simply connected. The constraints imposed by G enforce $c \notin U(W)$ and $V \subset \mathrm{B}(c,1) \cap U(W)$. In particular, for each $i \in \{1,2,3\}$, the set V_i^- is contained in the lens $L_i = I(\{c, w_i\})$. As $c \notin U(W)$, each L_i has diameter strictly smaller than $\sqrt{3}$. Any lens in $\mathbb{R}^2$ of diameter strictly smaller than $\sqrt{3}$ contains at most two points at a pairwise distance at least one. Thus, in particular, we have $v_i \notin L_i$, for all $i \in \{1,2,3\}$.

We analyze the interaction of the circles $S_i = \mathrm{S}(w_i, 1)$, for $i \in \{1,2,3\}$, with $C = \mathrm{S}(c,1)$. As $|L_i \cap V| \geq 2$ and as any two points in V are at distance strictly larger than one, we have $|S_i \cap C| = 2$, for all $i \in \{1,2,3\}$. As $c \notin U(W)$ and as $U(W)$ is simply connected, some

part of C is disjoint from $U(W)$. We use such a part to break up the circular sequence of intersection points between C and the S_i into a linear sequence $\mathcal{I}$ of six points (some but not all of which may coincide). On the one hand, as $|L_i \cap V| \geq 2$, for all $i \in \{1,2,3\}$, and as $|V| = 3$, we have $L_i \cap L_j \neq \emptyset$, for all $i,j \in \{1,2,3\}$. On the other hand, if $L_i \subseteq L_j$, for some $i,j \in \{1,2,3\}$ with $i \neq j$, then, given that $v_j \in V_i$ we also have $v_j \in L_j$, a contradiction. It follows that up to the indexing of W we have $\mathcal{I} = 1,2,3,1,2,3$, where we write an integer i to indicate an intersection $C \cap S_i$.

Let q denote the intersection point of $S_1 \cap S_3$ in $\mathrm{B}(c,1)$. Note that $q \notin L_2$ because $q \in L_2$ would imply $L_1 \cap L_3 \subseteq L_2$ and thus $v_2 \in L_2$, a contradiction. We continuously move a unit disk D starting from $D = \mathrm{B}(w_2,1)$ towards c such that the center d of D is on the line segment w_2c. We stop the movement as soon as $d \in \mathrm{S}(q,1)$. This process is well defined because $q \notin \mathrm{B}(w_2,1)$ and $q \in L_1 \cap L_3 \subset \mathrm{B}(c,1)$. Furthermore, during the whole movement we have $L_2 \subset D \subset U(W)$ and the disk D' at the end of the movement also contains $L_1 \cap L_3$. Thus, the lens $L = D' \cap \mathrm{B}(c,1)$ contains all of V and given that $L \subset U(W)$ it has diameter strictly smaller than $\sqrt{3}$, a contradiction. ◀

The examples discussed above demonstrate that the bound from Theorem 11 is tight in the worst case, that is, for *some* graphs. In the following, we will show that the bound is also tight for *all* graphs, up to a multiplicative constant. We first estimate the edge density of pandichotomous graphs in $\mathbb{R}^d$ and show that it is linearly bounded in d. In fact, we prove a stronger result, namely that every sufficiently dense graph G not only is not pandichotomous in $\mathbb{R}^d$, but in fact, asymptotically *almost all* partitions of the edge-set of G into short and long edges yield dichotomous ordinal graphs that are not realizable in $\mathbb{R}^d$.

▶ **Theorem 16.** *Let $d \in \mathbb{N}$, let $\varepsilon > 0$, and let $G = (V,E)$ be a graph with n vertices and m edges. Let $c < 7.182$ denote the unique positive root of the function $x - 3 - x \cdot H(1/x)$, where $H\colon (0,1) \to \mathbb{R}$ is the binary entropy function $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$.*

If $m > (c+\varepsilon)dn$, then for asymptotically almost all[2] partitions $E = E_s \cup E_\ell$ of the edge-set of G, the dichotomous ordinal graph $(V, E_s \cup E_\ell)$ has no geometric realization in $\mathbb{R}^d$. In particular, every graph that is pandichotomous in $\mathbb{R}^d$ has at most $(c+o(1))dn$ edges. The $o(1)$ term approaches zero quickly as $dn \to \infty$. For illustration, we also show that $m < \mu dn$, for all $n \in \mathbb{N}$ and $d \geq 2$ and some explicit constant $\mu < 7.23$.

In the proof of Theorem 16, we will use the following result by Warren [25] on the number of sign-patterns of a collection of multivariate polynomials.

▶ **Theorem 17** (Warren [25, Theorem 2])**.** *Let $q_1, \ldots, q_m$ be real polynomials in N variables, each of degree at most d. Then the number of connected components of the set*

$$\left\{ \mathbf{x} \in \mathbb{R}^N \;\middle|\; \forall i \in \{1,\ldots,m\} : q_i(\mathbf{x}) \neq 0 \right\}$$

is at most

$$2(2d)^N \sum_{k=0}^{N} 2^k \binom{m}{k}.$$

The following is an immediate consequence of the above.

[2] That is, a fraction of the partitions that tends to 1 as $dn \to \infty$.

▶ **Corollary 18.** *Let $q_1, \ldots, q_m$ be real polynomials in N variables, each of degree at most d and let $\mathcal{S} := \{(\mathrm{sgn}(q_1(\mathbf{x})), \ldots, \mathrm{sgn}(q_m(\mathbf{x}))) \,|\, \forall i \in \{1, \ldots, m\} : q_i(\mathbf{x}) \neq 0\} \subseteq \{+1, -1\}^m$. Then we have*

$$|\mathcal{S}| \leq 2(2d)^N \sum_{k=0}^{N} 2^k \binom{m}{k}.$$

Proof. Let $X := \{\mathbf{x} \in \mathbb{R}^N \,|\, \forall i \in \{1, \ldots, m\} : q_i(\mathbf{x}) \neq 0\}$. Consider the map $f : X \to \{+1, -1\}^m$, defined by $f(\mathbf{x}) = (\mathrm{sgn}(q_1(\mathbf{x})), \ldots, \mathrm{sgn}(q_m(\mathbf{x})))$. It is easy to see, using the continuity of the polynomials $q_1, \ldots, q_m$, that f is continuous. Let C be a topological component of X. The restriction of f to C is a continuous map on a connected topological space that takes on only finitely many (at most 2^m) values. Since every such map is constant, we find that f is constant on every component of X. Hence, the size of $\mathcal{S} = \mathrm{im}(f)$ is bounded by the number of components of X. The statement now follows from Theorem 17. ◀

Proof of Theorem 16. Assume w.l.o.g. that $G = (V, E)$ has vertex set $V = \{1, \ldots, n\}$. Let $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be an n-tuple of points in $\mathbb{R}^d$, no two of which are at distance exactly one. The *distance-pattern* of $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ w.r.t. G is the map $s_G(\mathbf{x}_1, \ldots, \mathbf{x}_n) : E \to \{+1, -1\}$, where

$$s_G(\mathbf{x}_1, \ldots, \mathbf{x}_n)(ij) := \begin{cases} +1, & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| > 1, \\ -1, & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < 1, \end{cases}$$

for every edge $ij \in E$. In other words, given a configuration of points $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in $\mathbb{R}^d$, the distance-pattern captures the information of which edges of G in a geometric realization placing vertices $1, \ldots, n$ at points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, respectively, are long or short.

Next, we estimate the size of the set $\mathcal{S}$ of all possible distance-patterns for G. Concretely, $\mathcal{S}$ is the set of all mappings $s_G(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ ranges over all n-tuples of points in $\mathbb{R}^d$, no two of which are at distance 1. Let us denote by M the number of partitions $E_s \cup E_\ell = E$ of the edge-set of G such that the dichotomous ordinal graph $(V, E_s \cup E_\ell)$ is realizable in $\mathbb{R}^d$. Let p denote the fraction of such partitions compared to all possible partitions of E. Note that $M = p2^m$ and that $p = 1$ if and only if G is pandichotomous.

▷ Claim 1. $|\mathcal{S}| \geq M$.

Proof of Claim 1. Consider any one of the M partitions $E_s \cup E_\ell$ of the edge-set E for which $(V, E_s \cup E_\ell)$ is realizable in $\mathbb{R}^d$. By Observation 1 we may assume a threshold of $\delta = 1$ and that we use a set of points with no unit distances. The distance-pattern of the corresponding tuple of n points in d-space then has value $+1$ for every long and -1 for every short edge.

Since any two distinct partitions of E into short and long edges result in different distance-patterns in $\mathcal{S}$ this way, it follows that $|\mathcal{S}| \geq M$, as claimed. ◁

Next, we obtain an upper bound on $|\mathcal{S}|$ using Theorem 17.

▷ Claim 2. $|\mathcal{S}| \leq 2 \cdot 4^{dn} \cdot \sum_{k=0}^{dn} 2^k \binom{m}{k}$.

Proof of Claim 2. For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have $\|\mathbf{x} - \mathbf{y}\| > 1$ if and only if $\|\mathbf{x} - \mathbf{y}\|^2 - 1 > 0$, and $\|\mathbf{x} - \mathbf{y}\| < 1$ if and only if $\|\mathbf{x} - \mathbf{y}\|^2 - 1 < 0$. Further, the expression

$$\|\mathbf{x} - \mathbf{y}\|^2 - 1 = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_d - y_d)^2 - 1$$

is a polynomial of degree two in the coordinates of $\mathbf{x}, \mathbf{y}$. It is therefore natural to associate with every edge $ij \in E$ of G a polynomial $q_{i,j}$, defined as $q_{i,j}(\mathbf{x}_1, \ldots, \mathbf{x}_n) := \|\mathbf{x}_i - \mathbf{x}_j\|^2 - 1$. The collection $\left(q_{i,j} \,\middle|\, ij \in E\right)$ then forms a set of m polynomials, each of degree 2, in the $N := dn$ variables corresponding to the coordinates of $\mathbf{x}_1, \ldots, \mathbf{x}_n$. For every tuple $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of points in $\mathbb{R}^d$, no two at unit distance, we have $s_G(\mathbf{x}_1, \ldots, \mathbf{x}_n)(ij) = \operatorname{sgn}(q_{i,j}(\mathbf{x}_1, \ldots, \mathbf{x}_n))$.

Hence, the set $\mathcal{S}$ defined before coincides with the set $\mathcal{S}$ defined in Corollary 18 for the collection $\left(q_{i,j} \,\middle|\, ij \in E\right)$ of m multivariate polynomials. We therefore obtain, using the degree bound 2 for the polynomials,

$$|\mathcal{S}| \leq 2 \cdot (2 \cdot 2)^N \cdot \sum_{k=0}^{N} 2^k \binom{m}{k} = 2 \cdot 4^{dn} \cdot \sum_{k=0}^{dn} 2^k \binom{m}{k}. \qquad \lhd$$

Using Claims 1 and 2, we obtain the inequality

$$p2^m = M \leq |\mathcal{S}| \leq 2 \cdot 4^{dn} \cdot \sum_{k=0}^{dn} 2^k \binom{m}{k}.$$

Since $c > 2$, we have $m \geq (c+\varepsilon)dn > 2dn$. This implies that the term $\binom{m}{k}$ is monotonically increasing for $k = 0, 1, \ldots, dn$. Hence, we may bound the right hand side above by

$$2 \cdot 4^{dn} \cdot \binom{m}{dn} \cdot \sum_{k=0}^{dn} 2^k = 2 \cdot 4^{dn} \cdot \binom{m}{dn} \cdot (2^{dn+1} - 1) < 4 \cdot 8^{dn} \cdot \binom{m}{dn}.$$

Let $x := \frac{m}{dn} \geq c + \varepsilon$. Using the estimate $\binom{a}{\lambda a} \leq 2^{aH(\lambda)}$, which holds for any $a \in \mathbb{N}$ and $\lambda \in (0,1)$, see e.g., [17, Lemma 9.2], we find

$$p2^{xdn} = p2^m < 4 \cdot 8^{dn} 2^{xdn \cdot H(1/x)}.$$

Taking logarithms and dividing by dn this simplifies to

$$x < \frac{2 - \log_2(p)}{dn} + 3 + x \cdot H(1/x). \tag{1}$$

Consider the function $f(x) = x - 3 - x \cdot H(1/x) = x - 3 - x \log_2 x + (x-1)\log_2(x-1)$. Observe that f is strictly monotonically increasing and continuous for $x \geq 2$ and that it has a unique positive root $c \approx 7.1815$.

It follows that $f(c+\varepsilon) > f(c) = 0$, and, since $x \geq c + \varepsilon$, Equation (1) above implies that

$$f(c+\varepsilon) \leq f(x) \leq \frac{2 - \log_2(p)}{dn} \Longrightarrow \log_2(p) \leq 2 - f(c+\varepsilon)dn.$$

Since the right hand side of the last inequality tends to $-\infty$ as $dn \to \infty$, it follows that $p \to 0$ for $dn \to \infty$. Hence, it is indeed true that asymptotically almost all partitions $E_s \cup E_\ell = E$ of the edge-set of G into short and long edges yield dichotomous ordinal graphs with no geometric representation in $\mathbb{R}^d$. This completes the proof of the first statement.

It remains to show that if G is pandichotomous ($p = 1$), then the explicit bound $m \leq \mu dn$ with $\mu := 7.2240208$ holds for all $n \in \mathbb{N}$ and $d \geq 2$. Set $\phi(z) = 2/z$ and observe that $\phi \to 0$ for $z \to \infty$. We can then express Equation (1) as $f(x) < \phi(dn)$. The trivial bound $m \leq \binom{n}{2}$ suffices to show that $m \leq \mu dn$, unless $\mu dn < \binom{n}{2}$, i.e., unless $\lceil 4\mu + 1 \rceil \leq n$. As ϕ is monotonically decreasing, it follows that $f(x) < \phi(dn) \leq \phi(2n) \leq \phi(2 \cdot \lceil 4\mu + 1 \rceil)$. For the value $\mu = 7.2240208$ we obtain $f(x) < \phi(60) = 1/30$, whereas $f(\mu) > 0.033333334 > 1/30$. Therefore, as f is monotonically increasing, it follows that $\frac{m}{dn} = x < \mu$, as claimed. $\blacktriangleleft$

▶ **Corollary 19.** *For some absolute constant* $\mu < 7.23$ *we have* $\frac{\mathrm{d}(G)}{2\mu} \leq \mathrm{ped}(G) \leq \mathrm{d}(G)$ *for every graph* G *and* $\frac{\mathrm{d}(G)}{2\mu} - 1 \leq \mathrm{psd}(G) \leq \mathrm{d}(G) - 1$ *for every graph* G *with* $\mathrm{d}(G) \geq 2$.

Proof. By Theorem 16 every graph G that is pandichotomous in $\mathbb{R}^d$ has at most μdn edges, for some absolute constant $\mu < 7.23$. Thus, the minimum degree of G is at most $\lfloor 2\mu d \rfloor$ and so G is k-degenerate, for $k = \lfloor 2\mu d \rfloor$. This implies $\mathrm{d}(G) \leq 2\mu \cdot \mathrm{ped}(G)$ and so $\frac{\mathrm{d}(G)}{2\mu} \leq \mathrm{ped}(G) \leq \mathrm{d}(G)$. Since $\mathrm{psd}(G) \geq \mathrm{ped}(G) - 1$ we obtain the lower bound for the spherical dimension. The upper bounds follow from Theorem 11. ◀

▶ **Corollary 20.** *Every* n*-vertex graph is pandichotomous in* $\mathcal{S}^{n-2}$ *and* $\mathbb{R}^{n-1}$*. But there exists an absolute constant* $\mu < 7.23$ *such that for every* $d < \frac{n-1}{4\mu}$ *there exist* n*-vertex bipartite graphs that are not pandichotomous in* $\mathbb{R}^d$ *or* $\mathcal{S}^{d-1}$*.*

Proof. Since n-vertex graphs are $(n-1)$-degenerate, the first statement is a direct consequence of Theorem 11. The complete bipartite n-vertex graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$ has minimum degree $\lfloor n/2 \rfloor$. Hence, Corollary 19 implies that $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$ is not pandichotomous in $\mathbb{R}^d$ for any dimension d such that $2\mu d < \lfloor n/2 \rfloor$, so in particular for $d < (n-1)/(4\mu)$. ◀

6 Conclusion

We study pandichotomous graphs with an emphasis on bipartite graphs on one hand and the relationship between the degeneracy and the pandichotomous dimension on the other hand. Some interesting open questions remain, such as:

- Is every planar graph, planar 3-tree, or planar bipartite graph pandichotomous in $\mathbb{R}^2$?
- Are bipartite dichotomous ordinal graphs realizable in $\mathbb{R}^2$ if the short graph is a 2-tree?
- Characterize the (complete) bipartite graphs that are pandichotomous in $\mathbb{R}^k$, for $k \geq 3$.
- Given $d \geq 3$, is there a bipartite d-degenerate graph that is not pandichotomous in $\mathbb{R}^{d-1}$? We believe that the answer to this question should be positive.

References

1 Md. Jawaherul Alam, Steven Chaplick, Gasper Fijavz, Michael Kaufmann, Stephen G. Kobourov, Sergey Pupyrev, and Jackson Toeniskoetter. Threshold-coloring and unit-cube contact representation of planar graphs. *Discret. Appl. Math.*, 216:2–14, 2017. doi:10.1016/J.DAM.2015.09.003.

2 Md. Jawaherul Alam, Stephen G. Kobourov, Sergey Pupyrev, and Jackson Toeniskoetter. Happy edges: Threshold-coloring of regular lattices. In *Proc. 7th Internat. Conf. Fun with Algorithms (FUN 2014)*, volume 8496 of *LNCS*, pages 28–39. Springer, 2014. doi:10.1007/978-3-319-07890-8_3.

3 Md. Jawaherul Alam, Stephen G. Kobourov, Sergey Pupyrev, and Jackson Toeniskoetter. Weak unit disk and interval representation of graphs. In *Proc. 41st Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG 2015)*, volume 9224 of *LNCS*, pages 237–251. Springer, 2015. doi:10.1007/978-3-662-53174-7_17.

4 Patrizio Angelini, Michael A. Bekos, Martin Gronemann, and Antonios Symvonis. Geometric representations of dichotomous ordinal data. In *Proc. 45th Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG 2019)*, volume 11789 of *LNCS*, pages 205–217. Springer, 2019. doi:10.1007/978-3-030-30786-8_16.

5 Joseph F. Bennett and William L. Hays. Multidimensional unfolding: Determining the dimensionality of ranked preference data. *Psychometrika*, 25(1):27–43, 1960.

6 Jiehua Chen, Kirk Pruhs, and Gerhard J. Woeginger. The one-dimensional Euclidean domain: finitely many obstructions are not enough. *Social Choice and Welfare*, 48(2):409–432, 2017. doi:10.1007/s00355-016-1011-y.

7 Jean-Paul Doignon and Jean-Claude Falmagne. A polynomial time algorithm for unidimensional unfolding representations. *J. Algorithms*, 16(2):218–233, 1994. doi:10.1006/jagm.1994.1010.

8 Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1987. doi:10.1007/978-3-642-61568-9.

9 Edith Elkind and Martin Lackner. Structure in dichotomous preferences. In *Proc. 24th IJCAI*, pages 2019–2025. AAAI Press, 2015. URL: https://ijcai.org/Abstract/15/286.

10 Timothy F. Havel. *The combinatorial distance geometry approach to the calculation of molecular conformation*. Ph.D. thesis, University of California, Berkeley, CA, 1982.

11 Eduard Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1932. URL: http://eudml.org/doc/145659.

12 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete Comput. Geom.*, 47(3):548–568, 2012. doi:10.1007/s00454-012-9394-8.

13 Joseph Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

14 Joseph Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

15 Hiroshi Maehara. Space graphs and sphericity. *Discrete Appl. Math.*, 7(1):55–64, 1984. doi:10.1016/0166-218X(84)90113-6.

16 Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, New York, NY, 2002. doi:10.1007/978-1-4613-0039-7.

17 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, New York, NY, 2005. doi:10.1017/CBO9780511813603.

18 Dominik Peters. Recognising multidimensional euclidean preferences. *CoRR*, abs/1602.08109, 2016. arXiv:1602.08109.

19 Dominik Peters. Recognising multidimensional Euclidean preferences. In *Proc. 21st AAAI Conf. Artificial Intelligence (AAAI'17)*, pages 642–648, 2017. doi:10.1609/AAAI.V31I1.10616.

20 Roger N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962.

21 Roger N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962.

22 Jacob Steiner. Einige Gesetze über die Theilung der Ebene und des Raumes. *J. für die reine und angewandte Mathematik*, 1:349–364, 1826. doi:10.1515/crll.1826.1.349.

23 Yoshikazu Terada and Ulrike von Luxburg. Local ordinal embedding. In *ICML*, volume 32 of *JMLR Workshop and Conf. Proc.*, pages 847–855. JMLR.org, 2014. URL: http://proceedings.mlr.press/v32/terada14.html.

24 Leena Chennuru Vankadara, Michael Lohaus, Siavash Haghiri, Faiz Ul Wahab, and Ulrike von Luxburg. Insights into ordinal embedding algorithms: A systematic evaluation. *J. Mach. Learn. Res.*, 24:191:1–191:83, 2023. URL: http://jmlr.org/papers/v24/21-1170.html.

25 Hugh Warren. Lower bounds for approximation by nonlinear manifolds. *Trans. Amer. Math. Soc.*, 133:167–178, 1968. doi:10.1090/S0002-9947-1968-0226281-1.

Steinhaus Filtration and Stable Paths in the Mapper

Dustin L. Arendt
Visual Analytics Group, Pacific Northwest National Laboratory, Richland, WA, USA

Matthew Broussard
TD Bank, Cherry Hill, NJ, USA TD Bank, Cherry Hill, NJ, USA

Bala Krishnamoorthy
Department of Mathematics and Statistics, Washington State University, Vancouver, WA, USA

Nathaniel Saul
Stripe, San Francisco, CA, USA

Amber Thrall
Department of Mathematics and Statistics, Washington State University, Pullman, WA, USA

Abstract

We define a new filtration called the *Steinhaus filtration* built from a single cover based on a generalized Steinhaus distance, a generalization of Jaccard distance. The homology persistence module of a Steinhaus filtration with infinitely many cover elements may not be q-tame, even when the covers are in a totally bounded space. While this may pose a challenge to derive stability results, we show that the Steinhaus filtration is stable when the cover is finite. We show that while the Čech and Steinhaus filtrations are not isomorphic in general, they are isomorphic for a finite point set in dimension one. Furthermore, the VR filtration completely determines the 1-skeleton of the Steinhaus filtration in arbitrary dimension.

We then develop a language and theory for stable paths within the Steinhaus filtration. We demonstrate how the framework can be applied to several applications where a standard metric may not be defined but a cover is readily available. We introduce a new perspective for modeling recommendation system datasets. As an example, we look at a movies dataset and we find the stable paths identified in our framework represent a sequence of movies constituting a gentle transition and ordering from one genre to another.

For explainable machine learning, we apply the Mapper algorithm for model induction by building a filtration from a single Mapper complex, and provide explanations in the form of stable paths between subpopulations. For illustration, we build a Mapper complex from a supervised machine learning model trained on the FashionMNIST dataset. Stable paths in the Steinhaus filtration provide improved explanations of relationships between subpopulations of images.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Cover and nerve, Jaccard distance, persistence stability, Mapper, recommender systems, explainable machine learning

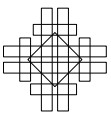
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.10

Related Version *Full Version*: `https://arxiv.org/abs/1906.08256`

Supplementary Material *Software*: `https://github.com/AmberThrall/SteinhausNotebooks/` archived at `swh:1:dir:cf2c816b09a7a3818f63a2e8efa0f0193b4b5669`

Acknowledgements Broussard, Krishnamoorthy, and Saul acknowledge funding from the US National Science Foundation through grants 1661348 and 1819229. Part of the research described in this paper was conducted under the Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory, a multiprogram national laboratory operated by Battelle for the U.S. Department of Energy.

© Dustin L. Arendt, Matthew Broussard, Bala Krishnamoorthy, Nathaniel Saul, and Amber Thrall;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

The need to rigorously seed a solution with a notion of stability in topological data analysis (TDA) has been addressed primarily using topological persistence [4, 11, 17]. Persistence arises when we work with a sequence of objects built on a dataset, a *filtration*, rather than with a single object. One line of focus of this work has been on estimating the homology of the dataset. This typically manifests itself as examining the persistent homology represented as a diagram or barcode, with interpretations of zeroth and first homology as capturing significant clusters and holes, respectively [1, 14, 15, 37]. In practice it is not always clear how to interpret higher dimensional homology (even holes might not make obvious sense in certain cases). A growing focus is to use persistence diagrams to help compare different datasets rather than interpret individual homology groups [2, 8, 34].

The implicit assumption in most such TDA applications is that the data is endowed with a natural metric, e.g., points exist in a high-dimensional space or pairwise distances are available. In certain applications, it is also not clear how one could assign a meaningful metric. For example, memberships of people in groups of interest is captured simply as sets specifying who belongs in each group. An instance of such data is that of recommendation systems, e.g., as used in Netflix to recommend movies to the customer. Graph based recommendation systems have been an area of recent research. Usually these systems are modeled as a bipartite graph with one set of nodes representing recommendees and the other representing recommendations. In practice, these systems are augmented in bespoke ways to accommodate whichever type of data is available. It is highly desirable to analyze the structure without prescribing some ill-fit or incomplete metric to the data.

Another TDA approach for structure discovery and visualization of high-dimensional data is based on a construction called *Mapper* [32]. Defined as the nerve of a refined pullback cover of the data (See Figure 1), Mapper has found increasing use in diverse applications in the past several years [23]. Attention has recently focused on interpreting parts of the 1-skeleton of the Mapper complex, which is a simplicial complex, as significant features of the data. Paths, flares, and cycles have been investigated in this context [21, 26, 33]. The framework of persistence has been applied to this construction to define a *multi-scale* Mapper, which permits one to derive results on stability of such features [13]. At the same time, the associated computational framework remains unwieldy and still most applications base their interpretations on a single Mapper object.

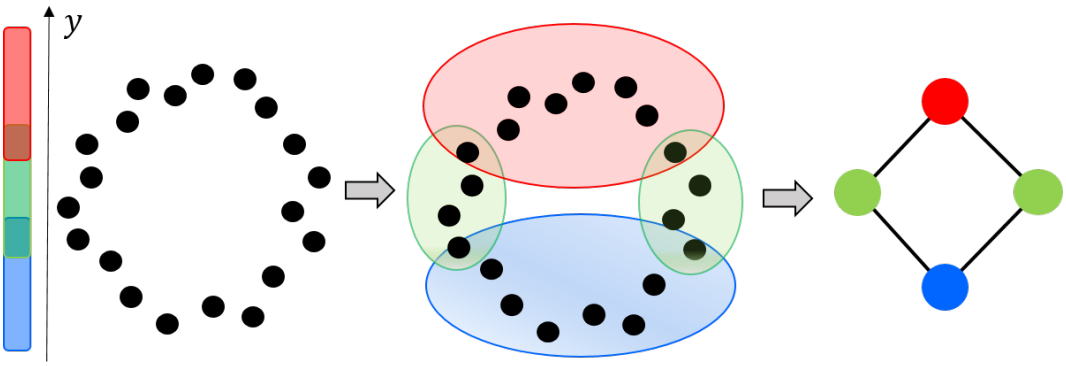


Figure 1 Mapper (Right) constructed on a noisy point set sampled from a circle (Left). The cover consists of three overlapping intervals (in blue, green, red) covering the range of y-coordinate values of the points.

The Mapper construction works with covers. By default, one starts with overlapping hypercubes that cover a parameter space (usually a subset of $\mathbb{R}^d$ for some dimension d) and consider the pullback of this cover to the space of data. In recommendation systems, the cover is just a collection of abstract sets providing membership info. Could we define a topological construction on such abstract covers that still reveals the topology of the dataset?

We could study paths in this construction, but as the topological constructions are noisy, we would want to define a notion of stability for such paths. With this goal in mind, could we define a *filtration* from the abstract cover? But unlike in the setting of, e.g., multiscale Mapper [13], we do not have a sequence of covers (called a tower of covers) – we want to work with a *single* cover. How do we define a filtration on a single abstract cover? And could we prove is is stable? Finally, could we use our construction on real data?

1.1 Our Contributions

We introduce a new type of filtration defined on a single abstract cover. Termed *Steinhaus filtration*, our construction uses Steinhaus distances between elements of the cover. We generalize the Steinhaus distance between two elements to those of multiple elements in the cover, and define a filtration on a single cover using the generalized Steinhaus distance as the filtration index. The homology persistence module of a Steinhaus filtration with infinitely many cover elements may not be q-tame, even when the covers are in a totally bounded space (Lemma 3.1). While this may pose a challenge to derive stability results, we show a stability result on the Steinhaus filtration when the cover is finite – the largest change in generalized Steinhaus distance provides an upper bound for the bottleneck distance of their homologies (Theorem 3.10). We prove that in 1-dimension the Steinhaus filtration is isomorphic to the standard Čech filtration built on the dataset (Theorem 4.1) and independently that the Vietoris-Rips (VR) filtration completely determines the 1-skeleton of the Steinhaus filtration in arbitrary dimensions (Lemma 4.2). We show by example that the Steinhaus filtration is not isomorphic to the Čech filtration in higher dimensions (Proposition 4.3).

This filtration is quite general, and enables the computation of persistent homology for datasets without requiring strong assumptions or defining ill-fit metrics. With real life applications in mind, we study paths in our construction. Paths provide intuitive explanations of the relationships between the objects that the terminal vertices represent. Our perspective of path analysis is that shortest may not be more descriptive – see Figure 2 for an illustration. Instead, we define a notion of *stability* of paths in the Steinhaus filtration. Under this notion, a stable path is analogous to a highly persistent feature as identified by persistent homology.

We demonstrate the utility of stable paths in Steinhaus filtrations on two real life applications: a problem in movie recommendation system and Mapper. We first show how recommendation systems can be modeled using the Steinhaus filtration, and then show how stable paths within this filtration suggest a sequence of movies that represent a "smooth" transition from one genre to another (Section 6.1). We then define an extension of the traditional Mapper [32] termed the *Steinhaus Mapper Filtration*, and show how its stable paths provide valuable explanations of populations in the Mapper complex, focusing on explainable machine learning (Section 6.2). Code for our computations is available at `https://github.com/AmberThrall/SteinhausNotebooks/`.

1.2 Related Work

Cavanna and Sheehy [7] developed theory for a *cover filtration*, built from a cover of a filtered simplicial complex. But we work from more general covers of arbitrary spaces.

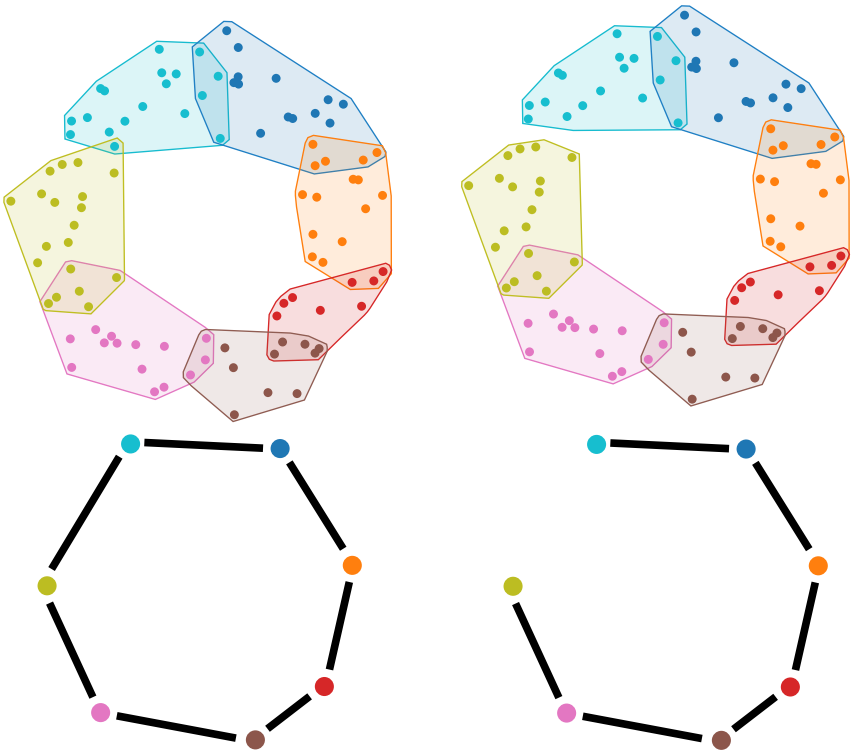


Figure 2 A cover with 7 elements, and its nerve (left column). The cyan and green vertices are connected by a single edge generated by a *single* point in intersection of cyan and green cover elements. Removing this point gives the cover and nerve shown in the right column. The path from cyan to green node now has six edges.

We are inspired by similar goals as those of previous work addressing stability of the Mapper construction by Dey, Mémoli, and Wang [13] and by Carrière and Oudot [6]. Our goal is to provide some consistency, and thus interpretability, to the Mapper construction. We incorporate ideas of persistence in a different manner into our construction using a *single* cover, which considerably reduces the effort of generating results.

The multi-scale Mapper defined by Dey, Mémoli, and Wang [13] builds a filtration on the Mapper complex by varying the parameters of a cover. This construction yields nice stability properties, but is unwieldy in practice and difficult to interpret. Carrière, Michel, and Oudot develop ways based on extended persistence to automatically select a hypercube cover that best captures the topology of the data [5]. This approach constructs one final Mapper that is easy to interpret, but is restricted to the use of hypercube covers, which is just one option of myriad potential covering schemes.

Kalyanaraman, Kamruzzaman, and Krishnamoorthy developed methods for tracking populations within the Mapper graph by identifying *interesting paths* [19] and *interesting flares* [20]. Interesting paths maximize an interestingness score, and are manifested in the Mapper graph as long paths that track particular populations that show trending behavior. Flares capture subpopulations that diverge, i.e., show branching behavior. In our context, we are interested in shorter paths, under the assumption that they provide the most succinct explanations for relationships between subpopulations.

Our work is similar to that of Parthasarathy, Sivakoff, Tian, and Wang [27] in that they use the Jaccard Index of an observed graph to estimate the geodesic distance of the underlying graph. We take an approach more akin to persistence and make far fewer assumptions.We cannot make rigorous estimates of distances but provide many possible representative paths.

The notion of S-paths [29] is similar to stable paths if we model covers as hypergraphs, and vice versa. Stable paths incorporate the size of each cover element (or hyperedges), normalizing the weights by relative size. This perspective allows us to compare different parts of the resulting structure which may have wildly difference sizes of covers. In this context, a large overlap of small elements is considered more meaningful than a proportionally small intersection of large elements.

In Section 6.1, we show how the Steinhaus filtration and stable paths can be applied in the context of recommendation systems. Our viewpoint on recommendation systems is similar to work of graph-based recommendation systems. This is an active area of research and we believe our new perspective of interpreting such systems as covers and filtrations will yield useful tools for advancing the field. The general approach of graph-based recommendation systems is to model the data as a bipartite graph, with one set of nodes representing the recommendation items and the other set representing the recommendees. We can interpret a bipartite graph as a cover, either with elements being the recommendees covering the items, or elements being the items covering the recommendees.

2 Steinhaus Filtrations

We define the Steinhaus distance, a generalization of the Jaccard distance between two sets, and further generalize it to an arbitrary collection of subsets of a cover. These generalizations take a measure μ and assume that all sets are taken mod differences by sets of measure 0.

▶ **Definition 2.1** (Steinhaus Distance [24]). *The* Steinhaus distance *between sets A, B given a measure μ is* $d_{St}(A,B) = 1 - \dfrac{\mu(A \cap B)}{\mu(A \cup B)} = \dfrac{\mu(A \cup B) - \mu(A \cap B)}{\mu(A \cup B)}$.

This distance lies in $[0,1]$, is 0 when sets are equal and 1 when they do not intersect.

▶ **Definition 2.2** (Generalized Steinhaus distance). *Define the* generalized Steinhaus distance *of a collection of sets $\{U_i\}$ as* $d_{St}(\{U_i\}) = 1 - \dfrac{\mu(\bigcap U_i)}{\mu(\bigcup U_i)}$.

We make use of this generalized distance to associate birth times to simplices in a nerve. Given a cover, we define the Steinhaus filtration as the filtration induced from sublevel sets of the generalized Steinhaus distance function. In other words, consider a cover of the space and the nerve of this cover. For each simplex in the nerve, we assign as birth time the value of its Steinhaus distance of its corresponding cover elements. This filtration captures information about similarity of cover elements and overall structure of the cover.

▶ **Definition 2.3** (Nerve). *A* nerve *of a cover $\mathcal{U} = \{U_i\}_{i \in C}$ is an abstract simplicial complex defined such that each subset $\{U_j\}_{j \in J} \subseteq \mathcal{U}$ with $J \subseteq C$, defines a simplex if $\bigcap_{j \in J}\{U_j\} \neq \emptyset$. In this construction, each cover element $U_i \in \mathcal{U}$ defines a vertex.*

▶ **Definition 2.4** (Steinhaus Nerve). *The* Steinhaus nerve *of a cover $\mathcal{U}$, denoted* $\mathrm{Nrv}_{St}(\mathcal{U})$*, is defined as the nerve of $\mathcal{U}$ with each simplex assigned their generalized Steinhaus distance as weight:* $w_\sigma = d_{St}(\{U_i \mid i \in \sigma\}) \quad \forall \sigma \in \mathcal{U}$.

Note that $w_\sigma < 1$ by definition for every simplex $\sigma \in \mathcal{U}$. We will use Nrv_{St} when the cover $\mathcal{U}$ is evident from context. The weighting scheme satisfies the conditions of a filtration.

▶ **Theorem 2.5.** *The Steinhaus nerve of a cover $\mathcal{U}$ is a filtered simplicial complex.*

Proof. This proof makes use of standard set theory results. Let $\mathcal{U}$ be an arbitrary cover of some set X and let Nrv_{St} be its Steinhaus nerve. We consider Nrv_{St} as a filtration by assigning as the birth time of simplex $\sigma \in \text{Nrv}_{St}$ its weight w_σ. To show this is indeed a filtration, we focus on a single simplex σ and a face $\tau \preceq \sigma$ to show that the face always appears in the filtration before the simplex.

Suppose σ is generated from cover elements $\{U_i\}_{i\in I}$ over some index set I. Let a face $\tau \preceq \sigma$ be generated by cover elements indexed by a subset $J \subset I$. The birth time of τ is

$$d_{St}(\{U_i\}_{i\in J}) = 1 - \frac{\mu(\cap_{i\in J} U_i)}{\mu(\cup_{i\in J} U_i)} \quad \text{and that of } \sigma \text{ is} \quad d_{St}\left(\{U_i\}_{i\in I}\right) = 1 - \frac{\mu(\cap_{i\in I} U_i)}{\mu(\cup_{i\in I} U_i)}.$$

Clearly, with $\{U_i\}_{i\in J} \subset \{U_i\}_{i\in I}$, we have that $\mu(\cap_{i\in J} U_i) \geq \mu(\cap_{i\in I} U_i)$ and $\mu(\cup_{i\in J} U_i) \leq \mu(\cup_{i\in I} U_i)$. It follows then that $d_{St}(\tau) \leq d_{St}(\sigma)$. With K_α denoting the subcomplex that includes all simplices in Nrv_{St} with birth time at most $\alpha \in [0, 1)$, for any $\alpha, \beta \in [0, 1)$ with $\alpha < \beta$, we have $K_\alpha \subseteq K_\beta$. Hence $\text{Nrv}_{St}(\mathcal{U})$ is a monotonic filtration. ◀

Following this result, we refer to the construction as the Steinhaus *filtration*.

We could study an adaptation of Steinhaus filtration to an analog of the VR complex by building a weighted clique rank filtration from the 1-skeleton of the Steinhaus filtration [28]. This adaptation drastically reduces the number of intersection and union checks required for the construction. The *weight rank clique filtration* is a way of generating a flag filtration from a weighted graph [28]. We can use this idea to build a VR analog of the Steinhaus filtration.

3 Stability

We provide a brief overview of persistence modules and the bottleneck distance. For details, see the book by Chazal, de Silva, Glisse, and Oudot [9].

Given a poset T, we define a *persistence module* $\mathbb{V}$ over T to be an indexed family of vector spaces $\{V_a\}_{a\in T}$ with a family of linear maps $\{v_a^b : V_a \to V_b\}$ that satisfy $v_b^c \circ v_a^b = v_a^c$ whenever $a \leq b \leq c$. The homologies of the Steinhaus filtered simplicial complex $H_*(\text{Nrv}_{St}(\mathcal{V}))$ defines a persistence module where each space $V_a \in H_*(\text{Nrv}_{St}(\mathcal{V}))$ is the homology of the simplicial complex consisting of simplices whose corresponding Steinhaus distance is at most a. Herein, we use simplicial homology with coefficients in some field k.

Given two persistence modules $\mathbb{U}$ and $\mathbb{V}$ over $\mathbb{R}$ and $\epsilon \in \mathbb{R}$, a *homomorphism of degree* ϵ is a collection of linear maps $\Phi = \{\phi_a : U_a \to V_{a+\epsilon}\}_{a\in\mathbb{R}}$ such that $v_{a+\epsilon}^{b+\epsilon} \circ \phi_a = \phi_b \circ u_a^b$. We denote the set of homomorphisms of degree ϵ by $\text{Hom}^\epsilon(\mathbb{U}, \mathbb{V})$. The composition of a homomorphism of degree ϵ and a homomorphism of degree ϵ' is a homomorphism of degree $\epsilon + \epsilon'$ given by the composition of their linear maps. For $\epsilon \geq 0$, a crucial endomorphism of degree ϵ is the shift map $1_{\mathbb{V}}^\epsilon \in \text{Hom}^\epsilon(\mathbb{V}, \mathbb{V})$ given by the maps

$$1_{\mathbb{V}}^\epsilon = \{v_a^{a+\epsilon} : V_a \to V_{a+\epsilon}\}_{a\in\mathbb{R}}.$$

We say that two persistence modules $\mathbb{U}$ and $\mathbb{V}$ are ϵ-*interleaved* if there are homomorphisms $\Phi \in \text{Hom}^\epsilon(\mathbb{U}, \mathbb{V})$ and $\Psi \in \text{Hom}^\epsilon(\mathbb{V}, \mathbb{U})$ such that $\Psi\Phi = 1_{\mathbb{U}}^{2\epsilon}$ and $\Phi\Psi = 1_{\mathbb{V}}^{2\epsilon}$.

When a persistence module $\mathbb{V}$ is q-tame, i.e., $\text{rank}(v_a^b) < \infty$ whenever $a < b$, then the persistence module decomposes into the direct sum of interval modules $\mathbb{V} = \bigoplus_{k\in K} \mathbb{I}(p_k^*, q_k^*)$. Each term $\mathbb{I}(p_k^*, q_k^*)$ represent the birth time and death time of a particular feature. This decomposition is often represented by the multiset of points (p_k, q_k) in the extended plane $\overline{\mathbb{R}}^2$. This multiset is called the *persistence diagram* of $\mathbb{V}$ and denoted $\text{dgm}(\mathbb{V})$.

In typical applications, the simplicial complex is finite. As a result, their homologies are finite and any map between their homologies will have finite rank. In other words, in typical cases the homologies of the Steinhaus filtration will be q-tame. However, if we allow for infinitely many cover elements then $H_*(\mathrm{Nrv}_{St}(\mathcal{V}))$ may not be q-tame.

▶ **Lemma 3.1.** *There is a Steinhaus filtered simplicial complex in a totally bounded space whose homology persistence module is not q-tame.*

Proof. Consider the Mapper complex in Figure 3. Each cover element is a rectangle of size

$$R = 11 \times 14, \; G = 6 \times 11, \quad \text{and} \quad B = 11 \times 12$$

with rectangular intersections of size $R \cap G = 2.5 \times 4$ and $B \cap G = 2.5 \times 3$. Under the Lebesgue measure, we get the Steinhaus distances of $d_{St}(R, G) = 200/210$ and $d_{St}(B, G) = 366/381$. Hence, under our Steinhaus filtration the cycles going around the red-green-red-green diamonds in the complex are born at $200/210$ and the remaining cycles are born at $366/381$. Since these cycles form a linearly independent set, we get that in the first homology persistence module the map from $200/210$ to $366/381$ has infinite rank.

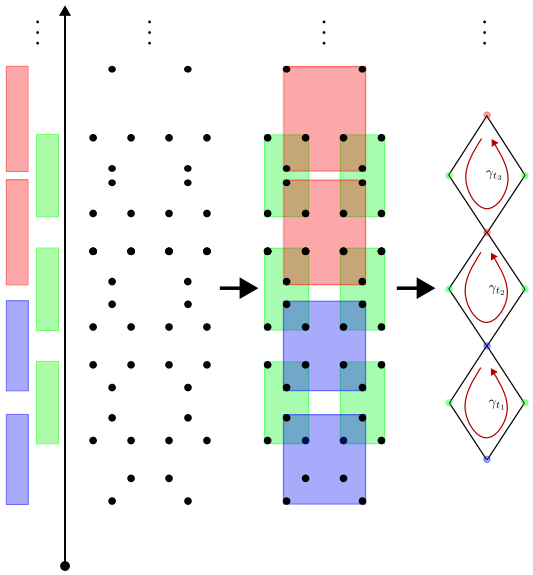


Figure 3 A non q-tame Steinhaus filtered Mapper complex. The center boxes follow the pattern of blue, blue, red, red, blue, blue, etc. For each diamond in the complex there is a cycle that traverses around the diamond.

We modify this construction to get a non q-tame example in a totally bounded space by shrinking each subsequent cover by factor of two. Thus, if we make the covers have sizes

$$R_n = 11 \times \frac{14}{2^n}, \; G_n = 6 \times \frac{11}{2^n}, \quad \text{and} \quad B_n = 11 \times \frac{12}{2^n}$$

and position them such that

$$R_n \cap G_n = 2.5 \times \frac{4}{2^n} \qquad R_n \cap G_{n+1} = 2.5 \times \frac{4}{2^{n+1}}$$
$$B_n \cap G_n = 2.5 \times \frac{3}{2^n} \qquad B_n \cap G_{n+1} = 2.5 \times \frac{3}{2^{n+1}}$$

then the covers are contained in a finite rectangle and the map in the first homology persistence module from $354/364$ to $630/645$ will have infinite rank. ◀

▶ **Definition 3.2.** *A partial matching between multisets A and B is a collection of pairs $M \subseteq A \times B$ such that for every $a \in A$, there is at most one $b \in B$ such that $(a, b) \in M$ and for every $b \in B$, there is at most one $a \in A$ such that $(a, b) \in M$.*

Let $\Delta \subset \overline{\mathbb{R}}^2$ be the diagonal. The cost of a partial matching M is given by

$$\text{cost}(M) := \max\left\{\max_{a\in A} \delta(a), \max_{b\in B} \delta(b)\right\}, \text{ where}$$

$$\delta(p) = \begin{cases} \|p - p'\|_\infty & \text{if there is some } p' \text{ such that } (p, p') \in M \text{ or } (p', p) \in M, \text{ and} \\ \inf_{q\in\Delta} \|p - q\|_\infty & \text{otherwise.} \end{cases}$$

▶ **Definition 3.3** (Bottleneck distance)**.** *The bottleneck distance between A and B is given by*

$$d_b(A, B) = \inf\{\text{cost}(M) : M \text{ is a partial matching between } A \text{ and } B\}.$$

▶ **Theorem 3.4** ([9])**.** *If $\mathbb{U}$ is a q-tame persistence module then it has a well-defined persistence diagram* $\text{dgm}(\mathbb{U})$*. If $\mathbb{U}, \mathbb{V}$ are q-tame persistence modules that are ϵ-interleaved then $d_b(\text{dgm}(\mathbb{U}), \text{dgm}(\mathbb{V})) \leq \epsilon$.*

We define a multivalued map C between X and Y, denoted $C : X \rightrightarrows Y$, to be a subset of $X \times Y$ such that the projection onto X is surjective. A multivalued map $C : X \rightrightarrows Y$ is said to be a *correspondence* if the projection onto Y is surjective, or equivalently,

$C^\top := \{(y, x) : (x, y) \in C\}$ is a multivalued map.

▶ **Definition 3.5** (ϵ-simplicial [10])**.** *Let $\mathbb{S} = \{S_\alpha\}$ and $\mathbb{T} = \{T_\beta\}$ be filtered simplicial complexes with vertex sets X and Y. A multivalued map $C : X \rightrightarrows Y$ is ϵ-simplicial if for any $\alpha \in \mathbb{R}$ and any simplex $\sigma \in S_\alpha$, every finite subset of $C(\sigma)$ is a simplex of $T_{\alpha+\epsilon}$.*

▶ **Proposition 3.6** ([10])**.** *Let $\mathbb{S}$ and $\mathbb{T}$ be filtered simplicial complexes with vertex sets X and Y. If $C : X \rightrightarrows Y$ is a correspondence such that C and $C^\top$ are both ϵ-simplicial, then they induce an ϵ-interleaving between $H_*(\mathbb{S})$ and $H_*(\mathbb{T})$ given by $H_*(C)$ and $H_*(C^\top)$.*

Let $\text{Nrv}_{St}(\mathcal{U})$ and $\text{Nrv}_{St}(\mathcal{V})$ be two Steinhaus filtered simplicial complexes. Given a correspondence $C : \mathcal{U} \rightrightarrows \mathcal{V}$, we define the *distortion* of C to be

$$\text{dis}(C) := \sup\{|d_{St}(\{U_i \mid i \in \sigma\}) - d_{St}(\{V_i \mid i \in \tau\})| : (\sigma, \tau) \in C\},$$

i.e., the largest change in birth times across each simplex. We then define the pseudometric by finding the correspondence with the smallest distortion:

$$d(\mathcal{U}, \mathcal{V}) = \inf\{\text{dis}(C) \mid C : \mathcal{U} \rightrightarrows \mathcal{V} \text{ is a correspondence}\}.$$

▶ **Lemma 3.7.** *$d(\mathcal{U}, \mathcal{V})$ is a pseudometric.*

$d(\mathcal{U}, \mathcal{V})$ does not satisfy the positivity requirement to be a metric. Consider the covers $\mathcal{U}$ and $\mathcal{V}$ given in Figure 4. The simplicial complexes are not isomorphic, but $d(\mathcal{U}, \mathcal{V}) = 0$.

▶ **Proposition 3.9.** *Let $\text{Nrv}_{St}(\mathcal{U})$ and $\text{Nrv}_{St}(\mathcal{V})$ be two Steinhaus filtered simplicial complexes. If $\epsilon > d(\mathcal{U}, \mathcal{V})$, then the persistence modules $H_*(\text{Nrv}_{St}(\mathcal{U}))$ and $H_*(\text{Nrv}_{St}(\mathcal{V}))$ are ϵ-interleaved.*

▶ **Theorem 3.10.** *Let $\text{Nrv}_{St}(\mathcal{U})$ and $\text{Nrv}_{St}(\mathcal{V})$ be two Steinhaus filtered simplicial complexes with q-tame homology persistence modules. Then*

$$d_b(\text{dgm}(H_*(\text{Nrv}_{St}(\mathcal{U}))), \text{dgm}(H_*(\text{Nrv}_{St}(\mathcal{V})))) \leq d(\mathcal{U}, \mathcal{V}).$$

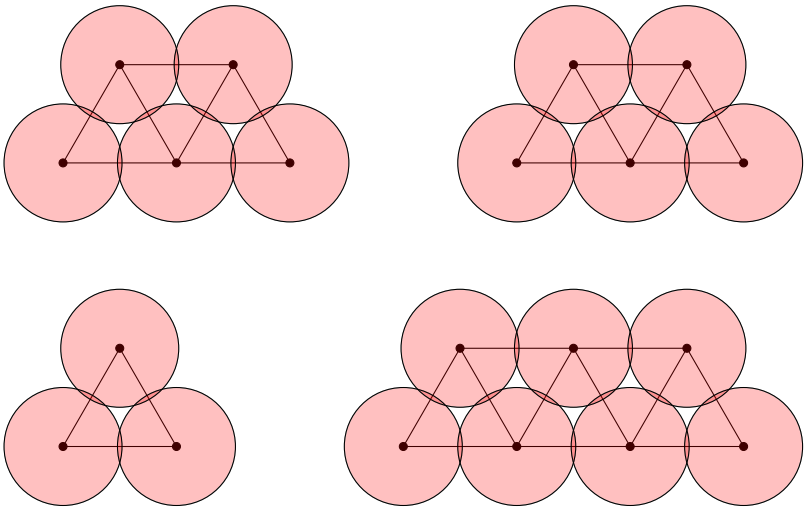


Figure 4 Nonisomorphic simplicial complexes for covers $\mathcal{U}, \mathcal{V}$ (top, bottom rows) with $d(\mathcal{U}, \mathcal{V}) = 0$.

4 Equivalence

To situate the Steinhaus filtration in the context of standard filtrations built on point clouds in Euclidean space, we wish to show the equivalence of the Steinhaus filtration to the Čech and VR filtrations under certain conditions. We first show that the Čech filtration on a finite set of points in dimension one, i.e., the nerve of balls with radius r around each point and over a sequence of r, and the Steinhaus Nerve constructed from the terminal cover of the Čech filtration are isomorphic (see Theorem 4.1).

We provide experimental evidence for the 1-skeletons of the Steinhaus filtration and the VR filtration being isomorphic. We prove one direction of this equivalence in arbitrary dimension – that the VR filtration completely determines the 1-skeleton of the Steinhaus filtration (see Lemma 4.2). We prove by example that the Steinhaus filtration and the Čech filtration in two (or higher) dimension(s) may not be isomorphic (see Proposition 4.3).

Let $\check{C}_r(X)$ be cover of X by balls of radius r centered on points in X. The Čech complex is the nerve of this cover. The Čech filtration is sequence of simplicial complexes for all r.

▶ **Theorem 4.1.** *Given a finite dataset $X \subset \mathbb{R}$ in dimension one and some radius $R > \texttt{diam}(X)$ the Čech filtration constructed from X is isomorphic to the Steinhaus filtration on X constructed from $\check{C}_R(X)$, given the Lebesgue (i.e., volume) measure.*

Proof. We set $\check{C}(\{v_i\})$ as birth radius of the simplex defined by $\{v_i\}$, computed as

$$\check{C}(\{v_i\}) = \frac{\max_i(v_i) - \min_i(v_i)}{2}$$

since in 1D space the simplex is born when balls around the two outermost points intersect.

Let $\{V_i\}$ be the set of balls of radius R centered on the set $\{v_i\}$. Recall that we are using the Steinhaus distance for Lebesgue measure, so μ computes volume here. Then the generalized Steinhaus distance for those balls is given by $d_{St}(\{V_i\}) = 1 - \frac{\min(v_i+R)-\max(v_i-R)}{\max(v_i+R)-\min(v_i-R)}$, since the mutual intersection of all the balls in this one dimensional space is the interval bounded by the minimum right endpoint of all the balls and the maximum of all left endpoints of all the balls and the union of all the balls has the minimum left endpoint amongst left endpoints and maximum right endpoint amongst right endpoints.

$$d_{St}(\{V_i\}) = 1 - \frac{\min(v_i) - \max(v_i) + 2R}{\max(v_i) - \min(v_i) + 2R}$$
$$= 1 - \frac{-2\check{C}(\{v_i\}) + 2R}{2\check{C}(\{v_i\}) + 2R} = 1 - \frac{R - \check{C}(\{v_i\})}{R + \check{C}(\{v_i\})}.$$

Solving for $\check{C}(\{v_i\})$, we get $\check{C}(\{v_i\}) = \dfrac{Rd_{St}(\{V_i\})}{2 - d_{St}(\{V_i\})}$, giving a bijection between birth times.

Now since $R > \texttt{diam}(X)$ and $\{v_i\} \subseteq X$, $0 \leq \check{C}(\{v_i\}) \leq R$. Also, $\frac{R-x}{R+x}$ decreases monotonically over the range $x \in [0, R]$. Thus, $1 - \frac{R-x}{R+x}$ increases monotonically on $x \in [0, R]$. Thus, if we order the subsets of X by increasing birth radius $(s_1, \ldots, s_n)$, then the Steinhaus distances $(d_{St}(s_1), \ldots, d_{St}(s_n))$ are also in increasing order, showing the isomorphism. ◀

▶ **Lemma 4.2.** *There is a map from the birth time of an edge in the VR filtration to its birth time in the Steinhaus filtration. Hence the VR filtration determines the* 1*-skeleton of the Steinhaus filtration in arbitrary dimensions.*

Proof. The volume of the intersection of two hyperspheres was derived by Li [22]. The volume of intersection of two hyperspheres of equal radius R in $\mathbb{R}^n$ with centers distance d apart is defined as

$$V_{\cap}^n(R, d) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2} + 1)} R^n I_{1-(d/2)/R^2}\left(\frac{n+1}{2}, \frac{1}{2}\right)$$

where Γ is the gamma function and I is the regularized incomplete beta function:

$$I_z(a, b) = \frac{\Gamma(a+b)\int_0^z u^{a-1}(1-u)^{b-1}du}{\Gamma(a)\Gamma(b)}.$$

We can reduce this equation to

$$V_{\cap}^n(R, d) = R^n \pi^{(n-1)/2} \frac{\int_0^{1-(d/2)/R^2} u^{(n-1)/2}(1-u)^{-1/2}du}{\Gamma(\frac{n+1}{2})}.$$

The volume of an n-sphere of radius R is $V_{\circ}(R) = \dfrac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} R^n$ and hence we get

$$V_{\cup}^n(R, d) = 2V_{\circ}^n(R) - V_{\cap}^n(R, d).$$

We then compute the Steinhaus distance with Lebesgue measure of two spheres in $\mathbb{R}^n$ and radius R with Euclidean distance d apart as

$$d_{St}^n(R, d) = \frac{2V_{\circ}^n(R) - 2V_{\cap}^n(R, d)}{2V_{\circ}^n(R) - V_{\cap}^n(R, d)}$$
$$= \frac{2\Gamma(\frac{n+1}{2}) - n\Gamma(\frac{n}{2})\pi^{-1/2}\int_0^{1-(d/2)/R^2} u^{(n-1)/2}(1-u)^{-1/2}du}{2\Gamma(\frac{n+1}{2}) - \frac{n}{2}\Gamma(\frac{n}{2})\pi^{-1/2}\int_0^{1-(d/2)/R^2} u^{(n-1)/2}(1-u)^{-1/2}du}.$$

This equation provides a mapping from the birth time of the edge in the VR filtration to the birth time of the edge in the Steinhaus filtration. Once an n and R are chosen, the equation readily reduces, producing the birth times of a simplex in the Steinhaus filtration. ◀

Next we provide detailed experimental results suggesting that the 1-skeletons of the Steinhaus and the VR filtrations are isomorphic. We estimate the area of intersection of 1-spheres using Monte Carlo integration with uniform sampling. The first plot in Figure 5 shows the 50 landmark points along with 20,000 points uniformly sampled around the landmarks for ease of computation. The middle plot shows the persistence diagrams of dimension 0 and 1 for the VR filtration on the landmarks. Finally, we show the Steinhaus filtration on the landmarks using balls with radii 0.5 as the covers.

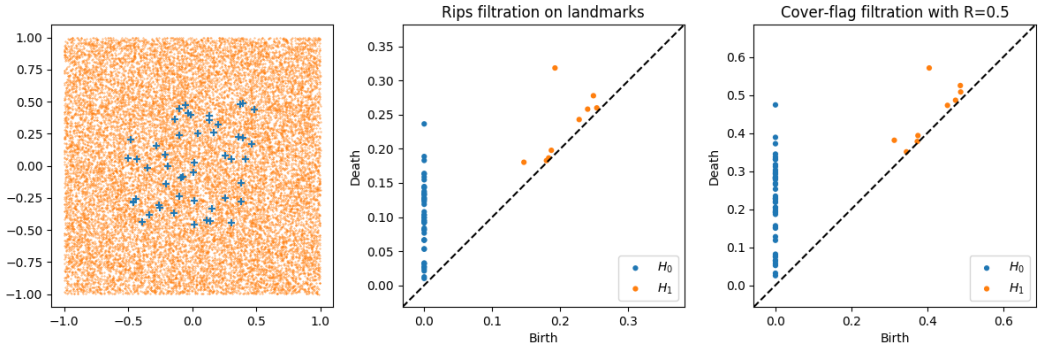


Figure 5 Persistence diagrams for the Vietoris-Rips filtration and the approximate Steinhaus filtration of a set of uniformly sampled points in the plane.

▶ **Proposition 4.3.** *The Čech and Steinhaus filtrations are not isomorphic in general.*

Proof. Let $X \subset \mathbb{R}^2$ be the four corners of a unit square. We construct a cover where at each point in X there is a ball of radius one (see Figure 6). In the Steinhaus filtration, four 2-simplices are born approximately at 0.935 before the 3-simplex is born around 0.959 under the volume measure. However, in the Čech complex the 2-simplices are born at the same time as the 3-simplex. ◀

5 Stable Paths

We develop a theory of stable paths within a Steinhaus filtration. We provide an algorithm for computing a most stable path from one vertex to another. Note that a most stable path might not be a shortest path in terms of number of edges. Conversely, a shortest path might not be highly stable. Since the two objectives are at odds with each other, we provide an algorithm to identify a family of shortest paths as we vary the stability level.

We were studying shortest paths in a Mapper graph constructed on a machine learning model as ways to illustrate the relations between the data as identified by the model. In this context, shortest paths found could have low Steinhaus distance, and thus could be considered noise. This motivated our desire to find stable paths, as they would intuitively be most representative of data and stable with respect to changing or data.

▶ **Definition 5.1** (ρ-Stable Path). *A path P is said to be* ρ-stable *for Steinhaus distance ρ if* $\max\{d_{St}(e) \mid e \in P\} \leq \rho$ *where $d_{St}(e)$ is the Steinhaus distance associated with edge e.*

It follows that a ρ_1-stable path is also ρ_2-stable for any $\rho_2 \geq \rho_1$. Also, a ρ_1-stable path P_1 is more stable than a ρ_2-stable path P_2 when $\rho_1 < \rho_2$. In this case, we have a higher confidence that the edges in P_1 do exist, and are not due to noise, than the edges in P_2.

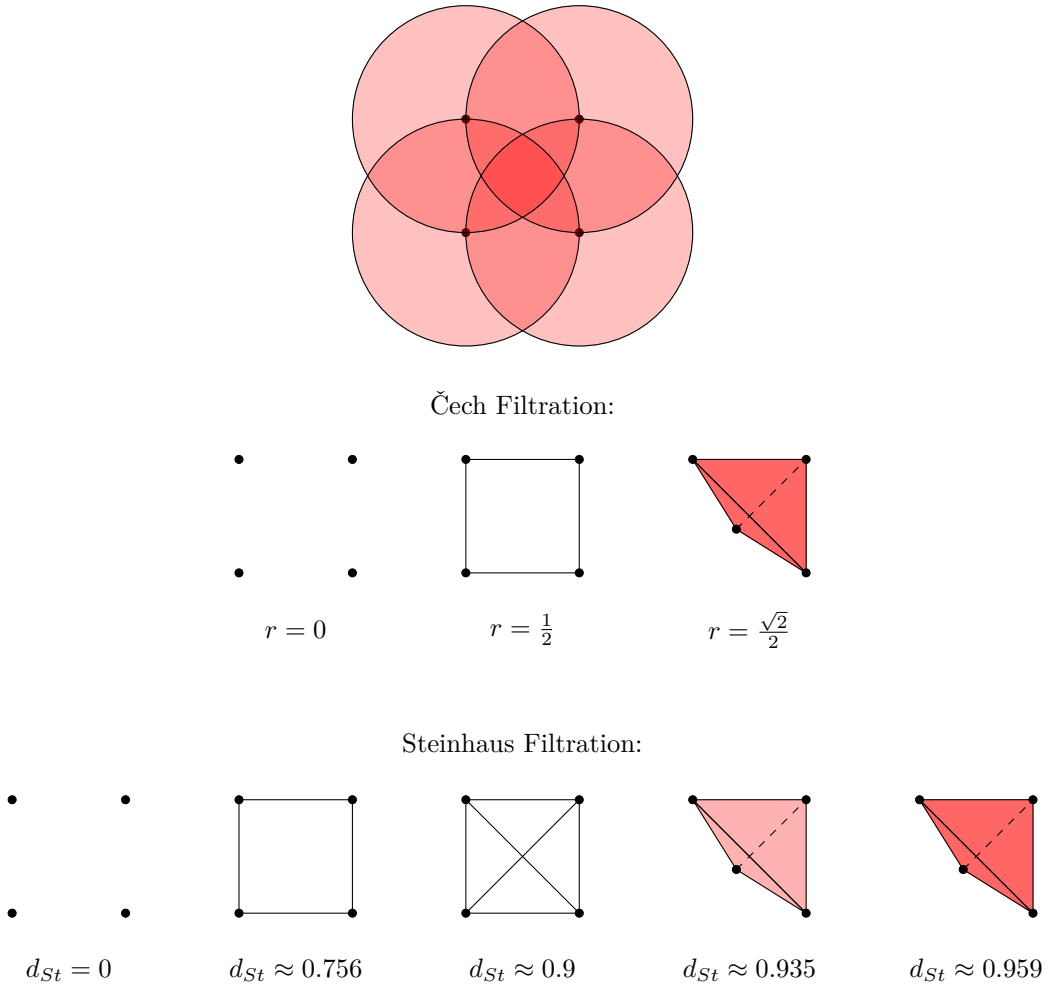


Figure 6 Example showing that the Čech filtration and Steinhaus filtration are not isomorphic in general. The Čech filtration (middle) goes from 1-simplices directly to a 3-simplex whereas the Steinhaus filtration (bottom) of the cover (top) has four 2-simplices born before the 3-simplex.

▶ **Definition 5.2** (Most Stable Path). *Given a pair of vertices s and t, a* most stable s-t path *is a ρ-stable path between s and t for the smallest value of ρ. If there are multiple s-t paths at the same minimum ρ value, a shortest path among them is defined as a most stable path.*

The problem of finding the most stable s-t path can be solved as a minimax path problem on an undirected graph, which can solved efficiently using, e.g., range minimum queries [12].

We are then left with two paths between vertices s and t, the shortest and the most stable. As these two notions, stable and short, are at odds with each other, we are interested in computing the entire Pareto frontier between the short and stable path. Algorithm 1 identifies the Pareto frontier, and its sample output is shown in Figure 7.

The blue points in Figure 7 are on the Pareto frontier, while the orange points are the pairs $[P', \rho']$ that get replaced from the LIST in the course of the algorithm. We then visualize the paths on the Pareto frontier in Figure 8. Continuing our analogy to persistence, the path corresponding to a point on the Pareto frontier which sees a steep rise to the left is considered highly persistent, e.g., the path with length 21 on the frontier.

Algorithm 1 Algorithm to identify the Pareto frontier between shortest and most stable paths.

Input: 1-skeleton G of Steinhaus filtration and vertices s, t
Output: A list LIST consisting of pairs $[P, \rho]$
set LIST $= \emptyset$
while s, t are connected in G **do**
  find a shortest path P between s and t
  $\rho = \max\{d_{St}(e) \mid e \in P\}$
  if LIST has no pair $[P', \rho']$ with $|P| = |P'|$ **then**
    add $[P, \rho]$ to LIST
  else
    if $\rho < \rho'$ for $[P', \rho'] \in$ LIST with $|P| = |P'|$ **then**
      replace $[P', \rho']$ with $[P, \rho]$ in LIST
    end if
  end if
  remove all edges e from G with $d_{St}(e) \geq \rho$
end while

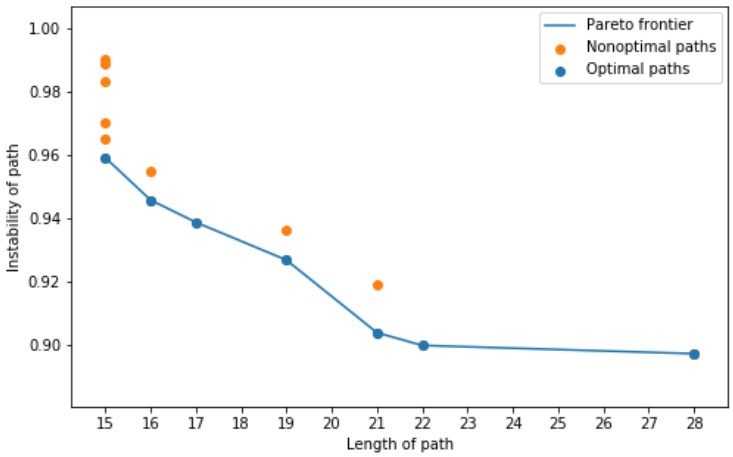


Figure 7 Pareto frontier between length of path and stability of path.

6 Applications

We apply the Steinhaus filtration and stable paths to recommendation systems and Mapper analysis. We show how recommendation systems can be modeled using the Steinhaus

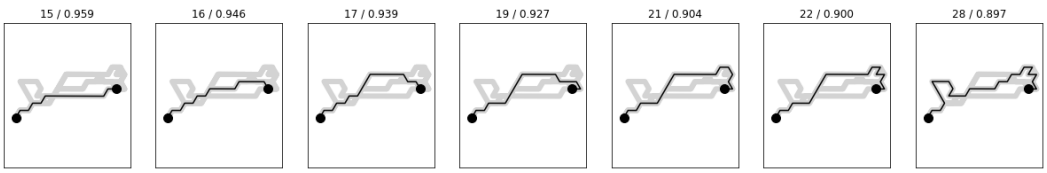


Figure 8 Visualization of each path on the Pareto frontier shown in Figure 7.

filtration and then show how stable paths in this filtration can answer the question *what movies should I show my friend first, to wean them into my favorite (but potentially weird) movie?* We then define an extension of the traditional Mapper complex [32, 6] called the Steinhaus Mapper Filtration, and show how stable paths within this filtration can provide valuable explanations of populations in the Mapper construction. As a direct illustration, we focus on the case of explainable machine learning, where the Mapper complex is constructed with a supervised machine learning model as the filter function, and address the question *what can we learn about the model?*

6.1 Recommendation Systems

We apply the Steinhaus filtration to a recommendation system dataset and employ the stable paths analysis to compute sequences of movies that ease viewers from one title to another title. Suppose you have only ever seen the movie *Mulan* and your partner wants to show you *Moulin Rouge*. It would be jarring to watch that movie, so your partner might gently build up to it by showing movies similar to both *Mulan* and *Moulin Rouge*. We compute stable paths that identify such a gentle sequence.

We use the MovieLens-20m dataset [18], comprised of 20 million ratings by 138,493 users of 27,278 movies. Often, these types of datasets are interpreted as bipartite graphs. We note that a bipartite graph can be equivalently represented as a covering of one node set with the other, and apply the Steinhaus filtration to build a filtration. In our case, we interpret each movie as a cover element of the users who have rated the movie. To reduce computational expenses and noise, we remove all movies with less than 10 ratings and then sample 4000 movies at random from the remaining movies.

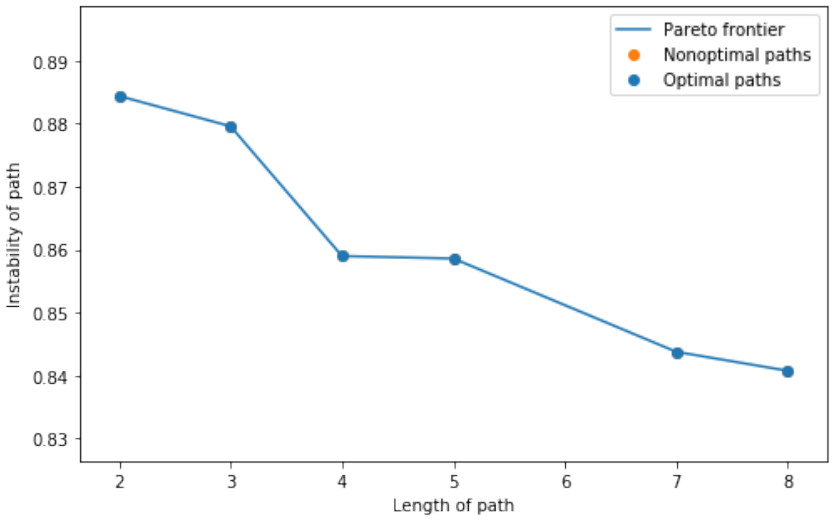


Figure 9 Pareto frontier of stable paths between *Mulan* and *Moulin Rouge*.

Figure 9 shows the computed Pareto frontier of stable paths for the case of *Mulan* and *Moulin Rouge*. In Table 1, we show two stable paths. The stable path with length 4 is found after a large drop in instability. As the length and stability must be traded off, we think this would be a decent path to choose if you want to optimize both. The second path is the most stable. For readers familiar with the movies, the relationship between each edge is clear.

Table 1 Two sequences of movie transitions.

Shortest Path	Most Stable Path
1. Mulan (1998)	1. Mulan (1998)
2. Dumbo (1941)	2. Robin Hood (1973)
3. The Sound of Music (1965)	3. Dumbo (1941)
4. Moulin Rouge (2001)	4. The Sound of Music (1965)
	5. Gone with the Wind (1939)
	6. Psycho (1960)
	7. High Fidelity (2000)
	8. Moulin Rouge (2001)

6.2 Steinhaus Mapper Filtration

We develop a method of model induction for inspecting a machine learning model. The goal is to develop an understanding of the model structure by characterizing the relationship between the feature space and the prediction space. The Mapper construction [32] is aptly suited for visualizing this functional structure.

This application is based on the work of using paths in the Mapper graph to provide explanations for supervised machine learning models [31]. In a similar approach, Rathore, Chalapathi, Palande, and Wang [30] used the Mapper construction to visually explore shapes of activations in deep learning. They focused mostly on bifurcations in the Mapper graph, and did not directly address the question of stability of the features. We build a Mapper from the predicted probability space of a logistic regression model. We then extend that Mapper to be a *Steinhaus Mapper Filtration* and analyze its stable paths.

Given topological spaces X, Y, function $f : X \to Y$, and a cover of Y, the Mapper complex is the nerve of the refined, i.e., with each cover element split into path-connected components, pullback cover of $f(Y)$. Typically, a clustering algorithm is used to compute the pullback of each cover element.

▶ **Definition 6.1** (Steinhaus Mapper filtration). *Given data X, a function $f : X \to Y$, and a cover U of Y, we define the* ***Steinhaus Mapper*** *as the Steinhaus nerve (Definition 2.4) of the refined pullback cover of $f(U)$:* $\mathrm{Nrv}_{St}(f^*U)$.

Figure 10 shows the Steinhaus Mapper filtration constructed from a logistic regression model built on the Fashion-MNIST dataset [36], with 70,000 images of clothing items from 10 classes. Each image is 28 pixels. The Fashion-MNIST dataset is typically considered to be a more difficult application than the MNIST handwritten digits.

We first reduce the dataset to 100 dimensions using Principal Components Analysis, and then a logistic regression classifier with l_1 regularization is trained on the reduced data using 5-fold cross validation on a training set of 60,000 images. The model evaluated at 93% accuracy on the remaining 10,000 images. We then extract the 10-dimensional predicted probability space and use UMAP [25] to reduce it to 2 dimensions. This 2-dimensional space is taken as the filter function of the Mapper complex, using a cover consisting of 40 bins along each dimension with 50% overlap between each bin. DBSCAN is used as the clustering algorithm in the refinement step [16]. KeplerMapper [35] is used to construct the Mapper complex. Finally, the cover is extracted and the Steinhaus Mapper filtration is constructed.

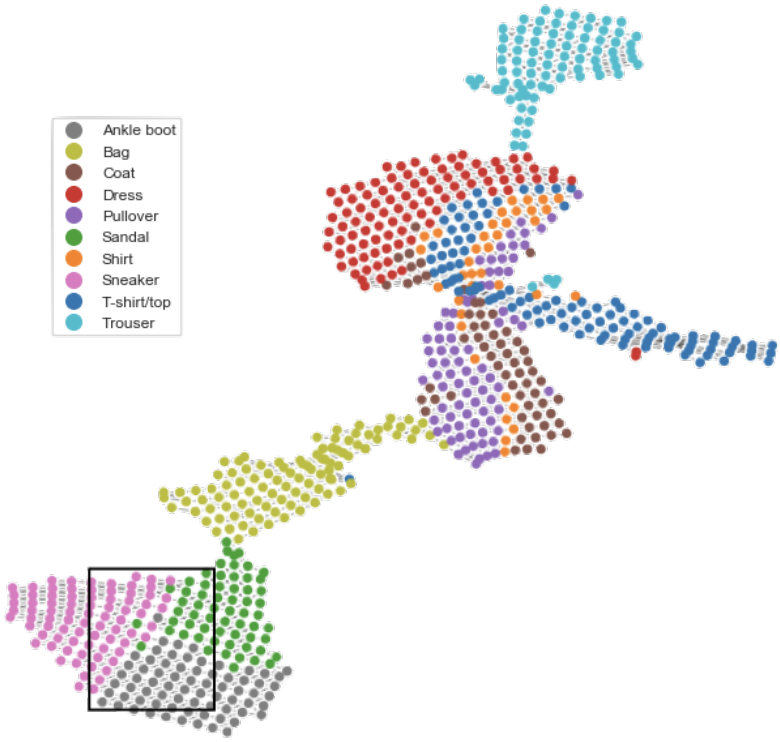


Figure 10 Mapper from logistic regression on Fashion-MNIST data. Window is frame of Fig. 11.

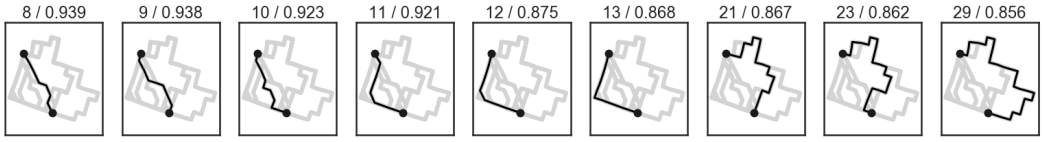


Figure 11 Depiction of stable paths found along Pareto frontier in Figure 12.

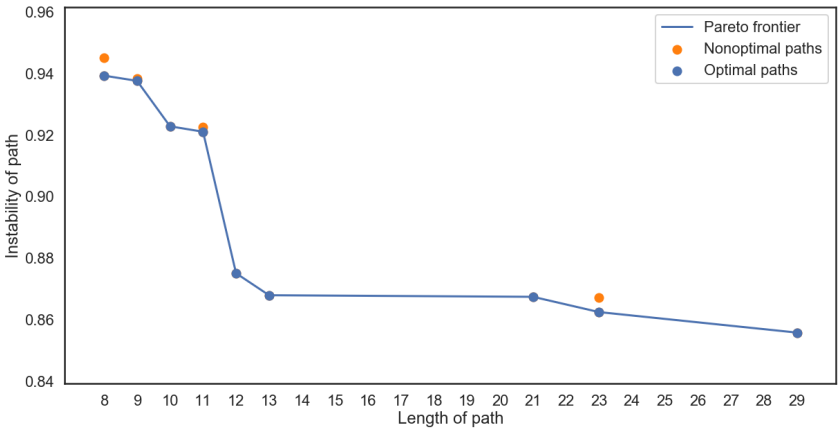


Figure 12 Pareto frontier of stable paths between predominately sneaker and ankle boot vertices.

To illustrate path explanations, we take two vertices selected from the sneaker and ankle boot regions of the resulting graph. The three regions of shoes (sneaker, ankle boot, and sandals) are understandably confusing to the model, and we are interested in where these confusions arise. Figure 11 shows the paths associated with the Pareto frontier (Figure 12).

Figure 13 shows representatives from each vertex in the path for the shortest and the stable path with length 12. Each row corresponds to one vertex and the columns show a representative from each class represented in the vertex. Each image shows the multiplicity of that type of shoe in the vertex.

In both paths, vertices start with mostly sneakers and sneaker-like sandals. They then transition to contain larger proportions of ankle boots, with all three classes showing higher cut tops or high heels.

Along each path we see the relationships between nodes change. In the most stable path (Right), we see slow transition from sneaker to ankle boot space, with some sandals spread throughout. Along the path, shoes from all three classes become taller. Near the middle, images from sneakers and ankle boots are indistinguishable.

7 Discussion

We introduced the Steinhaus filtration, a new kind of filtration that enables application of TDA to previously inaccessible types of data. We then developed a theory of stable paths in the Steinhaus filtration, and provide algorithms for computing the Pareto frontier between short and stable paths. As proof of their utility to real world applications, we show how these two ideas can be applied to the analysis of recommendation systems and of Mapper in the context of explainable machine learning.

Our results suggest many new questions. The application of recommendation systems leaves us curious if the Steinhaus filtration along with new results such as the one on predicting links in graphs using persistent homology [3] could help answer the main question in recommendation system research: *what item to recommend to the user next?*

While paths and connected components are most amenable to practical interpretations, could other structures in the Steinhaus filtration also suggest insights? What would holes and loops in the Steinhaus filtration for recommendation systems mean?

We showed that when the cover is finite, the Steinhaus filtration is stable to small changes within the cover. The example showing non q-tameness for instances with infinite covers even when it is in a totally bounded space motivates this following question: can we find a finite subcover that *approximates* such infinite covers?

References

1 Henry Adams and Gunnar Carlsson. On the nonlinear statistics of range image patches. *SIAM Journal on Imaging Sciences*, 2(1):110–117, 2009. doi:10.1137/070711669.

2 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.

3 Sumit Bhatia, Bapi Chatterjee, Deepak Nathani, and Manohar Kaul. A persistent homology perspective to the link prediction problem. In Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha, editors, *Complex Networks and Their Applications VIII*, pages 27–39, Cham, 2020. Springer International Publishing. arXiv:1811.04049.

Figure 13 Path visualizations for shortest path (left) and stable path with length 12 (right). Columns in the visualization are based on the class and each row represents a node in the Mapper graph. We show one representative for each node in each column. Columns with no shoes shown had no representative of that class in the node.

4 Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, January 2009. doi:10.1090/s0273-0979-09-01249-x.

5 Mathieu Carrière, Bertrand Michel, and Steve Oudot. Statistical analysis and parameter selection for Mapper. *Journal of Machine Learning Research*, 19(12):1–39, 2018. arXiv:1706.00204.

6 Mathieu Carrière and Steve Oudot. Structure and stability of the one-dimensional Mapper. *Foundations of Computational Mathematics*, 18:1333–1396, October 2018. doi:10.1007/s10208-017-9370-z.

7 Nicholas J Cavanna and Donald R Sheehy. Persistent nerves revisited. *Young Researchers Forum at Symposium of Computational Geometry*, 2017. URL: http://donsheehy.net/research/cavanna17peristent_yrf.pdf.

8 Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Mémoli, and Steve Y. Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1393–1403, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association. doi:10.1111/j.1467-8659.2009.01516.x.

9 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics. Springer Cham, 1 edition, 2016. doi:10.1007/978-3-319-42545-0.

10 Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173:193–214, 2014. doi:10.1007/s10711-013-9937-z.

11 Frédéric Chazal and Bertrand Michel. An introduction to Topological Data Analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4, 2021. doi:10.3389/frai.2021.667963.

12 Erik D. Demaine, Gad M. Landau, and Oren Weimann. On Cartesian trees and range minimum queries. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, pages 341–353, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02927-1_29.

13 Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Multiscale Mapper: Topological summarization via codomain covers. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 997–1013, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611974331.CH71.

14 Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, December 2009.

15 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002. doi:10.1007/S00454-002-2885-2.

16 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 Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996. URL: https://cdn.aaai.org/KDD/1996/KDD96-037.pdf.

17 Robert Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008. doi:10.1090/S0273-0979-07-01191-3.

18 F Maxwell Harper and Joseph A Konstan. The MovieLens datasets: History and context. *ACM transactions on interactive intelligent systems (TIIS)*, 5(4):19, 2016. doi:10.1145/2827872.

19 Ananth Kalyanaraman, Methun Kamruzzaman, and Bala Krishnamoorthy. Interesting paths in the Mapper complex. *Journal of Computational Geometry*, 10(1):500–531, 2019. doi:10.20382/jocg.v10i1a17.

20 Methun Kamruzzaman, Ananth Kalyanaraman, and Bala Krishnamoorthy. Detecting divergent subpopulations in phenomics data using interesting flares. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 155–164, New York, NY, USA, 2018. ACM. doi:10.1145/3233547.3233593.

21 Li Li, Wei-Yi Cheng, Benjamin S. Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P. Bottinger, and Joel T. Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science Translational Medicine*, 7(311):311ra174–311ra174, 2015. doi:10.1126/scitranslmed.aaa9364.

22 S Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011. doi:10.3923/ajms.2011.66.70.

23 Pek Y. Lum, Gurjeet Singh, Alan Lehman, Tigran Ishkanov, Mikael. Vejdemo-Johansson, Muthi Alagappan, John G. Carlsson, and Gunnar Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1236), 2013. doi:10.1038/srep01236.

24 Edward Marczewski and Hugo Steinhaus. On a certain distance of sets and the corresponding distance of functions. *Colloquium Mathematicum*, 6(1):319–327, 1958. URL: http://eudml.org/doc/210378.

25 Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, September 2020. arXiv:1802.03426.

26 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, 2011. doi:10.1073/pnas.1102826108.

27 Srinivasan Parthasarathy, David Sivakoff, Minghao Tian, and Yusu Wang. A quest to unravel the metric structure behind perturbed networks. In *Proceedings of the thirty-three annual symposium on Computational geometry*, pages 123–445. ACM, 2017. arXiv:1703.05475.

28 Giovanni Petri, Martina Scolamiero, Irene Donato, and Francesco Vaccarino. Topological strata of weighted complex networks. *PLoS One*, 8(6):e66506, 2013. doi:10.1371/journal.pone.0066506.

29 Emilie Purvine, Sinan Aksoy, Cliff Joslyn, Kathleen Nowak, Brenda Praggastis, and Michael Robinson. A topological approach to representational data models. In Sakae Yamamoto and Hirohiko Mori, editors, *Human Interface and the Management of Information. Interaction, Visualization, and Analytics*, pages 90–109, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-92043-6_8.

30 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.

31 Nathaniel Saul and Dustin L. Arendt. Machine learning explanations with topological data analysis. *Demo at the Workshop on Visualization for AI explainability (VISxAI)*, 2018. URL: https://sauln.github.io/blog/tda_explanations/.

32 Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Proceedings of the Symposium on Point Based Graphics*, pages 91–100, Prague, Czech Republic, 2007. Eurographics Association. doi:10.2312/SPBG/SPBG07/091-100.

33 Brenda Y. Torres, Jose Henrique M. Oliveira, Ann Thomas Tate, Poonam Rath, Katherine Cumnock, and David S. Schneider. Tracking resilience to infections by mapping disease space. *PLoS Biol*, 14(4):1–19, April 2016. doi:10.1371/journal.pbio.1002436.

34 Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet means for distributions of persistence diagrams. *Discrete and Computational Geometry*, 52(1):44–70, July 2014. doi:10.1007/s00454-014-9604-7.

35 Hendrik van Veen, Nathaniel Saul, David Eargle, and Sam Mangham. Kepler Mapper: A flexible python implementation of the Mapper algorithm. *The Journal of Open Source Software*, 4(42):1315, 2019. doi:10.21105/JOSS.01315.

36 Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv*, 2017. arXiv:1708.07747.

37 Afra J. Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, New York, NY, USA, 2005.

When Alpha-Complexes Collapse onto Codimension-1 Submanifolds

Dominique Attali
Université Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France

Mattéo Clémot
Universite Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS, Villeurbanne, France

Bianca B. Dornelas
Institute of Geometry, TU Graz, Austria
Institute for Medical Informatics, Statistics and Documentation, MedUni Graz, Austria

André Lieutier
No affiliation, Aix-en-Provence, France

Abstract

Given a finite set of points P sampling an unknown smooth surface $\mathcal{M} \subseteq \mathbb{R}^3$, our goal is to triangulate $\mathcal{M}$ based solely on P. Assuming $\mathcal{M}$ is a smooth orientable submanifold of codimension 1 in $\mathbb{R}^d$, we introduce a simple algorithm, *Naive Squash*, which simplifies the α-complex of P by repeatedly applying a new type of collapse called *vertical* relative to $\mathcal{M}$. Naive Squash also has a practical version that does not require knowledge of $\mathcal{M}$. We establish conditions under which both the naive and practical Squash algorithms output a triangulation of $\mathcal{M}$. We provide a bound on the angle formed by triangles in the α-complex with $\mathcal{M}$, yielding sampling conditions on P that are competitive with existing literature for smooth surfaces embedded in $\mathbb{R}^3$, while offering a more compartmentalized proof. As a by-product, we obtain that the restricted Delaunay complex of P triangulates $\mathcal{M}$ when $\mathcal{M}$ is a smooth surface in $\mathbb{R}^3$ under weaker conditions than existing ones.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Submanifold reconstruction, triangulation, abstract simplicial complexes, collapses, convexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.11

Related Version *Full Version*: `https://arxiv.org/pdf/2411.10388` [7]

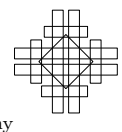
Funding *Bianca B. Dornelas*: Funded by the Austrian Science Fund (FWF), grant W1230.

1 Introduction

Given a finite set of points P that sample an unknown smooth surface $\mathcal{M} \subseteq \mathbb{R}^3$ (example in Figure 1a), we aim to approximate $\mathcal{M}$ based solely on P. This problem, known as *surface reconstruction*, has been widely studied [2,11,14,17,36–38,41]. Several algorithms based on computational geometry have been developed, such as Crust [3], PowerCrust [5], Cocone [4], Wrap [27] and variants based on flow complexes [13,25,34,35]. These algorithms rely on the Delaunay complex of P and offer theoretical guarantees, summarized in [23].

The most desirable guarantee is that the reconstruction outputs a triangulation of $\mathcal{M}$, that is, a simplicial complex whose support is *homeomorphic* to $\mathcal{M}$, in which case we call the algorithm *topologically correct*. That has been established for many of the aforementioned algorithms, assuming that P is noiseless ($P \subseteq \mathcal{M}$) and sufficiently dense. Specifically, let $\mathcal{R} > 0$ be a lower bound on the reach of $\mathcal{M}$, and $\varepsilon \geq 0$ an upper bound on the distance between any point of $\mathcal{M}$ and its nearest point in P. Both Crust and Cocone are topologically correct under the condition $\frac{\varepsilon}{\mathcal{R}} \leq 0.06$ [24], which, to our knowledge, is the weakest such constraint guaranteeing topological correctness for surface reconstruction algorithms in $\mathbb{R}^3$.

© Dominique Attali, Mattéo Clémot, Bianca B. Dornelas, and André Lieutier;
licensed under Creative Commons License CC-BY 4.0

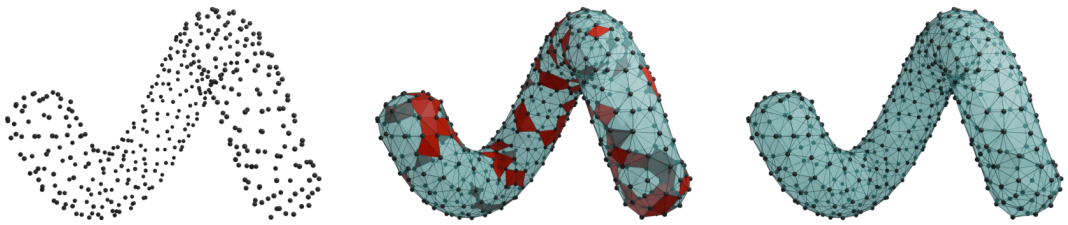


Surface reconstruction generalizes to approximating an unknown smooth submanifold $\mathcal{M} \subseteq \mathbb{R}^d$ from a finite sample P. One approach in that case, similar to the Wrap algorithm in $\mathbb{R}^3$, involves *collapses*, which are typically applied to complexes like the α-*complex* [12]. The α-complex of P [28,29,31] includes simplices whose circumspheres have radius $\leq \alpha$ and enclose no other points of P [30]. For well-chosen α, the α-complex has the same homotopy type as $\mathcal{M}$ [8,18,19,40], provided that P is sufficiently dense and has low noise relative to the reach of $\mathcal{M}$. However, it may still fail to capture the topology of $\mathcal{M}$, as illustrated in Figure 1b: for $\mathcal{M} \subseteq \mathbb{R}^3$, the α-complex of P includes *slivers*, tetrahedra that have one dimension more than $\mathcal{M}$, preventing the existence of a homeomorphism. Slivers complicate reconstructing k-dimensional submanifolds in $\mathbb{R}^d$ for $k \geq 2$ for all Delaunay-based reconstruction attempts.

Contributions. We introduce a simple algorithm, `NaiveVerticalSimplification`, which takes as input a simplicial complex K and simplifies it by applying collapses guided by the knowledge of $\mathcal{M}$. We call it naive because this knowledge is non-realistic in practice. We find conditions under which the algorithm is topologically correct for smooth orientable submanifolds $\mathcal{M}$ of $\mathbb{R}^d$ with codimension one. Its variant, `PracticalVerticalSimplification`, does not rely on $\mathcal{M}$ and remains topologically correct, though it requires stricter conditions. When applying both algorithms to the α-complex of P and returning the result, we obtain two reconstruction algorithms which we refer to as `NaiveSquash` and `PracticalSquash`, respectively. We determine conditions on the inputs P and α that guarantee the topological correctness of these squash algorithms. Moreover, for $d = 3$, we show that `PracticalSquash` is correct under the sampling condition $\frac{\varepsilon}{\mathcal{R}} \leq 0.178$ (see Figure 1c for an example output), while `NaiveSquash` is correct for $\frac{\varepsilon}{\mathcal{R}} \leq 0.225$, assuming suitable choice of α. We also show that the *restricted Delaunay complex* [16] is generically homeomorphic to $\mathcal{M}$ when $\frac{\varepsilon}{\mathcal{R}} \leq 0.225$.

In addition, while proving these results, we derive an upper bound for when triangles with vertices on a smooth submanifold $\mathcal{M} \subseteq \mathbb{R}^d$ form a small angle with $\mathcal{M}$: for a triangle abc with $a, b, c \in \mathcal{M}$, longest edge bc, and circumradius ρ, we show that the angle between the affine space spanned by abc and the tangent space to $\mathcal{M}$ at a satisfies:

$$\sin \angle \operatorname{Aff}(abc), \mathbf{T}_a\mathcal{M} \leq \frac{\sqrt{3}\,\rho}{\mathcal{R}}. \tag{1}$$



(a) **(b)** **(c)**

Figure 1 Points sampling a surface in $\mathbb{R}^3$ (a) with the corresponding α-complex, where tetrahedra are highlighted (b). Applying Practical Squash with parameter α outputs (c).

Techniques. Our proof of correctness for the squash algorithms is more compartmentalized than the ones present in the literature: we first consider a smooth orientable submanifold $\mathcal{M} \subseteq \mathbb{R}^d$ with codimension one and a general simplicial complex K embedded in $\mathbb{R}^d$ and contained within a small tubular neighborhood of $\mathcal{M}$. We introduce *vertical* collapses

(relative to $\mathcal{M}$) in K which remove d-simplices of K that either have no d-simplices of K above them in directions normal to $\mathcal{M}$ or no d-simplices of K below them in directions normal to $\mathcal{M}$. `NaiveVerticalSimplification` iteratively applies vertical collapses relative to $\mathcal{M}$. `PracticalVerticalSimplification` does not depend on knowledge of $\mathcal{M}$ and applies vertical collapses relative to a hyperplane, constructed dynamically based on the simplex currently considered for collapse.

We examine conditions for the correctness of these algorithms. Apart from the requirement that K has no vertical i-simplices relative to $\mathcal{M}$ for $0 < i < d$ and that its support projects onto $\mathcal{M}$ and fully covers it, we require the *vertical convexity* of K relative to $\mathcal{M}$. This means that each normal line to $\mathcal{M}$ at a point m (restricted to a small ball around m) intersects the support of K in a convex set. For `PracticalVerticalSimplification`, an additional requirement is that the $(d-1)$-simplices of K must form an angle of at most $\frac{\pi}{4}$ with $\mathcal{M}$.

Afterwards, we present `PracticalSquash` and `NaiveSquash`, which initialize the previous algorithms with K as the α-complex of a point set $P \subseteq \mathbb{R}^d$ that samples $\mathcal{M}$. We show that correctness is guaranteed when the i-simplices in the α-complex form small angles with $\mathcal{M}$ for $0 < i < d$. We provide explicit upper bounds for these angles, expressed in terms of ε, δ, and α, where ε and δ control the sample density and noise in P.

We analyze the case $d = 3$ and provide numerical bounds on the ratios $\frac{\varepsilon}{\mathcal{R}}$ and $\frac{\alpha}{\mathcal{R}}$ that ensure the correctness of both squash algorithms. Instrumental to this step, we derive (1) which enables us to upper bound the angles of triangles in the α-complex relative to the manifold and is of independent interest.

Related work. The Squash algorithms (both practical and naive) are similar to Wrap [12,31] in that they compute a subcomplex K of the Delaunay complex of P and then perform a sequence of collapses. However, the selection of K and the nature of the collapses differ between the two methods: in the naive squash, definitions are relative to $\mathcal{M}$, unlike Wrap, which uses flow lines derived from P to guide the collapsing sequence. This distinction allows us to address the general case first and then focus on the specific case of the α-complex. Moreover, while we guarantee correctness for a larger interval of the ratio $\frac{\varepsilon}{\mathcal{R}}$ compared to previous literature, most existing work addresses non-uniform sampling cases, whereas our work focuses on uniform sampling.

The vertical convexity assumption, crucial for the correctness of our algorithms, has been employed in various forms to establish collapsibility of certain classes of simplicial complexes [1,10,20]. Similarly, bounding the angle between the manifold and the simplices used for reconstruction has been essential in prior work [6,9,21–23]. Our bound (1) remains true when replacing $\mathcal{R}$ with the local feature size of a, as explained in [7, App. A]. The thus modified bound improves upon the known bound [23, Lemma 3.5].

At last, for $d = 3$, we show in the full version [7] that the restricted Delaunay complex is generically homeomorphic to $\mathcal{M}$ for $\frac{\varepsilon}{\mathcal{R}} \leq 0.225$. In contrast, it is proven to be homeomorphic to $\mathcal{M}$ only if $\frac{\varepsilon}{\mathcal{R}} \leq 0.09$ [22, Theorem 13.16], a result based on the Topological Ball Theorem [22, Theorem 13.1]. Our proof bypasses this requirement, relying instead on `NaiveSquash`.

Outline. After the preliminaries in Section 2, Section 3 defines vertically convex simplicial complexes. We introduce the concepts of upper and lower skins for these complexes and prove that both are homeomorphic to their orthogonal projection onto $\mathcal{M}$. Section 4 presents general conditions under which a simplicial complex K can be transformed into a triangulation of $\mathcal{M}$ through either Naive or Practical vertical simplification. Section 5 provides conditions ensuring the topological correctness of both the naive and practical Squash algorithms and the restricted Delaunay complex. All missing proofs can be found in the full version [7].

2 Preliminaries

Subsets and submanifold

Given a subset $X \subseteq \mathbb{R}^d$, we define several important geometric concepts. The convex hull of X is denoted as $\mathrm{conv}(X)$ and the affine space spanned by X as $\mathrm{Aff}(X)$. The interior of X is denoted as X°. The *relative interior* of X, denoted as $\mathrm{relint}(X)$, represents the interior of X within $\mathrm{Aff}(X)$. For any point x and radius r, we denote the closed ball with center x and radius r as $B(x,r)$. The r-offset of X, denoted as $X^{\oplus r}$, is the union of closed balls centered at each point in X with radius r: $X^{\oplus r} = \bigcup_{x\in X} B(x,r)$. The *medial axis* of X, denoted as $\mathrm{axis}(X)$, is the set of points in $\mathbb{R}^d$ that have at least two nearest points in X. The *reach* of X, denoted as $\mathrm{Reach}(X)$, is the infimum of distances between X and $\mathrm{axis}(X)$. Furthermore, we define the projection map $\pi_X : \mathbb{R}^d \setminus \mathrm{axis}(X) \to X$, which associates each point x with its unique closest point in X. This projection map is well-defined on every subset of $\mathbb{R}^d$ that does not intersect $\mathrm{axis}(X)$, particularly on every r-offset of X with $r < \mathrm{Reach}(X)$.

> Throughout the paper, we designate $\mathcal{M}$ as a compact C^2 submanifold of $\mathbb{R}^d$ of codimension one, and, therefore, orientable (see e.g. [42]).

Given $m \in \mathcal{M}$, we denote the affine tangent space to $\mathcal{M}$ at m as $\mathbf{T}_m\mathcal{M}$ and the affine normal space as $\mathbf{N}_m\mathcal{M}$. As $\mathcal{M}$ has codimension one, $\mathbf{T}_m\mathcal{M}$ is a hyperplane and $\mathbf{N}_m\mathcal{M}$ is a line. Additionally, since $\mathcal{M}$ is C^2, it has a positive reach [43]. For all real numbers r such that $0 < r < \mathrm{Reach}(\mathcal{M})$, the r-offset of $\mathcal{M}$ can be partitioned into the set of normal segments $\{\mathbf{N}_m\mathcal{M} \cap B(m,r)\}_{m\in\mathcal{M}}$ [26], that is,

$$\mathcal{M}^{\oplus r} = \dot{\bigcup}_{m\in\mathcal{M}} \mathbf{N}_m\mathcal{M} \cap B(m,r).$$

> We define $\mathbf{n} : \mathcal{M} \to \mathbb{R}^d$ as a differentiable field of unit normal vectors of $\mathcal{M}$ [26]. We let $\mathcal{R}$ be a finite arbitrary number such that $0 < \mathcal{R} \leq \mathrm{Reach}(\mathcal{M})$, fixed throughout.

Abstract simplicial complexes and collapses

We recall some classical definitions of algebraic topology [29,39]. An *abstract simplicial complex* is a collection K of finite non-empty sets with the property that if σ belongs to K, so does every non-empty subset of σ. Each element σ of K is called an *abstract simplex* and its *dimension* is one less than its cardinality: $\dim\sigma = \mathrm{card}\,\sigma - 1$. A simplex of dimension i is called an i-simplex and the set of i-simplices of K is denoted as $K^{[i]}$. If τ and σ are two simplices such that $\tau \subseteq \sigma$, then τ is called a *face* of σ, and σ is called a *coface* of τ. The $(d-1)$-dimensional faces of σ are the *facets* of σ. The *vertex set* of K is $\mathrm{Vert}\,K = \bigcup_{\sigma\in K}\sigma$. A *subcomplex* L of K is a simplicial complex whose elements belong to K. The *link* of σ in K, denoted $\mathrm{Lk}(\sigma,K)$, is the set of simplices τ in K such that $\tau\cup\sigma \in K$ and $\tau\cap\sigma = \emptyset$. It is a subcomplex of K. The *star* of σ in K, denoted as $\mathrm{St}(\sigma,K)$, is the set of cofaces of σ. The simplicial complex formed by all the faces of σ is the *closure* of σ, $\mathrm{Cl}\,\sigma$.

Consider next an abstract simplex $\sigma \subseteq \mathbb{R}^d$. One can associate it to the geometric simplex $\mathrm{conv}(\sigma) \subseteq \mathbb{R}^d$, called the *support* of σ. In general, $\dim(\mathrm{Aff}(\sigma)) \leq \dim(\sigma)$ and we say that σ is *non-degenerate* whenever $\dim(\mathrm{Aff}(\sigma)) = \dim\sigma$. Given a simplicial complex K with vertices in $\mathbb{R}^d$, we say that K is *canonically embedded* if the following two conditions are satisfied:

1. $\dim\sigma = \dim(\mathrm{Aff}(\sigma))$ for all $\sigma \in K$;
2. $\mathrm{conv}(\alpha\cap\beta) = \mathrm{conv}(\alpha)\cap\mathrm{conv}(\beta)$ for all $\alpha,\beta \in K$.

In this paper we consider exclusively abstract simplicial complexes K with vertex sets in $\mathbb{R}^d$ and which are canonically embedded.

Given such a simplicial complex, its *underlying space* (or *support*) is the point set $|K| = \bigcup_{\sigma \in K} \text{conv}(\sigma)$. If $|K|$ is homeomorphic to $\mathcal{M}$, then K is called a *triangulation* of $\mathcal{M}$ or is said to triangulate $\mathcal{M}$. Since K is canonically embedded, the link of every i-simplex of K falls into one of the following two categories: (1) it is a triangulation of the sphere of dimension $d-i-1$ or (2) it is a proper[1] subcomplex of such a triangulation. The *boundary complex* of a simplicial complex K is the subset of simplices in the second category, denoted ∂K, and it holds that $|\partial K| = \partial |K|$. Simplices in ∂K are referred to as *boundary simplices* of K. Given a set of abstract simplices Σ, if $\sigma \in \Sigma$ has no coface in Σ besides itself, then σ is said to be *inclusion-maximal* in Σ.

Suppose that $\tau \in K$ is a simplex whose star in K has a unique inclusion-maximal element $\sigma \neq \tau$. Then τ is said to be *free* in K. Equivalently, τ is free in K if and only if the link of τ in K is the closure of a simplex. Consequently, free simplices of K are always boundary simplices of K. However, not all boundary simplices of K are necessary free. There are instances where none of them are free, such as the famous example when K triangulates the 2-dimensional subspace of $\mathbb{R}^3$, known as the "house with two rooms". A *collapse* in K is the operation that removes from K a free simplex τ along with all its cofaces. This operation is known to preserve the homotopy-type of $|K|$.

Delaunay complexes, α-complexes, and α-shapes

Consider a finite collection of points $P \subseteq \mathbb{R}^d$. The Voronoi region of $q \in P$ is the collection of points $x \in \mathbb{R}^d$ that are closer to q than to any other points of P:

$$V(q,P) = \{x \in \mathbb{R}^d \mid \|x-q\| \leq \|x-p\|, \text{ for all } p \in P\}.$$

Given a subset $\sigma \subseteq P$, let $V(\sigma,P) = \bigcap_{q\in\sigma} V(q,P)$. The *Delaunay complex* is defined as

$$\text{Del}(P) = \{\sigma \subseteq P \mid \sigma \neq \emptyset \text{ and } V(\sigma,P) \neq \emptyset\}.$$

A simplex $\sigma \in \text{Del}(P)$ is called a *Delaunay simplex* of P and it is *dual* to its corresponding *Voronoi cell* $V(\sigma,P)$. Henceforth, we assume that the set of points P is in *general position*. This means that no $d+2$ points of P lie on the same d-dimensional sphere and no $k+2$ points of P lie on the same k-dimensional flat for $k < d$. In that case, $\text{Del}(P)$ is canonically embedded [33]. For $\alpha \geq 0$, the α-complex of P is the subcomplex of $\text{Del}(P)$ defined by:

$$\text{Del}(P,\alpha) = \{\sigma \subseteq P \mid \sigma \neq \emptyset \text{ and } V(\sigma,P) \cap P^{\oplus\alpha} \neq \emptyset\}.$$

Its underlying space $|\text{Del}(P,\alpha)| = \bigcup_{\sigma\in\text{Del}(P,\alpha)} \text{conv}(\sigma)$ is called the *α-shape* of P. It has the properties: (i) $|\text{Del}(P,\alpha)| \subseteq P^{\oplus\alpha}$ and (ii) $|\text{Del}(P,\alpha)|$ is homotopy equivalent to $P^{\oplus\alpha}$; see [28] for more details.

3 Vertically convex simplicial complexes

In this section, we define the concept of vertical convexity relative to $\mathcal{M}$ for both a set and a simplicial complex. We then study the boundary of a vertically convex simplicial complex K. Specifically, we divide the boundary of its underlying space into an upper and a lower skins,

[1] A *proper* subset A of B is such that $A \neq B$.

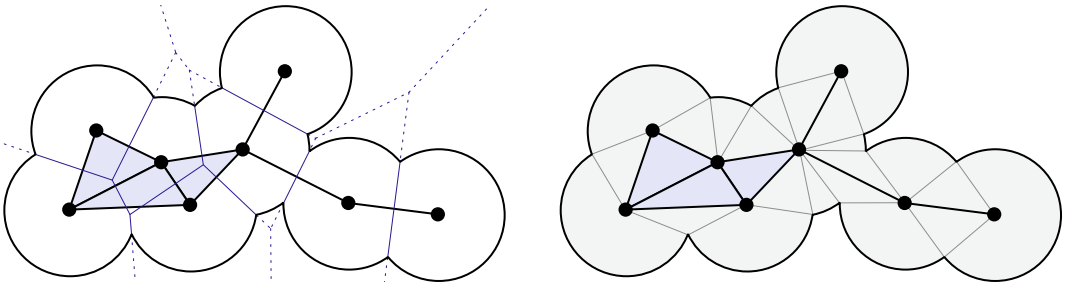


Figure 2 Left: P is such that neither $P^{\oplus\alpha}$ nor $\mathrm{Del}(P,\alpha)$ are vertically convex relative to a horizontal line. Right: Decomposition of $P^{\oplus\alpha} \setminus |\mathrm{Del}(P,\alpha)|^{\circ}$ in joins as described in [28].

enabling us to identify two boundary subcomplexes: an upper and a lower ones. Furthermore, we show that each of these subcomplexes triangulates the orthogonal projection of $|K|$ onto $\mathcal{M}$ (Lemma 6). We also extend the definitions for a single d-simplex.

▶ **Definition 1** (Vertical convexity). *A set $X \subseteq \mathbb{R}^d$ is* vertically convex *relative to $\mathcal{M}$ if $\exists\, r \in [0, \mathrm{Reach}(\mathcal{M}))$ such that*

1. *$X \subseteq \mathcal{M}^{\oplus r}$ and*
2. *$\forall\, m \in \mathcal{M}$, $\mathbf{N}_m\mathcal{M} \cap B(m,r) \cap X$ is convex.*

In other words, for any $m \in \mathcal{M}$, the set $\mathbf{N}_m\mathcal{M} \cap B(m,r) \cap X$ is either empty or a line segment (possibly of zero-length). A simplicial complex K is vertically convex *relative to $\mathcal{M}$ if its underlying space $|K|$ is.*

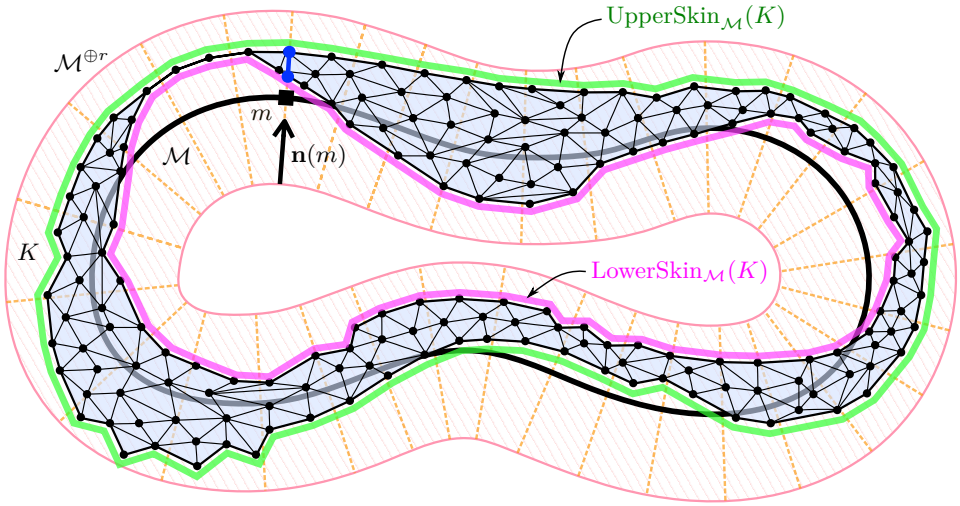


Figure 3 A simplicial complex K vertically convex relative to the curve $\mathcal{M}$. Each segment $\mathbf{N}_m\mathcal{M} \cap B(m,r)$ (in dashed orange) intersects $|K|$ in a line segment, as highlighted (blue) for the point m (represented by a black square). Lemma 4 shows that each of the two skins of $|K|$, depicted in green and pink according to the labeling arrows, is homeomorphic to $\mathcal{M}$.

Examples of a non-vertically convex and a vertically convex simplicial complexes are provided in Figures 2 and 3, respectively.

3.1 Upper and lower skins

Assume that $X \subseteq \mathbb{R}^d$ is vertically convex relative to $\mathcal{M}$ and let $m \in \pi_{\mathcal{M}}(X)$. The endpoints (possibly equal) of the segment $\mathbf{N}_m\mathcal{M} \cap B(m,r) \cap X$ are denoted by $\mathrm{low}_X(m)$ and $\mathrm{up}_X(m)$, with $\mathrm{up}_X(m)$ being above $\mathrm{low}_X(m)$ along the direction of the unit normal vector $\mathbf{n}(m)$. With this notation, X can be expressed as a union of disjoint normal segments:

$$X = \dot{\bigcup}_{m \in \pi_{\mathcal{M}}(X)} [\mathrm{low}_X(m), \mathrm{up}_X(m)].$$

The *upper skin* and *lower skin* of X are, respectively:

$$\mathrm{UpperSkin}_{\mathcal{M}}(X) = \{\mathrm{up}_X(m) \mid m \in \pi_{\mathcal{M}}(X)\},$$
$$\mathrm{LowerSkin}_{\mathcal{M}}(X) = \{\mathrm{low}_X(m) \mid m \in \pi_{\mathcal{M}}(X)\}.$$

Figure 3 displays an example. Our goal is to study the skins of $|K|$, for which we need two extra definitions.

▶ **Definition 2** (Vertical simplex). *A simplex $\sigma \subseteq \mathbb{R}^d$ such that $\mathrm{conv}(\sigma) \subseteq \mathbb{R}^d \setminus \mathrm{axis}(\mathcal{M})$ is* vertical *relative to $\mathcal{M}$ if there exists a pair of distinct points in $\mathrm{conv}(\sigma)$ sharing the same projection onto $\mathcal{M}$.*

▶ **Definition 3** (Non-vertical skeleton). *Assume that $|K| \subseteq \mathbb{R}^d \setminus \mathrm{axis}(\mathcal{M})$. We say that K has a* non-vertical skeleton *relative to $\mathcal{M}$ if K contains no vertical i-simplices relative to $\mathcal{M}$ for all integers $0 < i < d$.*

The next lemma is a key property of vertically convex simplicial complexes:

▶ **Lemma 4.** *Suppose that K is vertically convex and has a non-vertical skeleton relative to $\mathcal{M}$. Then, the upper and lower skins of $|K|$ are closed sets, each homeomorphic to $\pi_{\mathcal{M}}(|K|)$. The homeomorphism is realized in both cases by $\pi_{\mathcal{M}}$. In addition,*

$$\partial|K| = \mathrm{UpperSkin}_{\mathcal{M}}(|K|) \cup \mathrm{LowerSkin}_{\mathcal{M}}(|K|). \tag{2}$$

A simple consequence follows:

▶ **Lemma 5.** *Let K be vertically convex and with non-vertical skeleton relative to $\mathcal{M}$. If $\mathrm{UpperSkin}_{\mathcal{M}}(|K|) = \mathrm{LowerSkin}_{\mathcal{M}}(|K|)$, then $K = \partial K$.*

Aiming for a simplicial version of Equation (2), we define the *upper complex* of K and the *lower complex* of K relative to $\mathcal{M}$ as follows:

$$\mathrm{UpperComplex}_{\mathcal{M}}(K) = \{\nu \subseteq \partial K \mid \mathrm{conv}(\nu) \subseteq \mathrm{UpperSkin}_{\mathcal{M}}(|K|)\},$$
$$\mathrm{LowerComplex}_{\mathcal{M}}(K) = \{\nu \subseteq \partial K \mid \mathrm{conv}(\nu) \subseteq \mathrm{LowerSkin}_{\mathcal{M}}(|K|)\}.$$

By construction, both are subcomplexes of ∂K. A combinatorial equivalent of Lemma 4 is:

▶ **Lemma 6.** *Let K be vertically convex with non-vertical skeleton relative to $\mathcal{M}$. Then,*

$$|\,\mathrm{UpperComplex}_{\mathcal{M}}(K)| = \mathrm{UpperSkin}_{\mathcal{M}}(|K|),$$
$$|\,\mathrm{LowerComplex}_{\mathcal{M}}(K)| = \mathrm{LowerSkin}_{\mathcal{M}}(|K|) \quad \textit{and}$$
$$\partial K = \mathrm{UpperComplex}_{\mathcal{M}}(K) \cup \mathrm{LowerComplex}_{\mathcal{M}}(K).$$

Moreover, if $\pi_{\mathcal{M}}(|K|) = \mathcal{M}$, both $\mathrm{UpperComplex}_{\mathcal{M}}(K)$ and $\mathrm{LowerComplex}_{\mathcal{M}}(K)$ are triangulations of $\mathcal{M}$.

3.2 Upper and lower facets of a d-simplex

Let σ be a non-degenerate d-simplex of $\mathbb{R}^d$ such that $\mathrm{conv}(\sigma) \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$. In that case, $\mathrm{Cl}\,\sigma$ is embedded and vertically convex relative to $\mathcal{M}$. The facets of σ can be partitioned into *upper facets* and *lower facets* of σ relative to $\mathcal{M}$ as follows:

$$\mathrm{UpperFacets}_{\mathcal{M}}(\sigma) = \{\nu \text{ facet of } \sigma \mid \nu \in \mathrm{UpperComplex}_{\mathcal{M}}(\mathrm{Cl}\,\sigma)\}$$
$$\mathrm{LowerFacets}_{\mathcal{M}}(\sigma) = \{\nu \text{ facet of } \sigma \mid \nu \in \mathrm{LowerComplex}_{\mathcal{M}}(\mathrm{Cl}\,\sigma)\}.$$

An example can be seen in Figure 4, where one can also observe the following property:

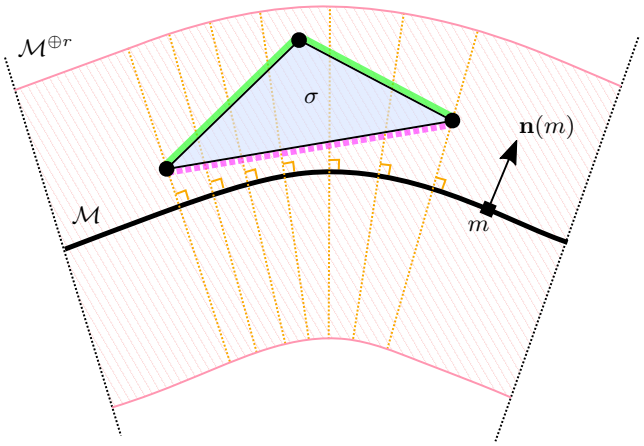


Figure 4 Upper (smooth green edges) and lower (dotted pink edge) facets of a 2-simplex $\sigma \subseteq \mathbb{R}^2$.

▶ **Lemma 7.** *Consider a non-degenerate d-simplex $\sigma \subseteq \mathbb{R}^d$ such that $\mathrm{conv}(\sigma) \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$. If σ has no vertical facets relative to $\mathcal{M}$, then $\mathrm{UpperFacets}_{\mathcal{M}}(\sigma)$ and $\mathrm{LowerFacets}_{\mathcal{M}}(\sigma)$ are non-empty sets that partition the facets of σ.*

4 Vertically collapsing simplicial complexes

In this section, assuming that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$, we introduce an algorithm for simplifying K using vertical collapses relative to $\mathcal{M}$ (Section 4.1) and establish conditions for when it outputs a triangulation of $\mathcal{M}$ (Section 4.2). We first present a naive version that requires the knowledge of $\mathcal{M}$ and then present a practical version (Section 4.3).

4.1 Naive algorithm

▶ **Definition 8** (Vertically free simplices). *A simplex τ is said to be* free from above (resp., free from below) *in K relative to $\mathcal{M}$ if*

- *τ is a free simplex of K;*
- *the unique inclusion-maximal simplex σ in $\mathrm{St}(\tau, K)$ has dimension d;*
- *the set of $(d-1)$-simplices in $\mathrm{St}(\tau, K)$ is exactly the set of upper (resp., lower) facets of σ relative to $\mathcal{M}$.*

We say that τ is vertically free *in K relative to $\mathcal{M}$ if τ is either free from above or from below in K relative to $\mathcal{M}$. See Figures 5 and 7 for a depiction.*

▶ Remark 9. Definition 8 can be naturally extended to non-compact submanifolds $\mathcal{M}$. In particular, it holds for hyperplanes, a fact that we use in Algorithm 2.

▶ **Definition 10** (Vertical collapse). *A* vertical collapse *of K relative to $\mathcal{M}$ is the operation of removing the star of a simplex $\tau \in K$ that is vertically free relative to $\mathcal{M}$.*

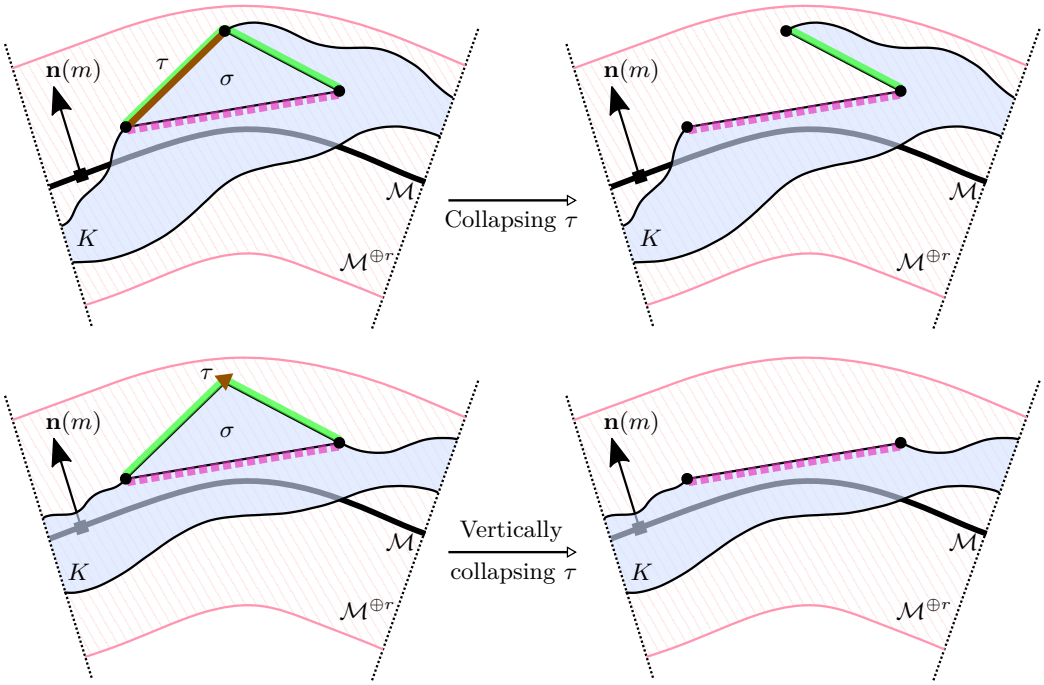


Figure 5 Schematic drawings of K in blue (smooth filled areas). Top row: the edge τ is free but not vertically free relative to $\mathcal{M}$ and collapsing τ does not preserve the vertical convexity of K. Bottom row: the vertex τ is free from above relative to $\mathcal{M}$, so that collapsing τ preserves the vertical convexity of K (Lemma 14). The $(d-1)$-simplices of K that disappear with τ are precisely the upper facets of σ (smooth edges, in green).

A vertical collapse of K can be seen as compressing the underlying space of K by shifting its upper or lower skin along directions normal to $\mathcal{M}$; see Figure 5. Our first algorithm, outlined in Algorithm 1, simplifies K by iteratively applying vertical collapses relative to $\mathcal{M}$. It is worth noting that the algorithm operates on any simplicial complex K with $|K| \subseteq \mathcal{M}^{\oplus r}$ for $r < \text{Reach}(\mathcal{M})$, irrespective of whether K is vertically convex relative to $\mathcal{M}$ or not.

Algorithm 1 `NaiveVerticalSimplification`(K).

while there is a simplex τ vertically free in K relative to $\mathcal{M}$ **do**
  Collapse τ in K;
end while

4.2 Correctness

We now establish conditions under which `NaiveVerticalSimplification`(K) transforms K into a triangulation of $\mathcal{M}$. For that, we introduce a binary relation over d-simplices:

▶ **Definition 11** (Below relation $\prec_{\mathcal{M}}$). *Let $\sigma_0, \sigma_1 \subseteq \mathbb{R}^d$ be two d-simplices sharing a common facet $\nu = \sigma_0 \cap \sigma_1$ and let $\text{conv}(\sigma_0) \cup \text{conv}(\sigma_1) \subseteq \mathcal{M}^{\oplus r}$ for some $r < \text{Reach}(\mathcal{M})$. We say that σ_0 is* below *σ_1 (or that σ_1 is* above *σ_0) relative to $\mathcal{M}$, denoted $\sigma_0 \prec_{\mathcal{M}} \sigma_1$, if ν is an upper facet of σ_0 and a lower facet of σ_1 relative to $\mathcal{M}$.*

Note that the relation $\prec_{\mathcal{M}}$ is not acyclic in general, see Figure 6.

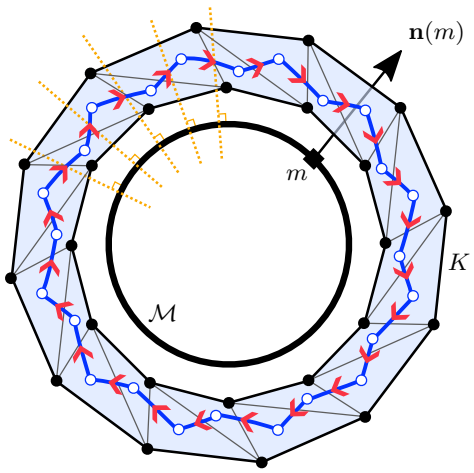


Figure 6 Non-Delaunay triangles that form a cycle in the $\prec_{\mathcal{M}}$ relation and their dual graph.

▶ **Theorem 12** (Correctness). *Consider K such that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$ and assume the following:*
Injective projection: *K has a non-vertical skeleton relative to $\mathcal{M}$.*
Covering projection: *$\pi_{\mathcal{M}}(|K|) = \mathcal{M}$.*
Vertical convexity: *K is vertically convex relative to $\mathcal{M}$.*
Acyclicity: *$\prec_{\mathcal{M}}$ is acyclic over d-simplices of K.*
Then, `NaiveVerticalSimplification`(K) *transforms K into a triangulation of $\mathcal{M}$.*

The remaining of this section aims to prove Theorem 12 and we consider K such that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$. Using the relation $\prec_{\mathcal{M}}$, associate to K its dual graph $G_{\mathcal{M}}(K)$ that has one node for each d-simplex of K and one arc for each pair of d-simplices $\sigma_0, \sigma_1 \in K$ that share a common facet $\sigma_0 \cap \sigma_1$. Direct an arc from σ_0 to σ_1 if $\sigma_0 \prec_{\mathcal{M}} \sigma_1$, and from σ_1 to σ_0 otherwise. Since either $\sigma_0 \prec_{\mathcal{M}} \sigma_1$ or $\sigma_1 \prec_{\mathcal{M}} \sigma_0$, this yields a well-defined orientation for each arc in the dual graph. Figures 6 and 7 show examples.

In a directed graph, a *source* is a node with only outgoing arcs, while a *sink* is a node with only incoming arcs. The next lemma states that a vertical collapse in K corresponds to the removal of either a sink or a source in $G_{\mathcal{M}}(K)$ and conversely. For that, given a finite set of abstract simplices $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, let $\bigcap \Sigma = \bigcap_{i=1}^{k} \sigma_i$ denote the set of vertices that belong to all simplices in Σ. If $\bigcap \Sigma \neq \emptyset$, it forms an abstract simplex.

▶ **Lemma 13** (Sinks and sources). *Consider K such that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$ and assume that K satisfies the injective projection, covering projection and vertical convexity assumptions of Theorem 12. Consider a d-simplex $\sigma \in K$ and let $\tau = \bigcap \mathrm{UpperFacets}_{\mathcal{M}}(\sigma)$ and $\tau' = \bigcap \mathrm{LowerFacets}_{\mathcal{M}}(\sigma)$. Then,*
- *τ is a free simplex of K from above relative to $\mathcal{M}$ $\iff$ σ is a sink of $G_{\mathcal{M}}(K)$.*
- *τ' is a free simplex of K from below relative to $\mathcal{M}$ $\iff$ σ is a source of $G_{\mathcal{M}}(K)$.*

The next lemma provides an invariant for the while-loop of Algorithm 1.

▶ **Lemma 14** (Loop invariant). *Consider K such that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$. Let τ be a vertically free simplex in K relative to $\mathcal{M}$. Let K' be obtained from K by collapsing τ in K. If K satisfies the assumptions of Theorem 12, then so does K'.*

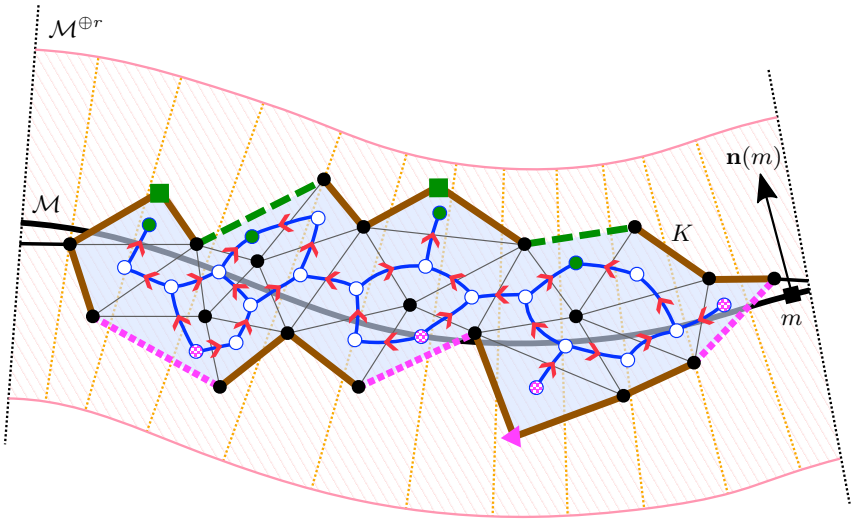


Figure 7 A vertically convex simplicial complex K relative to $\mathcal{M}$. All free simplices are highlighted by thickness. There are four simplices free from below (represented by three dotted edges and one triangular vertex, in pink) and four simplices free from above (represented by two dashed edges and two square vertices, in green). The dual graph (oriented edges, in blue) has four sources (vertices filled by dots, in pink) and four sinks (vertices with a smooth filling, in green), in one-to-one correspondence with the free simplices from below and above.

▶ **Lemma 15** (Upon termination). *Consider K such that $|K| \subseteq \mathcal{M}^{\oplus r}$ for some $r < \mathrm{Reach}(\mathcal{M})$ and assume that K satisfies the injective projection and vertical convexity assumptions of Theorem 12. If $G_{\mathcal{M}}(K) = \emptyset$, then* $\mathrm{LowerSkin}_{\mathcal{M}}(|K|) = \mathrm{UpperSkin}_{\mathcal{M}}(|K|)$.

Proof. We establish the contrapositive:

$$\mathrm{LowerSkin}_{\mathcal{M}}(|K|) \neq \mathrm{UpperSkin}_{\mathcal{M}}(|K|) \quad \Longrightarrow \quad G_{\mathcal{M}}(K) \neq \emptyset$$

Suppose that the two skins are distinct, in other words, that there exists $m \in \mathcal{M}$ such that $\mathrm{low}_{|K|}(m) \neq \mathrm{up}_{|K|}(m)$ and let us show that the segment $[\mathrm{low}_{|K|}(m), \mathrm{up}_K(m)]$ intersects the support of at least one d-simplex of K, implying $G_{\mathcal{M}}(K) \neq \emptyset$. Suppose, for a contradiction, that $[\mathrm{low}_{|K|}(m), \mathrm{up}_{|K|}(m)]$ only intersects the support of i-simplices of K for $i < d$. As $\mathrm{low}_{|K|}(m) \neq \mathrm{up}_{|K|}(m)$ and K has a finite number of simplices, at least one of these i-simplices, say ν, intersects $[\mathrm{low}_{|K|}(m), \mathrm{up}_{|K|}(m)]$ in a non-zero length segment containing distinct points $x, y \in \mathrm{conv}(\nu) \cap [\mathrm{low}_{|K|}(m), \mathrm{up}_{|K|}(m)]$. Hence, x and y share the same orthogonal projection m onto $\mathcal{M}$, implying that ν is vertical relative to $\mathcal{M}$. This contradicts the injective projection assumption on K and therefore establishes the contrapositive. ◀

We now prove the correctness of `NaiveVerticalSimplification`(K).

Proof of Theorem 12. The algorithm starts with K that satisfies the theorem assumptions. By Lemma 14, after each iteration of the while-loop we obtain a new K that continues to satisfy those assumptions. Since each iteration involves a vertical collapse of K relative to $\mathcal{M}$, the number of d-simplices of K is reduced. Thus, the algorithm must terminate. Upon termination, there are no vertically free simplices in K relative to $\mathcal{M}$. By Lemma 13, this implies that, when Algorithm 1 terminates, $G_{\mathcal{M}}(K)$ has no terminal node (neither a source nor a sink) and is therefore empty. By Lemma 15, it follows that $\mathrm{LowerSkin}_{\mathcal{M}}(|K|) = \mathrm{UpperSkin}_{\mathcal{M}}(|K|)$. By Lemma 5, we have $K = \partial K$ and Lemma 6 implies

$$K = \partial K = \mathrm{UpperComplex}_{\mathcal{M}}(K) = \mathrm{LowerComplex}_{\mathcal{M}}(K),$$

with K being a triangulation of $\mathcal{M}$. ◀

4.3 Practical version

Algorithm 1 relies on knowledge of $\mathcal{M}$, which renders it impractical for implementation since $\mathcal{M}$ is typically unknown. In this section, we introduce Algorithm 2, a feasible variant that is correct if the $(d-1)$-simplices of K form a sufficiently small angle with $\mathcal{M}$; see [7, App. A] for a definition of the angle between affine spaces. In this variant, we assign an affine space $\mathcal{H}_\tau$ to each $\tau \in K$: for a free simplex τ with a d-dimensional coface σ, $\mathcal{H}_\tau$ is defined as the hyperplane spanned by any facet of σ. Otherwise, set $\mathcal{H}_\tau = \emptyset$. We also use the notion of vertically free simplices relative to $\mathcal{H}_\tau$, extending Definition 8 as indicated in Remark 9.

Algorithm 2 `PracticalVerticalSimplification`(K).

while there is a simplex τ vertically free in K relative to $\mathcal{H}_\tau$ **do**
  Collapse τ in K;
end while

▶ **Theorem 16** (Correctness). *Suppose that K satisfies the assumptions of Theorem 12 and, in addition, for all $(d-1)$-simplices ν of K*

$$\max_{a \in \nu} \angle(\mathrm{Aff}(\nu), \mathbf{T}_{\pi_{\mathcal{M}}(a)}\mathcal{M}) < \frac{\pi}{4}. \tag{3}$$

Then, `PracticalVerticalSimplification`(K) *transforms K into a triangulation of $\mathcal{M}$.*

5 Correct reconstructions from α-complexes

In this section, we assume that $\mathcal{M}$ is sampled by a finite point set P and consider Algorithms 3 and 4, which apply vertical collapses to $\mathrm{Del}(P, \alpha)$ either straightforwardly or practically. We introduce two parameters, $\varepsilon \geq 0$ and $\delta \geq 0$, to control the sample density and noise of P, respectively, and a scale parameter $\alpha \geq 0$. Section 5.1 establishes conditions ensuring the correctness of Algorithms 3 and 4, with the proof outlined in Section 5.2. Section 5.3 shows how these conditions hold for a wide range of $\frac{\varepsilon}{\mathcal{R}}$ and $\frac{\alpha}{\mathcal{R}}$ when $\mathcal{M}$ is a surface in $\mathbb{R}^3$ and P is noiseless. This result is extended to the restricted Delaunay complex of P in Section 5.4.

Algorithm 3 `NaiveSquash`(P, α).

$K \leftarrow \mathrm{Del}(P, \alpha)$; `NaiveVerticalSimplification`$(K)$; **return** K;

Algorithm 4 `PracticalSquash`(P, α).

$K \leftarrow \mathrm{Del}(P, \alpha)$; `PracticalVerticalSimplification`$(K)$; **return** K;

5.1 Sampling and angular conditions in $\mathbb{R}^d$

The next definition enables us to express our results in $\mathbb{R}^d$ more concisely.

▶ **Definition 17** (Strict homotopy condition). *We say that $\varepsilon, \delta \geq 0$ satisfy the* strict homotopy condition *if $(\mathcal{R}-\delta)^2 - \varepsilon^2 > (4\sqrt{2}-5)\mathcal{R}^2$ for $\delta \leq \varepsilon$ and $\varepsilon + \sqrt{2}\delta < (\sqrt{2}-1)\mathcal{R}$ for $\delta \geq \varepsilon$.*

Let $I(\varepsilon, \delta)$ be an interval of α values so that $P^{\oplus\alpha}$ is vertically convex with relation to $\mathcal{M}$. The exact definition can be found in [7, App. D.1]. The fact that this interval guarantees vertical convexity follows from the specialization of Propositions 5 and 7 in [8] to the case where $\mathcal{M}$ has codimension-one:

▶ **Theorem 18** (Specialization of [8]). *Suppose that $\mathcal{M} \subseteq P^{\oplus\varepsilon}$ and $P \subseteq \mathcal{M}^{\oplus\delta}$ with $\varepsilon, \delta \geq 0$ that satisfy the strict homotopy condition. Then, for all $\alpha \in I(\varepsilon, \delta)$, $\pi_{\mathcal{M}}(P^{\oplus\alpha}) = \mathcal{M}$ and $P^{\oplus\alpha}$ is vertically convex relative to $\mathcal{M}$. Thus, $P^{\oplus\alpha}$ has associated upper and lower skins and deformation-retracts onto $\mathcal{M}$ along $\pi_{\mathcal{M}}$. In addition, the two skins partition $\partial P^{\oplus\alpha}$.*

The above concepts can be put together to state our main theorem:

▶ **Theorem 19.** *Assume $\mathcal{M} \subseteq P^{\oplus\varepsilon}$ and $P \subseteq \mathcal{M}^{\oplus\delta}$ for $\varepsilon, \delta \geq 0$ that satisfy the strict homotopy condition. Let $\alpha \in \left[\delta, \frac{2(\mathcal{R}-\delta)}{3}\right) \cap I(\varepsilon, \delta)$ and $\beta > 0$ such that $\mathcal{M}^{\oplus\beta} \subseteq P^{\oplus\alpha}$. Suppose that for all i-simplices $\tau \in \mathrm{Del}(P, \alpha)$, $0 < i < d$ and all $(d-1)$-simplices $\nu \in \mathrm{Del}(P, \alpha)$ it holds:*

$$\max_{x \in \mathrm{conv}(\tau)} \angle(\mathrm{Aff}(\tau), \mathbf{T}_{\pi_{\mathcal{M}}(x)}\mathcal{M}) < \frac{\pi}{2}, \tag{4}$$

$$\min_{a \in \tau} \angle(\mathrm{Aff}(\tau), \mathbf{T}_{\pi_{\mathcal{M}}(a)}\mathcal{M}) < \arcsin\left(\frac{(\mathcal{R}+\beta)^2 - (\mathcal{R}+\delta)^2 - \alpha^2}{2(\mathcal{R}+\delta)\alpha}\right) \text{ and} \tag{5}$$

$$\min_{x \in \mathrm{conv}(\nu)} \angle(\mathrm{Aff}(\nu), \mathbf{T}_{\pi_{\mathcal{M}}(x)}\mathcal{M}) < \frac{\pi}{2} - 2\arcsin\left(\frac{\alpha}{2(\mathcal{R}-\delta-\alpha)}\right). \tag{6}$$

Then, $\mathrm{Del}(P, \alpha)$ satisfies the injective projection, covering projection, vertical convexity and acyclicity assumptions of Theorem 12. Furthermore, both the upper and lower complexes of $\mathrm{Del}(P, \alpha)$ relative to $\mathcal{M}$ are triangulations of $\mathcal{M}$ and $\texttt{NaiveSquash}(P, \alpha)$ returns a triangulation of $\mathcal{M}$.

One can check that Conditions (4), (5) and (6) are well-defined; see Remark E1 in [7].

▶ **Corollary 20.** *Suppose the assumptions of Theorem 19 are satisfied and furthermore that for all $(d-1)$-simplices $\nu \in \mathrm{Del}(P, \alpha)$,*

$$\max_{a \in \nu} \angle(\mathrm{Aff}(\nu), \mathbf{T}_{\pi_{\mathcal{M}}(a)}\mathcal{M}) < \frac{\pi}{4}. \tag{7}$$

Then, $\texttt{PracticalSquash}(P, \alpha)$ returns a triangulation of $\mathcal{M}$.

5.2 Partial proof technique for Theorem 19

In this section, we establish the covering projection and vertical convexity of $\mathrm{Del}(P, \alpha)$, as guaranteed by Theorem 19. The complete proof of that theorem, which is unfortunately too lengthy to include here, can be found in [7, App. E].

▶ **Lemma 21.** *Assume $\mathcal{M} \subseteq P^{\oplus\varepsilon}$ and $P \subseteq \mathcal{M}^{\oplus\delta}$ for $\varepsilon, \delta \geq 0$ that satisfy the strict homotopy condition. Let $\alpha \in [\delta, \mathcal{R} - \delta) \cap I(\varepsilon, \delta)$ and $\beta > 0$ be such that $\mathcal{M}^{\oplus\beta} \subseteq P^{\oplus\alpha}$. Suppose that for all i-simplices $\tau \in \mathrm{Del}(P, \alpha)$ with $0 < i < d$, Conditions (4) and (5) hold. Then, $\pi_{\mathcal{M}}(|\mathrm{Del}(P, \alpha)|) = \mathcal{M}$ and $\mathrm{Del}(P, \alpha)$ is vertically convex relative to $\mathcal{M}$.*

Upper and lower joins. For proving this lemma, we introduce upper and lower joins. Consider $\mathcal{M}$, P, ε, δ, α and β that satisfy the assumptions of Lemma 21 and notice that they also meet the conditions of Theorem 18. Therefore, $P^{\oplus\alpha}$ has associated upper and lower skins and the two skins form a partition of $\partial P^{\oplus\alpha}$. Using that partition, we decompose the set difference $P^{\oplus\alpha} \setminus |\mathrm{Del}(P, \alpha)|^\circ$ into upper and lower joins, slightly adapting what is done in [28]; see Figures 2 and 8. For that, notice that $\partial P^{\oplus\alpha}$ can be decomposed into faces, each face being the restriction of $\partial P^{\oplus\alpha}$ to a Voronoi cell of P. There is a one-to-one correspondence

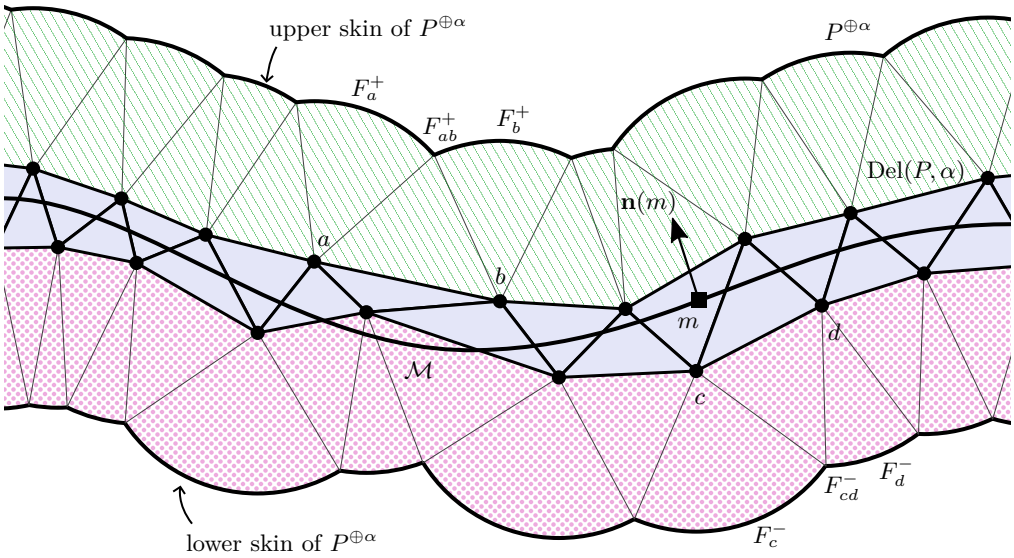


Figure 8 Decomposing $P^{\oplus\alpha} \setminus |\mathrm{Del}(P,\alpha)|^\circ$ into upper joins (hashed, in green) and lower joins (dotted, in pink).

between simplices of $\partial\,\mathrm{Del}(P,\alpha)$ and faces of $\partial P^{\oplus\alpha}$: the simplex $\tau \in \partial\,\mathrm{Del}(P,\alpha)$ corresponds to the face $F_\tau = V(\tau,P) \cap \partial P^{\oplus\alpha}$ and conversely. We can further partition each face F_τ of $\partial P^{\oplus\alpha}$ into a portion F_τ^+ that lies on the upper skin of $\partial P^{\oplus\alpha}$ and a portion F_τ^- that lies on the lower skin of $\partial P^{\oplus\alpha}$. Note that F_τ^+ or F_τ^- can be empty. We refer to F_τ^+ as an upper face and F_τ^- as a lower face. The set of upper faces decompose the upper skin, while the set of lower faces decompose the lower skin. A *join* $X * Y$ is defined as the set of segments $[x,y]$ where $x \in X$ and $y \in Y$ [28]. We call $F_\tau^+ * \mathrm{conv}(\tau)$ an *upper join* and $F_\tau^- * \mathrm{conv}(\tau)$ a *lower join*. The next lemma, proved in [7, App. E.2], identifies points in $\partial|\,\mathrm{Del}(P,\alpha)|$ that are connected to upper or lower joins.

▶ Remark 22. The collection of upper and lower joins cover the set $P^{\oplus\alpha} \setminus |\,\mathrm{Del}(P,\alpha)|^\circ$.

▶ Remark 23. If an upper join and a lower join have a non-empty intersection, the common intersection belongs to $|\,\mathrm{Del}(P,\alpha)|$.

▶ **Lemma 24.** *Under the assumptions of Lemma 21, let* $\gamma \in \mathrm{Del}(P,\alpha)$ *and* $x \in \mathrm{relint}(\mathrm{conv}(\gamma))$. *If for some* $\lambda > 0$ *(*resp. $\lambda < 0$*), the segment* $(x, x + \lambda\mathbf{n}(\pi_{\mathcal{M}}(x))]$ *lies outside* $|\,\mathrm{Del}(P,\alpha)|$, *then it intersects an upper (*resp. lower *join).*

Proof of Lemma 21. Consider $\mathcal{M}$, P, ε, δ, α and β that satisfy the assumptions of Lemma 21. As noted before, they also meet the conditions of Theorem 18. Therefore, $\pi_{\mathcal{M}}(P^{\oplus\alpha}) = \mathcal{M}$ and $P^{\oplus\alpha}$ is vertically convex relative to $\mathcal{M}$. Hence, there exists $r < \mathrm{Reach}(\mathcal{M})$ such that $P^{\oplus\alpha} \subseteq \mathcal{M}^{\oplus r}$ and $P^{\oplus\alpha} \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ is a line segment for any $m \in \mathcal{M}$. Fix $m \in \mathcal{M}$ arbitrarily. We show that $|\,\mathrm{Del}(P,\alpha)| \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ is also a line segment.

First, we show by contradiction that it is non-empty. Let u^+ (*resp.* u^-) be the endpoint of the segment $P^{\oplus\alpha} \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ that lies on the upper (*resp.* lower) skin of $P^{\oplus\alpha}$ and hence is contained in an upper (*resp.* lower) join. By Remark 22, the entire segment $[u^+,u^-]$ is covered by upper and lower joins. Thus, at some point c of $[u^+,u^-]$, an upper join and a lower join intersect. By Remark 23, such an intersection places c on $|\,\mathrm{Del}(P,\alpha)|$ as well.

Second, we show by contradiction that $|\,\mathrm{Del}(P,\alpha)| \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ is connected. Suppose that $a,b \in |\,\mathrm{Del}(P,\alpha)| \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ are such that $[a,b] \cap |\,\mathrm{Del}(P,\alpha)| = \{a,b\}$ with a being above b along the direction of $\mathbf{n}(m)$. Since $a,b \in |\,\mathrm{Del}(P,\alpha)| \subseteq P^{\oplus\alpha}$ and $P^{\oplus\alpha}$ is

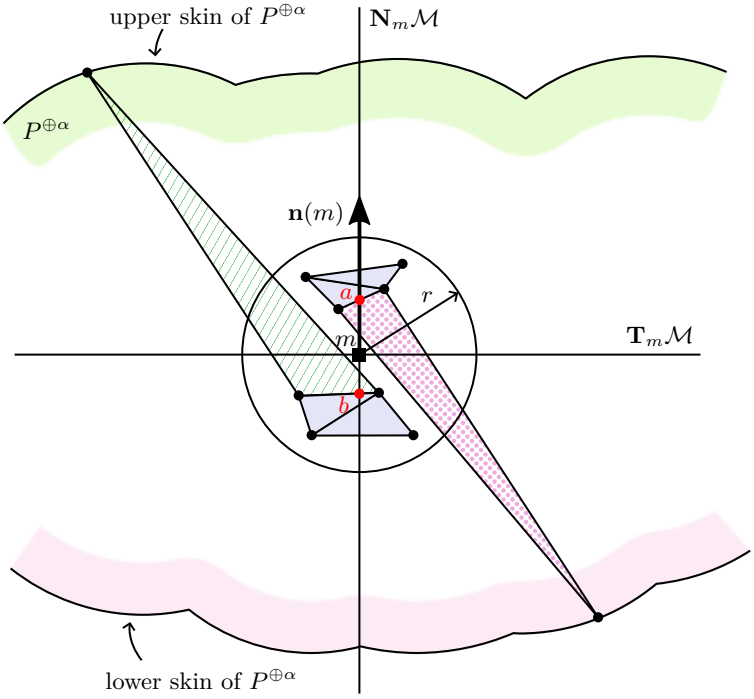


Figure 9 Reaching a contradiction in the proof of Lemma 21. We see the simplices of $\mathrm{Del}(P,\alpha)$ whose support intersects $\mathbf{N}_m\mathcal{M} \cap B(m,r)$ (smooth filling, in pale blue) and one upper (hashed, in green) and one lower (dotted, in pink) joins that intersect $[a,b]$.

vertically convex, the segment $[a,b]$ is contained in $P^{\oplus\alpha}$. By Remark 22, the entire segment $[a,b]$ is covered by upper and lower joins. Let γ_a and γ_b be the simplices of $\mathrm{Del}(P,\alpha)$ that contain a and b, respectively, in their relative interior. Letting $\lambda = \frac{\|a-b\|}{2}$, then both segments $(a, a-\lambda\mathbf{n}(m)]$ and $(b, b+\lambda\mathbf{n}(m)]$ lie outside $|\mathrm{Del}(P,\alpha)|$. It follows, by Lemma 24, that there are at least one lower and one upper joins among the joins that cover (a,b); see Figure 9. Hence, an upper and a lower joins intersect at a point c of the segment (a,b). By Remark 23, c lies in $|\mathrm{Del}(P,\alpha)|$, a contradiction. Therefore, for all $m \in \mathcal{M}$, $|\mathrm{Del}(P,\alpha)| \cap \mathbf{N}_m\mathcal{M} \cap B(m,r)$ is non-empty and connected, thus forming a line segment. ◀

5.3 Sampling conditions for surfaces in $\mathbb{R}^3$

Theorem 19 requires that the i-simplices of $\mathrm{Del}(P,\alpha)$ form a small angle with the manifold $\mathcal{M}$, for $0 < i < d$. Ensuring this can be challenging in practice, especially for $i \geq 3$. However, in the specific case of noiseless edges ($i = 1$) or triangles ($i = 2$), it is possible to upper bound the angle these simplices form with $\mathcal{M}$. For edges, it is known that:

▶ **Lemma 25** ([15, Lemma 7.8]). *If ab is a non-degenerate edge with $a, b \in \mathcal{M}$, then*

$$\sin\angle\,\mathrm{Aff}(ab), \mathbf{T}_a\mathcal{M} \;\leq\; \frac{\|b-a\|}{2\,\mathcal{R}}.$$

For triangles, let $\rho(\tau)$ be the radius of the smallest $(d-1)$-sphere circumscribing τ. We establish a simple bound that is tighter than the previous one (see [23, Lemma 3.5]):

▶ **Lemma 26.** *If abc is a non-degenerate triangle with longest edge bc, for $a, b, c \in \mathcal{M}$, then*

$$\sin \angle \operatorname{Aff}(abc), \mathbf{T}_a\mathcal{M} \leq \frac{\rho(abc)}{\mathcal{R}} \qquad \text{if } abc \text{ is an obtuse triangle and}$$

$$\sin \angle \operatorname{Aff}(abc), \mathbf{T}_a\mathcal{M} \leq \frac{\sqrt{3}\,\rho(abc)}{\mathcal{R}} \qquad \text{if } abc \text{ is an acute triangle.}$$

If abc is obtuse, the bound is tight and happens when $\mathcal{M}$ is a sphere of radius $\mathcal{R}$.

The proof is technical and is therefore provided in [7, App. A]. For the same reason, the proof of the following result, where we use the bounds for edges and triangles to establish sampling conditions for surfaces in $\mathbb{R}^3$, is in [7, App. F]. Let us define

$$\beta_{\varepsilon,\alpha} = -\frac{\varepsilon^2}{2\mathcal{R}} + \sqrt{\alpha^2 + \frac{\varepsilon^4}{4\mathcal{R}^2} - \varepsilon^2},$$

which is one particular value of β that guarantees $\mathcal{M}^{\oplus\beta} \subseteq P^{\oplus\alpha}$; see [7, App. D.2].

▶ **Theorem 27.** *Let $\mathcal{M}$ be a C^2 surface in $\mathbb{R}^3$ whose reach is at least $\mathcal{R} > 0$. Let P be a finite point set such that $P \subseteq \mathcal{M} \subseteq P^{\oplus\varepsilon}$.*

1. *For all $\varepsilon, \alpha \geq 0$ that satisfy $\frac{\sqrt{3}\,\alpha}{\mathcal{R}} < \min\left\{\frac{(\mathcal{R}+\beta_{\varepsilon,\alpha})^2 - \mathcal{R}^2 - \alpha^2}{2\mathcal{R}\alpha}, \cos\left(2\arcsin\left(\frac{\alpha}{2\mathcal{R}}\right)\right)\right\}$,*
   - *the upper and lower complexes of $\operatorname{Del}(P, \alpha)$ relative to $\mathcal{M}$ are triangulations of $\mathcal{M}$;*
   - *$\texttt{NaiveSquash}(P, \alpha)$ returns a triangulation of $\mathcal{M}$.*
2. *For all $\varepsilon, \alpha \geq 0$ that satisfy in addition $\frac{\sqrt{3}\,\alpha}{\mathcal{R}} < \sin\left(\frac{\pi}{4} - 2\arcsin\left(\frac{\alpha}{2\mathcal{R}}\right)\right)$,*
   - *$\texttt{PracticalSquash}(P, \alpha)$ returns a triangulation of $\mathcal{M}$.*

The pairs of $(\frac{\varepsilon}{\mathcal{R}}, \frac{\alpha}{\mathcal{R}})$ that satisfy **1** and **2** are depicted in Figures 10a and 10b, respectively.

▶ Remark 28. In particular, **1** holds for $\frac{\varepsilon}{\mathcal{R}} \leq 0.225$ and $\frac{\alpha}{\mathcal{R}} = 0.359$; and **2** for $\frac{\varepsilon}{\mathcal{R}} \leq 0.178$ and $\frac{\alpha}{\mathcal{R}} = 0.207$, better bounds than the previous existing ones [22, Theorem 13.16][24].

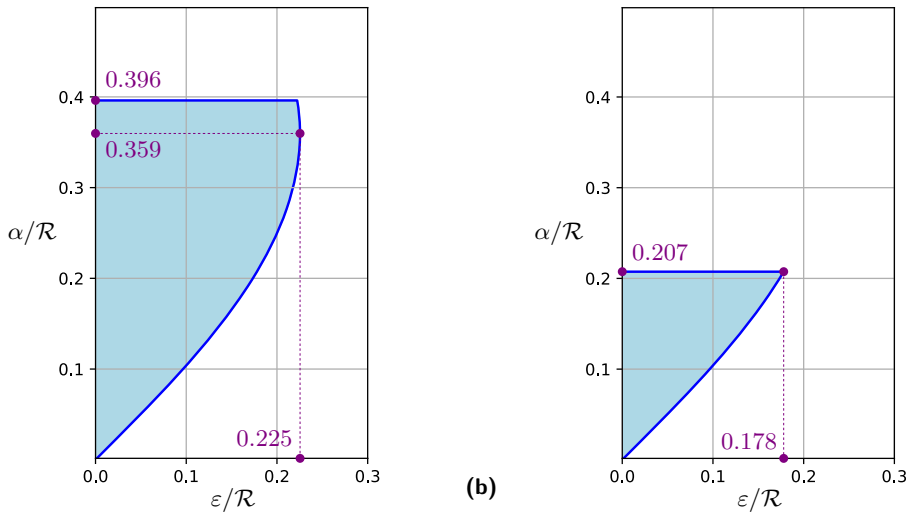


Figure 10 Pairs of $(\frac{\varepsilon}{\mathcal{R}}, \frac{\alpha}{\mathcal{R}})$ for which $\texttt{NaiveSquash}(P, \alpha)$ (a) and $\texttt{PracticalSquash}(P, \alpha)$ (b) are correct, for $P \subseteq \mathcal{M} \subseteq P^{\oplus\varepsilon}$ and $d = 3$.

5.4 The restricted Delaunay complex

We recall from [32] that the *restricted Delaunay complex* is

$$\mathrm{Del}_{\mathcal{M}}(P) = \{\sigma \subseteq P \mid \sigma \neq \emptyset \text{ and } V(\sigma, P) \cap \mathcal{M} \neq \emptyset\}.$$

▶ **Theorem 29.** *Let $\mathcal{M}$ be a C^2 surface in $\mathbb{R}^3$ whose reach is at least $\mathcal{R} > 0$. Let P be a finite set such that $P \subseteq \mathcal{M} \subseteq P^{\oplus\varepsilon}$ for $0 \leq \frac{\varepsilon}{\mathcal{R}} \leq 0.225$. Under the additional generic assumption that all Voronoi cells of P intersect $\mathcal{M}$ transversally, $\mathrm{Del}_{\mathcal{M}}(P)$ is a triangulation of $\mathcal{M}$.*

References

1 Karim Adiprasito and Bruno Benedetti. Barycentric subdivisions of convex complexes are collapsible. *Discrete & Computational Geometry*, 64(3):608–626, 2020. doi:10.1007/S00454-019-00137-3.

2 Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Point set surfaces. In *Proceedings Visualization, 2001. VIS'01.*, pages 21–29. IEEE, 2001. doi:10.1109/VISUAL.2001.964489.

3 N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999. doi:10.1007/PL00009475.

4 Nina Amenta, Sunghee Choi, Tamal K. Dey, and Naveen Leekha. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Comput. Geom. Appl.*, 12(1-2):125–141, 2002. doi:10.1142/S0218195902000773.

5 Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In David C. Anderson and Kunwoo Lee, editors, *Sixth ACM Symposium on Solid Modeling and Applications, Sheraton Inn, Ann Arbor, Michigan, USA, June 4-8, 2001*, pages 249–266. ACM, 2001. doi:10.1145/376957.376986.

6 D. Attali and A. Lieutier. Flat delaunay complexes for homeomorphic manifold reconstruction, 2022. arXiv:2203.05943.

7 Dominique Attali, Mattéo Clémot, Bianca B. Dornelas, and André Lieutier. When alpha-complexes collapse onto codimension-1 submanifolds, 2024. doi:10.48550/arXiv.2411.10388.

8 Dominique Attali, Hana Dal Poz Kouřimská, Christopher Fillmore, Ishika Ghosh, André Lieutier, Elizabeth Stephenson, and Mathijs Wintraecken. Tight bounds for the learning of homotopy à la Niyogi, Smale, and Weinberger for subsets of Euclidean spaces and of Riemannian manifolds. In *Proc. 26th Ann. Sympos. Comput. Geom.*, Athens, Greece, june 11-14 2024.

9 Dominique Attali and André Lieutier. Delaunay-like triangulation of smooth orientable submanifolds by ℓ_1-norm minimization. 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 8:1–8:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.8.

10 Dominique Attali, André Lieutier, and David Salinas. When convexity helps collapsing complexes. In *SoCG 2019-35th International Symposium on Computational Geometry*, page 15, 2019.

11 Håvard Bakke Bjerkevik. Tighter bounds for reconstruction from ε-samples. 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 9:1–9:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.9.

12 Ulrich Bauer and Herbert Edelsbrunner. The morse theory of čech and delaunay complexes. *Transactions of the American Mathematical Society*, 369(5):3741–3762, 2017.

13 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.

14 Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. doi:10.1109/2945.817351.

15 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and topological inference*, volume 57. Cambridge University Press, 2018.

16 Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, and Steve Oudot. Equating the witness and restricted Delaunay complexes. Research Report CGL-TR-24, CGL, November 2011. URL: https://inria.hal.science/hal-00772486.

17 Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)*, pages 67–76, 2001. doi:10.1145/383259.383266.

18 F. Chazal, D. Cohen-Steiner, and A. Lieutier. A sampling theory for compact sets in Euclidean space. *Discrete and Computational Geometry*, 41(3):461–479, 2009. doi:10.1007/S00454-009-9144-8.

19 F. Chazal and A. Lieutier. Smooth Manifold Reconstruction from Noisy and Non Uniform Approximation with Guarantees. *Computational Geometry: Theory and Applications*, 40:156–170, 2008. doi:10.1016/J.COMGEO.2007.07.001.

20 Aaron Chen, Florian Frick, and Anne Shiu. Neural codes, decidability, and a new local obstruction to convexity. *SIAM Journal on Applied Algebra and Geometry*, 3(1):44–66, 2019. doi:10.1137/18M1186563.

21 Siu-Wing Cheng, Tamal K Dey, and Edgar A Ramos. Manifold reconstruction from point samples. In *SODA*, volume 5, pages 1018–1027, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070579.

22 Siu-Wing Cheng, Tamal Krishna Dey, Jonathan Shewchuk, and Sartaj Sahni. *Delaunay mesh generation*. CRC Press Boca Raton, 2013.

23 Tamal K Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2006.

24 Tamal K Dey. Curve and surface reconstruction. In *Handbook of Discrete and Computational Geometry*, pages 915–936. Chapman and Hall/CRC, 2017.

25 Tamal K Dey, Joachim Giesen, Edgar A Ramos, and Bardia Sadri. Critical Points of the Distance to an ε-Sampling of a Surface and Flow-Ccomplex-Based Surface Reconstruction. *International Journal of Computational Geometry & Applications*, 18, 2008.

26 M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Upper Saddle River, New Jersey, 1976.

27 Herbert Edelsbrunner. Surface reconstruction by wrapping finite sets in space. *Discrete and computational geometry: the Goodman-Pollack Festschrift*, pages 379–404, 2003.

28 Herbert Edelsbrunner. Alpha shapes-a survey. In R. van de Weygaert, G. Vegter, J. Ritzerveld, and V. Icke, editors, *Tessellations in the Sciences: Virtues, Techniques and Applications of Geometric Tilings*. Springer, 2011.

29 Herbert Edelsbrunner and John L. Harer. *Computational topology*. American Mathematical Society, Providence, RI, 2010. An introduction. doi:10.1090/mbk/069.

30 Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory*, 29(4):551–558, 1983. doi:10.1109/TIT.1983.1056714.

31 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.

32 Herbert Edelsbrunner and Nimish R Shah. Triangulating topological spaces. *International Journal of Computational Geometry & Applications*, 7(04):365–378, 1997. doi:10.1142/S0218195997000223.

33 Steven Fortune. Voronoi diagrams and delaunay triangulations. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 513–528. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.ch23.

34 Joachim Giesen and Matthias John. Surface reconstruction based on a dynamical system. *Computer graphics forum*, 21(3):363–371, 2002. doi:10.1111/1467-8659.00596.

35 Joachim Giesen and Matthias John. The flow complex: a data structure for geometric modeling. In *SODA*, volume 3, pages 285–294, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644157.

36 Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. Noise-adaptive shape reconstruction from raw point sets. *Computer Graphics Forum*, 32(5):229–238, 2013. doi:10.1111/CGF.12189.

37 Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques (SIGGRAPH)*, pages 71–78, 1992. URL: https://dl.acm.org/citation.cfm?id=134011.

38 Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Computer Graphics Forum*, 7(4), 2006.

39 J.R. Munkres. *Elements of algebraic topology*. Perseus Books, 1993.

40 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.

41 Stefan Ohrhallinger, Jiju Peethambaran, Amal Dev Parakkat, Tamal K Dey, and Ramanathan Muthuganapathy. 2d points curve reconstruction survey and benchmark. *Computer Graphics Forum*, 40(2):611–632, 2021. doi:10.1111/CGF.142659.

42 Hans Samelson. Orientability of hypersurfaces in r^n. *Proceedings of the American Mathematical Society*, 22(1):301–302, 1969.

43 Sebastian Scholtes. On hypersurfaces of positive reach, alternating Steiner formulae and Hadwiger's problem. *arXiv preprint*, April 2013. arXiv:1304.4179.

Higher-Order Color Voronoi Diagrams and the Colorful Clarkson–Shor Framework

Sang Won Bae
Division of AI Computer Science and Engineering, Kyonggi University, Suwon, South Korea

Nicolau Oliver
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Evanthia Papadopoulou
Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Abstract

Given a set S of n colored sites, each $s \in S$ associated with a distance-to-site function $\delta_s \colon \mathbb{R}^2 \to \mathbb{R}$, we consider two distance-to-color functions for each color: one takes the minimum of δ_s for sites $s \in S$ in that color and the other takes the maximum. These two sets of distance functions induce two families of higher-order Voronoi diagrams for colors in the plane, namely, the minimal and maximal order-k color Voronoi diagrams, which include various well-studied Voronoi diagrams as special cases. In this paper, we derive an exact upper bound $4k(n-k)-2n$ on the total number of vertices in both the minimal and maximal order-k color diagrams for a wide class of distance functions δ_s that satisfy certain conditions, including the case of point sites S under convex distance functions and the L_p metric for any $1 \leqslant p \leqslant \infty$. For the L_1 (or, L_∞) metric, and other convex polygonal metrics, we show that the order-k minimal diagram of point sites has $O(\min\{k(n-k),(n-k)^2\})$ complexity, while its maximal counterpart has $O(\min\{k(n-k),k^2\})$ complexity. To obtain these combinatorial results, we extend the Clarkson–Shor framework to colored objects, and demonstrate its application to several fundamental geometric structures, including higher-order color Voronoi diagrams, colored j-facets, and levels in the arrangements of piecewise linear/algebraic curves/surfaces. We also present iterative algorithms to compute higher-order color Voronoi diagrams.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases higher-order Voronoi diagrams, color Voronoi diagrams, Hausdorff Voronoi diagrams, colored j-facets, arrangements, Clarkson–Shor technique

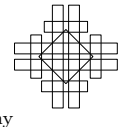
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.12

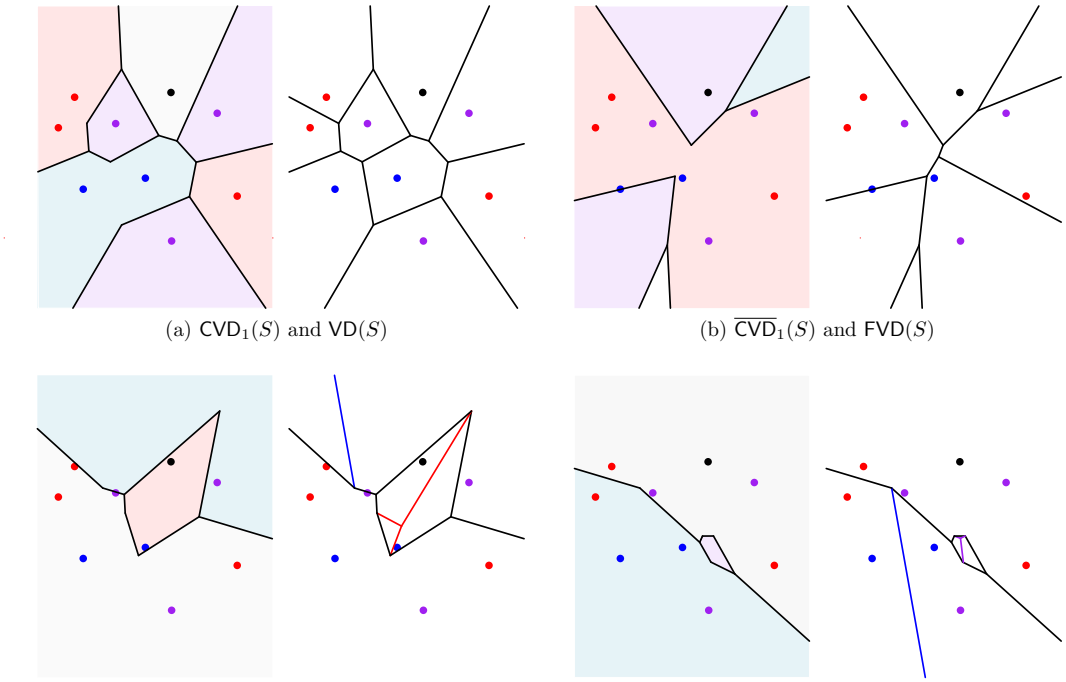
Related Version *Full Version*: `https://arxiv.org/pdf/2504.06960` [10]

Funding *Sang Won Bae*: Supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00251168), and in part by the University of Bayreuth Centre of International Excellence "Alexander von Humboldt" (Senior Fellowship 2023). *Evanthia Papadopoulou*: Supported in part by the Swiss National Science Foundation (SNF), project 200021E-201356.

Acknowledgements The authors would like to thank Otfried Cheong, Christian Knauer, and Fabian Stehn for valuable discussions and comments, in particular, at the beginning of this work. The research by the first author has been done partly during his visits to Universität Bayreuth, Bayreuth, Germany and to Università della Svizzera italiana, Lugano, Switzerland.

© Sang Won Bae, Nicolau Oliver, and Evanthia Papadopoulou;
licensed under Creative Commons License CC-BY 4.0





(a) $\mathsf{CVD}_1(S)$ and $\mathsf{VD}(S)$ (b) $\overline{\mathsf{CVD}}_1(S)$ and $\mathsf{FVD}(S)$

(c) $\mathsf{CVD}_3(S) = \mathsf{FCVD}(S)$ and $\mathsf{FCVD}^*(S)$ (d) $\overline{\mathsf{CVD}}_3(S) = \mathsf{HVD}(S)$ and $\mathsf{HVD}^*(S)$

Figure 1 Special cases of $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ for colored points S in the Euclidean plane.

1 Introduction

Let S be a set of n *sites*, each of which is assigned a color from a set $K = \{1, \ldots, m\}$ of m colors. Let $S_i \subseteq S$ be the set of sites in color $i \in K$. We consider two *distance-to-color* functions from any point $x \in \mathbb{R}^2$ to each color $i \in K$:

$$d_i(x) := \min_{s \in S_i} \delta_s(x) \quad \text{and} \quad \bar{d}_i(x) := \max_{s \in S_i} \delta_s(x),$$

where $\delta_s(x)$ denotes the prescribed distance-to-site function from $x \in \mathbb{R}^2$ to site $s \in S$.

Based on the two sets of point-to-color distances, we can define two different Voronoi diagrams of m colors in K. For each $1 \leqslant k \leqslant m-1$ and subset $H \subseteq K$ with $|H| = k$, define

$$R_k(H;S) := \{x \in \mathbb{R}^2 \mid d_i(x) < d_j(x) \text{ for all } i \in H \text{ and } j \in K \setminus H\}$$
$$\overline{R}_k(H;S) := \{x \in \mathbb{R}^2 \mid \bar{d}_i(x) > \bar{d}_j(x) \text{ for all } i \in H \text{ and } j \in K \setminus H\},$$

called the *minimal* and *maximal color Voronoi regions* of H with respect to S. We then define the *order-k minimal color Voronoi diagram* of S, $\mathsf{CVD}_k(S)$, to be the partition of $\mathbb{R}^2$ into the minimal regions $R_k(H;S)$, for all $H \subset K$ with $|H| = k$; the *order-k maximal color Voronoi diagram* of S, $\overline{\mathsf{CVD}}_k(S)$, to be the partition of $\mathbb{R}^2$ into the maximal regions $\overline{R}_k(H;S)$. In other words, $\mathsf{CVD}_k(S)$ partitions $\mathbb{R}^2$ by k *nearest colors* under the minimal distances $\{d_i\}_{i \in K}$, while $\overline{\mathsf{CVD}}_k(S)$ partitions $\mathbb{R}^2$ by k *farthest colors* under the maximal distances $\{\bar{d}_i\}_{i \in K}$.

These two families of color Voronoi diagrams generalize various conventional counterparts. For $k = 1$, $\mathsf{CVD}_1(S)$ and $\overline{\mathsf{CVD}}_1(S)$ correspond to the ordinary nearest-site and farthest-site Voronoi diagrams of S, $\mathsf{VD}(S)$ and $\mathsf{FVD}(S)$, respectively, where adjacent faces of a common color are merged. See Figure 1(a) and (b).

- If $m = n$, that is, each site in S has a distinct color, then $\mathsf{CVD}_k(S) = \mathsf{VD}_k(S)$, the ordinary order-$k$ Voronoi diagram without colors [13, 29, 30, 40]. In this case, it holds that $\overline{\mathsf{CVD}}_k(S) = \mathsf{VD}_{n-k}(S)$, also known as the *order-k farthest-site Voronoi diagram* or the *order-k maximal Voronoi diagram* [5].
- The *farthest color Voronoi diagram* $\mathsf{FCVD}(S)$ [1, 9, 24, 32] partitions the plane $\mathbb{R}^2$ into regions of colors $i \in K$ that consist of all points $p \in \mathbb{R}^2$ whose *farthest* color is i with respect to the minimal distances $\{d_i\}_{i\in K}$. The *Hausdorff Voronoi diagram* $\mathsf{HVD}(S)$, also known as the *min-max* or *cluster* Voronoi diagram [22, 37, 38], similarly partitions $\mathbb{R}^2$ into regions of *nearest* colors with respect to the maximal distances $\{\bar{d}_i\}_{i\in K}$. We thus have $\mathsf{FCVD}(S) = \mathsf{CVD}_{m-1}(S)$ and $\mathsf{HVD}(S) = \overline{\mathsf{CVD}}_{m-1}(S)$. These two diagrams are often refined so that the region of each color $i \in K$ is subdivided into subregions of sites in S_i that determine the distance-to-color d_i or $\bar{d}_i$. We denote these refined versions by $\mathsf{FCVD}^*(S)$ and $\mathsf{HVD}^*(S)$, respectively. See Figure 1(c) and (d).

In this paper, we initiate the study of higher-order color Voronoi diagrams, $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$, with general distance functions δ_s for $s \in S$. Our main results are as follows:

(1) We prove an exact upper bound $4k(n-k) - 2n$ on the total number of vertices of both order-k diagrams $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ for each $1 \leqslant k \leqslant m-1$, when the sites $s \in S$ are points and $\delta_s(x)$ is given as the *convex distance* from $x \in \mathbb{R}^2$ to s based on any convex and compact subset of $\mathbb{R}^2$, which includes the L_p distance for any $1 \leqslant p \leqslant \infty$. This result is in fact a corollary of our more general result: we derive an exact equation under certain conditions on functions δ_s, which only requires relations on the numbers of vertices and unbounded edges in $\mathsf{VD}(S')$ and $\mathsf{FVD}(S')$ for $S' \subseteq S$. The bound $4k(n-k) - 2n$ can be realized, for example, when $m = n$, so it is tight and best possible.

(2) Under the L_1 or the L_∞ metric, we prove that $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ consist of at most $\min\{4k(n-k) - 2n, 4(n-k)^2\}$ and $\min\{4k(n-k) - 2n, 2k^2\}$ vertices, respectively. Similar bounds are derived for any convex distance function based on a convex polygon.

(3) We present an iterative algorithm that computes color Voronoi diagrams of order 1 to k in $O(k^2 n + n \log n)$ expected or $O(k^2 n \log n)$ worst-case time, provided that S and $\{\delta_s\}_{s\in S}$ satisfy the requirements of abstract Voronoi diagrams [28] and an additional condition (see Section 5), which includes colored points S under any smooth convex distance function. For points S in the Euclidean plane, it can be reduced to $O(k^2 n + n \log n)$ worst-case time.

Our combinatorial results generalize previously known bounds for the ordinary higher-order Voronoi diagrams $\mathsf{VD}_k(S)$ of uncolored sites S, which is a special case of $m = n$ in our setting. The asymptotically tight bound $O(k(n-k))$ on the complexity of $\mathsf{VD}_k(S)$ has been proved not only for points [29, 30] under the L_p metric but also for line segments [40] and even in the abstract setting [13]. Under the L_1 (or L_∞) metric, the complexity of $\mathsf{VD}_k(S)$ is known to be $O(\min\{k(n-k), (n-k)^2\})$ for a set S of points or line segments [30, 40].

In particular, the same exact number $4k(n-k) - 2n$ can be derived for the ordinary order-k diagram $\mathsf{VD}_k(S)$ and order-$(n-k)$ diagram $\mathsf{VD}_{n-k}(S)$ under the Euclidean metric from previous results [20, 21, 29]. In his book [21, Chapter 13], Edelsbrunner showed that the number of vertices of $\mathsf{VD}_k(S)$ is *exactly* $(4k-2)n - 2k^2 - e_k - 2\sum_{i=1}^{k-1} e_i$, if S is in general position, where e_k denotes the number of unbounded edges in $\mathsf{VD}_k(S)$ (or, equivalently, *k-sets* in S). One can easily verify that the total number of vertices in $\mathsf{VD}_k(S)$ and $\mathsf{VD}_{n-k}(S)$ is exactly $4k(n-k) - 2n$ by using the identities: $e_k = e_{n-k}$ and $\sum_{k=1}^{n-1} e_k = n(n-1)$. This can also be verified from the inductive approach of Lee [29]. As a recent result related to ours, Biswas et al. [12] derived exact relations for the complexity of 3D Euclidean higher-order Voronoi diagrams with Morse theory, extending the inductive argument by Lee.

Another remarkable special case of our results is when $k = m - 1$, which yields the $O(m(n - m + 1))$ bound for the farthest color Voronoi diagram $\mathsf{FCVD}(S) = \mathsf{CVD}_{m-1}(S)$ and the Hausdorff Voronoi diagram $\mathsf{HVD}(S) = \overline{\mathsf{CVD}}_{m-1}(S)$. The worst-case complexity of $\mathsf{FCVD}(S)$ and $\mathsf{HVD}(S)$ is known to be $\Theta(mn)$ or $\Theta(n^2)$, if S is a set of points or line segments under the Euclidean or L_1 metric [1, 9, 22, 24, 38]. While the upper bounds $O(mn)$ and $O(n^2)$ are shown to be tight by matching lower bound constructions [1, 22, 24], they become significantly loose when n is close to m; if $n = m$, we have $\mathsf{FCVD}(S) = \mathsf{FVD}(S)$ and $\mathsf{HVD}(S) = \mathsf{VD}(S)$, where both have linear complexity. Hence, our new results not only prove tight upper bounds simultaneously on both diagrams, but also formally reveal a smooth extension upon the previous knowledge about the ordinary Voronoi diagrams for any $m \leqslant n$. Prior to our results, it was known that the complexity of $\mathsf{FCVD}(S)$ and $\mathsf{HVD}(S)$ can range from $\Theta(n)$ to $\Theta(m(n - m + 1))$ expressed in terms of geometric parameters, called *straddles* for $\mathsf{FCVD}(S)$ [33] and *crossings* for $\mathsf{HVD}(S)$ [38]. Conditions under which these diagrams have linear complexity have also been discussed [8, 33, 38].

Our combinatorial results are based on a color-augmented extension of the Clarkson–Shor framework [20], so-called the *colorful Clarkson–Shor framework*, which has its own interest with various applications. Our notions of colored objects and configurations are naturally inherited from any set system that fits in the original Clarkson–Shor framework, and yield a systematic scheme to deal with objects that are collections of primitive elements. Our new framework provides a unifying approach to derive the complexity of higher-order color Voronoi diagrams under general distances δ_s, including the well-studied diagrams $\mathsf{FCVD}(S)$, $\mathsf{HVD}(S)$, and the ordinary higher-order diagram $\mathsf{VD}_k(S)$. In deriving our results, we make use of a close relation between the color diagrams $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ and the *colored* k-facets of S. An analogous relation for the Euclidean ordinary case (without colors) has been shown by Clarkson and Shor [20] and Edelsbrunner [21]. We also derive lower and upper bounds on the number of colored k-facets in $\mathbb{R}^2$. We also demonstrate more applications of the colorful Clarkson–Shor framework that result in several new bounds on levels of arrangements of piecewise linear or algebraic curves and surfaces; see [10].

Our algorithm follows the principle of iteratively computing all order-i Voronoi diagrams for $1 \leqslant i \leqslant k$, and compares to the $O(k^2 n \log n)$-time counterpart of Lee [29] for computing $\mathsf{VD}_1(S), \ldots, \mathsf{VD}_k(S)$ for points in the Euclidean metric. After a series of improvements [2, 3, 7, 16, 18, 36, 41], the first optimal $O(n \log n + kn)$-time algorithm that computes $\mathsf{VD}_k(S)$ for points S in the Euclidean plane was eventually presented in 2024 by Chan et al. [17]. Efficient algorithms of different approaches are also known for computing $\mathsf{VD}_k(S)$ of line segments S [40] or under the model of abstract Voronoi diagrams [14, 15]; and for computing $\mathsf{FCVD}^*(S)$ [1, 9, 24, 33] and $\mathsf{HVD}^*(S)$ [4, 22, 38].

Due to space limit, proofs are omitted but can be found in the full version [10].

2 Color Voronoi diagrams and arrangements

Let S be a set of n *sites*, which can be any abstract objects, and $\delta_s \colon \mathbb{R}^2 \to \mathbb{R}$ for $s \in S$ be a given continuous function. We assume that the functions δ_s are in *general position*, similarly to [26]. More precisely, let γ_s be the graph of δ_s, that is, the xy-monotone surface $\{(p, \delta_s(p)) \mid p \in \mathbb{R}^2\}$ in $\mathbb{R}^3$, and $\Gamma := \{\gamma_s \mid s \in S\}$. By the general position of functions δ_s, we mean the following: no more than three surfaces in Γ meet at a common point, no more than two surfaces in Γ meet at a one-dimensional curve, no two surfaces in Γ are tangent to each other, and none of the surfaces in Γ is tangent to the intersection curve of two others in Γ.

We denote the minimization diagram of Γ by $\mathsf{VD}(S)$, the *nearest-site Voronoi diagram* of S, and the maximization diagram of Γ by $\mathsf{FVD}(S)$, the *farthest-site Voronoi diagram* of S. It is well known that the ordinary (uncolored) higher-order Voronoi diagrams $\mathsf{VD}_k(S)$ of S are determined by levels of the arrangement $\mathcal{A}(\Gamma)$ of Γ as established in earlier work [5, 23]. In the following, we discuss an analogous relation between order-k color Voronoi diagrams and levels of certain surfaces in $\mathbb{R}^3$.

Let us assume that the sites in S are colored with m colors in $K = \{1, \ldots, m\}$. Let $\kappa\colon S \to K$ denote a color assignment such that the color of $s \in S$ is $\kappa(s) \in K$. Let $S_i := \{s \in S \mid \kappa(s) = i\}$ and $\Gamma_i := \{\gamma_s \mid s \in S_i\}$ for $i \in K$. Define E_i and $\overline{E}_i$ to be the lower and upper envelopes of surfaces in Γ_i, respectively, and consider the two arrangements $\mathcal{A}_\Gamma = \mathcal{A}(\{E_1, \ldots, E_m\})$ and $\overline{\mathcal{A}}_\Gamma = \mathcal{A}(\{\overline{E}_1, \ldots, \overline{E}_m\})$. Note that E_i is the graph of the minimal distance function d_i of color $i \in K$, and $\overline{E}_i$ is the graph of the maximal distance $\bar{d}_i$. We then consider the levels of the arrangements $\mathcal{A}_\Gamma$ and $\overline{\mathcal{A}}_\Gamma$. For $1 \leqslant k \leqslant m$, let L_k be the k-level of $\mathcal{A}_\Gamma$ *from below* and $\overline{L}_k$ be the k-level of $\overline{\mathcal{A}}_\Gamma$ *from above*. So, L_1 is the lower envelope of $E_1, \ldots, E_m$, and $\overline{L}_1$ is the upper envelope of $\overline{E}_1, \ldots, \overline{E}_m$. Thus, projecting L_1 and $\overline{L}_1$ down onto $\mathbb{R}^2$ yields $\mathsf{VD}(S)$ and $\mathsf{FVD}(S)$, respectively. On the other hand, L_m is the upper envelope of the lower envelopes $E_1, \ldots, E_m$, and $\overline{L}_m$ is the lower envelope of the upper envelopes $\overline{E}_1, \ldots, \overline{E}_m$. Projecting L_m and $\overline{L}_m$ down onto $\mathbb{R}^2$ yields the refined diagrams $\mathsf{FCVD}^*(S)$ and $\mathsf{HVD}^*(S)$, respectively [22, 24].

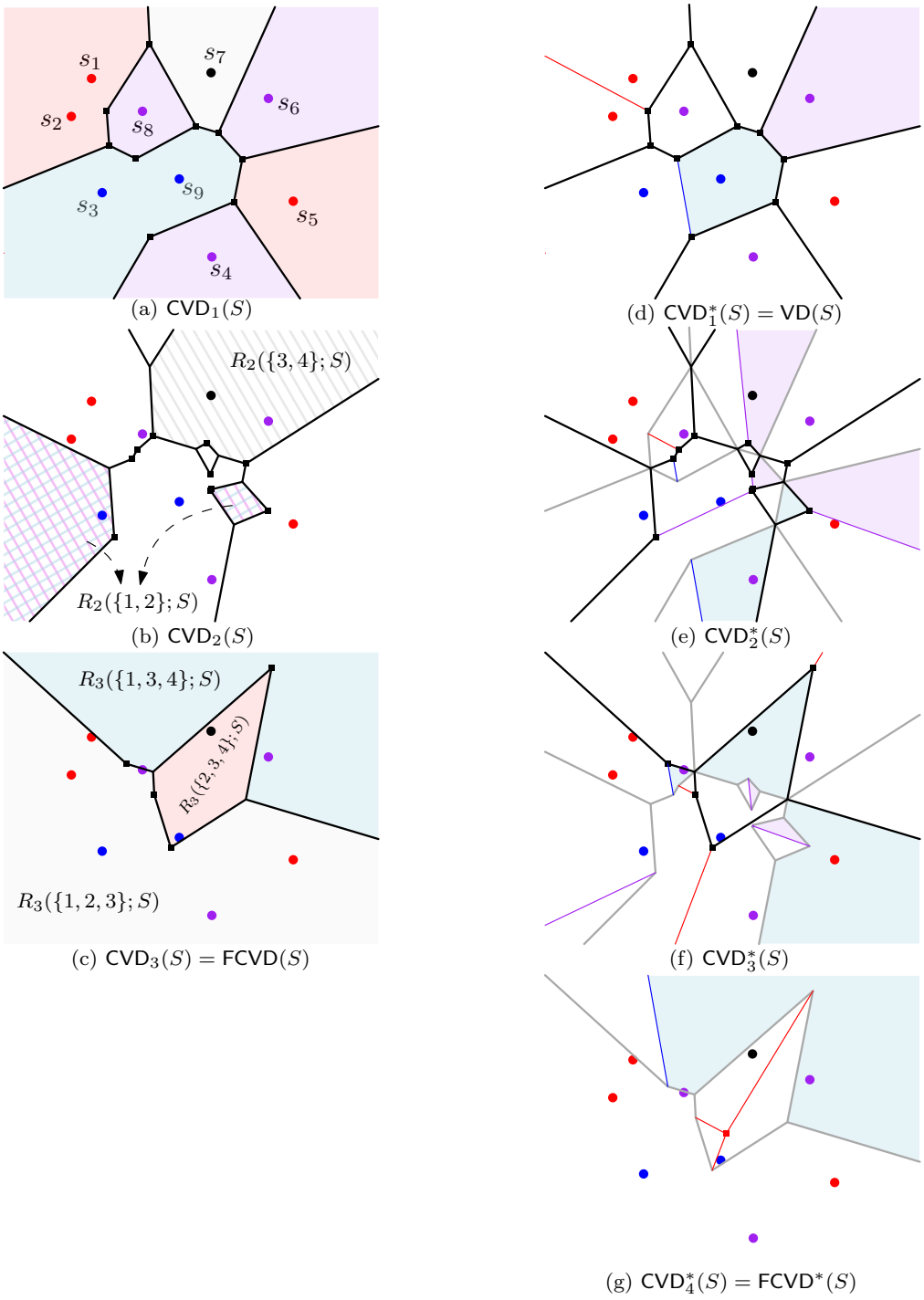
From the viewpoint of [5, 23], we observe the following.

- The order-k color Voronoi diagrams, $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$, for $1 \leqslant k \leqslant m-1$ are the projections of $L_k \cap L_{k+1}$ and of $\overline{L}_k \cap \overline{L}_{k+1}$, respectively, onto $\mathbb{R}^2$.
- For each $1 \leqslant k \leqslant m$, let $\mathsf{CVD}^*_k(S)$ denote the planar map obtained by projecting L_k down onto $\mathbb{R}^2$; analogously, let $\overline{\mathsf{CVD}}^*_k(S)$ be the map obtained by projecting $\overline{L}_k$ onto $\mathbb{R}^2$. By definition, $\mathsf{CVD}^*_k(S)$ and $\overline{\mathsf{CVD}}^*_k(S)$ refine $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$, respectively. Each face f of $\mathsf{CVD}^*_k(S)$ (or of $\overline{\mathsf{CVD}}^*_k(S)$) is *associated with* a site $s \in S_i$ such that, for any point $p \in f$, $d_i(p) = \delta_s(p)$ and $i \in K$ is the *k-th nearest color* from p with respect to $\{d_i\}_{i\in K}$ (resp. $\bar{d}_i(p) = \delta_s(p)$ and $i \in K$ is the *k-th farthest color* from p with respect to $\{\bar{d}_i\}_{i\in K}$). This way, the refined diagrams $\mathsf{CVD}^*_k(S)$ and $\overline{\mathsf{CVD}}^*_k(S)$ partition the plane $\mathbb{R}^2$ by the k-th nearest and k-th farthest colors, respectively.

From the construction, it is clear that $\mathsf{CVD}^*_1(S) = \mathsf{VD}(S)$ and $\overline{\mathsf{CVD}}^*_1(S) = \mathsf{FVD}(S)$. Note also that $\mathsf{CVD}^*_m(S) = \mathsf{FCVD}^*(S)$ and $\overline{\mathsf{CVD}}^*_m(S) = \mathsf{HVD}^*(S)$, whereas $\mathsf{FCVD}(S) = \mathsf{CVD}_{m-1}(S)$ and $\mathsf{HVD}(S) = \overline{\mathsf{CVD}}_{m-1}(S)$.

Figures 2 and 3 illustrate an example under the Euclidean metric, where S consists of $n = 9$ points and $m = 4$ colors: $S_1 = \{s_1, s_2, s_5\}$, $S_2 = \{s_3, s_9\}$, $S_3 = \{s_4, s_6, s_8\}$, and $S_4 = \{s_7\}$, in red, blue, purple, and black, respectively; selected regions of $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ are labeled in (a)–(c); faces associated with $s_6 \in S_3$ and those with $s_9 \in S_2$ in $\mathsf{CVD}^*_k(S)$ and $\overline{\mathsf{CVD}}^*_k(S)$ are shaded in purple and blue, respectively, in (d)–(g).

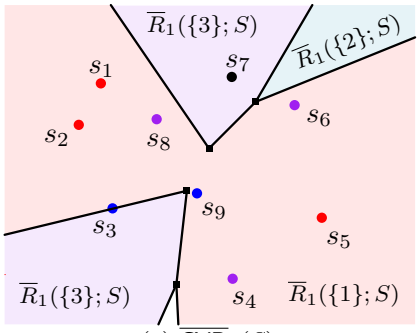
Now, consider the vertices and edges of the arrangements $\mathcal{A}_\Gamma$ and $\overline{\mathcal{A}}_\Gamma$. By the general position assumption, each vertex v of $\mathcal{A}_\Gamma$ or of $\overline{\mathcal{A}}_\Gamma$ is determined by exactly three sites $s, s', s'' \in S$ in such a way that v is a common intersection of three surfaces $\gamma_s, \gamma_{s'}, \gamma_{s''} \in \Gamma$. Such a vertex v is called *c-chromatic* for $c \in \{1, 2, 3\}$ if $|\{\kappa(s), \kappa(s'), \kappa(s'')\}| = c$. (Note that any 1-chromatic vertex is a vertex of some single envelope E_i or $\overline{E}_i$.) Similarly, each edge of $\mathcal{A}_\Gamma$ and of $\overline{\mathcal{A}}_\Gamma$ is determined by exactly two sites in S, and is either 1- or 2-chromatic according to the number of involved colors. We identify each vertex or edge of $\mathsf{CVD}^*_k(S)$ or of $\overline{\mathsf{CVD}}^*_k(S)$ by its original lifted copy in $\mathcal{A}_\Gamma$ or in $\overline{\mathcal{A}}_\Gamma$. Observe that any c-chromatic vertex or edge of $\mathcal{A}_\Gamma$ or of $\overline{\mathcal{A}}_\Gamma$ appears in c consecutive levels, as it lies in the intersection of



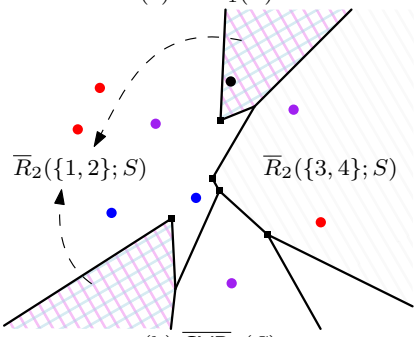
(a) $\mathsf{CVD}_1(S)$ (b) $\mathsf{CVD}_2(S)$ (c) $\mathsf{CVD}_3(S) = \mathsf{FCVD}(S)$

(d) $\mathsf{CVD}_1^*(S) = \mathsf{VD}(S)$ (e) $\mathsf{CVD}_2^*(S)$ (f) $\mathsf{CVD}_3^*(S)$ (g) $\mathsf{CVD}_4^*(S) = \mathsf{FCVD}^*(S)$

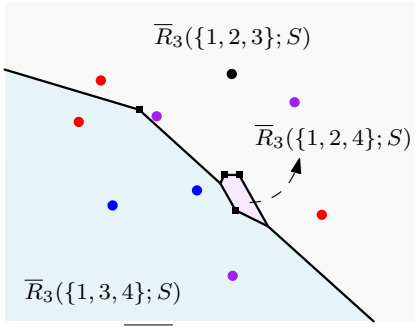
Figure 2 The minimal color Voronoi diagrams $\mathsf{CVD}_k(S)$ and the refined diagrams $\mathsf{CVD}_k^*(S)$.



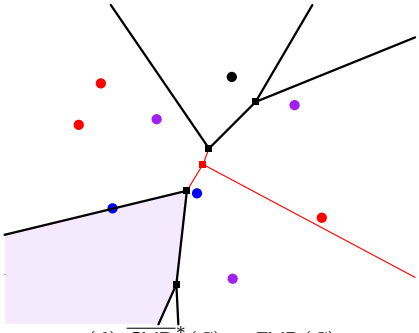
(a) $\overline{\mathsf{CVD}}_1(S)$



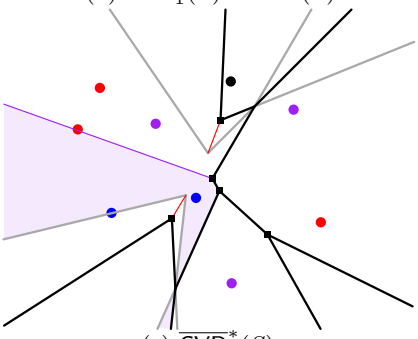
(b) $\overline{\mathsf{CVD}}_2(S)$



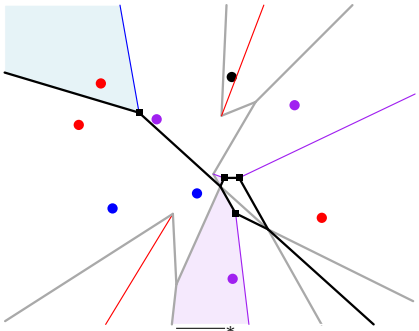
(c) $\overline{\mathsf{CVD}}_3(S) = \mathsf{HVD}(S)$



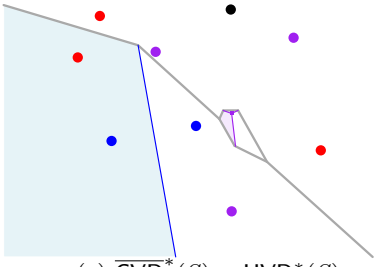
(d) $\overline{\mathsf{CVD}}_1^*(S) = \mathsf{FVD}(S)$



(e) $\overline{\mathsf{CVD}}_2^*(S)$



(f) $\overline{\mathsf{CVD}}_3^*(S)$



(g) $\overline{\mathsf{CVD}}_4^*(S) = \mathsf{HVD}^*(S)$

Figure 3 The maximal color diagrams $\overline{\mathsf{CVD}}_k(S)$ and the refined diagrams $\overline{\mathsf{CVD}}_k^*(S)$.

c surfaces from $\{E_i\}_{i\in K}$ or from $\{\overline{E}_i\}_{i\in K}$, respectively. Thus, c-chromatic vertices appear in c consecutive orders of the refined diagrams. We call a vertex or an edge of $\mathsf{CVD}^*_k(S)$ or of $\overline{\mathsf{CVD}}^*_k(S)$ *new* if it does not appear in $\mathsf{CVD}^*_{k-1}(S)$ or in $\overline{\mathsf{CVD}}^*_{k-1}(S)$, respectively; and *old*, otherwise. By definition, the vertices and edges of $\mathsf{CVD}^*_1(S)$ and $\overline{\mathsf{CVD}}^*_1(S)$ are all new. Note that every edge of $\mathsf{CVD}_k(S)$ (or, of $\overline{\mathsf{CVD}}_k(S)$) is 2-chromatic and new, and appears both in $\mathsf{CVD}^*_k(S)$ and $\mathsf{CVD}^*_{k+1}(S)$ (in $\overline{\mathsf{CVD}}^*_k(S)$ and $\overline{\mathsf{CVD}}^*_{k+1}(S)$, resp.), being first new and then old. See Figures 2 and 3, where new 2-chromatic edges are in black, old 2-chromatic edges in gray, 1-chromatic edges in their own color, and new vertices marked by small squares.

We define $v_{c,j} = v_{c,j}(S)$ for $1 \leqslant c \leqslant 3$ and $0 \leqslant j \leqslant m-1$ to be the number of c-chromatic vertices in $\mathcal{A}_\Gamma$ below which there are exactly j surfaces from $\{E_i\}_{i\in K}$; $\bar{v}_{c,j} = \bar{v}_{c,j}(S)$ to be the number of c-chromatic vertices in $\overline{\mathcal{A}}_\Gamma$ above which there are exactly j surfaces from $\{\overline{E}_i\}_{i\in K}$.

▶ **Lemma 1.** *For any $1 \leqslant c \leqslant 3$ and $1 \leqslant k \leqslant m$, the following hold:*

(i) *The number of new c-chromatic vertices of $\mathsf{CVD}^*_k(S)$ is exactly $v_{c,k-1}$, and the number of new c-chromatic vertices of $\overline{\mathsf{CVD}}^*_k(S)$ is exactly $\bar{v}_{c,k-1}$.*

(ii) *The number of vertices of $\mathsf{CVD}^*_k(S)$ is exactly $v_{3,k-1}+v_{3,k-2}+v_{3,k-3}+v_{2,k-1}+v_{2,k-2}+v_{1,k-1}$, and the number of vertices of $\overline{\mathsf{CVD}}^*_k(S)$ is exactly $\bar{v}_{3,k-1}+\bar{v}_{3,k-2}+\bar{v}_{3,k-3}+\bar{v}_{2,k-1}+\bar{v}_{2,k-2}+\bar{v}_{1,k-1}$, where $v_{c,j} = \bar{v}_{c,j} = 0$ for $j < 0$.*

(iii) *The number of vertices of $\mathsf{CVD}_k(S)$ is exactly $v_{3,k-1}+v_{3,k-2}+v_{2,k-1}$, and the number of vertices of $\overline{\mathsf{CVD}}_k(S)$ is exactly $\bar{v}_{3,k-1}+\bar{v}_{3,k-2}+\bar{v}_{2,k-1}$.*

3 The colorful Clarkson–Shor framework

The Clarkson–Shor technique [20] is based on a general framework dealing with so-called configurations or ranges defined by a set of objects. Specifically, the following three ingredients are supposed to be given with a constant integer parameter $d \geqslant 1$, see also Sharir [42]:

- A set S of n *objects*.
- A set $\mathcal{F}(S)$ of *configurations*, each of which is defined by exactly d objects in S.
- A *conflict relation* $\chi \subseteq S \times \mathcal{F}(S)$ between objects $s \in S$ and configurations $f \in \mathcal{F}(S)$ with the requirement that none of the d objects defining f are in conflict with f.

In the original framework, the number of objects that define a configuration does not have to be exactly d, but at most d. This restriction can be achieved by adding dummy objects to S.

Let us call such a triplet $(S, \mathcal{F}(S), \chi)$ a *CS-structure*. Given any CS-structure $(S, \mathcal{F}(S), \chi)$ with parameter d, we now impose a *color* assignment $\kappa\colon S \to K$ to the objects in S, where $K = \{1, 2, \ldots, m\}$ denotes the set of m colors with $m \leqslant n$. For each color $i \in K$, let $S_i := \{s \in S \mid \kappa(s) = i\}$. For $f \in \mathcal{F}(S)$ and set $D_f \subseteq S$ of d objects defining f, $\kappa(D_f) = \{\kappa(s) \mid s \in D_f\}$ is called a set of colors defining f. We build a *color-to-configuration conflict relation* $\chi_\kappa \subseteq K \times \mathcal{F}(S)$ such that a color $i \in K$ is in conflict with a configuration $f \in \mathcal{F}(S)$ if an object $s \in S_i$ is in conflict with f, that is, $(i, f) \in \chi_\kappa$ if and only if $(s, f) \in \chi$ for some $s \in S_i$. We are then interested in those configurations $f \in \mathcal{F}(S)$ such that none of its defining colors in $\kappa(D_f)$ are in conflict with f. Let $\mathcal{F}(S, \kappa) \subseteq \mathcal{F}(S)$ be the set of these configurations, called *colored configurations with respect to κ*. We call $f \in \mathcal{F}(S, \kappa)$ *c-chromatic* if $|\kappa(D_f)| = c$ for $1 \leqslant c \leqslant d$, and let the *weight* of f be the number of colors in K that are in conflict with f. Let $\mathcal{F}_{c,j}(S, \kappa) \subseteq \mathcal{F}(S, \kappa)$ be the set of c-chromatic weight-j colored configurations in $\mathcal{F}(S, \kappa)$.

Considering the colors in K as new *objects*, each of which is a collection of objects in S, observe that this color-augmented structure $(K, \bigcup_j \mathcal{F}_{c,j}(S, \kappa), \chi_\kappa)$ for each $1 \leqslant c \leqslant d$ is again a CS-structure with parameter c. Therefore, the main lemma of Clarkson and Shor [20, Lemma 2.1] automatically implies the following.

▶ **Lemma 2.** *With the above notations, let $1 \leqslant c \leqslant d$, $r \geqslant 0$, and $a \geqslant 0$ be integers, and $R \subseteq K$ be a random subset of r colors. Then,*

$$\binom{m}{r} \mathbf{E}[|\mathcal{F}_{c,a}(S_R, \kappa_R)|] \geqslant \sum_{j=0}^{m-c} |\mathcal{F}_{c,j}(S,\kappa)| \binom{j}{a} \binom{m-c-j}{r-c-a},$$

where $S_R = \bigcup_{i\in R} S_i$ and $\kappa_R \colon S_R \to R$ denotes the restriction of κ to S_R. The equality holds if each configuration in $\mathcal{F}(S,\kappa)$ is defined by a unique set of d objects in S.

In probabilistic arguments dealing with CS-structures, it is usually necessary to have an upper bound on the number of weight-0 configurations. For any subset $S' \subseteq S$, let $\mathcal{F}_0(S') \subseteq \mathcal{F}(S')$ be the set of (uncolored) configurations f of weight 0, that is, $(s,f) \notin \chi$ for all $s \in S'$. Let $T_0(n')$ be any nondecreasing function with $T_0(0) = 0$ that upper bounds the number $|\mathcal{F}_0(S')|$ of these configurations for any set S' of n' uncolored objects. The following is obvious by definition.

▶ **Lemma 3.** *Let $S' \subseteq S$ be any subset and κ' be any color assignment for S'. Then,*

$$\sum_{c=1}^{d} |\mathcal{F}_{c,0}(S',\kappa')| = |\mathcal{F}_0(S')| \leqslant T_0(|S'|).$$

In many applications, such an upper bound function T_0 is either known from previous work or relatively easy to obtain. Once we have T_0, we can derive general upper bounds on the number of corresponding colored configurations of weight at most k in such a procedural way as done for uncolored cases [20, 34, 43].

▶ **Theorem 4.** *With the above notations, suppose T_0 is a convex function. For each $1 \leqslant c \leqslant d$, the total number of $(\leqslant c)$-chromatic colored configurations is bounded by*

$$\sum_{b=1}^{c} \sum_{j=0}^{m-b} \binom{m-b-j}{c-b} |\mathcal{F}_{b,j}(S,\kappa)| \leqslant \binom{m}{c} \cdot \frac{1}{m} \sum_{i\in K} T_0(c|S_i|) = O(m^{c-1} \cdot T_0(cn)).$$

Also, for each $2 \leqslant c \leqslant d$ and $0 \leqslant k \leqslant \lfloor \frac{m}{c} \rfloor - 1$, it holds that

$$\sum_{j=0}^{k} |\mathcal{F}_{c,j}(S,\kappa)| = O\left(\frac{(k+1)^c}{m} \cdot \sum_{i\in K} T_0\left(\frac{m|S_i|}{k+1}\right)\right) = O\left(\frac{(k+1)^c}{m} \cdot T_0\left(\frac{mn}{k+1}\right)\right).$$

Remark that the left-hand side of the first bound can be seen as a "weighted" count of $(\leqslant c)$-chromatic colored configurations, and that the second bound in Theorem 4 for $n = m$ implies Clarkson and Shor's original bound, $O((k+1)^c \cdot T_0(n/(k+1)))$, for uncolored cases [20, Theorem 3.1]. Note that Theorem 4 still implies the same Clarkson–Shor bound if T_0 is linear. Moreover, if the colors are assigned in a favorably uniform way, we can derive similar bounds as well without assuming the convexity of T_0.

▶ **Theorem 5.** *With the above notations, suppose $|S_i| \leqslant \rho \cdot \frac{n}{m}$ for every $i \in K$ for some $\rho \geqslant 1$. Then, for $1 \leqslant c \leqslant d$,*

$$\sum_{b=1}^{c} \sum_{j=0}^{m-b} \binom{m-b-j}{c-b} |\mathcal{F}_{b,j}(S,\kappa)| \leqslant \binom{m}{c} T_0\left(c\rho \cdot \frac{n}{m}\right),$$

and for $2 \leqslant c \leqslant d$ and $0 \leqslant k \leqslant \lfloor \frac{m}{c} \rfloor - 1$,

$$\sum_{j=0}^{k} |\mathcal{F}_{c,j}(S,\kappa)| = O\left((k+1)^c \cdot T_0\left(\rho \cdot \frac{n}{k+1}\right)\right).$$

3.1 Colored j-facets

Let S be a set of n points in $\mathbb{R}^d$. A *j-facet* in S is an oriented $(d-1)$-simplex σ with its vertices chosen from S such that the open half-space on its positive side σ^+ contains exactly j points of S. Among the first applications of the original Clarkon–Shor framework was the tight upper bound $O(k^{\lceil d/2\rceil}n^{\lfloor d/2\rfloor})$ on the number of $(\leqslant k)$-facets [20], implying the same asymptotic bound on the $(\leqslant k)$-level in the arrangement of hyperplanes via the point-to-hyperplane duality [21]. Many variants of j-facets have been discussed in the literature; we refer to a survey article by Wagner [44].

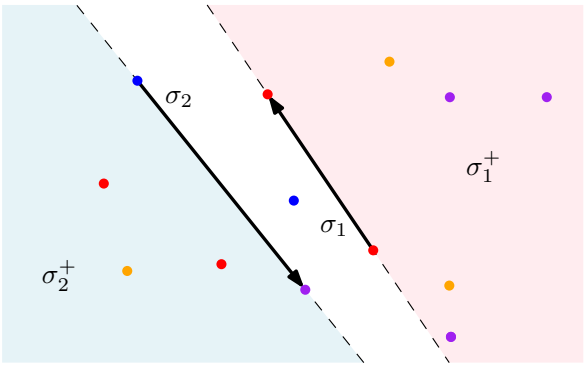


Figure 4 Colored j-facets in colored points in $\mathbb{R}^2$: a 1-chromatic 2-facet σ_1 and a 2-chromatic 2-facet σ_2 are shown, which choose half-planes σ_1^+ and σ_2^+ on their right side.

Now, we assume that the points in S are colored by a color assignment κ with m colors K. For any subset $A \subset \mathbb{R}^d$, we shall say that A *intersects* a color $i \in K$, if A contains some $s \in S_i$. According to our notion of colored configurations, a *colored j-facet* σ in S with respect to κ is an oriented simplex defined by d points $D_\sigma \subseteq S$ such that exactly j colors, but none of the defining colors in $\kappa(D_\sigma)$, are intersected by σ^+. See Figure 4. Notice that colored j-facets correspond to vertices of the arrangement of m lower/upper envelopes of hyperplanes dual to S_i for $i \in K$; see the full version [10] for a more detailed discussion.

For $1 \leqslant c \leqslant d$ and $j \geqslant 0$, let $e_{c,j}(S)$ be the number of c-chromatic j-facets in S and $e_j(S) := \sum_c e_{c,j}(S)$ be the number of all j-facets. Katoh and Tokuyama [27, Proposition 15] proved that $e_k(S) = O(k^{1/3}n)$ in $\mathbb{R}^2$ and $e_k(S) = O(k^{2/3}n^2)$ in $\mathbb{R}^3$ based on a generalized Lovász's Lemma. Theorems 4 and 5 directly imply the following bounds.

▶ **Corollary 6.** *For a set S of n colored points in $\mathbb{R}^d$ with m colors and any $0 \leqslant k \leqslant \lfloor \frac{m}{d} \rfloor - 1$, the number of $(\leqslant k)$-facets in S is $\sum_{j=0}^{k} e_j(S) = O(m^{\lfloor d/2\rfloor - 1}k^{\lceil d/2\rceil}n^{\lfloor d/2\rfloor})$. If there is a constant $\rho \geqslant 1$ such that $|S_i| \leqslant \rho \cdot \frac{n}{m}$ for every $i \in K$, then the bound is improved to $\sum_{j=0}^{k} e_j(S) = O(k^{\lceil d/2\rceil}n^{\lfloor d/2\rfloor})$.*

Note that for large k with $k \geqslant \lfloor \frac{m}{d} \rfloor$, the above bound on the number of $(\leqslant k)$-facets is asymptotically the same as the total number of configurations (see Theorems 4 and 5).

We continue our discussion for the case of $d = 3$. Lemma 3 implies that $e_0(S)$ counts the number of facets of the convex hull of S in $\mathbb{R}^3$. Using this fact, we observe the following.

▶ **Lemma 7.** *Let $2 \leqslant r \leqslant m$ and $R \subseteq K$ be a random set of r colors. It holds that*

$$\binom{m}{r}\mathbf{E}[e_0(S_R)] \leqslant 2\binom{m-1}{r-1}n - 4\binom{m}{r}$$

with equality when the points in S are in convex and general position.

Now, suppose that S is in convex and general position. Then, Lemmas 2 and 7 provide two different ways of exactly counting $\binom{m}{r}\mathbf{E}[e_0(S_R)]$ for $2 \leqslant r \leqslant m$, resulting in:

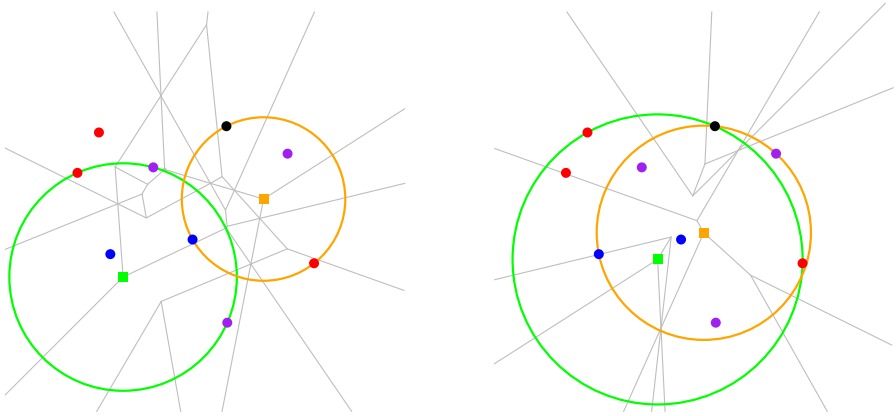
▶ **Theorem 8.** *Let $S \subset \mathbb{R}^3$ be a set of n points in convex and general position, each of which is colored by one of m colors. Then, it holds that for each $0 \leqslant j \leqslant m-2$*

$$e_{3,j}(S) + \sum_{i=0}^{j} e_{2,i}(S) + \sum_{i=0}^{j}(j-i+1)e_{1,i}(S) = 2(j+1)(n-j-2).$$

Note that Theorem 8 reveals an exact equation on $e_j(S) = e_{3,j}(S) + e_{2,j}(S) + e_{1,j}(S)$ for each $0 \leqslant j \leqslant m-2$. If $m = n$, that is, $|S_i| = 1$ for all $i \in K$, we have $e_{1,j}(S) = e_{2,j}(S) = 0$ for every j, so the equality $e_j(S) = 2(j+1)(n-j-2)$ holds. This exact number for the case of $m = n$ was proved earlier by Clarkson and Shor [20, Theorem 3.5].

3.2 Euclidean color Voronoi diagrams

Suppose that S consists of n points in general position in $\mathbb{R}^2$, with a given color assignment $\kappa\colon S \to K = \{1, \ldots, m\}$, and $\delta_s(x) = \|x - s\|_2$ is the Euclidean distance for each $s \in S$ and any $x \in \mathbb{R}^2$. Consider $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ for $1 \leqslant k \leqslant m-1$ in this setting.



(a) $\mathsf{CVD}_2^*(S)$ (b) $\overline{\mathsf{CVD}}_2^*(S)$

Figure 5 Selected new vertices (small squares) in $\mathsf{CVD}_2^*(S)$ and $\overline{\mathsf{CVD}}_2^*(S)$ and their corresponding circles. Green vertices are 2-chromatic, while orange ones are 3-chromatic. Observe that exactly one color is intersected by the interior $\hat{C}$ of each circle C in (a) and the exterior $\overline{C}$ of each circle C in (b).

We consider all circles through any three points in S with no regards of colors and let $\mathcal{F}(S)$ and $\overline{\mathcal{F}}(S)$ be the sets of the interiors and exteriors, respectively, of these circles. Also, define two conflict relations $\chi \subseteq S \times \mathcal{F}(S)$ and $\bar{\chi} \subseteq S \times \overline{\mathcal{F}}(S)$ to be the inclusion relation. We then consider colored configurations $\mathcal{F}(S,\kappa)$ and $\overline{\mathcal{F}}(S,\kappa)$ with respect to the given color assignment κ. Observe that each colored configuration of weight j in $\mathcal{F}(S,\kappa)$ or in $\overline{\mathcal{F}}(S,\kappa)$ corresponds to a new vertex of $\mathsf{CVD}_{j+1}^*(S)$ or of $\overline{\mathsf{CVD}}_{j+1}^*(S)$, respectively, by Lemma 1 and the discussions in Section 2, see Figure 5 illustrating the case of $j = 1$. Therefore, for each $1 \leqslant c \leqslant 3$ and $0 \leqslant j \leqslant m-c$, we have $v_{c,j}(S) = |\mathcal{F}_{c,j}(S,\kappa)|$ and $\bar{v}_{c,j}(S) = |\overline{\mathcal{F}}_{c,j}(S,\kappa)|$.

Now, consider the well-known lifting that maps points $p = (x,y)$ in $\mathbb{R}^2$ onto the unit paraboloid $U = \{z = x^2+y^2\}$ in $\mathbb{R}^3$: $p = (x,y) \mapsto p^\cup = (x,y,x^2+y^2)$. Let $S^\cup = \{s^\cup \mid s \in S\}$ be the set of lifted colored points in $\mathbb{R}^3$. (The horizontal plane $\{z = 0\}$ is identified as the original plane $\mathbb{R}^2$.) Consider colored j-facets in $S^\cup$ as in the previous section, and recall that $e_{c,j}(S^\cup)$ denotes the number of c-chromatic j-facets in $S^\cup$. We then observe the following.

▶ **Lemma 9.** *For $1 \leqslant c \leqslant 3$ and $0 \leqslant j \leqslant m - c$, we have $v_{c,j}(S) + \bar{v}_{c,j}(S) = e_{c,j}(S^{\cup})$.*

Since $S^{\cup}$ is in convex and general position, Theorem 8, together with Lemmas 1 and 9, implies an exact equation on the number of vertices in $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$.

▶ **Theorem 10.** *Let S be a set of n points with m colors in general position in the Euclidean plane, and $1 \leqslant k \leqslant m-1$. The total number of vertices in $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ is exactly*

$$4k(n-k) - 2n - 2\sum_{i=0}^{k-2} e_{2,i}(S^{\cup}) - \sum_{i=0}^{k-1}(2k-2i-1)e_{1,i}(S^{\cup}).$$

Theorem 10 implies the $O(k(n-k))$ bound on the complexity of $\mathsf{CVD}^*_k(S)$ and $\overline{\mathsf{CVD}}^*_k(S)$ by Lemma 1. An interesting special case of the above result is the following.

▶ **Corollary 11.** *Given a set S of n colored points with m colors in the Euclidean plane, the complexity of both $\mathsf{FCVD}^*(S)$ and $\mathsf{HVD}^*(S)$ is bounded by $O(m(n-m+1))$.*

4 Color Voronoi diagrams under general distance functions

We extend our results for the Euclidean case to general distance functions. We continue the discussions from Section 2, so S is a set of n sites, colored with m colors from K by a color assignment κ, and the functions $\delta_s \colon \mathbb{R}^2 \to \mathbb{R}$ for $s \in S$ are in general position.

Notice that the vertices of the arrangements $\mathcal{A}_\Gamma$ and $\overline{\mathcal{A}}_\Gamma$ form two set systems that fit in our colorful framework. More precisely, let $\mathcal{F}(S)$ be the set of vertices of the arrangement of n surfaces in Γ, and consider two conflict relations $\chi, \bar{\chi} \subseteq S \times \mathcal{F}(S)$ such that $(s,v) \in \chi$ if $v \in \mathcal{F}(S)$ lies above surface $\gamma_s \in \Gamma$ and $(s,v) \in \bar{\chi}$ if v lies below γ_s. Based on two CS-structures $(S, \mathcal{F}(S), \chi)$ and $(S, \mathcal{F}(S), \bar{\chi})$, we consider their colored configurations with respect to κ, denoted by $\mathcal{F}(S,\kappa)$ and $\overline{\mathcal{F}}(S,\kappa)$, respectively. By this construction, we have $v_{c,j}(S) = |\mathcal{F}_{c,j}(S,\kappa)|$ and $\bar{v}_{c,j}(S) = |\overline{\mathcal{F}}_{c,j}(S,\kappa)|$, counting c-chromatic weight-j colored configurations in $\mathcal{F}(S,\kappa)$ and in $\overline{\mathcal{F}}(S,\kappa)$, respectively, and, simultaneously, counting new c-chromatic vertices in $\mathsf{CVD}^*_{j+1}(S)$ and in $\overline{\mathsf{CVD}}^*_{j+1}(S)$ by Lemma 1.

For each $S' \subseteq S$, let $v_0(S')$ and $u_0(S')$ denote the numbers of vertices and unbounded edges[1], respectively, in $\mathsf{VD}(S')$; let $\bar{v}_0(S')$ and $\bar{u}_0(S')$ denote the numbers of vertices and unbounded edges, respectively, in $\mathsf{FVD}(S')$. We consider the following conditions.

V1 $v_0(S') = 2|S'| - 2 - u_0(S')$ for any $S' \subseteq S$.
V2 $\bar{v}_0(S') = \bar{u}_0(S') - 2$ for any $S' \subseteq S$.

Note that if every region in $\mathsf{VD}(S')$ is nonempty and simply connected, then Euler's formula and the general position assumption imply condition **V1**. If $\mathsf{FVD}(S')$ forms a tree, then every face of $\mathsf{FVD}(S')$ is unbounded and thus condition **V2** holds by Euler's formula.

In addition, for $c \in \{1,2\}$ and $j \geqslant 0$, let $u_{c,j} = u_{c,j}(S)$ be the number of c-chromatic unbounded edges in $\mathcal{A}_\Gamma$ that lie *above* exactly j surfaces in $\{E_i\}_{i\in K}$, and $\bar{u}_{c,j} = \bar{u}_{c,j}(S)$ be the number of c-chromatic unbounded edges in $\overline{\mathcal{A}}_\Gamma$ that lie *below* exactly j surfaces in $\{\overline{E}_i\}_{i\in K}$. From the discussion in Section 2, observe that $u_{c,j}$ and $\bar{u}_{c,j}$ are equal to the numbers of new c-chromatic unbounded edges in $\mathsf{CVD}^*_{j+1}(S)$ and in $\overline{\mathsf{CVD}}^*_{j+1}(S)$, respectively. Further, as for $v_{c,j}$ and $\bar{v}_{c,j}$, note that $u_{c,j}$ and $\bar{u}_{c,j}$ indeed count c-chromatic weight-j colored configurations based on two CS-structures for unbounded edges in the arrangement of n surfaces in Γ. Hence, assuming **V1** and **V2**, Lemma 3 implies: for any subset $R \subseteq K$,

[1] Hereafter, by counting unbounded edges, we mean counting vertices at infinity. So, if an unbounded edge separates the plane $\mathbb{R}^2$, then it is counted twice.

$$\sum_{c=1}^{3} v_{c,0}(S_R) = 2|S_R| - 2 - \sum_{c=1}^{2} u_{c,0}(S_R) \quad \text{and} \quad \sum_{c=1}^{3} \bar{v}_{c,0}(S_R) = \sum_{c=1}^{2} \bar{u}_{c,0}(S_R) - 2,$$

since $\mathsf{CVD}_1^*(S_R) = \mathsf{VD}(S_R)$ and $\overline{\mathsf{CVD}}_1^*(S_R) = \mathsf{FVD}(S_R)$. Combining these equations and the others obtained by Lemma 2, we have two systems of linear equations that can be solved in a similar way as done in Theorem 8. For $0 \leqslant j \leqslant m-1$, define

$$V_j := v_{3,j} + \sum_{i=0}^{j}(v_{2,i} + (j-i+1)v_{1,i}), \qquad U_j := \sum_{i=0}^{j}(u_{2,i} + (j-i+1)u_{1,i}),$$

$$\overline{V}_j := \bar{v}_{3,j} + \sum_{i=0}^{j}(\bar{v}_{2,i} + (j-i+1)\bar{v}_{1,i}), \quad \text{and} \quad \overline{U}_j := \sum_{i=0}^{j}(\bar{u}_{2,i} + (j-i+1)\bar{u}_{1,i}).$$

▶ **Lemma 12.** *With the above notations, let $0 \leqslant j \leqslant m-2$. Condition **V1** implies $V_j + U_j = (j+1)(2n-j-2)$; condition **V2** implies $\overline{V}_j - \overline{U}_j = -(j+1)(j+2)$.*

▶ **Theorem 13.** *Given S and $\{\delta_s\}_{s\in S}$ in general position as above, let $1 \leqslant k \leqslant m-1$ be an integer. If condition **V1** is true, then the number of vertices in $\mathsf{CVD}_k(S)$ is exactly*

$$2k(2n-k) - 2n - 2\sum_{i=0}^{k-2} v_{2,i}(S) - \sum_{i=0}^{k-1}(2k-2i-1)v_{1,i}(S) - U_{k-1} - U_{k-2};$$

*if condition **V2** is true, the number of vertices in $\overline{\mathsf{CVD}}_k(S)$ is exactly*

$$\overline{U}_{k-1} + \overline{U}_{k-2} - 2k^2 - 2\sum_{i=0}^{k-2} \bar{v}_{2,i}(S) - \sum_{i=0}^{k-1}(2k-2i-1)\bar{v}_{1,i}(S).$$

Remark that condition **V1** already implies the asymptotic complexity $O(k(n-k))$ of $\mathsf{CVD}_k(S)$ for $k \leqslant \frac{n}{2}$ as $2n-k \leqslant 3(n-k)$, while we have $O(kn)$ for $k > \frac{n}{2}$. Further, to show the $O(k(n-k))$ bound for any value of k for both $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$, it suffices to show that $U_j \geqslant (j+1)(j+2) - o(j^2)$ and $\overline{U}_j \leqslant (j+1)(2n-j-2) + o(j^2)$. In this way, Theorem 13 reduces the problem of bounding the complexity of higher-order color Voronoi diagrams to that of bounding the number of their unbounded edges. Also, note that Lemma 12 implies $U_j \leqslant (j+1)(2n-j-2)$ and $\overline{U}_j \geqslant (j+1)(j+2)$, if conditions **V1** and **V2** hold.

Remark also that if $\mathsf{VD}(S')$ and $\mathsf{FVD}(S')$ fall under the umbrella of abstract Voronoi diagrams, then conditions **V1** and **V2** hold [28,35], so Theorem 13 implies:

▶ **Corollary 14.** *Suppose that S and $\{\delta_s\}_{s\in S}$ imply a bisector system that satisfies the conditions of abstract Voronoi diagrams [28]. Then, the complexity of $\mathsf{CVD}_k(S)$ and $\mathsf{CVD}_k^*(S)$ is $O(k(n-k))$ for $1 \leqslant k \leqslant \lfloor \frac{n}{2} \rfloor$ and $O(kn)$ for $\lfloor \frac{n}{2} \rfloor + 1 \leqslant k \leqslant m$.*

The quantities U_j and $\overline{U}_j$, related to the number of unbounded edges, often turn out to be equal; the very typical example is the Euclidean case for point sites S, where the equality $u_{c,j}(S) = \bar{u}_{c,j}(S) = e_{c,j}(S)$ holds. This inspires us to consider the following third condition:

V3 $u_0(S') = \bar{u}_0(S')$ for any $S' \subseteq S$.

▶ **Lemma 15.** *Condition **V3** implies $U_j = \overline{U}_j$ for any $0 \leqslant j \leqslant m-1$.*

Assuming conditions **V1**–**V3**, we obtain the same exact number as in Theorem 10.

▶ **Theorem 16.** *Given S and $\{\delta_s\}_{s\in S}$ in general position as above, if conditions **V1**–**V3** hold, then the total number of vertices in $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ for $1 \leqslant k \leqslant m-1$ is exactly*

$$4k(n-k) - 2n - 2\sum_{i=0}^{k-2}(v_{2,i}(S) + \bar{v}_{2,i}(S)) - \sum_{i=0}^{k-1}(2k-2i-1)(v_{1,i}(S) + \bar{v}_{1,i}(S)).$$

Below, we discuss some specific cases of functions δ_s for a set S of points in the plane $\mathbb{R}^2$, in which new results are derived by applying Theorems 13 and 16.

Convex distance functions. From now on, suppose S consists of n colored points in $\mathbb{R}^2$. Let $B \subset \mathbb{R}^2$ be any convex and compact body whose interior contains the origin. Define $\delta_s(x) = \|x-s\|_B$ for point $s \in S$ to be the *convex distance* from $x \in \mathbb{R}^2$ to s based on B [19,31]. Since Voronoi diagrams of point sites under a convex distance function fall under the model of abstract Voronoi diagrams [28,35], conditions **V1** and **V2** hold.

Condition **V3**, however, is not guaranteed in general; a popular example is the L_1 or L_∞ metric, under which $\mathsf{VD}(S')$ may have $\Theta(|S'|)$ parallel unbounded edges while $\mathsf{FVD}(S')$ has at most four. In the following, we first assume that B is *smooth* that is, there is a unique line tangent to B at each point on its boundary [31]. We then make the following observation for a smooth B, stronger than condition **V3**, which has been known for the Euclidean metric even in higher dimensions [11].

▶ **Lemma 17.** *Given S and δ_s for $s \in S$ as above, suppose B is smooth. For $c \in \{1,2\}$ and $0 \leqslant j \leqslant m-1$, we have $u_{c,j}(S) = \bar{u}_{c,j}(S) = e_{c,j}(S)$, the number of c-chromatic j-facets in S.*

By Lemma 17, Theorem 16 implies the same upper bound $4k(n-k)-2n$ on the total number of vertices in both $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ under any smooth convex distance function.

We then relax the smoothness of B by a limit argument, so let B be any convex and compact body. Consider a sequence of smooth and convex bodies $B_0, B_1, \ldots$ that converges to B. Obviously, there exists $\hat{B} = B_i$ sufficiently close to B so that: (a) the functions $\hat{\delta}_s(x) = \|x-s\|_{\hat{B}}$ for $s \in S$ are in general position (see Section 2), and (b) for any scaled and translated copy B_{pqr} of B having three points $p, q, r \in S$ on its boundary, there is also a scaled and translated copy $\hat{B}_{pqr}$ of $\hat{B}$, whose boundary goes through p, q, r and the separation of S by $\hat{B}_{pqr}$ is the same as that by B_{pqr}.

This implies that the vertices in $\mathsf{CVD}_k(S)$ and in $\overline{\mathsf{CVD}}_k(S)$ under the convex distance function based on B are preserved in their counterpart diagrams under the convex distance function based on $\hat{B}$. Furthermore, the general position assumption can be relaxed, as it does not decrease the number of vertices in the diagrams. Hence, we conclude:

▶ **Corollary 18.** *Let S be a set of n colored points in $\mathbb{R}^2$ with m colors. For $1 \leqslant k \leqslant m-1$, the total number of vertices in $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ under any L_p metric for $1 \leqslant p \leqslant \infty$ or any convex distance function is at most $4k(n-k)-2n$.*

It is worth noting that Lemmas 12 and 17 imply bounds for colored j-facets in $\mathbb{R}^2$.

▶ **Corollary 19.** *Let S be a set of n colored points in $\mathbb{R}^2$ with m colors. For $0 \leqslant k \leqslant m-2$,*

$$(k+1)(k+2) \leqslant \sum_{j=0}^{k} e_{2,j}(S) + \sum_{j=0}^{k}(k-j+1)e_{1,j}(S) \leqslant (k+1)(2n-k-2).$$

More on polygonal convex distance functions. First, we consider the L_∞ metric, so B is the unit square centered at the origin. Liu, Papadopoulou, and Lee [30] proved an upper bound of $O((n-k)^2)$ for ordinary order-k Voronoi diagrams of n points under the L_∞ metric using the *Hanan grid*. An analogous argument can also be applied to our color diagrams.

▶ **Lemma 20.** *Under the L_∞ metric, the number of vertices in $\mathsf{CVD}_k(S)$ is at most $4(n-k)^2$.*

Hence, the complexity of $\mathsf{CVD}_k(S)$ under the L_∞ metric is $O(\min\{k(n-k),(n-k)^2\})$. For the maximal counterpart $\overline{\mathsf{CVD}}_k(S)$, we prove the following.

▶ **Lemma 21.** *Under the L_∞ metric, for any $0 \leqslant j \leqslant m-2$, we have $\overline{U}_j \leqslant 2(j+1)(j+2)$. Therefore, the number of vertices of $\overline{\mathsf{CVD}}_k(S)$ for $1 \leqslant k \leqslant m-1$ is at most $2k^2$.*

Summarizing, we obtain:

▶ **Theorem 22.** *Let S be a set of n colored points with m colors in the L_∞ or L_1 plane. For $1 \leqslant k \leqslant m-1$, the number of vertices in $\mathsf{CVD}_k(S)$ is at most $\min\{4k(n-k)-2n, 4(n-k)^2\}$ and the number of vertices in $\overline{\mathsf{CVD}}_k(S)$ is at most $\min\{4k(n-k)-2n, 2k^2\}$.*

The above approach also works for polygonal convex distances, concluding the following.

▶ **Corollary 23.** *Let B be a convex $2b$-gon with $2b \geqslant 4$, centrally symmetric around the origin, and S a set of n colored points with m colors. For $1 \leqslant k \leqslant m-1$, $\mathsf{CVD}_k(S)$ and $\overline{\mathsf{CVD}}_k(S)$ under the convex distance function based on B consist of at most $\min\{4k(n-k)-2n, 2(b^2-b)(n-k)^2\}$ and $\min\{4k(n-k)-2n, 2(b^2-b-1)k^2\}$ vertices, respectively.*

Remark that a more careful analysis could reduce the factor depending on b, and relax the central symmetry of B.

5 Iterative algorithms for color Voronoi diagrams

In this section, we present an iterative approach to compute the order-k color Voronoi diagrams and their refined counterparts for an increasing order of k. Recall that S is a set of n sites associated with distance functions δ_s for $s \in S$. We assume the general position assumption on $\{\delta_s\}_{s\in S}$ given in Section 2. We first establish some key structural properties, which add the concept of color to well-known properties of order-k Voronoi diagrams.

Consider a face f of an order-k Voronoi region $R_k(H;S)$ of $\mathsf{CVD}_k(S)$, or a region $\overline{R}_k(H;S)$ of $\overline{\mathsf{CVD}}_k(S)$, where $H \subseteq K$ with $|H| = k$. Recall that $S_H \subseteq S$ is the set of sites whose colors are included in H. Let $S_f \subseteq S$ be the set of sites that, together with sites in S_H, define the edges along the boundary of f. The following property is directly derived from the definitions.

▶ **Lemma 24.** *Let $f \subseteq R_k(H;S)$ be a face of $\mathsf{CVD}_k(S)$ for $1 \leqslant k \leqslant m-1$. It holds that:*

- **(i)** $\mathsf{CVD}_1(S_f) \cap f = \mathsf{CVD}_{k+1}(S) \cap f$ *and* $\mathsf{VD}(S_f) \cap f = \mathsf{CVD}^*_{k+1}(S) \cap f$.
- **(ii)** $\mathsf{FCVD}(S_H) \cap f = \mathsf{CVD}_{k-1}(S) \cap f$ *and* $\mathsf{FCVD}^*(S_H) \cap f = \mathsf{CVD}^*_k(S) \cap f$.

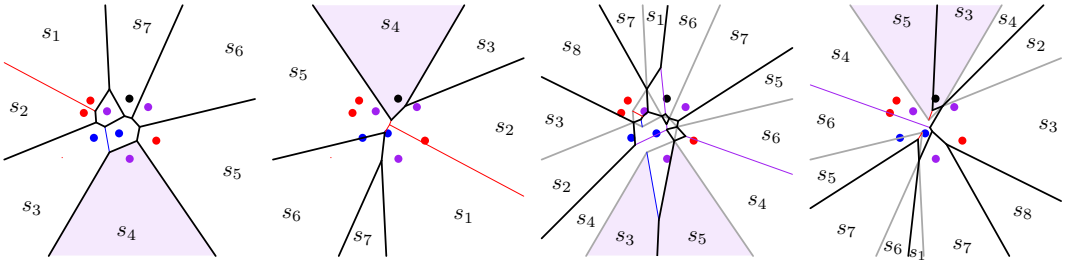
Analogous claims hold for the maximal diagrams, however, for an unbounded face f of $\overline{\mathsf{CVD}}_k(S)$, the set S_f is no longer adequate to derive the portion of $\overline{\mathsf{CVD}}_{k+1}(S)$ that lies within f. We need the set $S_f^+ \subseteq S \setminus S_H$ that defines the unbounded faces of $\overline{\mathsf{CVD}}_{k+1}(S) \cap f$.

▶ **Lemma 25.** *Let $f \subseteq \overline{R}_k(H;S)$ be a face of $\overline{\mathsf{CVD}}_k(S)$ for $1 \leqslant k \leqslant m-1$. Let $S_f^+ \subseteq S \setminus S_H$ be the set of sites that define unbounded faces in $\overline{\mathsf{CVD}}_{k+1}(S) \cap f$; $S_f^+ = \emptyset$ if f is bounded. The following hold:*

(i) $\overline{\mathsf{CVD}}_1(S_f \cup S_f^+) \cap f = \overline{\mathsf{CVD}}_{k+1}(S) \cap f$ *and* $\mathsf{FVD}(S_f \cup S_f^+) \cap f = \overline{\mathsf{CVD}}^*_{k+1}(S) \cap f$.
(ii) $\mathsf{HVD}(S_H) \cap f = \overline{\mathsf{CVD}}_{k-1}(S) \cap f$ *and* $\mathsf{HVD}^*(S_H) \cap f = \overline{\mathsf{CVD}}^*_k(S) \cap f$.

Using Lemma 24(i), we can iteratively compute $\mathsf{CVD}^*_{k+1}(S)$, given $\mathsf{CVD}^*_k(S)$. However, to iteratively compute $\overline{\mathsf{CVD}}_{k+1}(S)$, we first need to identify the sites that define the new unbounded edges of $\overline{\mathsf{CVD}}_{k+1}(S)$. This information, however, is not encoded in $\overline{\mathsf{CVD}}_k(S)$. We give a condition, related to condition **V3**, under which we can use $\mathsf{CVD}^*_{k+1}(S)$ to derive the information missing from $\overline{\mathsf{CVD}}^*_{k+1}(S)$. This condition is satisfied if, for example, S is a set of points and δ_s for $s \in S$ is a convex distance based on a smooth body.

V3′ The unbounded faces of $\mathsf{VD}(S')$ and $\mathsf{FVD}(S')$ are defined by the same sequence of sites, for any $S' \subseteq S$.



(a) $\mathsf{CVD}^*_1(S) = \mathsf{VD}(S)$ (b) $\overline{\mathsf{CVD}}^*_1(S) = \mathsf{FVD}(S)$ (c) $\mathsf{CVD}^*_2(S)$ (d) $\overline{\mathsf{CVD}}^*_2(S)$

Figure 6 Illustration of condition **V3′** and Lemma 26 with the same set $S = \{s_1, \ldots, s_9\}$ as in Figures 2–3 under the Euclidean metric. In (a)(b), $\mathsf{VD}(S')$ and $\mathsf{FVD}(S')$ have the same sequence of sites that define unbounded faces for any $S' \subseteq S$, so condition **V3′** holds. The shaded region in (a) is a face f of $\mathsf{CVD}_1(S)$ corresponding to face f' of $\overline{\mathsf{CVD}}_1(S)$ shaded in (b). In (c)(d), shaded regions show how the portions of f and f' at infinity are subdivided in $\mathsf{CVD}^*_2(S)$ and $\overline{\mathsf{CVD}}^*_2(S)$.

▶ **Lemma 26.** *Condition **V3′** implies that the unbounded faces of* $\mathsf{CVD}^*_k(S)$ *and of* $\overline{\mathsf{CVD}}^*_k(S)$ *are defined by the same sequence of sites for any* $1 \leqslant k \leqslant m$. *(See Figure 6.)*

Now, we assume that the sites S and their distance functions δ_s fall under the model of *abstract Voronoi diagrams* [28]. Furthermore, we assume that any distance or bisector can be computed in $O(1)$ time. We then conclude the following using procedures in [3, 6, 25, 39].

▶ **Theorem 27.** *Let* S *and* $\{\delta_s\}_{s \in S}$ *be a set of* n *colored sites with* m *colors and distance functions that satisfy the conditions of abstract Voronoi diagrams. Then, for* $1 \leqslant k \leqslant m$, *in* $O(k^2 n + n \log n)$ *expected time or in* $O(k^2 n \log n)$ *worst-case time, we can compute* $\mathsf{CVD}^*_1(S), \ldots, \mathsf{CVD}^*_k(S)$. *If in addition condition **V3′** holds, then we can also compute* $\overline{\mathsf{CVD}}^*_1(S), \ldots, \overline{\mathsf{CVD}}^*_k(S)$ *in the same time bound. If* S *consists of points and* $\delta_s(x)$ *is the Euclidean distance to* $s \in S$, *the time bound is reduced to* $O(k^2 n + n \log n)$ *in the worst case.*

The convex distance functions satisfy the conditions of Theorem 27, hence we have:

▶ **Corollary 28.** *Let* B *be a convex and compact body in* $\mathbb{R}^2$ *of a constant complexity that contains the origin in its interior. Given a set* S *of* n *colored points in* $\mathbb{R}^2$ *with* m *colors and an integer* $1 \leqslant k \leqslant m$, *we can compute* $\mathsf{CVD}^*_1(S), \ldots, \mathsf{CVD}^*_k(S)$ *in* $O(k^2 n + n \log n)$ *expected time or in* $O(k^2 n \log n)$ *worst-case time. If* B *is smooth, then we can also compute* $\overline{\mathsf{CVD}}^*_1(S), \ldots, \overline{\mathsf{CVD}}^*_k(S)$ *in the same time bound.*

References

1 Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. The farthest color Voronoi diagram and related problems. Technical Report 002, Rheinische Friedrich–Wilhelms–Universität Bonn, 2006.

2 Pankaj K. Agarwal, Mark de Berg, Jiří Matoušek, and Otfried Schwarzkopf. Constructing levels in arrangements and higher order Voronoi diagrams. *SIAM J. Comput.*, 27(3):654–667, 1998. doi:10.1137/S0097539795281840.

3 Alok Aggarwal, Leonidas J. Guibas, James B. Saxe, and Peter W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(1):591–604, 1989. doi:10.1007/BF02187749.

4 Elena Arseneva and Evanthia Papadopoulou. Randomized incremental construction for the Hausdorff Voronoi diagram revisited and extended. *Journal of Combinatorial Optimization*, 37(2):579–600, February 2019. doi:10.1007/s10878-018-0347-x.

5 Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16(1):78–96, 1987. doi:10.1137/0216006.

6 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, Singapore, 2013. doi:10.1142/8685.

7 Franz Aurenhammer and Otfried Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order Voronoi diagram. *Int. J. Comput. Geom. Appl.*, 2(4):363–381, 1992. doi:10.1142/S0218195992000214.

8 Sang Won Bae. On linear-sized farthest-color Voronoi diagrams. *IEICE Trans. Inf. Syst.*, 95-D(3):731–736, 2012. doi:10.1587/transinf.E95.D.731.

9 Sang Won Bae. Tight bound and improved algorithm for farthest-color Voronoi diagrams of line segments. *Comput. Geom.: Theory Appl.*, 47(8):779–788, 2014. doi:10.1016/j.comgeo.2014.04.005.

10 Sang Won Bae, Nicolau Oliver, and Evanthia Papadopoulou. Higher-order color Voronoi diagrams and the colorful Clarkson–Shor framework, 2025. arXiv:2504.06960.

11 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*, 72(3):1304–1332, May 2023. doi:10.1007/s00454-023-00492-2.

12 Ranita Biswas, Sebastiano Cultrera di Montesano, Herbert Edelsbrunner, and Morteza Saghafian. Counting cells of order-k Voronoi tessellations in $\mathbb{R}^3$ with Morse theory. In Kevin Buchin and Éric Colin de Verdière, editors, *Proceedings of the 37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPIcs*, pages 16:1–16:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SoCG.2021.16.

13 Cecilia Bohler, Panagiotis Cheilaris, Rolf Klein, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. On the complexity of higher order abstract Voronoi diagrams. *Comput. Geom.: Theory Appl.*, 48(8):539–551, 2015. doi:10.1016/j.comgeo.2015.04.008.

14 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.

15 Cecilia Bohler, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. A randomized divide and conquer algorithm for higher-order abstract Voronoi diagrams. *Comput. Geom.: Theory Appl.*, 59:26–38, 2016. doi:10.1016/j.comgeo.2016.08.004.

16 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000. doi:10.1137/S0097539798349188.

17 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 David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4451–4463. SIAM, 2024. doi:10.1137/1.9781611977912.156.

18 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete Comput. Geom.*, 56(4):866–881, 2016. doi:10.1007/s00454-016-9784-4.

19 L. Paul Chew and Robert L.S. Drysdale III. Voronoi diagrams based on convex distance functions. In *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 235–244. ACM, 1985. doi:10.1145/323233.323264.

20 Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989. doi:10.1007/BF02187740.

21 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

22 Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete Comput. Geom.*, 4(4):311–336, 1989. doi:10.1007/BF02187733.

23 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1:25–44, 1986. doi:10.1007/BF02187681.

24 Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Comput. Geom.*, 9:267–291, 1993. doi:10.1007/BF02189323.

25 Kolja Junginger and Evanthia Papadopoulou. Deletion in abstract Voronoi diagrams in expected linear time and related problems. *Discrete & Computational Geometry*, 69(4):1040–1078, March 2023. doi:10.1007/s00454-022-00463-z.

26 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

27 Naoki Katoh and Takeshi Tokuyama. k-Levels of concave surfaces. *Discrete Comput. Geom.*, 27:567–584, 2002. doi:10.1007/s00454-001-0086-z.

28 Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1989. doi:10.1007/3-540-52055-4.

29 Der-Tsai Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Computers*, 31(6):478–487, 1982. doi:10.1109/TC.1982.1676031.

30 Chih-Hung Liu, Evanthia Papadopoulou, and Der-Tsai Lee. The k-nearest-neighbor Voronoi diagram revisited. *Algorithmica*, 71(2):429–449, 2015. doi:10.1007/s00453-013-9809-9.

31 Lihong Ma. *Bisectors and Voronoi Diagams for Convex Distance Functions*. PhD thesis, Fern Unversität Hagen, 2000.

32 Ioannis Mantas, Evanthia Papadopoulou, Vera Sacristán, and Rodrigo I. Silveira. Farthest color Voronoi diagrams: Complexity and algorithms. In *Proc. LATIN 2020*, volume 12118 of *Lecture Notes in Computer Science*, pages 283–295, 2020. doi:10.1007/978-3-030-61792-9_23.

33 Ioannis Mantas, Evanthia Papadopoulou, Rodrigo I. Silveira, and Zeyu Wang. The farthest color Voronoi diagram in the plane. Algorithmica to appear, preprint available at Research Square. doi:10.21203/rs.3.rs-4644060/v1.

34 Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

35 Kurt Mehlhorn, Stefan Meiser, and Ronald Rasch. Furthest site abstract Voronoi diagrams. *Int. J. Comput. Geom. Appl.*, 11(6):583–616, 2001. doi:10.1142/S0218195901000663.

36 Ketan Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete Comput. Geom.*, 6(1):307–338, 1991. doi:10.1007/BF02574692.

37 Evanthia Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 20(5):583–597, 2001. doi:10.1109/43.920683.

38 Evanthia Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004. doi:10.1007/s00453-004-1095-0.

39 Evanthia Papadopoulou. Abstract Voronoi-like Graphs: Extending Delaunay's Theorem and Applications. In *Proc. 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 52:1–52:16, 2023. doi:10.4230/LIPIcs.SoCG.2023.52.

40 Evanthia Papadopoulou and Maksym Zavershynskyi. The higher-order Voronoi diagram of line segments. *Algorithmica*, 74(1):415–439, 2016. doi:10.1007/s00453-014-9950-0.

41 Edgar A. Ramos. On range reporting, ray shooting, and k-level construction. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry, Miami Beach, Florida, USA, June 13-16, 1999*, pages 390–399. ACM, 1999. doi:10.1145/304893.304993.

42 Micha Sharir. The Clarkson–Shor technique revisited and extended. *Comb. Probab. Comput.*, 12(2):191–201, 2003. doi:10.1017/S0963548302005527.

43 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

44 Uli Wagner. k-sets and k-facets. In Jacob E. Goodman, János Pach, and Richard Pollack, editors, *Surveys on Discrete and Computational Geometry*, volume 453 of *Contemporary Mathematics*, pages 443–513. Amer. Math. Soc., 2008.

The Erdős–Szekeres Conjecture Revisited

Jineon Baek
Department of Mathematics, University of Michigan, Ann Arbor, MI, USA

Martin Balko
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Abstract

The famous and still open *Erdős–Szekeres Conjecture* from 1935 states that every set of at least $2^{k-2}+1$ points in the plane with no three being collinear contains k points in *convex position*, that is, k points that are vertices of a convex polygon. In this paper, we revisit this conjecture and show several new related results.

First, we prove a relaxed version of the Erdős–Szekeres Conjecture by showing that every set of at least $2^{k-2}+1$ points in the plane with no three being collinear contains a *split k-gon*, a relaxation of k-tuple of points in convex position. Moreover, we show that this is tight, showing that the value $2^{k-2}+1$ from the Erdős–Szekeres Conjecture is exactly the right threshold for split k-gons.

We obtain an analogous relaxation in a much more general setting of ordered 3-uniform hypergraphs where we also show that, perhaps surprisingly, a corresponding generalization of the Erdős–Szekeres Conjecture is not true. Finally, we prove the Erdős–Szekeres Conjecture for so-called *decomposable sets* and provide new constructions of sets of 2^{k-2} points without k points in convex position, generalizing all previously known constructions of such point sets and allowing us to computationally tackle the Erdős–Szekeres Conjecture for large values of k.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Computational geometry; Mathematics of computing → Combinatorics

Keywords and phrases convex position, Erdős–Szekeres theorem, point set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.13

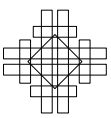
Funding *Jineon Baek*: acknowledges support from the Korea Foundation for Advanced Studies during the completion of this research.
Martin Balko: supported by grant no. 23-04949X of the Czech Science Foundation (GAČR) and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE 24/SCI/008).

1 Introduction

A finite set of points on a plane is in *general position* if there are no three points on a single line and in *convex position* if they form the vertices of a convex polygon. We consider finite point sets in general position where no two points have the same x- or y-coordinate.

Motivated by a problem posed by Esther Klein, Erdős and Szekeres [7] showed that for any positive integer k there is a minimum integer $ES(k)$ such that any set of $ES(k)$ points in the plane in general position contains k points in convex position. This famous result, called the *Erdős–Szekeres Theorem*, was a starting point of both discrete geometry and Ramsey theory and was obtained as one of the first applications of Ramsey's Theorem, which was independently discovered by Erdős and Szekeres [7] in the same paper. Ramsey's Theorem, however, yields a poor bound on the minimum size $ES(k)$ of a planar point set in general position that guarantees the existence of k points in convex position. Thus, Erdős and Szekeres [7] also provided an alternative proof that yields a much better estimate on $ES(k)$. This alternative argument uses more geometry and is based on so-called *caps* and *cups*.

© Jineon Baek and Martin Balko;
licensed under Creative Commons License CC-BY 4.0



For a positive integer a, we say that a points in the plane form an *a-cap* if they lie on a graph of a concave function. Similarly, for $u \in \mathbb{N}$, a set of u points in the plane forms a *u-cup* if its points lie on a graph of a convex function. Using this notion, Erdős and Szekeres [7] proved the following result known as the *Erdős–Szekeres Cap-Cup Theorem*.

▶ **Theorem 1** (The Erdős–Szekeres Cap-Cup Theorem [7]). *For all integers $a, u \geq 2$, every set of at least $\binom{a+u-4}{a-2}+1$ points in the plane in general position contains an a-cap or a u-cup.*

Moreover, this is tight. That is, for all integers $a, u \geq 2$, there are sets of $\binom{a+u-4}{a-2}$ points in the plane in general position with neither an a-cap nor a u-cup.

Since points of an a-cap and a u-cup are in convex position, Theorem 1 gives

$$ES(k) \leq \binom{2k-4}{k-2} + 1 \tag{1}$$

for any integer $k \geq 2$. In fact, observe that every set of k points in convex position is a union of an *a-cap* and a *u-cup* that share common endpoints and satisfy $a + u = k + 2$.

The bound (1) gives roughly $ES(k) \leq 4^k$. Erdős–Szekeres believed that this could be significantly improved and posed the following conjecture.

▶ **Conjecture 2** (The Erdős–Szekeres Conjecture [7]). *For every integer $k \geq 2$, we have*

$$ES(k) = 2^{k-2} + 1.$$

When posed in 1935, the conjecture was known to be true only for $k \leq 4$. Later, it was proved for $k = 5$, and Peters and Szekeres [19] proved it also for $k = 6$ in 2006 using a computer-assisted approach. In the 1960s, Erdős and Szekeres [8] showed $ES(k) \geq 2^{k-2} + 1$ for any integer $k \geq 2$, further supporting the conjecture. Despite many efforts, this conjecture is still open for $k \geq 7$, and Erdős himself offered \$500 reward for its solution. For many decades, the bound (1) was the best known, but then several smaller-order improvements appeared [5, 14, 16, 18, 21, 22]. In 2017, Suk [20] made a breakthrough on this conjecture by proving $ES(k) \in 2^{k+O(k^{2/3}\log k)}$ and his bound was then slightly improved by Holmsen, Mojarrad, Pach, and Tardos [13] to $ES(k) \in 2^{k+O(\sqrt{k \log k})}$. Although these bounds are fairly close to $2^{k-2} + 1$, there is still a gap between the known lower and upper bounds, and this famous conjecture thus remains open.

2 Our results

In this paper, we introduce a natural relaxed variant of the Erdős–Szekeres Conjecture for which the number $2^{k-2} + 1$ is exactly the right threshold. To our knowledge, this is the first version of this famous conjecture where the conjectured value $2^{k-2} + 1$ appears as the right threshold. We discuss abstract combinatorial extensions of this result and connections to the original Erdős–Szekeres Conjecture, some of which indicate that this conjecture might be false. We also prove the Erdős–Szekeres Conjecture restricted to special point sets, and we provide new examples of 2^{k-2} point sets with no k points in convex position.

2.1 Split polygons

For a positive integer N, let P be a set of N points in the plane in general position. For a positive integer k, a *split k-gon* in P is a set of points from P that is formed by an a-cap and a u-cup that share the rightmost point and that satisfy $a + u = k + 2$; see Figure 1. Note that a split k-gon can contain k or $k + 1$ points and that if the a-cap and the u-cup share also the leftmost point, then the points of the split k-gon are in convex position.

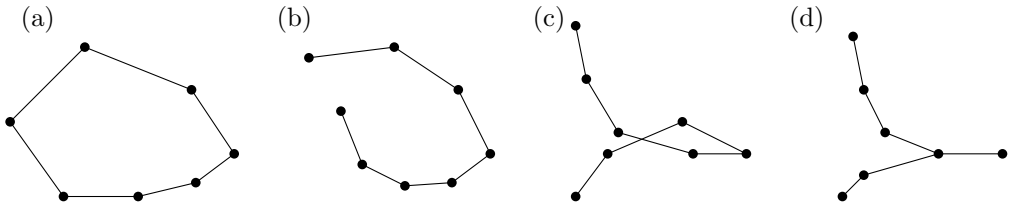


Figure 1 (a–d) Examples of split 7-gons. Note that the split 7-gons in (a) and (d) have only 7 points while the remaining split 5-gons contain 8 points.

We consider a variant of the Erdős–Szekeres Theorem for split polygons. More specifically, for a positive integer k, let $ES_{split}(k)$ be the minimum positive integer N such that every finite set of at least N points in the plane in general position contains a split k-gon. Since $ES_{split}(k) \leq ES(k)$, we have $ES_{split}(k) \leq 2^{(1+o(1))k}$ by the result of Suk [20].

As our first main result, we prove that the numbers $ES_{split}(k)$ attain exactly the value $2^{k-2}+1$ from the Erdős–Szekeres Conjecture.

▶ **Theorem 3.** *For every integer $k \geq 2$, we have $ES_{split}(k) = 2^{k-2}+1$.*

In fact, we can prove a stronger statement in the setting corresponding to a refinement of the Erdős–Szekeres Conjecture as introduced by Erdős, Tuza, and Valtr [9]. In their refinement, one tries to find k-tuples of points in convex position in point sets that do not contain a-caps and u-cups of prescribed sizes a and u.

We state this formally in our setting. For integers $a, u, k \geq 2$, let $ES_{split}(a, u, k)$ be the minimum positive integer N such that every finite set of at least N points in the plane in general position contains an a-cap, a u-cup, or a split k-gon. We are mostly interested in the cases $\max\{a, u\} \leq k \leq a + u - 2$ as otherwise at least one parameter from a, u, k is unnecessary. Therefore, we define

$$T = \{(a, u, k) : a, u \geq 2 \text{ and } \max\{a, u\} \leq k \leq a + u - 2\}.$$

For any triple $(a, u, k) \in T$, we can then determine the numbers $ES_{split}(a, u, k)$ exactly.

▶ **Theorem 4.** *For every $(a, u, k) \in T$, we have*

$$ES_{split}(a, u, k) = 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

Note that $ES_{split}(k) = ES_{split}(k, k, k)$. Thus, Theorem 3 follows from Theorem 4 by setting $a = u = k$ as then

$$ES_{split}(k) = 1 + \sum_{i=2}^{k} \binom{k-2}{i-2} = 2^{k-2} + 1.$$

Also, observe that Theorem 1 is a special case of Theorem 4 for $k = a + u - 2$.

We also remark that if $ES(a, u, k)$ is the minimum positive integer N such that every finite set of at least N points in the plane in general position contains an a-cap, a u-cup, or k points in convex position, then it is conjectured that

$$ES(a, u, k) = 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2} \tag{2}$$

for all $(a, u, k) \in T$. However, this equality is known to be true only in the trivial case $k \geq a + u - 3$ and Baek [1] proved it when a or u equals 4. Surprisingly, Erdős, Tuza, and Valtr [9] showed that the Erdős–Szekeres conjecture is equivalent to (2).

2.2 Abstract split polygons

The proof of the upper bound in Theorem 4 is based on techniques used by Moshkovitz and Shapira [17] when estimating ordered Ramsey numbers of so-called monotone paths. It turns out that the proof can be generalized to this abstract combinatorial setting. To state this extension, we need to introduce some definitions.

For a positive integer N, let $\mathcal{K}_N^3$ be the ordered complete 3-uniform hypergraph with the vertex set $[N] = \{1, \dots, N\}$ ordered by the standard ordering $<$ of the positive integers. A *2-coloring* of $\mathcal{K}_N^3$ is a function χ that assigns either a red or blue color to each hyperedge of $\mathcal{K}_N^3$. A *monotone path* $\mathcal{P}_n^3$ is an ordered 3-uniform hypergraph on $n \geq 2$ vertices where the hyperedges are formed by triples of vertices that are consecutive in the vertex ordering of $\mathcal{P}_n^3$.

Let $P = \{p_1, \dots, p_N\}$ be a set of points in the plane in general position with $x(p_1) < \cdots < x(p_N)$ where $x(p_i)$ is the x-coordinate of a point p_i. The *2-coloring induced by* P is a 2-coloring χ_P of $\mathcal{K}_N^3$ that colors a hyperedge $\{i, j, k\}$ with $i < j < k$ red if the points p_i, p_j, p_k form a 3-cap and blue if p_i, p_j, p_k form a 3-cup. Observe that, for each n, red monotone paths on n vertices in χ_P correspond to n-caps in P and that blue monotone paths on n vertices in χ_P correspond to n-cups in P.

In 2014, Moshkovitz and Shapira [17] proved the following result, whose first part vastly generalizes the first part of the Erdős–Szekeres Cap-Cup Theorem (Theorem 1) as it deals with all 2-colorings of $\mathcal{K}_N^3$ and not only with those induced by a point set of size N.

▶ **Theorem 5** ([7, 11, 17]). *For all integers $a, u \geq 2$ and $N \geq \binom{a+u-4}{a-2} + 1$, every 2-coloring of $\mathcal{K}_N^3$ contains either a red copy of $\mathcal{P}_a^3$ or a blue copy of $\mathcal{P}_u^3$.*

Moreover, this is tight. That is, there are 2-colorings of $\mathcal{K}_N^3$ with $N = \binom{a+u-4}{a-2}$ that do not contains red copy of $\mathcal{P}_a^3$ nor a blue copy of $\mathcal{P}_u^3$.

In other words, Theorem 5 states that the *ordered Ramsey number* [3, 6] of $\mathcal{P}_a^3$ and $\mathcal{P}_u^3$ equals $\binom{a+u-4}{a-2} + 1$. We extend this result similarly as Theorem 4 extends Theorem 1.

We start by generalizing split polygons. Let χ be a 2-coloring of $\mathcal{K}_N^3$ with colors red and blue. For $k \in \mathbb{N}$, an *abstract split k-gon* in χ is an ordered subhypergraph of $\mathcal{K}_N^3$ that is formed by a red monotone path $\mathcal{P}_a^3$ and a blue monotone path $\mathcal{P}_u^3$ that share the rightmost vertex and that satisfy $a + u = k + 2$. Observe that an abstract split k-gon can have at most $k+1$ vertices. Also, note that, for every abstract split k-gon C in a coloring χ_P induced by a point set P, the two monochromatic monotone paths in C can share only their endpoint pair and that abstract split k-gons in χ_P are in one-to-one correspondence with split k-gons in P.

Generalizing the definition of $ES_{split}(a, u, k)$, we let $C_{split}(a, u, k)$ be the minimum positive integer N such that every 2-coloring of $\mathcal{K}_N^3$ with colors red and blue contains a red copy of $\mathcal{P}_a^3$, a blue copy of $\mathcal{P}_u^3$, or an abstract split k-gon. Observe that considering the 2-colorings induced by P gives, for every $(a, u, k) \in T$,

$$ES_{split}(a, u, k) \leq C_{split}(a, u, k). \tag{3}$$

As our next result, we fully determine the numbers $C_{split}(a, u, k)$ by showing that $ES_{split}(a, u, k) = C_{split}(a, u, k)$ for every $(a, u, k) \in T$.

▶ **Theorem 6.** *For every $(a, u, k) \in T$, we have*

$$C_{split}(a, u, k) = 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

Similarly as before, note that Theorem 5 is a special case of Theorem 6 for $k = a + u - 2$.

2.3 Finding polygons that are not split

The only difference between the setting in the Erdős–Szekeres Conjecture and the setting in Theorem 3 is that in the latter one, we allow the leftmost endpoints of the a-cap and the u-cup to be different. Enforcing these two endpoints to meet without increasing the size of the considered point sets would resolve the Erdős–Szekeres Conjecture. However, the combinatorial structure used to prove Theorem 3 does not seem strong enough to ensure this. Moreover, we will show that this strengthening is not true in the abstract combinatorial setting from Subsection 2.2.

First, we generalize the concept of convex position. For a positive integer N, let χ be a 2-coloring of $\mathcal{K}_N^3$. For a positive integer k, a *weak k-gon* in χ is an ordered subhypergraph of $\mathcal{K}_N^3$ that is formed by a red monotone path $\mathcal{P}_a^3$ and a blue monotone path $\mathcal{P}_u^3$ that share both their end-vertices and satisfy $a + u = k + 2$. That is, a weak k-gon is a particular case of an abstract split k-gon. Note that the red and the blue monotone path can share other vertices, not only their two end-vertices. If we exclude this possibility and allow the two paths to share only their end-vertices, then we call such ordered subhypergraph a *strong k-gon* in χ. If χ_P is a 2-coloring induced by a point set P, then every weak k-gon is a strong k-gon in χ_P and corresponds to a set of k points from P that are in convex position.

Let $C_{weak}(k)$ be the minimum $N \in \mathbb{N}$ such that every 2-coloring of $\mathcal{K}_N^3$ contains a weak k-gon. An analogous number for strong k-gons is denoted by $C_{strong}(k)$. Clearly,

$$C_{strong}(k) \geq C_{weak}(k) \geq C_{split}(k, k, k) = 2^{k-2} + 1,$$

where the equality follows from Theorem 6. Moreover, if $C_{weak}(k) \leq 2^{k-2} + 1$, then the Erdős–Szekeres Conjecture is true.

We do not have $C_{strong}(k) \leq 2^{k-2} + 1$ as Balko and Valtr [4] proved $C_{strong}(7) > 33$ with SAT solvers, disproving a conjecture of Peters and Szekeres [19]. Our SAT-solver-based proofs show that this is also the case for weak k-gons.

▶ **Theorem 7.** *We have $C_{weak}(7) > 33$.*

Thus, there is no hope for extending the Erdős–Szekeres Conjecture to the abstract setting with weak k-gons. This also suggests that the Erdős–Szekeres Conjecture might be false and that one might look for counterexamples. A natural candidate for a counterexample is a subset of the point sets that were used by Erdős–Szekeres to prove the tightness in Theorem 1 for a-caps and u-cups since such a set used by Erdős–Szekeres [7] contains a subset of 2^{k-2} points with no k points in convex position.

However, we prove that it is impossible to find a counterexample to Erdős–Szekeres Conjecture by restricting ourselves to these sets. In fact, we show this for a larger class of so-called *decomposable sets* (see Section 4 for their definition). These sets include, for example, the point sets with no long caps nor cups used by Erdős–Szekeres [7] as well as the *Horton sets* defined by Valtr [23].

▶ **Theorem 8.** *Let a, u, k be positive integers satisfying $a, u \leq k$. Then, every decomposable set of more than $\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ points contains an a-cap, a u-cup, or k points in convex position.*

In particular, since every non-empty subset of a decomposable set is decomposable, we obtain, by choosing $a = u = k$, that every subset of more than 2^{k-2} points from a decomposable set contains k points in convex position. Therefore, if we restrict ourselves to decomposable sets, the Erdős–Szekeres Conjecture is true. The proof of Theorem 8 is omitted.

Finally, when exploring the validity of the Erdős–Szekeres Conjecture, we discovered new constructions of sets of 2^{k-2} points in general position with no k points in convex position. As our last main result, we introduce a "blow-up" method that generalizes all previously known constructions of such point sets; see Section 4 for their brief description. Our new method is captured by Lemma 14 in Section 6. We show that all previously known examples follow rather easily from this lemma and that it also gives new examples of such point sets.

We also tried to computationally tackle the Erdős–Szekeres conjecture for large k using a non-linear optimization solver based on a genetic algorithm and this new construction. However, despite finding numerous new examples of 2^{k-2} points without k points in convex position even for $k = 20$, our experiments never produced such points sets with more than 2^{k-2} points; see Section 6 for more details.

3 Open problems

We showed that the variant of the Erdős–Szekeres Conjecture is true for split k-gons, even in an abstract combinatorial setting, while the original Erdős–Szekeres Conjecture does not admit such an abstract strengthening; these results are summarized in Table 1. Each cell corresponds to a variant statement of the Erdős–Szekeres Conjecture.

Table 1 Variants of the Erdős–Szekeres Conjecture and their status.

	Point sets	**Ordered hypergraphs**
Split k-gons	True (Theorem 6)	True (Theorem 3)
Weak k-gons	Open (Erdős–Szekeres [7])	False (Theorem 7)

It also follows from Theorem 7 that any proof of the Erdős–Szekeres Conjecture has to use some assumptions on the 2-colorings. Natural candidates for such 2-colorings are so-called *signotopes* that appear under many different names in the literature (see [2, 10], for example). For $N \in \mathbb{N}$, a 2-coloring χ of $\mathcal{K}_N^3$ is a *signotope* if for every 4-tuple $a < b < c < d$ of vertices of $\mathcal{K}_N^3$ there is at most one change of color in the sequence $(\chi(a,b,c), \chi(a,b,d), \chi(a,c,d), \chi(b,c,d))$.

Signotopes on $\mathcal{K}_N^3$ are in one-to-one correspondence with pseudoline arrangements determined by N points [10] and thus offer a natural and frequently used combinatorial generalization of point sets. Moreover, if we have $\chi(a,b,c) = \chi(b,c,d)$ for some vertices $a < b < c < d$ in a signotope χ, then $\chi(a,b,c) = \chi(a,b,d) = \chi(a,c,d) = \chi(b,c,d)$. It follows that any weak k-gon in χ is also a strong k-gon in χ.

In our computer experiments, we were not able to find a signotope on at least $1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ vertices that would contain neither a red copy of $\mathcal{P}_a^3$, nor a blue copy of $\mathcal{P}_u^3$, nor a weak k-gon. Thus, it is still possible that the Erdős–Szekeres Conjecture is true even if we do not restrict ourselves to 2-colorings induced by point sets but we consider all signotopes. Such a generalization is equivalent to a conjecture of Goodman and Pollack [12].

Another natural open problem is to obtain better bounds on $C_{weak}(k)$ and $C_{strong}(k)$. We know that both numbers can be larger than $2^{k-2} + 1$, but the strongest upper bound that we are aware of follows from Theorem 5 for $a = u = k$ and gives

$$C_{weak}(k) \leq C_{strong}(k) \leq \binom{2k-4}{k-2} + 1.$$

4 Preliminaries and notation

In this section, we introduce some useful notation and definitions and briefly describe all known constructions of sets of 2^{k-2} that avoid subsets of k points in convex position. Some of these constructions are used later in our proofs.

For a point p in the plane, we use $x(p)$ and $y(p)$ to denote the x- and y-coordinate of p, respectively. Let A and B be two sets of points in the plane in general position. We write $x(A) < x(B)$ (or $y(A) < y(B)$) if A is to the left (or below, respectively) of B, that is, $x(a) < x(b)$ ($y(a) < y(b)$, respectively) for every $a \in A$ and $b \in B$. We say that A is *deep below* B if $y(A) < y(B)$ and all points from B lie above every line determined by two points from A and all points from A lie below every line determined by two points from B.

For integers $a, u \geq 2$, we now describe the sets of $\binom{a+u-4}{a-2}$ points without a-caps and u-cups that were used by Erdős and Szekeres [7] to prove the tight lower bound in Theorem 1. We denote these sets by $P(a, u)$ and we construct them by induction on $a + u$. If $a = 2$ or $u = 2$, then we let $P(a, u)$ be a set of one point. For $a, u \geq 3$, we let $P(a, u)$ be the point set that is obtained by adding a translated copy of $P(a, u-1)$ to $P(a-1, u)$ that is to the left of $P(a-1, u)$ and deep below $P(a-1, u)$. It is then easy to show that $P(a, u)$ does not contain a-cap nor a u-cup [7, 15].

A set P is *decomposable* if either $|P| = 1$ or if $|P| \geq 2$ and P can be partitioned into two decomposable sets A and B such that A is deep below B. Note that the sets $P(a, u)$ are decomposable and that every non-empty subset of a decomposable set is decomposable.

Let a, u, and k be integers satisfying $(a, u, k) \in T$. We use $S(a, u, k)$ to denote the sum $\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$. For $a, u, k \in \mathbb{N}$ with $a, u \leq k$, we also define $S(a, u, k) = 0$ if $a = 1$ or $u = 1$. These cases agree with the formula $\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ as the sum is then empty.

We now describe all known constructions of $S(a, u, k)$ points in general position without a-caps, u-cups, or k points in general position. Let Q be an arbitrary set of $a + u - k - 1$ points $q_{k-a+2}, \ldots, q_u$ in the plane in general position sorted by increasing y-coordinate. Let P be the point set obtained by replacing each q_i with a set Q_i of $\binom{k-2}{i-1}$ points without $(k - i + 2)$-cap and i-cup in a sufficiently small neighborhood of q_i such that Q_i is deep below Q_{i+1} for every $i \in \{k - a + 2, \ldots, u - 1\}$. Then, it can be shown that P does not contain a-cap, u-cup, nor k points in convex position [8, 9]. We call the resulting point set *Valtr's construction* as it was described to us by Valtr [24]. To our knowledge, it is not described in the literature. If, in addition, Q is a $(a + u - k - 1)$-cup and each Q_j is the set $P(k + 1 - j, j + 1)$, then we call the set P the *Erdős–Szekeres construction* and we denote it by $P(a, u, k)$. For $a = u = k$, this is the point set constructed by Erdős and Szekeres [8] to prove $ES(k) \geq 2^{k-2} + 1$.

5 Proofs of Theorems 4 and 6

Here, we prove Theorems 4 and 6 by determining the values $ES_{split}(a, u, k)$ and $C_{split}(a, u, k)$ for every $(a, u, k) \in T$. It follows from (3) that in order to prove both these results, it suffices to prove the upper bound $C_{split}(a, u, k) \leq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ from Theorem 6 and the lower bound $ES_{split}(a, u, k) \geq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ from Theorem 4.

5.1 Upper bound from Theorem 6

Let m and n be positive integers. We use $\preceq$ to denote the partial ordering on $[m]\times[n]$ satisfying $(x_1, y_1) \preceq (x_2, y_2)$ if and only if $x_1 \leq x_2$ and $y_1 \leq y_2$ for all $(x_1, y_1), (x_2, y_2) \in [m] \times [n]$. A subset D of $[m] \times [n]$ is a *down-set* if $y \in D$ implies $x \in D$ for every $x \preceq y$; see Figure 2.

There is a one-to-one correspondence between down-sets in $[m] \times [n]$ and $m \times n$ *line partitions* with entries from $\{0, 1, \ldots, n\}$ which are sequences $(a_1, \ldots, a_m)$ of numbers satisfying $n \geq a_1 \geq a_2 \geq \cdots \geq a_m \geq 0$; see [17]. It is well-known and easy to see that the number of such line partitions (and thus also the number of down-sets in $[m] \times [n]$) is $\binom{m+n}{m}$.

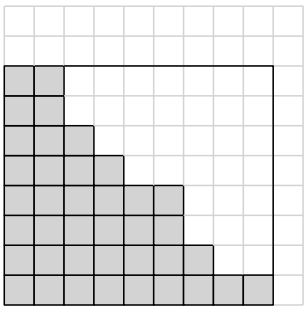


Figure 2 An example of a down-set in $[10] \times [10]$ with the bounding box 9×8.

Given a non-empty down-set D in $[m] \times [n]$, we say that D has the *bounding box* $r \times s$ if r is the maximum integer such that $a_r > 0$ and $a_1 = s$. We say that the empty down-set has the bounding box 0×0.

▶ **Lemma 9.** *For $(a, u, k) \in T$, the number of down-sets in $[a-2] \times [u-2]$ with the bounding box $r \times s$ where $r + s \leq k - 1$ is*

$$\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

The proof of Lemma 9 is omitted. We now prove the upper bound on $C_{split}(a, u, k)$.

▶ **Lemma 10.** *For every $(a, u, k) \in T$, we have*

$$C_{split}(a, u, k) \leq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

Proof. For a positive integer N, let χ be a 2-coloring of $\mathcal{K}_N^3$ with colors red and blue such that there is neither a red copy of $\mathcal{P}_a^3$, a blue copy of $\mathcal{P}_u^3$, nor an abstract split k-gon. We show that $N \leq \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$.

For vertices u and v of $\mathcal{K}_N^3$ with $u < v$, we let $D(u, v) = (n_r - 1, n_b - 1)$ where n_r (n_b, respectively) is the maximum number of vertices of a red (blue, respectively) monotone path ending with u and v. By the choice of χ, we have $D(u, v) \in [a-2] \times [u-2]$. Next, we define

$$D(v) = \{x \in [a-2] \times [u-2] \colon x \preceq D(u, v) \text{ for some } u < v\}.$$

By definition, $D(v)$ is a down-set in $[a-2] \times [u-2]$. Moreover, since there is no abstract split k-gon in χ, $D(v)$ has the bounding box $r \times s$ where $r + s \leq k - 1$.

Consider the mapping $v \mapsto D(v)$. We show that this mapping is injective. Suppose for contradiction that $D(u) = D(v)$ for some vertices $u < v$. By definition, we have $D(u, v) \in D(v)$. Since $D(u) = D(v)$, we obtain $D(u, v) \in D(u)$ and thus, by the definition of $D(u)$, there is a vertex t such that $t < u$ and $D(u, v) \preceq D(t, u)$. However, if $\chi(t, u, v)$ is red, the longest red monotone path ending at u, v is longer than the longest red monotone path ending at t, u. An analogous claim holds for blue monotone paths if $\chi(t, u, v)$ is blue. In either case, we cannot have $D(u, v) \preceq D(t, u)$ and we obtain a contradiction.

Thus, N is at most as large as the number of down-sets in $[a-2]\times[u-2]$ with the bounding box $r \times s$ where $r+s \leq k-1$. By Lemma 9, this number equals $\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$. ◀

By combining Lemma 10 with (3), we obtain the upper bounds from Theorems 4 and 6.

5.2 Lower bound from Theorem 4

Now, we prove the lower bounds. First, we provide a general construction of 2-colorings that avoid long monochromatic monotone paths and abstract split k-gons, obtaining the lower bound $C_{split}(a,u,k) \geq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ from Theorem 6. Then, we show that a special case of this construction can be realized by points in the plane, obtaining a tight lower bound $ES_{split}(a,u,k) \geq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ from Theorem 4.

▶ **Lemma 11.** *For every $(a,u,k) \in T$, we have*

$$C_{split}(a,u,k) \geq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

Proof. We use $\mathcal{E} := \mathcal{D}(a,u)$ to denote the set of all down-sets in $[a-2]\times[u-2]$. In this proof, we let $\mathcal{D} := \mathcal{D}(a,u,k)$ be the set of all down-sets in $[a-2]\times[u-2]$ with the bounding box $r \times s$ where $r+s \leq k-1$ and we set $N = |\mathcal{D}|$. Note that $\mathcal{D} \subseteq \mathcal{E}$. We know, by Lemma 9, that $N = \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$. The set $\mathcal{E}$ is partially ordered by inclusion $\subseteq$. We let $\leq_{\mathcal{E}}$ be an arbitrary linear extension on $\mathcal{E}$ that extends $\subseteq$ on $\mathcal{E}$.

For two down-sets $A, B \in \mathcal{E}$ with $A <_{\mathcal{E}} B$, choose an arbitrary element $\delta(A,B)$ of $B \setminus A$. This is possible, as the fact that $\leq_{\mathcal{E}}$ extends $\subseteq$ implies that the set $B \setminus A$ is non-empty if $A <_{\mathcal{E}} B$. For $i \in \{1,2\}$, we let $\delta(A,B)_i$ be the ith coordinate of $\delta(A,B)$.

We let $\mathcal{K}^3_{|\mathcal{E}|}$ be the ordered complete 3-uniform hypergraph with the vertex set $\mathcal{E}$ ordered by $\leq$. We now define the following 2-coloring $\chi_{a,u} = \chi_{a,u}(\delta, \leq_{\mathcal{E}})$ of $\mathcal{K}^3_{|\mathcal{E}|}$, which is used later in the paper. The color $\chi_{a,u}(A,B,C)$ of an edge $\{A,B,C\}$ of $\mathcal{K}^3_{|\mathcal{E}|}$ with $A<B<C$ is red if $\delta(A,B)_1 < \delta(B,C)_1$ and blue otherwise. Note that $\delta(A,B) \in B$ and $\delta(B,C) \notin B$ and thus $\delta(A,B) \neq \delta(B,C)$. Also, if $\chi_{a,u}(A,B,C)$ is blue, then $\delta(A,B)_2 < \delta(B,C)_2$ as we cannot have $\delta(B,C) \preceq \delta(A,B)$, because otherwise the fact that B is a down-set in $[a-2]\times[u-2]$ together with $\delta(A,B) \in B$ implies $\delta(B,C) \in B$, which is impossible as $\delta(B,C) \in C \setminus B$. Let $\chi = \chi(\delta, \leq_{\mathcal{E}})$ be the coloring $\chi_{a,u}$ restricted to the copy of $\mathcal{K}^3_N$ in $\mathcal{K}^3_{|\mathcal{E}|}$ induced by $\mathcal{D}$.

It remains to show that χ contains neither a red copy of $\mathcal{P}^3_a$, a blue copy of $\mathcal{P}^3_u$, nor an abstract split k-gon as then we have $C_{split}(a,u,k) \geq N+1$. To do so, we first prove by induction on $\ell \geq 2$ that any monochromatic copy of $\mathcal{P}^3_\ell$ in χ with last two vertices $B<C$ satisfies $\delta(B,C)_1 \geq \ell - 1$ if all its hyperedges are red in χ and $\delta(B,C)_2 \geq \ell - 1$ if they are blue. If $\ell = 2$, then the claim is trivially true as $(1,1) \preceq \delta(B,C)$. Thus, consider $\ell \geq 3$. Let $A<B<C$ be the last three vertices of the monochromatic copy of $\mathcal{P}_\ell$ in χ. If all the hyperedges are red, then the induction hypothesis and the definition of χ give $\delta(B,C)_1 > \delta(A,B)_1 \geq \ell - 2$. Similarly, if the hyperedges are blue, we obtain $\delta(B,C)_2 > \delta(A,B)_2 \geq \ell - 2$. Since all down-sets from $\mathcal{D}$ are contained in $[a-2]\times[u-2]$, it follows that there is no red copy of $\mathcal{P}^3_a$ nor a blue copy of $\mathcal{P}^3_u$ in χ. Since every down-set from $\mathcal{D}$ has the bounding box $r \times s$ where $r+s \leq k-1$, we also see that there is no abstract split k-gon in χ. Thus, we have $C_{split}(a,u,k) \geq N+1$. ◀

We now prove $ES_{split}(a,u,k) \geq N+1$. This can be shown directly using the Erdős–Szekeres construction $P(a,u,k)$ from Section 4. We prove this claim by showing that a special case of the construction from the proof of Lemma 11 corresponds to $\chi_{P(a,u,k)}$.

▶ **Lemma 12.** *For every $(a, u, k) \in T$, we have*

$$ES_{split}(a, u, k) \geq 1 + \sum_{i=k-a+2}^{u} \binom{k-2}{i-2}.$$

The proof of Lemma 12 is omitted. Finally, we remark that using the Erdős–Szekeres construction in the proof of the tight lower bound on $ES_{split}(a, u, k)$ is important. For example, in the case $a = u = k$, some of the sets P of 2^{k-2} points constructed in Section 4 do not have k points in convex position but contain split $(2k-4)$-gons.

6 Avoiding k points in convex position

Here, we introduce new constructions of sets of 2^{k-2} points in general position with no k points in convex position, generalizing all previously known constructions of such point sets.

For integers $a, u, k \geq 2$, let $P'(a, u)$ be an arbitrary set of $\binom{a+u-4}{a-2}$ points in general position in the plane with no a-cap and no u-cup. If $(a, u, k) \in T$, let $P'(a, u, k)$ and $P(a, u, k)$ be sets of $\sum_{i=k-a+2}^{u} \binom{k-2}{i-2}$ points in general position in the plane with no a-cap, no u-cup, and no k points in convex position where $P(a, u, k)$ is the point configuration given by the Erdos-Szekeres construction and $P'(a, u, k)$ is an arbitrary such point configuration. Both sets are described in Section 4.

We let S be a set of points $p_1, \ldots, p_N$ sorted by increasing x-coordinates. We use the term *factor* for an integer k that is used as a parameter that corresponds to the forbidden size of a subset in convex position of an enlarged point set obtained from S. For integers i and j with $1 \leq i < j \leq N$, we let $s_{i,j}$ be the maximum number of points from S that are in convex position and have p_i and p_j as their leftmost and rightmost points, respectively.

We now define a blow-up operation that is the core of our construction and will help us obtain large sets with few points in convex position; see Figure 3 for an illustration.

▶ **Definition 13** ((X, Y)-blow-up of S)**.** *Let $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_N)$ be sequences of non-negative integers. For a factor $k \in \mathbb{N}_0$, an (X, Y)-*blow-up *$b_{X,Y,k}(S)$ of S with respect to k is a point set obtained from S by replacing each point p_i of S with $i \in \{2, 3, \ldots, N-1\}$ with an affine image C_{p_i} of $P'(y_i + 2, x_i + 2)$ that is rotated clockwise by $\pi/2$, is almost vertical and lies in a small neighborhood of p_i. The point p_1 is replaced with an affine image C_{p_1} of $P'(k, x_1 + 2, k)$, where this copy is rotated clockwise by $\pi/2$, is almost vertical and lies in a small neighborhood of p_1. Similarly, the point p_N is analogously replaced with an affine image C_{p_N} of $P'(y_N + 2, k, k)$. We choose the neighborhoods small enough so that they are disjoint for different sets C_{p_i} and for all triples i_1, i_2, i_3 with $1 \leq i_1 < i_2 < i_3 \leq N$, the triple $(p_{i_1}, p_{i_2}, p_{i_3})$ has the same orientation as each triple $(q_{i_1}, q_{i_2}, q_{i_3})$ where $q_{i_j} \in C_{p_{i_j}}$. Also, C_{p_i} is almost vertical, meaning that for every $p_j \in S$ with $j \neq i$ any line determined by two points of C_{p_i} has all points of C_{p_j} on the same side as the vertical line containing p_i. We call the sets C_{p_i}* clusters.

For each p_i, every a-cap in the pre-image of C_{p_i} corresponds to a *left convex chain* of size a in C_{p_i} and every u-cup in the pre-image of C_{p_i} corresponds to a *right convex chain* of size u in C_{p_i}. We note that the constructions could be stated without the rotation by $\pi/2$, but this version will be more natural in later constructions. In the following result, we state how (X, Y)-blows-ups of arbitrary point sets affect the size of the largest subset in convex position in the resulting point sets and also describe how large these point sets are.

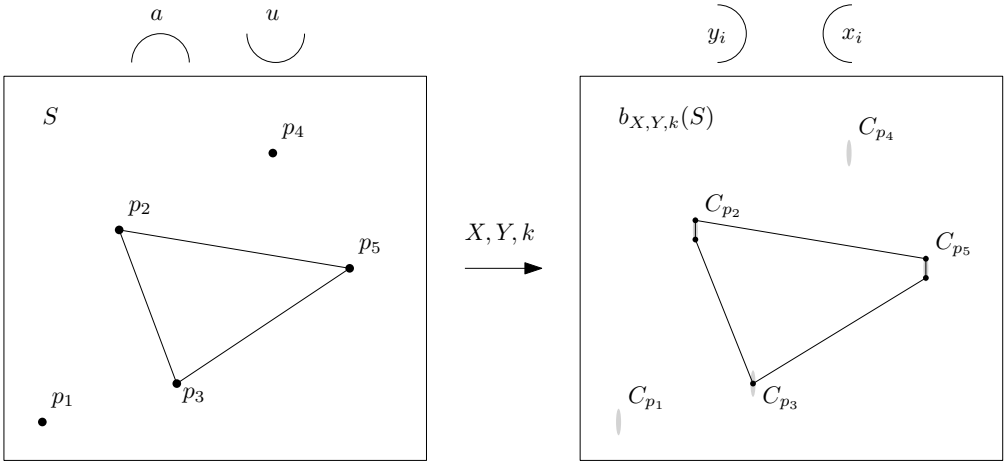


Figure 3 An illustration of an (X, Y)-blow-up of a point set S and its subset in convex position.

▶ **Lemma 14** (General blow-up construction). *For $k, N \in \mathbb{N}$, and a set S of N points in the plane in general position, let $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_N)$ be sequences of non-negative integers satisfying*

(a) *$x_i + y_i \leq k - 1$ for every $i \in [N]$ and*

(b) *$x_i + y_j \leq k - 1 - s_{i,j}$ for all $i, j \in [N]$ with $i < j$.*

Then, the (X, Y)-blow-up $b_{X,Y,k}(S)$ of S does not contain k points in convex position and

$$|b_{X,Y,k}(S)| = \sum_{l=0}^{x_1} \binom{k-2}{l} + \sum_{i=2}^{N-1} \binom{x_i + y_i}{x_i} + \sum_{l=0}^{y_N} \binom{k-2}{l}.$$

The proof of Lemma 14 is omitted. Observe that it follows from the second condition that the factor k is larger than the size of the largest convex subset of S as the numbers x_i and y_i must be non-negative.

6.1 All known constructions

We now show that for each set S and factor $k = |S| + 1$ there are sequences X and Y such that $|b_{X,Y,k}(S)| = 2^{k-2}$ with no k points in convex position. It will follow that all known constructions of sets of 2^{k-2} points in general position with no k points in convex position are a special case of an (X, Y)-blow-up. We recall that all such known constructions fall within Valtr's construction. Now, let S be an arbitrary set of $N \leq k - 1$ points in general position. We set $x_i = i - 1$, and $y_i = k - i - 1$ for every $i \in [N]$. Then, clearly, $x_i + y_i = k - 2 \leq k - 1$ and $x_i, y_i \geq 0$ for every $i \in [N]$ as $|S| = N \leq k - 1$. We have $x_i + y_j \leq k - 1 - s_{i,j}$ for all $i, j \in [N]$ with $i < j$, as $s_{i,j} \leq j - i + 1$ and $(i-1)+(k-j-1) = k-1-(j-i+1) \leq k-1-s_{i,j}$. Thus, all assumptions in Lemma 14 are met, and $b_{X,Y,k}(S)$ does not contain k points in convex position. If $N = k - 1$, then we also get

$$\begin{aligned} |b_{X,Y,k}(S)| &= \sum_{l=0}^{0} \binom{k-2}{l} + \sum_{i=2}^{N-1} \binom{x_i + y_i}{x_i} + \sum_{l=0}^{0} \binom{k-2}{l} \\ &= 1 + \sum_{i=2}^{k-2} \binom{k-2}{i-1} + 1 = 2^{k-2} \end{aligned}$$

and $b_{X,Y,k}(S)$ corresponds to Valtr's construction rotated by $\pi/2$.

6.2 New construction

The previous choices of X and Y are not the only ones that work, we describe another choice of an (X, Y)-blow-up to obtain a set of 2^{k-2} points with no k-tuple in convex position for any $k \geq 3$. This time, the sequence X will not be increasing, and we allow $k < |S|$.

We now proceed with the construction. For each point p_i, each subset of S in convex position with p_i as its rightmost point (leftmost point) is called a *left-gon at* p_i (*right-gon at* p_i, respectively). For a *threshold* $L \in \{0, 1, \ldots, N\}$, we call points p_i with $i \in [L]$ *left* and points p_j with $j \in [N] \setminus [L]$ *right*. For each left (right, respectively) point p_i, we define s_i as the maximum size of left-gon at p_i (right-gon at p_i, respectively). Note that if $L \in [N-1]$ there are exactly two points p_i of S with $s_i = 1$, namely, the leftmost and the rightmost point of S. For every positive integer j, let $v_{j,L}(S)$ be the number of points p_i of S with $s_i = j$.

If $x_i = x$ and $y_i = k - 1 - x - s_i$ for every $i \in [L]$ and $x_i = k - 1 - x - s_i$ and $y_i = x$ for each $i \in [N] \setminus [L]$, then we call the set $b_{X,Y,k}(S)$ an *x-blow-up of S with respect to threshold L and factor k* and denote it by $b_{x,k,L}(S)$. We now show that the size of the x-blow-up of S can be expressed in terms of the numbers $v_{j,L}(S)$.

▶ **Corollary 15.** *For a set S of N points in the plane in general position, let x be a non-negative integer and choose $m = k - 2x$ for some threshold $L \in [N-1]$ and factor $k \geq 1$. If S does not contain m points in convex position, then the x-blow-up $b_{x,k,L}(S)$ of S does not contain k points in convex position and*

$$|b_{x,k,L}(S)| = 2\sum_{l=0}^{x} \binom{k-2}{l} + \sum_{j=2}^{k-2x-1} v_{j,L}(S) \cdot \binom{k-j-1}{x}.$$

The proof of Corollary 15 is omitted. It follows from Corollary 15 that in order to produce large x-blow-ups without k points in convex position, point sets S with suitable values $v_{j,L}(S)$ for some threshold L are useful. We now provide a simple formula for the values $v_{j,N}(P)$ in the Erdős–Szekeres construction $P = P(k, k, k)$ where all points are labeled as left.

▶ **Lemma 16.** *For every integer $k \geq 2$, if we set the threshold $L = 2^{k-2}$, then, for $P = P(k, k, k)$, we have*

$$v_{j,L}(P) = \begin{cases} 1, & \text{for } j = 1, \\ 2^{j-2}, & \text{for } j \in \{2, \ldots, k-1\}, \\ 0, & \text{otherwise.} \end{cases}$$

The proof of Lemma 16 is omitted. Analogously, we get the same values $v_{j,0}(P)$ if all points of $P(k, k, k)$ are labeled as right, that is, if we set $L = 0$.

We now construct a point set M with $|M| = 2^{m-2}$ and without m points in convex position that has larger x-blow-ups than $P = P(m, m, m)$.

▶ **Definition 17** (Point set M)**.** *We construct the following set M that contains $2m-2$ points $\ell_1, \ldots, \ell_{m-1}, , r_1, \ldots, r_{m-1}$ in the plane in general position. We let $\ell_1 = (-m, 0)$, $r_1 = (m, \varepsilon)$, $\ell_2 = (-m+1, 1)$, and $r_2 = (m-1, 1+\varepsilon)$ for some small $\varepsilon > 0$. We now specify the placement of the remaining points $\ell_3, \ldots, \ell_{m-1}$ and $r_3, \ldots, r_{m-1}$ inductively; see Figure 4. Assume that we have placed points $\ell_1, \ldots, \ell_i$ and $r_1, \ldots, r_i$ for some $i \geq 2$. We now let ℓ_{i+1} be a point above the lines $\overline{\ell_{i-1}\ell_i}$ and $\overline{r_{i-1}r_i}$ to the right of ℓ_{i-1} and to the left of the line $x = 0$ and we let r_{i+1} be a point above the lines $\overline{\ell_{i-1}\ell_i}$ and $\overline{r_{i-1}r_i}$ to the left of r_i and to the right of the line $x = 0$ so that $y(r_{i+1}) > y(\ell_{i+1})$.*

We let P' be a point set obtained by replacing each point ℓ_i with $i \in [m-1]$ with a copy of the set $P(m-i+1, i)$ that lies in a sufficiently small neighborhood of ℓ_i. We denote this copy as C_{ℓ_i}. Note that $C_{\ell_1} = \emptyset$. We then let P'' be a point set obtained by replacing each point r_i with $i \in [m-1]$ with a copy of the set $P(m-i, i+1)$ that lies in a sufficiently small neighborhood of r_i. We use C_{r_i} to denote this copy and note that $C_{r_{m-1}} = \emptyset$. Note that the point sets P' and P'' then form affine copies of the point set $P(m-1, m-1, m-1)$. Let $M = P' \cup P''$ and note that $|M| = 2^{m-2}$.

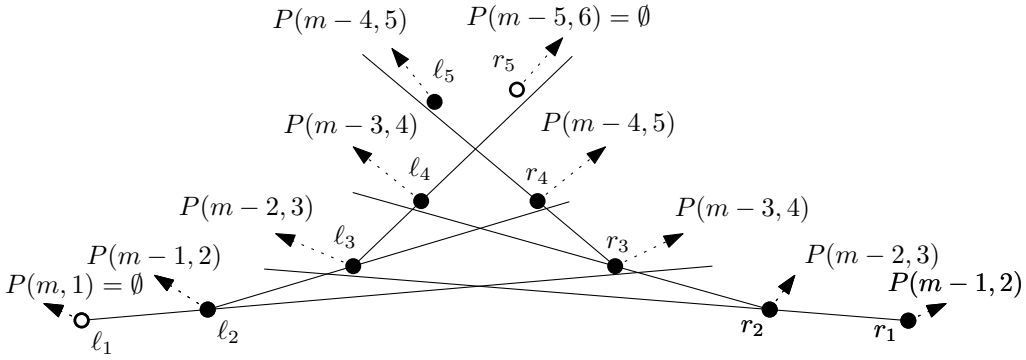


Figure 4 Example of the construction of the point set M for $m = 6$.

Now, if we set the threshold $L = 2^{m-3}$, then the left points of M are the points from P', and the right points of M are the points of P''. We also obtain the following formulas.

▶ **Lemma 18.** *For every integer $m \geq 3$, the set M does not contain m points in convex position and, for the threshold $L = 2^{m-3}$, satisfies*

$$v_{j,L}(M) = \begin{cases} 2, & \text{for } j = 1, \\ 2^{j-1}, & \text{for } j \in \{2, \dots, m-2\}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The proof of Lemma 18 is omitted. We now prove that suitable x-blow-ups of M produce new sets of 2^{k-2} points in the plane in general position that do not contain k points in convex position, obtaining a new construction of such point sets.

▶ **Theorem 19.** *For each positive integer x and $k \geq 2x+3$, let $m = k - 2x$. The x-blow-up $b_{x,k,L}(M)$ of the point set M with respect to the threshold $L = 2^{m-3}$ does not contain k points in convex position and $|b_{x,k,L}(M)| = 2^{k-2}$.*

Proof. By applying the x-blow-up from Corollary 15 with parameters x, k, and $L = 2^{m-3}$ to M, we obtain a set $b_{x,k,L}(M)$ of

$$|b_{x,k,L}(M)| = \sum_{l=0}^{x} \binom{k-2}{l} + \sum_{j=2}^{k-2x-2} 2^{j-1} \cdot \binom{k-j-1}{x} + \sum_{l=k-x-2}^{k-2} \binom{k-2}{l}$$

points in the plane in general position that contains no k points in convex position.

To show that $|b_{x,k,L}(M)|$ equals 2^{k-2} it suffices to prove that

$$\sum_{l=x+1}^{k-x-3} \binom{k-2}{l} = \sum_{j=2}^{k-2x-2} 2^{j-1} \cdot \binom{k-j-1}{x}$$

if $k \geq 2x+3$, as then $|b_{x,k,L}(S)| = 2^{k-2}$ follows by the binomial theorem. This equality is true because both sides count the number of subsets of $[k-2]$ of size at least $x+1$ and at most $k-x-3$. This is trivial for the left-hand side. To prove the right-hand side, let $\mathcal{F}$ be the set of subsets of $[k-2]$ of size at least $x+1$ and at most $k-x-3$. For $F \in \mathcal{F}$, we let j be the minimum element from $[k-2]$ such that either F or its complement contains exactly $x+1$ elements in $\{j-1,\ldots,k-2\}$. Note that such j is uniquely determined for each $F \in \mathcal{F}$ and that $j \in \{2,\ldots,k-2x-2\}$. On the other hand, we can uniquely determine set F from $\mathcal{F}$ by selecting $j-1$, deciding whether F or its complement has exactly $x+1$ elements in $\{j-1,\ldots,k-2\}$ and then select the remaining x of them from $\{j,\ldots,k-2\}$ to F (or to its complement, respectively) and, finally, select an arbitrary subset of $[j-2]$ to F (or its complement, respectively). ◀

6.3 Computational experiments

Since there are other possible choices for X and Y, one might wonder whether some of them yield constructions of more than 2^{k-2} points without k points in convex position. We implemented a simple program in *Matlab* based on a non-linear optimization solver to find as large (X,Y)-blow-up of a given point set as possible. Given a set of N points in the plane in general position and a factor k, this solver tries to find a solution $(x_1,\ldots,x_N,y_1,\ldots,y_N)$ that satisfies the linear constraints from Lemma 14 and that has as large value of the function

$$\sum_{l=0}^{x_1}\binom{k-2}{l} + \sum_{i=2}^{N-1}\binom{x_i+y_i}{x_i} + \sum_{l=0}^{y_N}\binom{k-2}{l}$$

as possible. Since the objective function is not linear, we used a solver based on a genetic algorithm that does not guarantee finding a true optimum solution. We tried various point sets on the input, including subsets of $P(a,u)$ and random point sets. Using this program, we obtained many examples of 2^{k-2} points for various choices of S and k up to 20. Even though there is some freedom in choosing X and Y, the size of the (X,Y)-blow-up never exceeded 2^{k-2}. So the Erdős–Szekeres conjecture remains open.

References

1 J. Baek. On the Erdős-Tuza-Valtr conjecture. *European J. Combin.*, 124:Paper No. 104085, 15, 2025. doi:10.1016/j.ejc.2024.104085.

2 M. Balko. Ramsey numbers and monotone colorings. *J. Combin. Theory Ser. A*, 163:34–58, 2019. doi:10.1016/j.jcta.2018.11.013.

3 M. Balko, J. Cibulka, K. Král, and J. Kynčl. Ramsey numbers of ordered graphs. *Electron. J. Combin.*, 27(1), 2020. doi:10.37236/7816.

4 M. Balko and P. Valtr. A SAT attack on the Erdős–Szekeres conjecture. *European Journal of Combinatorics*, 66:13–23, 2017. doi:10.1016/j.ejc.2017.06.010.

5 F. R. K. Chung and R. L. Graham. Forced convex n-gons in the plane. *Discrete Comput. Geom.*, 19(3):367–371, 1998. doi:10.1007/PL00009353.

6 D. Conlon, J. Fox, C. Lee, and B. Sudakov. Ordered Ramsey numbers. *J. Combin. Theory Ser. B*, 122:353–383, 2017. doi:10.1016/J.JCTB.2016.06.007.

7 P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compos. Math.*, 2:463–470, 1935.

8 P. Erdős and G. Szekeres. On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 3–4:53–62, 1960–61.

9 P. Erdős, Z. Tuza, and P. Valtr. Ramsey-remainder. *European J. Combin.*, 17(6):519–532, 1996. doi:10.1006/EUJC.1996.0045.

10 S. Felsner and H. Weil. Sweeps, arrangements and signotopes. *Discrete Appl. Math.*, 109(1–2):67–94, 2001. doi:10.1016/S0166-218X(00)00232-8.

11 J. Fox, J. Pach, B. Sudakov, and A. Suk. Erdős–Szekeres-type theorems for monotone paths and convex bodies. *Proc. London Math. Soc.*, 105(5):953–982, 2012.

12 J. E. Goodman and R. Pollack. A combinatorial perspective on some problems in geometry. In *Proceedings of the Twelfth Southeastern Conference on Combinatorics, Graph Theory and Computing, Vol. I*, volume 32 of *Congressus Numerantium*, pages 383–394, 1981.

13 A. F. Holmsen, H. N. Mojarrad, J. Pach, and G. Tardos. Two extensions of the Erdős–Szekeres problem. *Journal of the European Mathematical Society*, 22:3981–3995, 2020. doi:10.4171/JEMS/1000.

14 D. Kleitman and L. Pachter. Finding convex sets among points in the plane. *Discrete Comput. Geom.*, 19(3):405–410, 1998. doi:10.1007/PL00009358.

15 J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

16 H. N. Mojarrad and G. Vlachos. An improved upper bound for the Erdős-Szekeres conjecture. *Discrete Comput. Geom.*, 56(1):165–180, 2016. doi:10.1007/S00454-016-9791-5.

17 G. Moshkovitz and A. Shapira. Ramsey theory, integer partitions and a new proof of the Erdős-Szekeres theorem. *Adv. Math.*, 262:1107–1129, 2014.

18 S. Norin and Y. Yuditsky. Erdős-Szekeres without induction. *Discrete Comput. Geom.*, 55(4):963–971, 2016. doi:10.1007/S00454-016-9778-2.

19 L. Peters and G. Szekeres. Computer solution to the 17-point Erdős–Szekeres problem. *ANZIAM J.*, 48(2):151–164, 2006.

20 A. Suk. On the Erdős-Szekeres convex polygon problem. *J. Amer. Math. Soc.*, 30(4):1047–1053, 2017.

21 G. Tóth and P. Valtr. Note on the Erdős–Szekeres theorem. *Discrete Comput. Geom.*, 19(3):457–459, 1998.

22 G. Tóth and P. Valtr. The Erdős–Szekeres theorem: upper bounds and related results. In *Combinatorial and computational geometry*, volume 52 of *Math. Sci. Res. Inst. Publ.*, pages 557–568, Cambridge, 2005. Cambridge University Press.

23 P. Valtr. Convex independent sets and 7-holes in restricted planar point sets. *Discrete & Computational Geometry*, 7(2):135–152, 1992. doi:10.1007/BF02187831.

24 P. Valtr. Private communication, 2024.

Extremal Betti Numbers and Persistence in Flag Complexes

Lies Beers
Vrije Universiteit Amsterdam, The Netherlands

Magnus Bakke Botnan
Vrije Universiteit Amsterdam, The Netherlands

Abstract

We investigate several problems concerning extremal Betti numbers and persistence in filtrations of flag complexes. For graphs on n vertices, we show that $\beta_k(X(G))$ is maximal when $G = \mathcal{T}_{n,k+1}$, the Turán graph on $k+1$ partition classes, where $X(G)$ denotes the flag complex of G. Building on this, we construct an edgewise (one edge at a time) filtration $\mathcal{G} = G_1 \subseteq \cdots \subseteq \mathcal{T}_{n,k+1}$ for which $\beta_k(X(G_i))$ is maximal for all graphs on n vertices and i edges. Moreover, the persistence barcode $\mathcal{B}_k(X(G))$ achieves a maximal number of intervals, and total persistence, among all edgewise filtrations with $|E(\mathcal{T}_{n,k+1})|$ edges.

For $k = 1$, we consider edgewise filtrations of the complete graph K_n. We show that the maximal number of intervals in the persistence barcode is obtained precisely when $G_{\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor} = \mathcal{T}_{n,2}$. Among such filtrations, we characterize those achieving maximal total persistence. We further show that no filtration can optimize $\beta_1(X(G_i))$ for all i, and conjecture that our filtrations maximize the total persistence over all edgewise filtrations of K_n.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological data analysis, Extremal graph theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.14

Related Version *Full Version*: https://arxiv.org/abs/2502.21294 [3]

Acknowledgements We thank Ulrich Bauer for suggesting the problem of maximizing the number of intervals in the barcode of a Vietoris–Rips filtration.

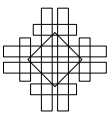
1 Introduction

A central theme in topological data analysis (TDA) is the computation of homological invariants from simplicial complexes. These complexes are often constructed from point cloud data, with the *Vietoris–Rips* complex being one of the most widely used constructions. For a finite set P in a metric space (M, d), the Vietoris–Rips complex at scale r, denoted $\mathrm{VR}_r(P)$, includes all simplices $\sigma = \{p_0, \ldots, p_n\}$ such that $d(p_i, p_j) \leq 2r$ for all $0 \leq i, j \leq n$. This complex is a *flag complex*, meaning it is the largest simplicial complex with a given underlying graph (the 1-skeleton). In particular, the edges in the connectivity graph of P at scale r fully determine the higher-dimensional simplices in $\mathrm{VR}_r(P)$. Varying the scale parameter r induces a filtration of simplicial complexes:

$$\mathrm{VR}(P)_{r_0} \hookrightarrow \mathrm{VR}(P)_{r_1} \hookrightarrow \cdots \hookrightarrow \mathrm{VR}(P)_{r_m},$$

where $r_0 < r_1 < \cdots < r_m$. Applying k-dimensional homology over a field $\mathbf{k}$ to this filtration yields a persistence barcode in degree k, denoted $\mathcal{B}_k(\mathrm{VR}(P))$. This barcode, consisting of intervals $[a, b)$, encodes the birth and death of topological features across scales and serves as a powerful tool for extracting topological information from P. When P is a sufficiently dense sampling of an underlying space X, the number of "long bars" in $\mathcal{B}_k(\mathrm{VR}(P))$ determines the k-th Betti number $\beta_k(X)$, which describes the k-dimensional topological features of X [7]. For a comprehensive introduction to topological data analysis, we refer the reader to [8].

© Lies Beers and Magnus Bakke Botnan;
licensed under Creative Commons License CC-BY 4.0



Given the central role of such filtrations in TDA, this paper addresses a fundamental question: how many topological features can arise in a data set of n points? Specifically, we investigate bounds on quantities such as the maximal value of $\beta_k(\mathrm{VR}_r(P))$, the maximal number of intervals in $\mathcal{B}_k(\mathrm{VR}(P))$, the length of the longest interval, and the sum of the lengths of the intervals (the total persistence).

▶ Remark 1. In this paper, we focus on edgewise filtrations of graphs, i.e., sequences of graphs $\mathcal{G} = G_1 \subseteq G_2 \subseteq \cdots \subseteq G_m = \{G_i\}_{i=1}^m$, on n vertices, where G_{i+1} and G_i differ by precisely one edge. Notably, every filtration of flag complexes can be realized as the Vietoris–Rips filtration of an appropriate metric on the vertices of G_m; see the full version of this paper [3, A]. Thus, our results are directly applicable to data analysis using persistent homology.

1.1 Overview and contributions

In Section 2, we introduce the relevant background material and compute the Betti numbers of flag complexes on Turán graphs.

In Section 3, we examine extremal Betti numbers and establish tight upper bounds on $\beta_k(F)$, where F is a flag complex on n vertices (Theorem 10). This upper bound is achieved by the Turán graph $\mathcal{T}_{n,k+1}$.

In Section 4, we study filtrations of flag complexes on n vertices with at most e edges, where e is the number of edges in $\mathcal{T}_{n,k+1}$. We prove that our filtration maximizes β_k at all filtration steps (Theorem 16).

In Section 5, we analyze the longest possible interval in the barcode of a filtration of flag complexes on n vertices (Corollary 21).

Additionally, in Section 6, we focus on degree 1 homology and show that any filtration of flag complexes on n vertices will achieve the maximal number of intervals in its degree 1 barcode if and only if the filtration contains $\mathcal{T}_{n,2}$ (Corollary 25).

In Section 7, we explore flag filtrations in homology degree 1 that achieve both a maximal number of intervals and maximal total persistence. Equivalently, we maximize the total persistence of all filtrations of graphs containing $G_{\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor} = \mathcal{T}_{n,2}$. Given the importance of Turán graphs in extremal graph theory, we believe that identifying extremal filtrations of Turán graphs is interesting in its own right. Our result relies on an elaborate combinatorial analysis that precisely classifies the extremal filtrations (Theorem 31).

Finally, the paper concludes with a discussion in which we outline several important conjectures for future work.

1.2 Related work

The study of extremal values of $\mathbb{Z}$-linear functions on the f- and β-vectors of simplicial complexes with n vertices has a rich history. Classical questions include determining the extremal Euler characteristic and the maximal sum of Betti numbers. These problems were addressed for general simplicial complexes in [4] and later extended to arbitrary $\mathbb{Z}$-linear functions in [11]. Specifically, [11, Theorem 3.4] shows that for flag complexes, extremal values are always realized by $\mathcal{T}_{n,k}$ for some k. Consequently, Theorem 10 follows directly from [11, Theorem 3.4].

Our proof of Theorem 10 adapts the approach of [1, Theorem 1.1], which gives an alternative proof for the extremal total Betti number of flag complexes. The key observation in our proof of Theorem 10 plays a central role in our results on extremal interval lengths (Section 5).

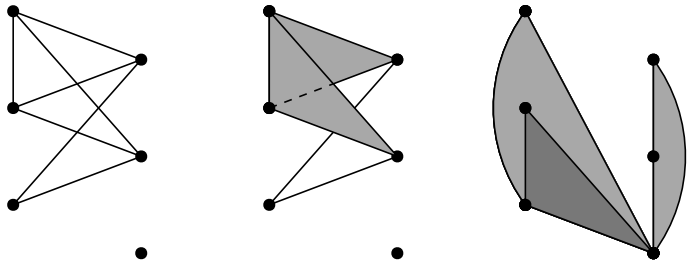


Figure 1 A graph (left), its flag complex (middle) and independence complex (right).

Another related work is [10], which examines asymptotic bounds for β_k in Vietoris–Rips complexes of point samples in $\mathbb{R}^d$. Similar questions have been explored for Čech complexes; see, e.g., [9].

Our work diverges from earlier work by considering *filtered* flag complexes. This approach is closely tied to the problem of finding extremal values for complexes with precisely n vertices and e edges; we return to this in Section 8. Importantly, and in contrast to classical work, graphs achieving extremal values need not have $\mathcal{T}_{n,k}$ as a subgraph.

2 Background

Graphs and Simplicial Complexes. Let $G = (V, E)$ be a graph. We write $\{v, w\}$ for the edge connecting the vertices v and w. For a vertex $v \in V$, the *neighborhood* is $N_G(v) \coloneqq \{w \colon \{v, w\} \in E\}$, and the *closed neighborhood* is $N_G[v] \coloneqq N_G(v) \cup \{v\}$. The *degree* of v is $\deg(v) \coloneqq \deg_G(v) \coloneqq |N_G(v)|$. The complete graph on n vertices is denoted K_n.

The *complement* of G, denoted $\overline{G}$, is the graph on the same vertex set as G, where two distinct vertices of $\overline{G}$ are adjacent if and only if they are non-adjacent in G. The *join* of disjoint graphs G_1 and G_2 is the graph $G_1 \vee G_2$ with vertex set $V(G_1) \sqcup V(G_2)$ and edge set $E(G_1) \cup E(G_2) \cup \big\{\{v_1, v_2\} \colon v_1 \in V(G_1), v_2 \in V(G_2)\big\}$. The *complete bipartite graph* K_{n_1,n_2} is defined as the join $G_1 \vee G_2$, where each G_i is an empty graph on n_i vertices. The sets $V(G_1)$ and $V(G_2)$ are called the *partition classes* of K_{n_1,n_2}.

For a vertex v in a simplicial complex K, the *link* and *(closed) star* of v are the simplicial complexes given respectively by

$$\mathrm{lk}_K(v) = \{\tau \in K : v \notin \tau, \{v\} \cup \tau \in K\} \qquad \mathrm{st}_K(v) = \{\tau \in K : \{v\} \cup \tau \in K\}.$$

We write $K - v$ for the simplicial complex with simplices $\tau \in K$ for which $v \notin K$.

For a graph G, we let $X(G)$ denote the simplicial complex with m-simplices given by the $(m+1)$-cliques in G. A simplicial complex K is called a *flag complex* if $K = X(G)$ for some graph G. The *independence complex* of G is the simplicial complex $\mathrm{Ind}(G) = X(\overline{G})$. Examples of flag and independence complexes are given in Figure 1.

Homology. For a simplicial complex K, we let $\beta_k(K)$ denote the dimension of the reduced homology group $\tilde{H}_k(K; \mathbf{k})$, and let $C_k(K)$ denote the vector space of k-chains. We use coefficients in a fixed, but arbitrary, field $\mathbf{k}$; our optimal constructions are torsion-free and thus our results do not depend on the choice of coefficient field. For that reason, we simply write $\tilde{H}_k(K)$. We also employ the notation $\beta_k^{\mathrm{FL}}(G) = \beta_k(X(G))$ for a graph G.

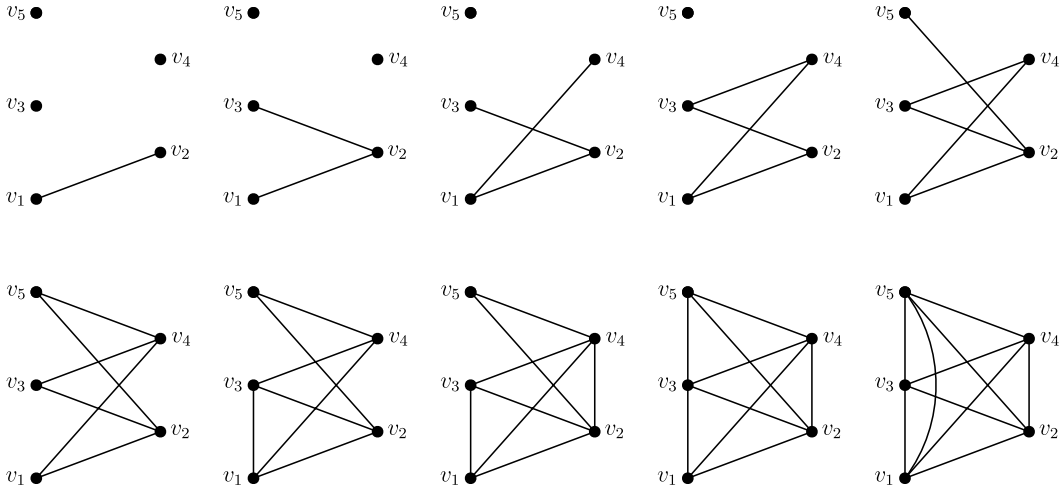


Figure 2 An edgewise filtration of K_5.

Persistent Homology. A filtration $\mathcal{K}$ of simplicial complexes is a collection of simplicial complexes $\{K_i\}_{i=1}^m$ such that $K_i \subseteq K_j$ for $i \leq j$. Applying $\tilde{H}_k(-;\mathbf{k})$ to a filtration yields a sequence of vector spaces and linear maps $\tilde{H}_k(\mathcal{K}) : \tilde{H}_k(K_1) \to \cdots \to \tilde{H}_k(K_m)$ called a *persistence module*. Provided all the vector spaces are finite-dimensional, $\tilde{H}_k(\mathcal{K})$ is uniquely described by a collection of intervals in $\{1,\ldots,m\}$, called the *degree k barcode of $\mathcal{K}$*. We shall denote this barcode by $\mathcal{B}_k(\mathcal{K})$. The *total persistence* of $\mathcal{K}$ is given by

$$T\beta_k(\mathcal{K}) = \sum_{[a,b)\in\mathcal{B}_k(\mathcal{K})} \big(b-a\big) = \sum_{i=1}^{m} \beta_k(K_i).$$

For a more thorough introduction to persistent homology, (generalized) persistence modules, and examples, see, e.g., [8, Chapter 3] or [5].

For a filtration $\mathcal{G}$ of graphs (1-dimensional simplicial complexes), we get a filtration $X(\mathcal{G})$ of simplicial complexes by taking the flag complex at every index. If $G_{i+1} - G_i$ is a single edge for all i, then we say that the filtration $\mathcal{G}$ is *edgewise*. We shall employ the notation $\mathcal{B}_k^{\mathrm{FL}}(G) = \mathcal{B}_k(X(\mathcal{G}))$.

▶ **Example 2.** An edgewise filtration of K_5 can be found in Figure 2.

2.1 Elementary homological properties

The following two lemmas are well-known and important in combinatorial topology. For completeness, their proofs can be found in the full version [3, B.1].

▶ **Lemma 3.** *Let K be a simplicial complex and $v \in V(K)$ a vertex. Then, for all $k \geq 1$,*

$$\beta_k(K) \leq \beta_k(K - v) + \beta_{k-1}(\mathrm{lk}_K(v)).$$

▶ **Lemma 4.** *For any two graphs G and H, and $k \geq -1$,*

$$\beta_k(\mathrm{Ind}(G \sqcup H)) = \sum_{i,j\geq -1; i+j=k-1} \beta_i(\mathrm{Ind}(G))\beta_j(\mathrm{Ind}(H)).$$

The following observation is essential for our work in Section 7.

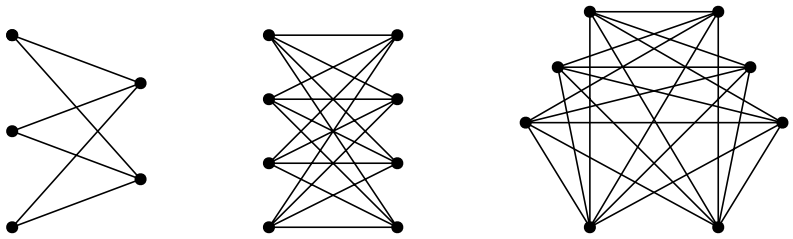


Figure 3 From left to right: the Turán graphs $\mathcal{T}_{5,2}$, $\mathcal{T}_{8,2}$ and $\mathcal{T}_{8,3}$.

▶ **Proposition 5.** *Let $V(G) = V_1 \sqcup V_2$ be the vertex set of a graph G containing all edges of the form $\{v, w\}$ where $v \in V_1$ and $w \in V_2$. Let $d_i \geq 1$ denote the number of connected components of G restricted to V_i. Then, $\beta_1^{\mathrm{FL}}(G) = \beta_0^{\mathrm{FL}}(G_1)\beta_0^{\mathrm{FL}}(G_2) = (d_1 - 1)(d_2 - 1)$.*

Proof. Let $\overline{G}$ denote the complement graph of G and observe that $\overline{G} = \overline{G}_1 \sqcup \overline{G}_2$ where G_1 and G_2 are the full subgraphs of G with vertices V_1 and V_2, respectively. From Lemma 4,

$$\beta_1^{\mathrm{FL}}(G) = \beta_1(\mathrm{Ind}(\overline{G})) = \beta_0(\mathrm{Ind}(\overline{G}_1))\beta_0(\mathrm{Ind}(\overline{G}_2)) = \beta_0(X(G_1))\beta_0(X(G_2)).$$ ◀

2.2 Turán graphs

▶ **Definition 6.** *Let $n \geq 0$ and $k \geq 1$, and let n' be the smallest positive integer such that $n' \equiv n \bmod k$. Then, the (n, k)* graph $\mathcal{T}_{n,k}$ *is the complement graph of the graph*

$$\overline{\mathcal{T}_{n,k}} = \bigsqcup_{i=1}^{k} K_{n_i}, \qquad n_i = \begin{cases} \lceil n/k \rceil & \text{if } 1 \leq i \leq n', \\ \lfloor n/k \rfloor & \text{otherwise.} \end{cases}$$

▶ **Example 7.** See Figure 3 for some examples of Turán graphs.

The proof of the following can be found in the full version [3, B.1].

▶ **Proposition 8.** *For all integers $n \geq 1$ and $k \geq 1$, we let n' be the smallest positive integer such that $n' \equiv n \bmod k$. We have*

$$\beta_i^{\mathrm{FL}}(\mathcal{T}_{n,k}) = \begin{cases} (\lceil n/k \rceil - 1)^{n'} \cdot (\lfloor n/k \rfloor - 1)^{k-n'} & \text{if } i = k - 1, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, if n is a multiple of k, then $\beta_{k-1} = (n/k - 1)^k$.

3 The maximum value of $\beta_k(X(G))$

The following is well-known; see the full version [3, B.2] for a proof.

▶ **Lemma 9.** *Let S be a positive integer, and $x_i \geq 1$ be integers satisfying $\sum_{i=1}^{n} x_i = S$. Then, $\prod_{i=1}^{n}(x_i - 1) \leq \prod_{i=1}^{n}(y_i - 1)$ where*

$$y_i = \begin{cases} \lceil S/n \rceil & \text{if } 1 \leq i \leq S \bmod n \\ \lfloor S/n \rfloor & \text{otherwise.} \end{cases} \tag{1}$$

As said in Section 1.2, the following proof is a modification of the proof of [1, Theorem 1.1].

▶ **Theorem 10.** *Let G be a graph on n vertices. Then, for all $k \geq 0$,*

$$\beta_k^{\mathrm{FL}}(G) \leq (\lceil n/(k+1) \rceil - 1)^{(n \bmod k+1)} \cdot (\lfloor n/(k+1) \rfloor - 1)^{(k+1)-(n \bmod k+1)}.$$

Proof. We shall work inductively on n. The result is trivially true for $n = 1$, so assume that it holds for all $n' < n$.

Note that $X(G) = \mathrm{Ind}(\overline{G})$. Let d denote the minimal degree over all $v \in V(\overline{G})$. Assume that v is a vertex with $\deg(v) = d$, and let $\{v_1, \ldots, v_d\}$ denote the neighbours of v in $\overline{G}$. Moreover, let $\overline{G}_i = \overline{G} - \{v_1, \ldots, v_i\}$ for $i = 1, \ldots, d$ and $\overline{G}_0 = \overline{G}$. Applying Lemma 3,

$$\begin{aligned}
\beta_k(\mathrm{Ind}(\overline{G})) &\leq \beta_k(\mathrm{Ind}(\overline{G}_1)) + \beta_{k-1}(\mathrm{Ind}(\overline{G} - N_{\overline{G}}[v_1]) \\
&\leq \beta_k(\mathrm{Ind}(\overline{G}_2)) + \beta_{k-1}(\mathrm{Ind}(\overline{G}_1 - N_{\overline{G}_1}[v_2])) + \beta_{k-1}(\mathrm{Ind}(\overline{G} - N_{\overline{G}}[v_1]))) \\
&\vdots \\
&\leq \beta_k(\mathrm{Ind}(\overline{G}_d)) + \sum_{i=0}^{d-1} \beta_{k-1}(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}]))
\end{aligned}$$

Here $\overline{G}_d$ has an isolated vertex and thus $\mathrm{Ind}(\overline{G}_d)$ is a cone, and therefore $\beta_k(\mathrm{Ind}(\overline{G}_d)) = 0$. Since every vertex of $\overline{G}$ has degree at least d, it follows that $\overline{G}_i - N_{\overline{G}_i}(v_{i+1})$ contains at most $n - (d+1) = n - d - 1$ vertices. By the induction hypothesis it follows that

$$\beta_k(\mathrm{Ind}(\overline{G})) \leq d \cdot (x_1 - 1) \cdots (x_k - 1) = ((d+1) - 1) \cdot (x_1 - 1) \cdots (x_k - 1)$$

for integers $x_i \geq 1$ for which $(d+1) + \sum_{i=1}^{k} x_i = d + 1 + n - d - 1 = n$. Hence, by Lemma 9,

$$\beta_k(\mathrm{Ind}(\overline{G})) \leq (\lceil n/(k+1) \rceil - 1)^{(n \bmod k+1)} \cdot (\lfloor n/(k+1) \rfloor - 1)^{(k+1)-(n \bmod k+1)}.$$ ◀

Let $\mathbf{G}(n) = \{G\colon \text{ graph on } n \text{ vertices}\}$. Combining Theorem 10 and Proposition 8, we have

▶ **Corollary 11.** *Of all graphs on n vertices, $\mathcal{T}_{n,k+1}$ maximizes the k-th Betti number. I.e.,*

$$\max_{G \in \mathbf{G}(n)} \beta_k^{\mathrm{FL}}(G) = \beta_k^{\mathrm{FL}}(\mathcal{T}_{n,k+1}).$$

In particular, if n is a multiple of k, then $\max_{G \in \mathbf{G}(n)} \beta_k^{\mathrm{FL}}(G) = (n/k - 1)^k$.

4 The maximum value of $\beta_k(X(G))$ for $|E(G)| \leq |E(\mathcal{T}_{n,k+1})|$

In this section, we fix $n, k \geq 1$, and the notation

$$e_{n,k+1} \coloneqq |E(\mathcal{T}_{n,k+1})| \text{ and } \Delta_{m-1,m}^{k+1} \coloneqq e_m^{k+1} - e_{m-1}^{k+1}. \tag{2}$$

The following result follows from combining Lemma 3 and Corollary 11, and will be integral in proving the main result of this section.

▶ **Corollary 12.** *Let v be a vertex of degree $d \geq 1$ in a graph G. Then,*

$$\beta_k^{\mathrm{FL}}(G) \leq \beta_k^{\mathrm{FL}}(G - v) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}).$$

Proof. If $K = X(G)$, then $K - v = X(G - v)$, and Lemma 3 implies that

$$\beta_k^{\mathrm{FL}}(G) \leq \beta_k^{\mathrm{FL}}(G - v) + \beta_{k-1}(\mathrm{lk}_{X(G)}(v)).$$

Now observe that $\mathrm{lk}_{X(G)}(v) = X(N_G[v])$, and since $\mathrm{lk}_{X(G)}(v)$ has d vertices, it follows from Corollary 11 that $\beta_{k-1}(\mathrm{lk}_K(v)) \leq \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k})$. ◀

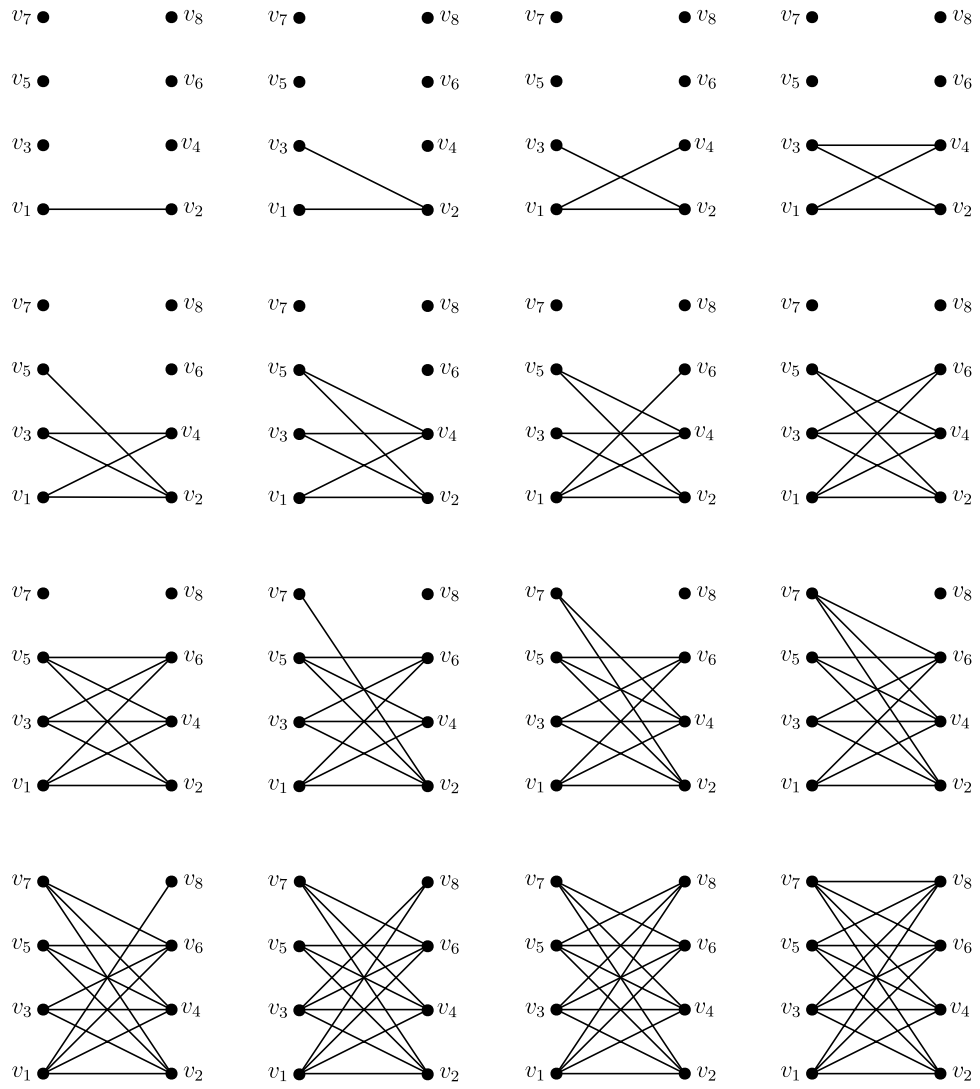


Figure 4 The filtration $\mathcal{H}$ of $\mathcal{T}_{8,2}$ from Definition 13.

We now define an edgewise filtration $\mathcal{H}^{n,k+1} = \mathcal{H} = \{H_i\}_{i=1}^{e_{n,k+1}} = \{H_i^{n,k+1}\}_{i=1}^{e_{n,k+1}}$ of the Turán graph $\mathcal{T}_{n,k+1}$. In this section, we shall show, for $m = 1, \ldots, |\mathcal{T}_{n,k+1}|$, that $\beta_k^{\mathrm{FL}}(H_m)$ is maximal over all graphs with m edges, i.e., that $\mathcal{H}$ is *fiberwise optimal*.

▶ **Definition 13.** *Let V denote the vertices of $\mathcal{T}_{n,k+1}$ and label the elements of V such that v_i and v_{i+k+1} are in the same partition. Writing each edge as $e_{i,j} = \{v_i, v_j\}$ for $i < j$, we order the edges by the following co-lexicographic order: $e_{i,j} < e_{k,l}$ if $\max\{i,j\} < \max\{k,l\}$, or if $i = k$ and $j < l$, or if $j = l$ and $i < k$. Following this order, let $H_i = H_i^{n,k+1}$ denote the subgraph of $\mathcal{T}_{n,k+1}$ with the first i edges.*

Note that, since $\mathcal{T}_{n,k+1}$ contains no $k+1$-simplices, $\beta_k(H_i)$ is increasing as a function of i.

▶ **Example 14.** See Figure 2 for an edgewise filtration $\mathcal{G} = \{G_i\}_{i=1}^{6}$ of K_5, where $G_i \cong H_i^{5,2}$ for $i = 1, \ldots, 6$. Furthermore, in Figure 4 one can find the filtration $\mathcal{H}^{8,2}$ of $\mathcal{T}_{8,2}$.

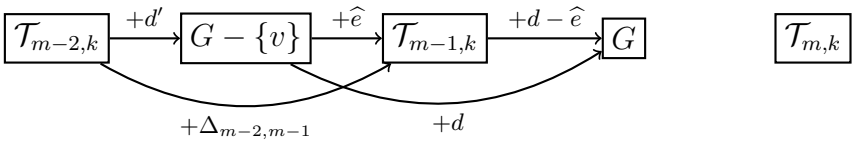


Figure 5 An illustration of Case 2 from the proof of Theorem 16.

The next lemma shows that, once a vertex has been connected to the growing component, and until the next vertex gets connected, the change in β_k^{FL} from adding a single edge increases with the number of edges already added. Its proof, similar to that of Corollary 12, can be found in the full version [3, B.3].

▶ **Lemma 15.** *For e edges, let m be maximal such that $\mathcal{T}_{m,k+1}$ is a subgraph of H_e. Then,*

$$\beta_k^{\mathrm{FL}}(H_e) = \beta_k^{\mathrm{FL}}(\mathcal{T}_{m,k+1}) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{e-e_m^{k+1},k}).$$

In particular, for $e_m^{k+1} \le e_1 < e_2 < e_{m+1}^{k+1}$, we have that

$$\beta_k^{\mathrm{FL}}(H_{e_1+1}) - \beta_k^{\mathrm{FL}}(H_{e_1}) \le \beta_k^{\mathrm{FL}}(H_{e_2+1}) - \beta_k^{\mathrm{FL}}(H_{e_2}).$$

▶ **Theorem 16.** *Let $e \le e_{n,k+1}$, let H_e be as above, and let G be any other graph on n vertices and e edges. Then, $\beta_k^{\mathrm{FL}}(G) \le \beta_k^{\mathrm{FL}}(H_e)$.*

Proof. We prove this inductively on the number of edges e. The statement is clearly true for $e = 1$, so let's assume that the result holds for all $e' < e$.

Let m be the number of vertices in H_e with positive degree, and let n' denote the number of vertices in G with positive degree. If $n' < m$, then by Theorem 10,

$$\beta_k^{\mathrm{FL}}(G) \le \beta_k^{\mathrm{FL}}(\mathcal{T}_{m-1,k+1}) \le \beta_k^{\mathrm{FL}}(H_e).$$

Hence, we may assume that $n' \ge m$. In particular, the average degree of the positive-degree-vertices in G is no larger than the average degree of positive-degree-vertices in H_e.

Choose a vertex v in G with minimal positive degree d, and observe that

$$d \le e_m^{k+1} - e_{m-1}^{k+1} = \Delta_{m-1,m},$$

since $\mathcal{T}_{m-1,k+1} \subsetneq H_e \subseteq \mathcal{T}_{m,k+1}$. In fact, if $d = \Delta_{m-1,m}$, then we must have that the average degree in H_e is at least d. Importantly, this happens if and only if $H_e = \mathcal{T}_{m,k+1}$ and m is a multiple of $k+1$.

By the induction assumption,

$$\beta_k^{\mathrm{FL}}(G - v) \le \beta_k^{\mathrm{FL}}(H_{e-\deg(v)}), \text{ and thus, by Corollary 12,}$$
$$\beta_k^{\mathrm{FL}}(G) \le \beta_k^{\mathrm{FL}}(H_{e-\deg(v)}) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}).$$

If $d = \Delta_{m-1,m}$, then this becomes, by Lemma 15,

$$\beta_k^{\mathrm{FL}}(G) \le \beta_k^{\mathrm{FL}}(\mathcal{T}_{m-1,k+1})) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}) = \beta_k^{\mathrm{FL}}(\mathcal{T}_{m,k+1}) = \beta_k^{\mathrm{FL}}(H_e).$$

We may therefore assume that $d \le \Delta_{m-1,m} - 1 \le \Delta_{m-2,m-1}$.

Let $\hat{e}$ denote the number of edges that is added to H_{e-d} before a new vertex gets positive degree in the filtration $\mathcal{H}$. Let d' be the degree of the last vertex in H_{e-d} that obtained positive degree in $\mathcal{H}$. We consider two cases.

- **Case 1:** $\mathcal{T}_{m-1,k} \subseteq H_{e-d}$**.** Then,

$$\begin{aligned}\beta_k^{\mathrm{FL}}(G) &\leq \beta_k^{\mathrm{FL}}(H_{e-d}) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}) \\ &= \beta_k^{\mathrm{FL}}(H_{e-d}) + \beta_k^{\mathrm{FL}}(H_{e_m^{k+1}+d}) - \beta_k^{\mathrm{FL}}(H_{e_m^{k+1}}) \\ &\leq \beta_k^{\mathrm{FL}}(H_{e-d}) + \beta_k^{\mathrm{FL}}(H_e) - \beta_k^{\mathrm{FL}}(H_{e-d}) \\ &= \beta_k^{\mathrm{FL}}(H_e).\end{aligned}$$

- **Case 2:** $\mathcal{T}_{m-2,k} \subseteq H_{e-d} \subset \mathcal{T}_{m-1,k}$**.** Note that $d' + \hat{e} = \Delta_{m-2,m-1} \geq d$ and $d - \hat{e} \geq 0$ (see Figure 5). In particular,

$$\begin{aligned}\beta_k^{\mathrm{FL}}(G) &\leq \beta_k^{\mathrm{FL}}(H_{e-d}) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}) \\ &= \beta_k^{\mathrm{FL}}(H_{e-d}) + \left(\beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d,k}) - \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d-\hat{e},k})\right) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d-\hat{e},k}) \\ &\leq \beta_k^{\mathrm{FL}}(H_{e-d}) + \left(\beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d'+\hat{e},k}) - \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d',k})\right) + \beta_{k-1}^{\mathrm{FL}}(\mathcal{T}_{d-\hat{e},k}) \\ &= \beta_k^{\mathrm{FL}}(H_e).\end{aligned}$$

In both cases, the inequality follows from Corollary 12 and Lemma 15. ◀

5 Tight bounds on the vanishing of homology and extremal interval lengths

For a graph G with n vertices, it is guaranteed that there exists a vertex of degree $n-1$ when the average degree exceeds $n-2$. Specifically, if the number of edges satisfies $m > \frac{n(n-2)}{2}$, then $X(G)$ must be a cone, implying that $\beta_k^{\mathrm{FL}}(G) = 0$ for all k. In practical scenarios, however, the primary interest is with $\beta_k^{\mathrm{FL}}(G)$ for small k. In this section, we provide tight bounds for the vanishing of $\beta_k^{\mathrm{FL}}(G)$ for a fixed k.

▶ **Lemma 17.** *Let G be a graph with minimum degree u. Let v be a vertex of degree u and let $N_G(v) = \{v_1, \ldots, v_u\}$. Let $d_i := \deg(v_i)$. If G has n vertices and m edges, then*

$$\left|V\left(G - N_G[v_i]\right)\right| = n - d_i - 1 \text{ and } \left|E\left(G - N_G[v_i]\right)\right| \leq m - \left(d_i + \left\lceil \frac{(u-1)d_i}{2} \right\rceil\right).$$

Proof. Let $\widehat{G} := G - N_G[v_i]$. The fact that $\left|V\left(\widehat{G}\right)\right| = n - d_i - 1$ is trivial. For showing the inequality, note that, by removing $N_G[v_i]$ from G, we remove the d_i edges containing v_i, and all edges containing the vertices of $N_G(v_i)$. Those vertices all have degree at least u, which means that they have at least $u-1$ neighbors apart from v_i. Because it might be the case that $N_G\left(N_G[v_i]\right) = N_G[v_i]$, it follows that

$$\left|E\left(\widehat{G}\right)\right| \leq m - \left(d_i + \left\lceil \frac{(u-1)d_i}{2} \right\rceil\right).$$ ◀

▶ **Theorem 18.** *Let G be a graph with n vertices and $m > \binom{n-1}{2} + k$ edges. Then $\beta_k^{\mathrm{FL}}(G) = 0$.*

Proof. Let us first consider the case $k = 0$. This case is immediate from the fact that the maximal number of edges in a graph on n vertices with an isolated vertex is $\binom{n-1}{2}$.

Working inductively on k, assume that result holds for all $k' < k$, and that $m > \binom{n-1}{2} + k$. Let $\overline{G}$ be the complement graph of G, and note that the number of edges $\overline{m} = |E(\overline{G})|$ satisfies

$$\overline{m} = \binom{n}{2} - m < \binom{n}{2} - \binom{n-1}{2} - k = n - 1 - k.$$

From the proof of Theorem 10, we have that

$$\beta_k^{\mathrm{FL}}(G) = \beta_k(\mathrm{Ind}(\overline{G})) \leq \sum_{i=0}^{d-1} \beta_{k-1}(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}])),$$

where $\overline{G}_i = \overline{G} - \{v_1, \ldots, v_i\}$, and $\{v_1, \ldots, v_u\}$ are the neighbors of a vertex $v \in V(\overline{G})$ with minimal degree u. We shall show that all the terms in the sum are zero.

If we let $u' \geq u - i$ denote the minimal degree of a vertex in $V(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}])$,

$$\begin{aligned}
&\left|E\big(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}]\big)\right| \\
&\overset{(i)}{\leq} |E(\overline{G}_i)| - \left(\deg_{\overline{G}_i}(v_{i+1}) + \left\lceil \frac{(u'-1)\deg_{\overline{G}_i}(v_{i+1})}{2} \right\rceil\right) \\
&\overset{(ii)}{\leq} |E(\overline{G}_i)| - \left(u - i + \left\lceil \frac{(u-i-1)(u-i)}{2} \right\rceil\right) = |E(\overline{G}_i)| - \sum_{j=1}^{u-i} j \\
&\overset{(iii)}{\leq} |E(\overline{G})| - \sum_{j=u-i+1}^{u} j - \sum_{j=1}^{u-i} j = |E(\overline{G})| - \sum_{j=1}^{u} j = \overline{m} - \left(\frac{u(u+1)}{2}\right).
\end{aligned}$$

Here, (i) follows from Lemma 17, (ii) from $\deg_{\overline{G}_i}(v_i) \geq u'$, and (iii) from the fact that $\overline{G}_i$ is obtained by removing i vertices from $\overline{G}$ with degree at least u in $\overline{G}$. Now write

$$n' = |V(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}]))| \leq n - u - 1,$$

and observe that

$$|E(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}]))| \geq \binom{n'}{2} - \overline{m} + u(u+1)/2.$$

It follows that,

$$\begin{aligned}
&|E(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}]))| - \binom{n'-1}{2} \\
&\geq \binom{n'}{2} - \overline{m} + u(u+1)/2 - \binom{n'-1}{2} \\
&= n' - 1 - \overline{m} + u(u+1)/2 > n' - 1 - (n-1-k) + u(u+1)/2 \geq k + u(u+1)/2 \geq k.
\end{aligned}$$

Hence, $\beta_{k-1}(\mathrm{Ind}(\overline{G}_i - N_{\overline{G}_i}[v_{i+1}])) = 0$ by the induction hypothesis. ◀

▶ Remark 19. Observe that the difference between the bound for a cone and the bound given in Theorem 18 is $n/2 - k - 1$. While this difference is linear in the number of vertices, it can result in a significant reduction in the number of higher-dimensional simplices; a reduction which has the potential to speed up current implementations of persistent homology for flag complexes, e.g., Ripser [2].

In the following example, we show that that bounds in the previous theorem are tight.

▶ **Example 20.** Let $n = (k+1)p$, and let $K_{1,p-1}$ be a star graph on p vertices and $p-1$ edges, i.e., a central vertex connected to all other vertices. Observe that $\beta_0(\mathrm{Ind}(K_{1,p-1})) = 1$ as one vertex is completely disconnected in the complement graph. If, $H = \bigsqcup_{i=1}^{k+1} K_{1,p-1}$, then it follows from repeated application of Lemma 4 that $\beta_k(\mathrm{Ind}(H)) \geq \prod_{i=1}^{k+1} \beta_0(\mathrm{Ind}(K_{1,p-1})) = 1$. The number of edges in $\mathrm{Ind}(H)$ is

$$\binom{n}{2} - (k+1)(p-1) = \binom{n}{2} - n + (k+1) = \binom{n}{2} - (n-1) + k = \binom{n-1}{2} + k.$$

▶ **Corollary 21.** *Let $\mathcal{G}$ be an edgewise filtration on n vertices. Then for any $[a,b) \in \mathcal{B}_k^{\mathrm{FL}}(\mathcal{G})$, we have*

$$2k(k+1) \leq a < b \leq \binom{n-1}{2} + k.$$

Proof. The bound on a follows from Theorem 16, and the bound on b from Theorem 18. ◀

It is straightforward to define a filtration $\mathcal{F}$ of the complement graph of H from Example 20 such that $\mathcal{B}_k^{\mathrm{FL}}(\mathcal{F})$ contains the interval $[a,b)$ from Corollary 21.

6 The maximum value of $|\beta_1^{\mathrm{FL}}(\mathcal{G})|$

▶ **Proposition 22.** *Let $\mathcal{G} = \{G_i\}_{i=1}^m$ be an edgewise filtration. Then, there exists a triangle-free subgraph H of G_m such that $\beta_1^{\mathrm{FL}}(H) \geq |\mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})|$.*

Proof. Choose a representative cycle $c_{[a,b]}$ for each non-empty interval $[a,b) \in \mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})$, e.g., by running the standard algorithm for persistent homology. For a cycle c, let $e(c)$ denote the set of edges on which the cycle has a non-zero coefficient, and let H be the subgraph of G_m with edges given by

$$\bigcup_{[a,b)\in\mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})} e(c_{[a,b)]}).$$

Note that the cycles $\{c_{[a,b)} : [a,b) \in \mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})\}$ are linearly independent (as elements of $C_1(G)$) by virtue of being representatives for non-trivial intervals.

If $X(H)$ contains a 2-simplex $\tau = \{v_1, v_2, v_3\}$, then any cycle $c_{[a,b)}$ supported on the edge $\{v_2, v_3\}$ is homologous in $\tilde{H}_1(X(H))$ to the cycle $c'_{[a,b)} = c_{[a,b)} - \partial_2(\tau)$. Hence, we can remove $\{v_2, v_3\}$ from H, without reducing the 1st Betti number. Doing this for all triangles, we end up with a triangle-free graph $\hat{H}$ containing at least $|\mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})|$ linearly independent 1-cycles. In particular, $\beta_1^{\mathrm{FL}}(\hat{H}) \geq |\mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})|$. ◀

▶ Remark 23. This argument does not apply to homology in higher degrees. For instance, the removal of an edge can result in the removal of many 2-simplices, some of which are faces of 3-simplices, and others that are not.

The previous result actually gives a short proof of Corollary 11 for the case $k = 1$. We shall include this proof here, as it ensures uniqueness of the extremal complex.

▶ **Proposition 24.** *For any graph on n vertices, we have $\beta_1^{\mathrm{FL}}(G) \leq \beta_1^{\mathrm{FL}}(\mathcal{T}_{n,2})$ with equality if and only if $G = \mathcal{T}_{n,2}$.*

Proof. Define any edgewise filtration $\mathcal{G} = \{G_i\}_{i=1}^m$ for which $G_m = G$. Then, by Proposition 22, there exists a triangle-free graph H such that $\beta_1^{\mathrm{FL}}(H) \geq \mathcal{B}_1^{\mathrm{FL}}(\mathcal{G}) \geq \beta_1^{\mathrm{FL}}(G)$. Hence, it suffices to maximize $\beta_1^{\mathrm{FL}}(G)$ over triangle-free graphs. By the Euler-Poincaré formula,

$$\beta_1^{\mathrm{FL}}(G) = -\beta_0^{\mathrm{FL}}(G) - 1 - |V(G)| + |E(G)|.$$

In conclusion, we are seeking a triangle-free graph with a maximal number of edges, but this is well-known to be uniquely the graph $\mathcal{T}_{n,2}$ by Turán's theorem [6, Theorem 11.17]. ◀

Combined we arrive at the following.

▶ **Corollary 25.** *If $\mathcal{G}$ is a filtration of K_n, then $|\mathcal{B}_1^{\mathrm{FL}}(\mathcal{G})| \leq \beta_1^{\mathrm{FL}}(\mathcal{T}_{n,2})$ with equality if and only if $G_{\lceil n/2\rceil \cdot \lfloor n/2\rfloor} = \mathcal{T}_{n,2}$.*

7 Extremal filtrations for degree 1 persistent homology

In this section, we consider edgewise filtrations $\mathcal{G} = G_1 \subseteq \cdots \subseteq G_{\binom{n}{2}}$ of the complete graph K_n that maximize the total persistence and the number of bars for degree 1 homology. We shall answer the following question:

For a fixed number of vertices n and for $k = 1$, which edgewise filtrations of the complete graph K_n with a maximal number of bars achieve maximal total persistence?

By Corollary 25, this translates to finding

$$\max\left\{\sum_{i=1}^{n(n-1)/2} \beta_1^{\mathrm{FL}}(G_i) \colon \mathcal{G} = \left\{G_i \colon 1 \le i \le \binom{n}{2}, G_{|E(\mathcal{T}_{n,2})|} = \mathcal{T}_{n,2} \text{ and } G_{n(n-1)/2} = K_n\right\}\right\}.$$

First, we give some preliminaries and establish some notation.

- All graphs $G = (V, E)$ have n vertices. The join of two graphs G_1 and G_2, notation $G_1 \vee G_2$, is as in Section 2.
- For a filtration $\mathcal{G} = \{G_i\}_{i=1}^m$, we write $T\beta_1^{\mathrm{FL}}(\mathcal{G}) = \sum_{i=1}^m \beta_1^{\mathrm{FL}}(G_i)$ for its total persistence of degree 1.
- Denote $e_n := |E(\mathcal{T}_{n,2})|$. If $|E(G)| \ge e_n$, then G has $\mathcal{T}_{n,2}$ as a subgraph, with partition classes V_1 and V_2 of sizes $|V_1| = \lceil n/2 \rceil$ and $|V_2| = \lfloor n/2 \rfloor$.
- Let d_i be the number of connected components of $G|_{V_i}$. Then, by Proposition 5,

$$\beta_1^{\mathrm{FL}}(G) = (d_1 - 1)(d_2 - 1). \tag{3}$$

We also define what it means for two graphs and two filtrations to be (fiberwise) isomorphic.

▶ **Definition 26.** *Two graphs G_1 and G_2 with vertex sets $V(G_i)$ and edge sets $E(G_i)$ are* isomorphic, *notation $G_1 \cong G_2$, if there is a bijection $\varphi \colon V(G_1) \to V(G_2)$, such that $\{u_1, u_2\} \in E(G_1)$ if and only if $\{\varphi(u_1), \varphi(u_2)\} \in E(G_2)$. Furthermore, two filtrations $\mathcal{G}^1 = \{G_i^1\}_{i=1}^m$ and $\mathcal{G}^2 = \{G_i^2\}_{i=1}^m$ are* fiberwise isomorphic *if $G_i^1 \cong G_i^2$ for all $i = 1, \ldots, m$.*

First, by Corollary 25, any filtration with a maximal number of bars must contain the Turán graph $\mathcal{T}_{n,2}$. Moreover, Theorem 16 establishes that the filtration $\mathcal{H} = \{H_i\}_{i=1}^{e_n}$ from Definition 13 is fiberwise optimal, reducing our problem to maximizing the total persistence of an edgewise filtration of the complete graph, beginning with the Turán graph plus one edge. This is a bit more involved, because we cannot find a fiberwise optimal filtration of the complete graph, as is shown in the following example.

▶ **Example 27.** If $|E| = e_n + 2$, then one gets optimal degree 1 homology by adding one edge to V_1 and one edge to V_2 (left graph in Figure 6). However, if $|E| = e_n + 3$, then optimal degree 1 homology is obtained by adding all three edges to V_1 to form a K_3 (second-to-left graph in Figure 6). It is clear that a filtration of K_n cannot contain both graphs.

We can prove the following about the structure of the graphs in an optimal filtration of K_n.

▶ **Lemma 28.** *Let $\mathcal{G}$ be an edgewise filtration $\{G_i\}_{i=1}^{\binom{n}{2}}$ of K_n with maximal total persistence that includes $\mathcal{T}_{n,2}$. Then, for $t = 1, 2$ and $i = e_n + 1, \ldots, \binom{n-1}{2} + 1$, one of the subgraphs $G_i|_{V_t}$ consists of isolated vertices and a single component K_m, and the other subgraph $G_i|_{V_t}$ consists of isolated vertices, and a single component with l vertices, such that K_{l-1} is a subgraph of this component.*

▶ **Example 29.** Cf. Figure 6 for four examples of graphs with the properties from Lemma 28.

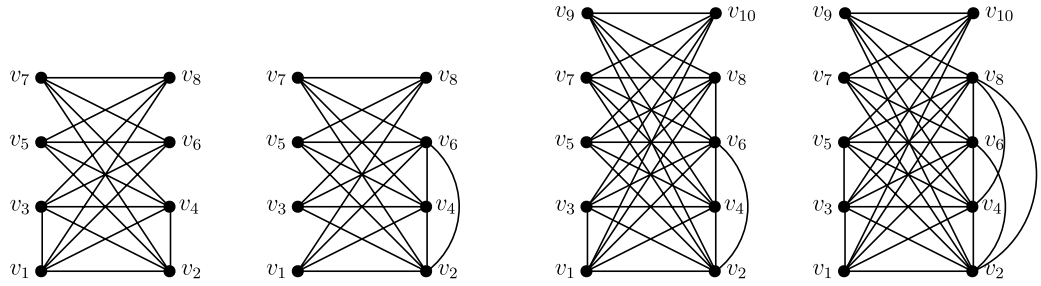


Figure 6 The graphs from Example 27 and 29.

Proof. Suppose that for some i, $E(G_i) \setminus E(G_{i-1}) = \{w_1, w_2\}$, where w_1 and w_2 are isolated vertices in $G_{j-1}|_{V_1}$ (without loss of generality). Also assume that $G_{j-1}|_{V_1}$ contains a component C of size $|C| > 1$. To maximize total persistence, it is more optimal to connect w_1 to C, since this allows us to add $|C| - 1$ edges of the form $\{w_1, c\}$ (where $c \in C$) without increasing the number of connected components (cf. (3)). Because these additional edges do not decrease the degree 1 homology, we should add them as early in the filtration as possible to maximize total persistence. Hence, by this same argument, $G_{j-1}|_{V_2}$ should be isomorphic to K_m and some isolated vertices.

Furthermore, note that we added the restriction $j \leq \binom{n-1}{2} + 1$ because Theorem 18 ensures that beyond this point, $\beta_1^{\mathrm{FL}}(G_j) = 0$. This means that

$$G_{\binom{n-1}{2}+1} = (K_1 \sqcup K_{\lceil n/2 \rceil - 1}) \vee (K_1 \sqcup K_{\lfloor n/2 \rfloor - 1}), \tag{4}$$

but if $j > \binom{n-1}{2} + 1$, then G_j could be any graph containing $G_{\binom{n-1}{2}+1}$ as a subgraph. ◀

Because of this, we restrict our focus to edgewise filtrations of the graph $G_{\binom{n-1}{2}+1}$ from (4).

Moreover, by Lemma 28, we only have to consider filtrations of the described form. We can represent them by a sequence of tuples of the form

$$(K_1, K_1) = (K_{l_1}, K_{r_1}), (K_{l_2}, K_{r_2}), \ldots, (K_{l_c}, K_{r_c}) = (K_{\lceil n/2 \rceil - 1}, K_{\lfloor n/2 \rfloor - 1}), \tag{5}$$

such that $l_i \geq l_{i-1}$ and $r_i \geq r_{i-1}$. The representation from (5) corresponds to the filtration $\mathcal{G} = \{G_i\}_{i=1}^m$, where, for $i \geq 1$,

$$G_{e_n + |E(K_{l_i})| + |E(K_{r_i})|} \cong \left(K_{l_i} \sqcup \bigsqcup_{j=l_i+1}^{\lceil n/2 \rceil} K_1\right) \bigvee \left(K_{r_i} \sqcup \bigsqcup_{j=r_i+1}^{\lfloor n/2 \rfloor} K_1\right) \text{ and}$$

$$G_{e_n + |E(K_{l_{i+1}})| + |E(K_{r_i})|} \cong \left(K_{l_{i+1}} \sqcup \bigsqcup_{j=l_i+1}^{\lceil n/2 \rceil} K_1\right) \bigvee \left(K_{r_i} \sqcup \bigsqcup_{j=r_i+1}^{\lfloor n/2 \rfloor} K_1\right).$$

In other words, the representation from (5) means that, starting from the Turán graph $\mathcal{T}_{n,2}$:

1. we begin adding edges by forming a K_{l_2} in V_1,
2. then add edges to form a K_{r_2} in V_2,
3. increase the sizes of the complete subgraphs in V_1 and V_2 by constructing a K_{l_3} (containing the smaller K_{l_2}) in V_1, a K_{r_3} in V_2, and so on.

Note that this representation is not unique ((K_3, K_1) is equivalent to $(K_2, K_1), (K_3, K_1)$).

▶ **Example 30.** In Figure 7, one finds a filtration and a corresponding representation.

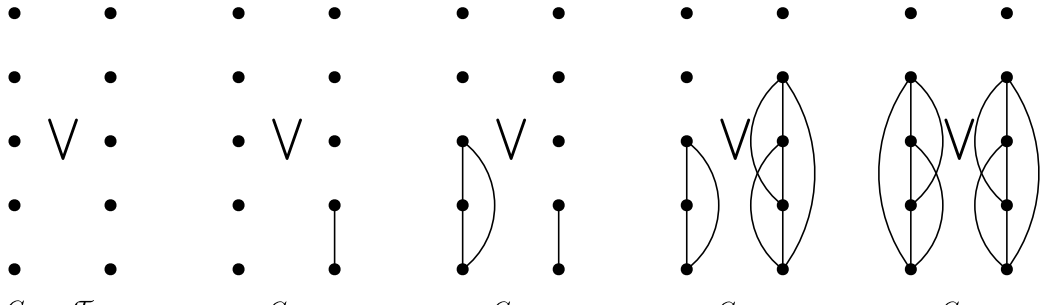


Figure 7 A filtration and its representation. Here, $\bigvee$ denotes the join of the left and right graphs.

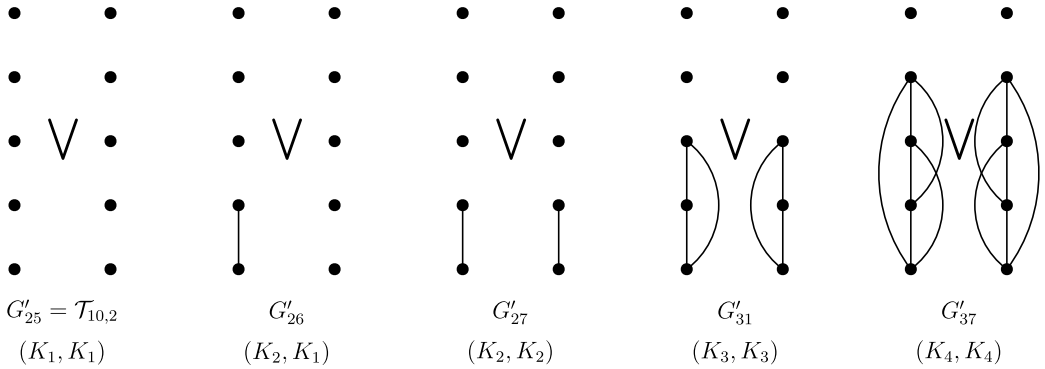


Figure 8 A filtration with the same total persistence as the one from Figure 7.

Surprisingly, the optimal strategy is to first construct two large cliques in V_1 and V_2 of sizes approximately $\frac{3}{4}|V_1|$ and $\frac{3}{4}|V_2|$, respectively, and then, alternating between V_1 and V_2, increase the sizes of the cliques one by one. We prove this in the main result of this section.

▶ **Theorem 31.** *Let $n \geq 4$ and let $j_n := \lfloor (3n-2)/8 \rfloor$ and $k_n := \lfloor (3n+7)/8 \rfloor$. Up to fiberwise isomorphism, the edgewise filtration $\mathcal{G}_{n,\max} = \{G_i\}_{i=1}^{\binom{n-1}{2}+1}$ of $G_{\binom{n-1}{2}+1}$ with maximal total persistence and a maximum number of bars is given by $G_i \cong H_i$ (Definition 13) for $i = 1, \ldots, e_n$. After $G_{e_n} = \mathcal{T}_{n,2}$, for $n \not\equiv 0 \mod 8$, the filtration is unique up to fiberwise isomorphism and represented by a sequence of tuples (cf. (5)) as follows*

$$\begin{cases} (K_1,K_1),(K_{j_n},K_{j_n}),(K_{j_n+1},K_{j_n+1}),\ldots,(K_{n/2-1},K_{n/2-1}), & \text{if } n \equiv 2,4,6 \mod 8, \\ (K_1,K_1),(K_{k_n},K_{k_n-1}),(K_{k_n+1},K_{k_n}),\ldots,(K_{(n-1)/2},K_{(n-3)/2}), & \text{if } n \equiv 1 \mod 2. \end{cases}$$

If $n \equiv 0 \mod 8$, then there are, up to fiberwise isomorphism, two optimal filtrations, represented by

$$(K_1,K_1),(K_{j_n},K_{j_n}),(K_{j_n+1},K_{j_n+1}),\ldots,(K_{n/2-1},K_{n/2-1}) \text{ and}$$
$$(K_1,K_1),(K_{j_n+1},K_{j_n+1}),(K_{j_n+2},K_{j_n+2}),\ldots,(K_{n/2-1},K_{n/2-1}).$$

▶ Remark 32. We prove the even case. The proof of the odd case is more or less the same (although the values are slightly different). However, we need to prove that, for any representation as in (5), $l_i \geq r_i$ for $i = 1, \ldots, c$. In the even case, we may assume this.

See Figure 8 for an example of and the full version [3, B.4] for a proof of this assumption.

Proof. It follows from Corollary 25 that, for a filtration $\mathcal{G}$ to have a maximum number of bars, we must have $G_{e_n} = \mathcal{T}_{n,2}$. For a filtration $\mathcal{G}$ to also have maximal total persistence, it immediately follows by Theorem 16 that $G_i \cong H_i$ for $i = 1, \dots, e_n$.

Any filtration $\mathcal{G}$ that we consider in this proof will therefore be such that $G_i \cong H_i$ for $i = 1, \dots, e_n$. We are interested in the remainder of the filtration, which we can represent by a sequence of tuples as in (5). Within the proof, we will further reduce it by writing

$$\big(l_1, r_1\big), \big(l_2, r_2\big), \dots, \big(l_c, r_c\big) \text{ instead of } \big(K_{l_1}, K_{r_1}\big), \big(K_{l_2}, K_{r_2}\big), \dots, \big(K_{l_c}, K_{r_c}\big). \tag{6}$$

Furthermore, given a filtration $\mathcal{G}$ represented by $\big(l_1, r_1\big), \big(l_2, r_2\big), \dots, \big(l_c, r_c\big)$, we define its *alternation depth* $d_{\mathcal{G}}$ as follows:

- We let $d_{\mathcal{G}} \coloneqq 1$ if $\big(l_i, r_i\big) = \big(l_{i-1} + 1, r_{i-1} + 1\big)$ for all $i = 2, \dots, c$.
- Otherwise, let $d_{\mathcal{G}} \geq 2$ be such that $\big(l_i, r_i\big) = \big(l_{i-1} + 1, r_{i-1} + 1\big)$ for all $i = d_{\mathcal{G}} + 1, \dots, c$, but $\big(l_{d_{\mathcal{G}}}, r_{d_{\mathcal{G}}}\big) \neq \big(l_{d_{\mathcal{G}}-1} + 1, r_{d_{\mathcal{G}}-1} + 1\big)$.

The idea of the proof is to start with a filtration $\mathcal{G} = \big\{G_i\big\}_{i=1}^{\binom{n-1}{2}+1}$, with a representation as in (6), and show that we can change the filtration in steps, to obtain in every step a filtration $\mathcal{G}'$ such that $T\beta_1^{\mathrm{FL}}\big(\mathcal{G}'\big) \geq T\beta_1^{\mathrm{FL}}\big(\mathcal{G}\big)$, and end up with an optimal solution.

Now, writing $n = 2p$, let $\mathcal{G}$ be a filtration, represented by

$$\big(1,1\big) = \big(l_1, r_1\big), \big(l_2, r_2\big), \dots, \big(l_c, r_c\big) = \big(n/2 - 1, n/2 - 1\big) = \big(p-1, p-1\big).$$

We shall show that

1. If the alternation depth $d_{\mathcal{G}} > j_n$, then we can change $\mathcal{G}$ to a filtration $\mathcal{G}'$ such that $d_{\mathcal{G}'} < d_{\mathcal{G}}$ and $T\beta_1^{\mathrm{FL}}\big(\mathcal{G}'\big) \geq T\beta_1^{\mathrm{FL}}\big(\mathcal{G}\big)$.
2. If $l_2 < j_n$, then we can change $\mathcal{G}$ to a filtration $\mathcal{G}'$ represented by $\big(l'_1, r'_1\big), \dots, \big(l'_{c'}, r'_{c'}\big)$ with $l'_2 = l_2 + 1$, such that $c' = c - 1$ or $c' = c$ (depending on $\mathcal{G}$) and $T\beta_1^{\mathrm{FL}}\big(\mathcal{G}'\big) \geq T\beta_1^{\mathrm{FL}}\big(\mathcal{G}\big)$.

For the first part, let $\mathcal{G}$ be a filtration with alternation depth $d_{\mathcal{G}} > j_n$. Its representation, substituting n by $2p$, is of the form

$$(1,1), \dots, (t,u), (v,w), \big(d_{\mathcal{G}}, d_{\mathcal{G}}\big), \big(d_{\mathcal{G}}{+}1, d_{\mathcal{G}}{+}1\big), \dots, \big(p{-}1, p{-}1\big) \text{ with } \big(v,w\big) \neq \big(d_{\mathcal{G}}{-}1, d_{\mathcal{G}}{-}1\big).$$

Since we assumed that $v \geq w$, we must have $w < d_{\mathcal{G}} - 1$. We transform $\mathcal{G}$ into a filtration $\mathcal{G}'$ which is the same as $\mathcal{G}$, except that we add an extra alternation step. We consider two cases.

- Case 1: $v < d_{\mathcal{G}} - 1$. In this case, $\mathcal{G}'$ is represented by

$$(1,1), \dots (t,u), (v,w), \big(d_{\mathcal{G}} - 1, d_{\mathcal{G}} - 1\big), \big(d_{\mathcal{G}}, d_{\mathcal{G}}\big), \big(d_{\mathcal{G}} + 1, d_{\mathcal{G}} + 1\big), \dots, \big(p-1, p-1\big).$$

- Case 2: $v = d_{\mathcal{G}} - 1$. In this case, $\mathcal{G}'$ is represented by

$$(1,1), \dots (t,u), \big(v = d_{\mathcal{G}} - 1, d_{\mathcal{G}} - 1\big), \big(d_{\mathcal{G}}, d_{\mathcal{G}}\big), \big(d_{\mathcal{G}} + 1, d_{\mathcal{G}} + 1\big), \dots, \big(p-1, p-1\big).$$

In both cases, we see that $d_{\mathcal{G}'} < d_{\mathcal{G}}$. Moreover, as we show in more detail in [3, B.4],

$$T\beta_1^{\mathrm{FL}}\big(\mathcal{G}'\big) - T\beta_1^{\mathrm{FL}}\big(\mathcal{G}\big) = \frac{1}{6}\big(d_{\mathcal{G}} - 1 - w\big)\big(d_{\mathcal{G}} - w\big)\big(4d_{\mathcal{G}} + 2w - 3p - 2\big) \geq 0. \tag{7}$$

To see that this difference is non-negative, first recall that $w < d_{\mathcal{G}} - 1$, so the first two terms are strictly positive. Furthermore, since $d_{\mathcal{G}} > \lfloor 3p/4 - 1/4 \rfloor$, we have $d_{\mathcal{G}} \geq \lceil 3p/4 \rceil \geq 3p/4$, so

$$4d_{\mathcal{G}} + 2w - 3p - 2 \geq 2w - 2 \geq 0,$$

since $w \geq 1$. Note that $T\beta_1^{\mathrm{FL}}\big(\mathcal{G}'\big) - T\beta_1^{\mathrm{FL}}\big(\mathcal{G}\big) = 0$ if and only if $w = 1$ and $d_{\mathcal{G}} = 3p/4$.

We now show the second part. We let $\mathcal{G}$ be a filtration with representation

$(1,1),(j,k),(l,m),\ldots,(p-1,p-1)$ with $j \leq 3p/4-1$.

We transform $\mathcal{G}$ into a filtration $\mathcal{G}'$ such that its start is different from $\mathcal{G}$'s, but from the tuple (l,m) onwards, $\mathcal{G}$ and $\mathcal{G}'$ are the same. We again consider two cases.

- Case 1: $l > j+1$. In this case, $\mathcal{G}'$ is represented by $(1,1),(j+1,k),(l,m),\ldots,(p-1,p-1)$.
- Case 2: $l = j+1$. In this case, $\mathcal{G}'$ is represented by $(1,1),(j+1=l,m),\ldots,(p-1,p-1)$.

In both cases, as we show in more detail in the full version [3, B.4], we have

$$T\beta_1^{\mathrm{FL}}(\mathcal{G}') - T\beta_1^{\mathrm{FL}}(\mathcal{G}) = (k-1)\big(j(p-j-1) - \frac{k}{6}(3p-2k-2)\big) \overset{(i)}{\geq} 2\big((j-k+1)(j-k)\big) \overset{(ii)}{\geq} 0, \quad (8)$$

where we use in (i) that $j \leq 3p/4-1$, and in (ii) that $j \geq k$. If $n \equiv 2,4,6 \mod 8$, then $j \leq 3p/4-1$ implies that $j \leq \lfloor 3p/4-1 \rfloor = \lfloor 3p/4-5/4 \rfloor = j_n - 1$. If $n \equiv 0 \mod 8$, then $j \leq 3p/4-1 = j_n$. Only in this case, and if $j = k = 3p/4-1$, then the inequality is sharp.

Hence, if we start with an arbitrary filtration $\mathcal{G}$, then we can transform it step by step into a more optimal filtration $\mathcal{G}^1$ with $d_{\mathcal{G}^1} \leq j_n$, by the first part. Then, by the second part, we can transform $\mathcal{G}^1$ step by step into a more optimal filtration $\mathcal{G}^2$ that starts with $(1,1),(j_n,j_n)$, with alternation depth $d_{\mathcal{G}^2} = j_n$. Hence, $\mathcal{G}^2$ is the optimal filtration as described in the statement for $n \equiv 2,4,6 \mod 8$.

In the case that $n \equiv 0 \mod 8$, we saw in the proof of case 1 and case 2, that the filtration $\mathcal{G}^3$ starting with $(1,1),\big(j_n+1,j_n+1\big)$ and having alternation depth $d_{\mathcal{G}^3} = j_n+1$ satisfies $T\beta_1^{\mathrm{FL}}(\mathcal{G}^3) = T\beta_1^{\mathrm{FL}}(\mathcal{G}^2)$, hence both $\mathcal{G}^2$ and $\mathcal{G}^3$ are optimal. ◀

8 Discussion

In this paper, we have answered several key questions in the context of topological data analysis. However, many questions remain open, and we propose the following conjectures.

▶ **Conjecture 33.** *If $\mathcal{G}$ is a filtration on n vertices, then the number of intervals in the barcode of $\mathcal{G}$ in homology degree k satisfies $|\mathcal{B}_k^{\mathrm{FL}}(\mathcal{G})| \leq \beta_k^{\mathrm{FL}}(\mathcal{T}_{n,k+1})$.*

▶ **Conjecture 34.** *The extremal filtrations described in Section 7 achieve the maximal total persistence over any edgewise filtration of flag complexes in homology degree 1.*

Proving the latter conjecture likely requires new ideas, as no filtration $\mathcal{G}$ can maximize $\beta_1^{\mathrm{FL}}(G_i)$ for all i. In fact, extremal values need not be realized by graphs containing Turán graphs as spanning subgraphs, as illustrated by the following example.

▶ **Example 35.** Let $n = 8$ and $m = 16+5$. Suppose G contains the Turán graph $\mathcal{T}_{8,2}$. By Proposition 5, we seek to add 5 edges to the Turán graph to maximize the product $(d_1-1)(d_2-1)$. It is easy to see that this product can be at most 1. However, by partitioning the vertices as $V = V_1 \cup V_2$, with $|V_1| = 3$ and $|V_2| = 5$, and adding all edges between V_1 and V_2, we need to add 6 more edges. Adding these as a K_4 in the larger partition gives $d_1 = 2$ and $d_2 = 1$; see Figure 9. The value 2 is extremal; see Figure 10.

Finding extremal values of $\mathbb{Z}$-linear functions defined on simplicial complexes with *precisely* n vertices and m edges presents a challenging and interesting direction for future research.

▶ **Conjecture 36.** *Let $\mathbf{G}(n,m)$ denote the collection of graphs on n vertices and m edges. If $G \in \mathbf{G}(n,m)$ and $\beta_1^{\mathrm{FL}}(G) = \max_{H \in \mathbf{G}(n,m)} \beta_1^{\mathrm{FL}}(H)$, then G contains a complete bipartite spanning subgraph.*

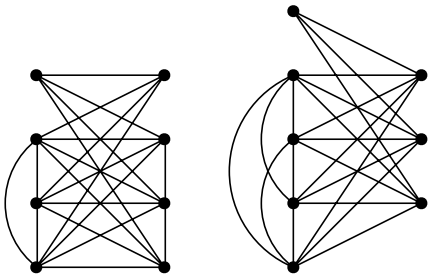


Figure 9 Two graphs G_l (left) and G_r (right) such that $\beta_1^{\mathrm{FL}}(G_l) = 1$ and $\beta_1^{\mathrm{FL}}(G_r) = 2$.

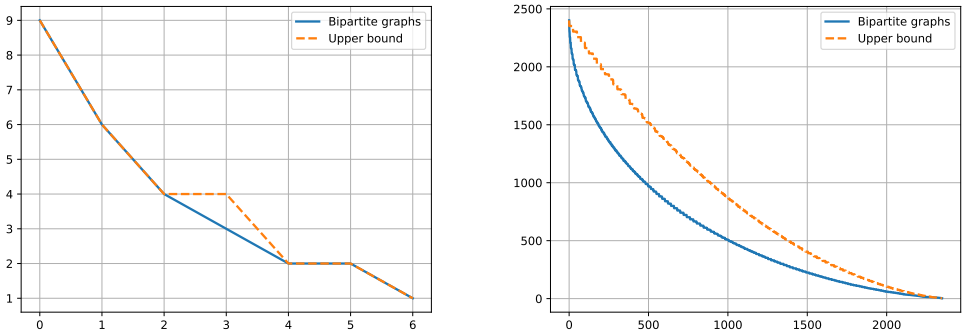


Figure 10 The vertical axis is β_1^{FL}, and the value k along the x-axis represents $(n/2)^2 + k$ edges. Here n is the number of vertices in the underlying graph; $n = 8$ (left) and $n = 100$ (right). The solid curves give the optimal value of $\beta_1^{\mathrm{FL}}(G)$ for any graph G containing a complete bipartite spanning subgraph, and the dashed line is an upper bound derived from the proof of Theorem 10 in conjuction with Lemma 17 (details omitted).

References

1. Michał Adamaszek. Extremal problems related to Betti numbers of flag complexes. *Discrete Applied Mathematics*, 173:8–15, 2014. doi:10.1016/J.DAM.2014.04.006.
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. Lies Beers and Magnus Bakke Botnan. Extremal Betti Numbers and Persistence in Flag Complexes, 2025. arXiv:2502.21294.
4. Anders Björner and Gil Kalai. An extended Euler-Poincaré theorem, 1988.
5. Magnus Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of Algebras and Related Structures*, pages 77–150. EMS Press, 2023.
6. Gary Chartrand and Ping Zhang. *A first course in graph theory*. Courier Corporation, 2013.
7. Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 232–241, 2008.
8. Tamal Krishna Dey and Yusu Wang. *Computational topology for data analysis*. Cambridge University Press, 2022.
9. 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.

10 Michael Goff. Extremal Betti numbers of Vietoris–Rips complexes. *Discrete and Computational Geometry*, 46(1):132, 2011. doi:10.1007/s00454-010-9274-z.

11 Dmitry N Kozlov. Convex hulls of f-and β-vectors. *Discrete & Computational Geometry*, 18:421–431, 1997. doi:10.1007/PL00009326.

When Distances Lie: Euclidean Embeddings in the Presence of Outliers and Distance Violations

Matthias Bentert
University of Bergen, Norway

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

Distance geometry explores the properties of distance spaces that can be exactly represented as the pairwise Euclidean distances between points in $\mathbb{R}^d$ ($d \geq 1$), or equivalently, distance spaces that can be isometrically embedded in $\mathbb{R}^d$. In this work, we investigate whether a distance space can be isometrically embedded in $\mathbb{R}^d$ after applying a limited number of modifications. Specifically, we focus on two types of modifications: *outlier deletion* (removing points) and *distance modification* (adjusting distances between points). The central problem, Euclidean Embedding Editing, asks whether an input distance space on n points can be transformed, using at most k modifications, into a space that is isometrically embeddable in $\mathbb{R}^d$.

We present several fixed-parameter tractable (FPT) and approximation algorithms for this problem. Our first result is an algorithm that solves Euclidean Embedding Editing in time $(dk)^{\mathcal{O}(d+k)} + n^{\mathcal{O}(1)}$. The core subroutine of this algorithm, which is of independent interest, is a polynomial-time method for compressing the input distance space into an equivalent instance of Euclidean Embedding Editing with $\mathcal{O}((dk)^2)$ points.

For the special but important case of Euclidean Embedding Editing where only outlier deletions are allowed, we improve the parameter dependence of the FPT algorithm and obtain a running time of $\min\{(d+3)^k, 2^{d+k}\} \cdot n^{\mathcal{O}(1)}$. Additionally, we provide an FPT-approximation algorithm for this problem, which outputs a set of at most $2 \cdot \mathsf{Opt}$ outliers in time $2^d \cdot n^{\mathcal{O}(1)}$. This 2-approximation algorithm improves upon the previous $(3+\varepsilon)$-approximation algorithm by Sidiropoulos, Wang, and Wang [SODA '17]. Furthermore, we complement our algorithms with hardness results motivating our choice of parameterizations.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Theory of computation → Fixed parameter tractability

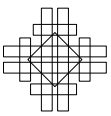
Keywords and phrases Parameterized Complexity, Euclidean Embedding, FPT-approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.15

Related Version *Full Version*: http://arxiv.org/abs/2503.19093

Funding *Matthias Bentert*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416).
Fedor V. Fomin: Supported by the Research Council of Norway under BWCA project (grant no. 314528).
Petr A. Golovach: Supported by the Research Council of Norway under BWCA project (grant no. 314528).

© Matthias Bentert, Fedor V. Fomin, Petr A. Golovach, M. S. Ramanujan, and Saket Saurabh;
licensed under Creative Commons License CC-BY 4.0



M. S. Ramanujan: Supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/V044621/1.
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

The Euclidean Distance Matrix (EDM) is a matrix containing the squared Euclidean distances between points in a set. A central problem in Distance Geometry is determining whether a given matrix is an EDM [5, 25, 11, 13]. That is, the task is to identify whether a given distance space can be isometrically embedded into ℓ_2-spaces, or equivalently, the pairwise Euclidean distances among points in $\mathbb{R}^d$ ($d \geq 1$). This problem has a rich history, originating with Cayley [7], whose observations were formalized by Menger [28], leading to Cayley-Menger determinants. Schoenberg [30] further advanced the field by characterizing ℓ_2-embeddable distances using negative type inequalities. Blumenthal's monograph [5] remains a foundational text, documenting the theoretical underpinnings of this area.

This problem has numerous applications, including sensor network localization [29, 14], molecular conformation [21], data visualization [6], statistics [17], psychology [34, 31], learning image manifolds [36], handwriting recognition [24], studying musical rhythms [12], and signal processing [15]. The complexity of determining whether a matrix is an EDM is well understood, and it can be efficiently addressed using Singular Value Decomposition (SVD) [6, 35].

In many practical applications involving EDMs, errors due to noise, missing values, or approximation inaccuracies are common. To address these challenges, several heuristic approaches based on Multidimensional Scaling, Low-Rank Matrix Approximation, and Semidefinite Programming have been developed to reconstruct the matrix and create an embedding that minimizes the impact of these errors. We refer readers to [6, 15] for a comprehensive overview of the extensive literature on this topic.

Despite the importance and numerous practical applications of the EDM recognition problem with noise and errors, little was known about its computational complexity until recently. A notable exception is the work of Sidiropoulos, Wang, and Wang [32], which initiates the study of EDM in the presence of outliers. The work of Sidiropoulos et al. serves as the initial foundation for our studies.

In this paper, we address the algorithmic question of minimizing the number of edits required to transform a given distance matrix into EDM. Specifically, we aim to apply the minimum number of modifications to a given *distance space* (X, ρ) with distance function ρ, such that the resulting distance space can be isometrically embedded into a d-dimensional Euclidean space $\mathbb{R}^d$. We consider two types of editing operations.

The first operation is *element deletion*. In matrix terms, this operation corresponds to deleting the row and column associated with the element we choose to remove. Following Sidiropoulos, Wang, and Wang [32], we refer to the elements removed from the distance space as *outliers*.

The second operation is *distance modification*. Let $X^{(2)}$ denote the set of unordered pairs of distinct elements in $X = \{x_1, x_2, \ldots\}$. For a pair of elements $x_i, x_j \in X$, the modification operation alters the distance $\rho(x_i, x_j)$. In terms of the distance matrix, this operation changes the ij-th and the ji-th entries. The problem of minimizing the number of modification operations to embed the resulting distance space in general metric spaces was studied in [18, 9], and embedding into ultrametric spaces was investigated in [9, 8].

Our primary algorithmic question is as follows: given integers k_O and k_M, can a distance space (X, ρ) be transformed into a distance space that is embeddable in $\mathbb{R}^d$ by removing at most k_O outliers and performing at most k_M modifications? In fact, we address an even more general weighted version of this problem, where each operation (outlier deletion or distance modification) is assigned a specific weight. More precisely, we study:

Euclidean Embedding Editing

Input: Distance space $\mathcal{X} = (X, \rho)$, weight functions $w_O \colon X \to \mathbb{Z}_{\geq 0}$ and $w_M \colon X^{(2)} \to \mathbb{Z}_{\geq 0}$, integers $W, k_O, k_M \geq 0$, and $d \geq 1$.

Question: Is there a subset of at most k_O outliers $O \subseteq X$ and a set $D \subseteq X^{(2)}$ of at most k_M distances with $w_O(O) + w_M(D) \leq W$ such that the distance subspace $(X \setminus O, \rho')$ where ρ' is obtained by modifying the values $\rho(x, y)$ for $\{x, y\} \in D$ is isometrically embeddable into $\mathbb{R}^d$?

Two special cases of Euclidean Embedding Editing are of particular importance. The variant of the problem with $k_M = 0$, that is, of placing all but k_O outlier points of a distance space (X, ρ) into a Euclidean space $\mathbb{R}^d$ of a given dimension d such that the Euclidean distance between any pair of points x, is equal to $\rho(x, y)$, is called Euclidean Embedding with Outliers [32]. For the variant with $k_O = 0$, that is, without deletion of outliers, following [18], we use the name Euclidean Metric Violation Distance.

EXAMPLE: The distance space $\mathcal{X} = (X, \rho)$ with $X = \{1, \ldots, 9\}$ and ρ defined by distance matrix D, where the ij-th entry of D is $\rho^2(i, j)$

$$D = \begin{bmatrix} 0 & 7 & 1 & 2 & 4 & 5 & 1 & 4 & 5 \\ 7 & 0 & 2 & 1 & 1 & 2 & 10 & 5 & 4 \\ 1 & 2 & 0 & 1 & 11 & 4 & 4 & 1 & 2 \\ 2 & 1 & 1 & 0 & 2 & 1 & 8 & 2 & 1 \\ 4 & 1 & 11 & 2 & 0 & 1 & 12 & 8 & 5 \\ 5 & 2 & 4 & 1 & 1 & 0 & 7 & 5 & 2 \\ 1 & 10 & 4 & 8 & 12 & 7 & 0 & 8 & 9 \\ 4 & 5 & 1 & 2 & 8 & 5 & 8 & 0 & 1 \\ 5 & 4 & 2 & 1 & 5 & 2 & 9 & 1 & 0 \end{bmatrix},$$

and $k_O = 1$, $k_M = 2$, $d = 2$, all weights equal to one, and $W = 3$, is a yes-instance of Euclidean Embedding Editing. Indeed, the distance space $\mathcal{X}' = (X', \rho')$ defined by distance matrix D' in Fig. 1 that can be obtained from $\mathcal{X}$ by modifying distances between $\{1, 2\}$ and $\{3, 5\}$ and deleting outlier 7, is isometrically embeddable in $\mathbb{R}^2$.

Our results. First, we present a compression for Euclidean Embedding Editing– a polynomial-time algorithm that reduces an instance of the problem to an equivalent instance with the number of points bounded by $\mathcal{O}(k^2 d^2)$ (Theorem 8), where $k = k_O + k_M$. As part of this compression, we also propose a $(d + 3)$-approximation algorithm for Euclidean Embedding with Outliers, which runs in polynomial time (Lemma 10). Using this compression algorithm, we design an FPT algorithm for Euclidean Embedding Editing with a running time of $(dk)^{\mathcal{O}(d+k)} + n^{\mathcal{O}(1)}$ (Theorem 20).

For Euclidean Embedding with Outliers, a particular but important special case of Euclidean Embedding Editing – where the goal is to determine whether the distance space, excluding up to k_O outliers, can be embedded in $\mathbb{R}^d$ – we obtain better dependence on the parameters with a running time of $\min\{(d + 3)^{k_O}, 2^{d+k_O}\} \cdot n^{\mathcal{O}(1)}$ (Theorem 21).

$$D' = \begin{bmatrix} 0 & 1 & 1 & 2 & 4 & 5 & 4 & 5 \\ 1 & 0 & 2 & 1 & 1 & 2 & 5 & 4 \\ 1 & 2 & 0 & 1 & 5 & 4 & 1 & 2 \\ 2 & 1 & 1 & 0 & 2 & 1 & 2 & 1 \\ 4 & 1 & 5 & 2 & 0 & 1 & 8 & 5 \\ 5 & 2 & 4 & 1 & 1 & 0 & 5 & 2 \\ 4 & 5 & 1 & 2 & 8 & 5 & 0 & 1 \\ 5 & 4 & 2 & 1 & 5 & 2 & 1 & 0 \end{bmatrix}$$

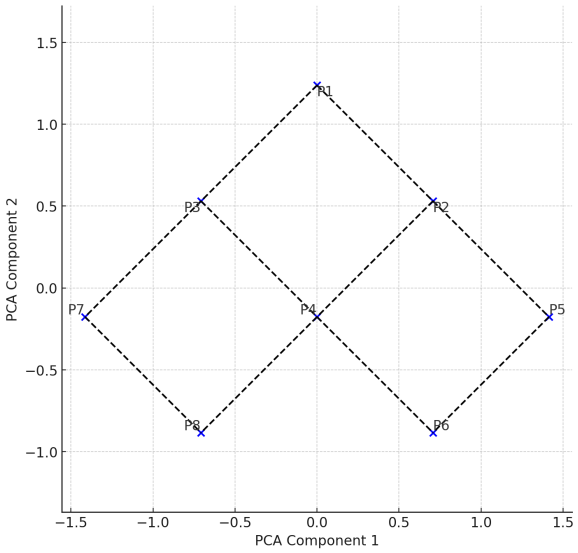


Figure 1 Matrix D' and an embedding of its distance space in $\mathbb{R}^2$.

Furthermore, for this problem, we propose a 2-approximation algorithm (Theorem 22). This randomized algorithm, with a running time of $2^d \cdot n^{\mathcal{O}(1)}$, guarantees a solution with at most $2 \cdot \mathsf{Opt}$ outliers. As is common 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.

We also complement our algorithmic results with lower-bound proofs, establishing that both Euclidean Embedding with Outliers and Euclidean Metric Violation Distance are NP-hard even when $d = 1$ (Theorems 26 and 27). So, FPT algorithms for these problems parameterized by d alone appears to be out of reach, motivating our FPT-approximation algorithm. Additionally, we prove that Euclidean Embedding with Outliers is W[1]-hard when parameterized by k_O alone (Theorem 28). These lower bounds indicate that to get an FPT algorithm for Euclidean Embedding Editing, it is important that we parameterize by *both* d and the solution size $k_O + k_M$. Importantly, our lower-bound results remain valid even in the unweighted case.

Related work. The computational complexity study of Euclidean Embedding with Outliers was initiated by Sidiropoulos, Wang, and Wang [32]. They demonstrated that, assuming the Unique Games Conjecture, the problem of computing a minimum outlier embedding into d-dimensional Euclidean space for any $d \geq 2$ is NP-hard to approximate within a factor of $2 - \varepsilon$ for any $\varepsilon > 0$. On the algorithmic side, they showed that a 2-approximation can be achieved in $\mathcal{O}(n^{d+3})$ time. Additionally, they established that a $(3 + \varepsilon)$-approximation is achievable in $(2/\varepsilon)^d d^{\mathcal{O}(1)} n^2 \log n$ time. For exact solutions, they noted an algorithm with a runtime of $\mathcal{O}(n^{d+3}) + 2^{k_O} n^2$.

They also obtained results related to bicriteria approximation involving nonisomorphic embeddings, approximation of the number of outliers, and distortion. The first result, is $(\mathcal{O}(\sqrt{\delta}), (2d+2)k_O)$-relative outlier embedding in $\mathbb{R}^d$ in time $\frac{1}{\delta^{\mathcal{O}(1)}} 2^{\mathcal{O}(d)} n^2 \log n$. The second result, $(\mathcal{O}(\sqrt{\delta}), 2k_O)$-relative outlier embedding in time $\frac{1}{\delta^{\mathcal{O}(1)}} {k_O}^{\mathcal{O}(d)} n^2$. (They define an algorithm as an $\mathcal{O}(f(\delta), g(k_O))$-relative outlier embedding of $\mathcal{X} = (X, \rho)$ to $\mathbb{R}^d$, for some functions f and g, if it either correctly decides that no embedding with distortion at most δ exists after removing k_O outliers, or outputs a set Y of size $g(k_O)$ such that there is an

embedding $X \setminus Y$ into $\mathbb{R}^d$ of distortion $\mathcal{O}(f(\delta))$.) Since allowing distortion could significantly decrease the number of outliers, our 2-approximation algorithm for Euclidean Embedding with Outliers, Theorem 22, is incomparable with these results.

We are not aware of any algorithms with guaranteed performance for Euclidean Metric Violation Distance. Parameterized and approximation algorithms for metric violation distance problems for general metric, tree distances and ultrametric could be found in [18, 8, 9, 20]. More generally, embeddings of various metric spaces are a fundamental primitive in the design of algorithms [22, 23, 27, 26, 2, 3], though prior work often focuses on minimizing embedding distortion.

2 Preliminaries

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*. If, in addition ρ satisfies a triange inequality: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$, for any $x, y, z \in X$, then ρ is called a semimetric on X. And if $\rho(x, y) = 0$ only for $x = y$, then ρ is called a *metric*, and (X, ρ) a *metric space*.

More precisely, recall that for two points $p, q \in \mathbb{R}^d$, the Euclidean distance between p and q is $\|p - q\|_2 = \sqrt{\langle p, p\rangle + \langle q, q\rangle - 2\langle p, q\rangle}$. We say that 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 $x, y \in X$. 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. We use $X^{(2)}$ to denote the set of unordered pairs of two distinct elements of X. As convention, we assume that the empty set of points is d-embeddable for every d.

Let $\mathcal{X} = (X, \rho)$ be a distance space where $X = \{x_1, \ldots, x_n\}$, and let $\rho_{i,j} = \rho(x_i, x_j)$ for all $i, j \in \{1, \ldots, n\}$. Then $\mathcal{X}$ is equivalently defined by the *distance matrix* $D(\rho)$, where $D(\rho)_{i,j} = \rho_{i,j}^2$.

We drop the explicit reference to ρ when it is clear from the context and instead of $D(\rho)$, simply write D. Suppose that $\mathcal{X}$ 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 $i \in \{1, \ldots, n\}$. We use the well-known property (see e.g. [15]) that a realization is unique up to *rigid transformations* of $\mathbb{R}^d$, that is, distance-preserving transformations of Euclidean space (rotations, reflections, translations).

▶ **Proposition 1** ([15]). *Let $(p_1, \ldots, p_n)$ and $(q_1, \ldots, q_n)$ be ordered set of points in $\mathbb{R}^d$. Then $\|p_i - p_j\|_2 = \|q_i - q_j\|_2$ for all $i, j \in \{1, \ldots, n\}$ if and only if there is a rigid transformation of $\mathbb{R}^d$ mapping p_i to q_i for all $i \in \{1, \ldots, n\}$.*

A realization can be constructed (if it exists) in polynomial time.

▶ **Proposition 2** ([1, 33]). *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 realization can be constructed in this running time.*

▶ **Definition 3** (Metric basis). *Let (X, ρ) be a d-embeddable distance space. 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.*

We will use the well-known characteristic of strong embeddability of a distance space into a Euclidean space. For $r+1$ points $x_0, x_1, \ldots, x_r$ of distance space (X, ρ) the *Cayley-Menger determinant* is the determinant of the matrix obtained from the distance matrix by prepending a row and a column whose first element is zero and the other elements are one. Formally, let $\rho_{i,j} = \rho(x_i, x_j)$, $i, j \in \{0, \ldots, r\}$. Then the Cayley-Menger determinant is

$$CM(x_0, x_1, \ldots, x_r) = \det \begin{pmatrix} 0 & 1 & 1 & 1 & \ldots & 1 \\ 1 & 0 & \rho_{0,1}^2 & \rho_{0,2}^2 & \ldots & \rho_{0,r}^2 \\ 1 & \rho_{0,1}^2 & 0 & \rho_{1,2}^2 & \ldots & \rho_{1,r}^2 \\ 1 & \rho_{0,2}^2 & \rho_{1,2}^2 & 0 & \ldots & \rho_{2,r}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \rho_{0,r}^2 & \rho_{1,r}^2 & \rho_{2,r}^2 & \ldots & 0 \end{pmatrix}$$

▶ **Proposition 4** ([5, 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* $1 \le j \le d$, *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, [32]), $\mathcal{X}$ 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.

Thus, the embedding of the distance space $\mathcal{X} = (X, \rho)$ into $\mathbb{R}^d$ is essentially characterized by $d+1$ "anchor" points of X. Note that in a realization of $\mathcal{X}$ that is strongly d-embeddable, the anchor points correspond to a set of $d+1$ points in general position in $\mathbb{R}^d$.

Following [5], we define sets of *independent points*.

▶ **Definition 5.** *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.*

Then our "anchors" are independent sets of size $d+1$. We use the following fact that the family of independent sets has matroid-like properties (implicit in [5, Chapter IV]).

▶ **Proposition 6** ([5, Chapter IV]). *Let (X, ρ) be a strongly d-embeddable distance space. Then, (i) any single-element set is independent, (ii) if $Y \subseteq X$ is independent, then any $\emptyset \subset Z \subseteq Y$ is independent, (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. Furthermore, the maximum size of an independent set is $d+1$ and any independent set Y of size $d+1$ is a metric basis.*

▶ **Lemma 7** (⋆). *Suppose (X, ρ) is a d-embeddable distance space, $Y \subset X$ is independent and $x \in X \setminus Y$. Then, $(Y \cup \{x\}, \rho)$ is $|Y|$-embeddable.*

For statements marked with (⋆), the proofs are omitted in this extended abstract and is deferred to the full version.

3 Compression for Euclidean Embedding Editing

In this section, we show that, given an instance of Euclidean Embedding Editing, one can in polynomial time construct an equivalent instance where the number of points is upper-bounded by a polynomial of $k = k_O + k_M$ and d and, moreover, the encoding of the weights is also polynomial in these parameters.

▶ **Theorem 8.** *There is a polynomial-time algorithm that, given an instance* $(\mathcal{X} = (X, \rho), w_O, w_M, W, k_O, k_M, d)$ *of* Euclidean Embedding Editing*, either solves the problem or constructs an equivalent instance* $(\mathcal{X}' = (X', \rho), w'_O, w'_M, W', k'_O, k'_M, d)$ *such that* $X' \subseteq X$*, and for* $k = k_O + k_M$*, it holds that* $|X'| = \mathcal{O}((kd)^2)$*,* $W' = 2^{\mathcal{O}((kd)^{12})}$*,* $w'_O(x) = 2^{\mathcal{O}((kd)^{12})}$ *for* $x \in X'$*, and* $w'_M(x, y) = 2^{\mathcal{O}((kd)^{12})}$ *for all* $\{x, y\} \in X'^{(2)}$*.*

We remark that Theorem 8 does not provide a kernelization algorithm according to the standard definition [10] as the size of the output instance is not upper-bounded by a function of the parameter – the distances between the points remain the same as in the input instance. Moreover, since Euclidean Embedding Editing is a decision problem, the word "solve" in the statement of Theorem 8 technically only refers to deciding correctly whether the instance is a yes-instance. However, if the instance is determined to be a yes-instance, then our algorithm can also output a solution that witnesses this, i.e., the outlier set and modified new distance values (if any). This additional feature of our compression algorithm will be used later in our FPT algorithm for Euclidean Embedding Editing.

In the proof of Theorem 8, we first design a polynomial-time $(d + 3)$-approximation algorithm (Section 3.1). This algorithm allows us, given an instance of the problem, either find a set of at most $(d+3)k$ points A such that $(X \setminus A, \rho)$ is d-embeddable or conclude that we have a no-instance.

In the first case, we iteratively construct a sequence of metric bases $Y_1, \ldots, Y_\ell$ for (Z, ρ) using Proposition 6, where initially $Z := X \setminus A$ and in each iteration, we delete the points of the constructed basis from Z. If $X \setminus A$ is sufficiently large, then there is a subsequence of bases of the same size with at least $2(k_O + 2k_M) + 1$ elements. We select the first subsequence with this property. The crucial property exploited by our algorithm is that this subsequence uniquely defines embeddings of the majority of the points assuming that we have a yes-instance of the problem. We use this to give a series of reduction rules that identify points in A that should be outliers in any solution and irrelevant points of $X \setminus A$ that could be deleted.

3.1 Bootstrapping the Compression through an Approximation

For a distance space $\mathcal{X} = (X, \rho)$, a *d-outlier set* is a set A of points such that $(X \setminus A, \rho)$ is d-embeddable. In unweighted Euclidean Embedding with Outliers (which we call UEEO), the function w_O assigns unit weight to every point. Sidiropoulos et al. [32] observed that this problem admits a simple greedy $(d + 3)$-approximation running in $n^{\mathcal{O}(d)}$. We show that this algorithm can be sped up to run in $n^{\mathcal{O}(1)}$ time, i.e., independent of d in the exponent. To do so, we give a polynomial-time subroutine that produces a small hitting set for all minimal solutions. This subroutine will later be used as the first step of our compression algorithm for Euclidean Embedding Editing.

▶ **Lemma 9.** *There is a polynomial-time algorithm that, given a distance space* $\mathcal{X} = (X, \rho)$ *and an integer* $d \geq 1$*, outputs a set of points* $\hat{A}$ *of size at most* $(d + 3)$ *such that if* $\mathcal{X}$ *is not* d*-embeddable, then* $\hat{A}$ *intersects every inclusionwise minimal* d*-outlier set of* $\mathcal{X}$*.*

Proof. Assume that the distance space $\mathcal{X} = (X, \rho)$ does not admit a d-embedding. Note that, in particular, this means that $|X| \geq 3$. We then do the following.

1. Set $A := \{p\}$ for an arbitrary point $p \in X$.
2. While
   $|A| \leq d$, do the following:
   a. If there is a point $x \in X \setminus A$ such that $(A \cup \{x\}, \rho)$ is not $|A|$-embeddable, then set $\hat{A} := A \cup \{x\}$ and return $\hat{A}$.
   b. If there is a point $x \in X \setminus A$ such that $A \cup \{x\}$ is independent, then set $A := A \cup \{x\}$ and continue the loop, else exit the loop.
3. If there are two distinct points $x, y \in X \setminus A$ such that $(A \cup \{x, y\}, \rho)$ is not $(|A| - 1)$-embeddable, then $\hat{A} := A \cup \{x, y\}$ and return $\hat{A}$.

Clearly, the algorithm runs in polynomial time. Since the while loop has at most d iterations, the set A when the loop is exited has size at most $d+1$ and so, the returned set $\hat{A}$ has size at most $d + 3$. ◀

We can now use Lemma 9 as a subroutine in the greedy approximation for the minimization variant of *unweighted* Euclidean Embedding with Outliers.

▶ **Lemma 10** (⋆). *There is a polynomial-time algorithm that, given a distance space $\mathcal{X} = (X, \rho)$ and an integer $d \geq 1$, outputs a set of points A of size at most $(d + 3) \cdot \mathsf{Opt}$ such that $(X \setminus A, \rho)$ is embeddable into $\mathbb{R}^d$ where Opt is the size of a smallest d-outlier set.*

3.2 Compression

We are now ready to give the compression algorithm for Euclidean Embedding Editing.

Proof of Theorem 8. Let $(\mathcal{X} = (X, \rho), w_O, w_M, W, k_O, k_M, d)$ be an instance of Euclidean Embedding Editing. First, we preprocess the instance to reduce the number of points and then we explain how to reassign the weights.

Notice that if $(\mathcal{X}, w_O, w_M, W, k_O, k_M, d)$ is a yes-instance, then it is possible to delete $k = k_O + k_M$ points to obtain a distance space embeddable in $\mathbb{R}^d$. In the first step of our algorithm, we call the algorithm from Lemma 10 for the instance $(\mathcal{X}, k, d)$ of Euclidean Embedding with Outliers *without* weights. This algorithm outputs a set of points A of size at most $(d+3) \cdot \mathsf{Opt}$ such that $(X \setminus A, \rho)$ is embeddable into $\mathbb{R}^d$ where Opt is the minimum number of outliers. If $|A| > (d + 3)k$ then we conclude that $(\mathcal{X}, w_O, w_M, W, k_O, k_M, d)$ is a no-instance of Euclidean Embedding Editing and stop. From now on, we assume that $|A| \leq (d + 3)(k_O + k_M)$.

Let $Y = X \setminus A$. We first show that if Y has bounded size, then we can return the original instance of the problem and stop.

▶ **Reduction Rule 1.** *If $|Y| \leq 2(k_O + 2k_M)(d + 1)^2$ then return $(\mathcal{X}, w_O, w_M, W, k_O, k_M, d)$.*

We can apply this rule because if $|Y| \leq 2(k_O + 2k_M)(d + 1)^2$ then $|X| = |Y| + |A| \leq 2(k_O + 2k_M)(d + 1)^2 + (k_O + k_M)(d + 3) \leq 5(k_O + k_M)(d + 3)^2 = \mathcal{O}((kd)^2)$. From now on, we assume that $|Y| > 2(k_O + 2k_M)(d + 1)^2$.

Because A is a feasible solution to Euclidean Embedding with Outliers, (Y, ρ) can be embedded into $\mathbb{R}^d$. In the subsequent steps, we may delete some points in A and reduce the parameter k_O. This may result in a trivial instance which we solve by applying the following rule whenever possible.

▶ **Reduction Rule 2.** *If $k_O < 0$ then return a no-answer and stop. If $k_O \geq 0$ and $A = \emptyset$ then return a yes-answer and stop.*

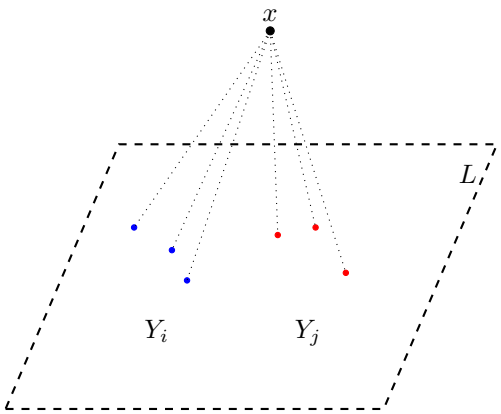


Figure 2 A pair (Y_i, Y_j) of x-compatible sets. Notice that if $j \geq i$ and $|Y_j| \leq |Y_i| = t+1$ then $(Y_i \cup Y_j, \rho)$ admits a strong embedding into $\mathbb{R}^t$ by Proposition 6 because Y_i is a metric basis for $(Y_i \cup Y_j, \rho)$. Furthermore, given a realization of (Y_i, ρ) such that the points are mapped into an affine subspace L of dimension t, the points of Y_j are mapped into L and the mapping is unique. If $(Y_i \cup \{x\}, \rho)$ is strongly t-embeddable then x is embedded in L and the embedding is unique. If $(Y_i \cup \{x\}, \rho)$ is strongly $(t+1)$-embeddable, that is, $Y_i \cup \{x\}$ is independent then by Proposition 1, the distance space is embedded into a subspace L' of dimension $t+1$ containing L and the embedding of x is unique up to the reflection with respect to L.

We greedily partition Y into sets $Y_1, \ldots, Y_\ell$ of size at most $d+1$ such that each Y_i is an inclusion-maximal independent set of points of $Y \setminus \bigcup_{j=1}^{i-1} Y_j$, that is, Y_i is an inclusion-maximal subset such that (Y_i, ρ) is strongly $(|Y_i|-1)$-embeddable.

Assume that $i \geq 1$ and the sets $Y_1, \ldots, Y_{i-1}$ are already constructed. We set $Z := Y \setminus \bigcup_{j=1}^{i-1} Y_j$. If $Z = \emptyset$, then the partition is constructed. If $Z \neq \emptyset$, then we do the following:

1. To initiate the construction of Y_i, we set $Y_i = \{x\}$ for an arbitrary point $x \in Z$ and set $h = 1$.
2. While there is a point $y \in Z \setminus Y_i$ such that $Y_i \cup \{y\}$ is independent, set $Y_i := Y_i \cup \{y\}$ and $h := h + 1$.

Because Y is d-embeddable, Z is strongly t-embeddable for some $t \leq d$ and contains an independent set of size t by Proposition 4. By Proposition 6, the described procedure will construct a set Y_i of size $t+1 \leq d+1$ such that (Y_i, ρ) is strongly t-embeddable. Notice that by Proposition 6, Y_i is a metric basis for (Z, ρ).

Let $x \in A$. We say that Y_i for $i \in \{1, \ldots, \ell\}$ is *x-compatible* if $(Y_i \cup \{x\}, \rho)$ is embeddable in $\mathbb{R}^d$. Otherwise, Y_i is *x-incompatible*. Similarly, for two distinct sets Y_i, Y_j for $i, j \in \{1, \ldots, \ell\}$, the pair (Y_i, Y_j) is *x-compatible* if $(Y_i \cup Y_j \cup \{x\}, \rho)$ is embeddable in $\mathbb{R}^d$, and (Y_i, Y_j) is *x-incompatible* otherwise (see Figure 2).

Notice that if Y_i is x-incompatible then for each solution to the instance either x is an outlier, or Y_i contains an outlier, or at least one distance between the points $Y_i \cup \{x\}$ should be modified.

For each $i \in \{1, \ldots, l\}$, $1 \leq |Y_i| \leq d+1$. We partition the family $\{Y_1, \ldots, Y_\ell\}$ of sets into parts $C_1, \ldots, C_{d+1}$ according to their size (some parts may be empty). Formally, for $h \in \{1, \ldots, d+1\}$, $C_h = \{Y_i \mid 1 \leq i \leq \ell \text{ and } |Y_i| = h\}$.

Consider the part C_h for some $h \in \{1, \ldots, d+1\}$. For a point $x \in A$, we say that two x-compatible sets $Y_i, Y_j \in C_h$ are *x-equivalent* if the pair (Y_i, Y_j) is x-compatible, that is, $(Y_i \cup Y_j \cup \{x\}, \rho)$ is embeddable into $\mathbb{R}^d$. We show the following claim.

▷ Claim 11 (⋆). x-equivalency is an equivalence relation on the family of x-compatible sets in C_h. Furthermore, if $Y_i, Y_j \in C_h$ are x-equivalent and the pair (Y_j, Y_s) for $Y_s \in C_{h'}$ with $h' \leq h$ is x-compatible then (Y_i, Y_s) is x-compatible.

The definition of x-equivalency implies the following property.

▷ Claim 12. Let R_1 and R_2 be two distinct classes of x-equivalent x-compatible sets. Then for any solution (O, D) to the considered instance, either (i) $x \in O$, or (ii) for each set Y_i in R_1 there is a $y \in Y_i$ such that $y \in O$ or there exists a z such that $\{y, z\} \in D$, or (iii) for each set Y_j in R_2 there is a $y \in Y_j$ such that $y \in O$ or there exists a z such that $\{y, z\} \in D$.

We use this property in the following rule. Because $|Y| > 2(k_O + 2k_M)(d+1)^2$ and each Y_i is of size at most $d+1$, there is $h \in \{1, \ldots, d+1\}$ with C_h being of size at least $2(k_O + 2k_M) + 1$. Let h^* be the maximum value of $h \in \{1, \ldots, d+1\}$ with this property.

▶ **Reduction Rule 3.** *If there is $x \in A$ such that each x-equivalence class R of x-compatible sets of C_{h^*} has size at most $|C_{h^*}| - (k_O + 2k_M) - 1$, then set $X := X \setminus \{x\}$, $A := A \setminus \{x\}$, and $k_O := k_O - 1$.*

After the exhaustive application of Reduction Rule 3, we have that for each $x \in A$, there is an x-equivalence class R_x of x-compatible sets in C_{h^*} that contains at least $|C_{h^*}| - k_O - 2k_M \geq (k_O + 2k_M) + 1$ sets and all other classes in C_{h^*} contain at most $k_O + 2k_M$ elements combined. We say that R_x is *large*. We exhaustively apply the following rule using the fact that R_x contains at least $(k_O + 2k_M) + 1$ sets.

▶ **Reduction Rule 4.** *If there is $x \in A$ such that there are at least $k_O + k_M + 1$ sets $Y_j \in C_h$ for any $h \leq h^*$ such that (Y_i, Y_j) is x-incompatible for some $Y_i \in R_x$, then set $X := X \setminus \{x\}$, $A := A \setminus \{x\}$, and $k_O := k_O - 1$.*

In the next crucial step of our algorithm, we identify *important* sets Y_i for $\in \{1, \ldots, \ell\}$ and delete all other sets that are irrelevant.

We apply the following *marking* procedure that labels important sets Y_i for $i \in \{1, \ldots, \ell\}$. For every $x \in A$, we do the following:

- mark every Y_i in C_h for $h > h^*$,
- for each $x \in A$, mark arbitrary $(k_O + 2k_M) + 1$ sets of R_x,
- for each $x \in A$ and each $h \in \{1, \ldots, h^*\}$, mark each set $Y_j \in C_h$ such that there is a set $Y_i \in R_x$ such that the pair (Y_i, Y_j) is x-incompatible.

We set $I = \{i \mid 1 \leq i \leq \ell \text{ and } Y_i \text{ is marked}\}$ and $Y' = \bigcup_{i \in I} Y_i$, and define $X' = A \cup Y'$.

▷ Claim 13 (⋆). The instance $(\mathcal{X} = (X, \rho), w_O, w_M, W, k_O, k_M, d)$ of Euclidean Embedding Editing is equivalent to the instance $(\mathcal{X}' = (X', \rho), w_O, w_M, W, k_O, k_M, d)$.

We have that the original instance $(\mathcal{X} = (X, \rho), w_O, w_M, W, k_O, k_M, d)$ and the obtained instance $(\mathcal{X}' = (X', \rho), w_O, w_M, W, k_O, k_M, d)$ of Euclidean Embedding Editing are equivalent. We prove that $|X'| = \mathcal{O}(k^2 d^2)$.

▷ Claim 14 (⋆). $|X'| \leq 9(k_O + k_M)^2 (d+3)^2$.

▷ Claim 15 (⋆). The overall running time of our algorithm is polynomial.

To compress the weights, we use the standard technique (see [16]) based on the algorithm of Frank and Tardos [19]. ◀

4 FPT algorithm for Euclidean Embedding Editing parameterized by solution size and dimension

In this section, we give our FPT algorithm for Euclidean Embedding Editing.

We prepare for it by recalling the relevant definitions and facts from [4]. In what follows, let R be a real closed field and $\ell \in \mathbb{N}$. Let $\mathcal{P}_\ell \subset R[X_1, \ldots, X_t]$ be a finite set of s polynomials each of degree at most ℓ.

▶ **Definition 16** ([4]). *A $\mathcal{P}_\ell$-*atom *is one of $P = 0, P \neq 0, P \geq 0, P \leq 0$, where P is a polynomial in $\mathcal{P}_\ell$ and a* quantifier-free $\mathcal{P}_\ell$-formula *is a formula constructed only from $\mathcal{P}_\ell$-atoms together with the logical connectives $\wedge$, $\vee$ and $\neg$.*

▶ **Proposition 17** (Theorem 13.13, [4]). *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 $\mathcal{P}_\ell$-formula. There exists an algorithm to decide the truth of the sentence with complexity*[1] *$s^{t+1} \cdot \ell^{\mathcal{O}(t)}$ in D where D is the ring generated by the coefficients of the polynomials in $\mathcal{P}_\ell$.*

▶ **Definition 18.** *Consider points $x_0, x_1, \ldots, x_r$ of distance space (X, ρ) and a set Z of pairs in X. For every $i < j \in \{0, \ldots, r\}$, if $\{x_i, x_j\} \notin Z$, then $\hat{\rho}_{i,j} = \rho(x_i, x_j)$, otherwise $\hat{\rho}_{i,j} = z_{i,j}$ where $z_{i,j}$ is an indeterminate. Then the* Z-Augmented Cayley-Menger determinant *is obtained from the Cayley-Menger determinant by replacing each $\rho_{i,j}$ with $\hat{\rho}_{i,j}$.*

▶ **Observation 19.** *Consider $r+1$ points $x_0, x_1, \ldots, x_r$ of distance space (X, ρ) and a set Z of pairs in X. The Z-Augmented Cayley-Menger determinant is a multi-variate polynomial with real coefficients, over the set $\{z_{i,j} \mid i < j,\ \{x_i, x_j\} \in Z\}$ of indeterminates and where each monomial has degree at most $2 \cdot (r+1)$.*

▶ **Theorem 20.** *Euclidean Embedding Editing is solvable in $(d(k_O + k_M))^{\mathcal{O}(d+k_M+k_O)} + n^{\mathcal{O}(1)}$ time.*

Proof. Our FPT algorithm for Euclidean Embedding Editing works as follows. We begin by running the compression algorithm (Theorem 8). Following this, based on insights from Proposition 4, we reduce the task of solving the resulting instance to the task of testing a bounded number (in terms of k, d) of sentences with purely existential quantifications as described in Proposition 17, where s, ℓ, t are all bounded by functions of k and d.

We now formalize this idea. Let $(\mathcal{X} = (X, \rho), w_O, w_M, W, k_O, k_M, d)$ be the given instance of Euclidean Embedding Editing. We run the algorithm of Theorem 8 on this instance. Recall that this algorithm either solves the instance or produces an equivalent instance $(\mathcal{X}' = (X', \rho), w'_O, w'_M, W', k'_O, k'_M, d)$ such that $X' \subseteq X$, and for $k = k_O + k_M$ and a bound $\tau(k, d) = 2^{\mathcal{O}((kd)^{12})}$, we have that $|X'| = \mathcal{O}((kd)^2)$. Note that since X' has size bounded by $\mathcal{O}((kd)^2)$, we may assume that k'_O is bounded by $\mathcal{O}((kd)^2)$ and k'_M is bounded by $\mathcal{O}((kd)^4)$.

Since we now have a bounded number of points in X', it is straightforward to guess a set $Z_O \subseteq X'$ of at most k'_O outliers and a subset Z_M of pairs from $X'' = X' \setminus Z_O$ of size at most k'_M such that (i) $w'_O(Z_O) + w'_M(Z_M) \leq W'$ and (ii) if there is a solution to the given instance, then there is a modification of the pairs in Z_M such that (X'', ρ'') is embeddable into $\mathbb{R}^d$, where ρ'' differs from ρ (restricted to X'') only among the pairs in Z_M. We next guess the embedding dimension of (X'', ρ''), denoted by r. Note that $r \leq d$.

[1] The measure of complexity here is the number of field operations.

By Proposition 4, we know that (X'', ρ'') is r-embeddable if and only if there is a subset $\hat{X} = \{x_0, \ldots, x_r\}$ of X'' such that (i) $(-1)^{j+1} CM(x_0, x_1, \ldots, x_j) > 0$ for $1 \leq j \leq r$, and (ii) for any $x, y \in X'' \setminus \hat{X}$, $CM(x_0, x_1, \ldots, x_r, x) = CM(x_0, x_1, \ldots, x_r, y) = CM(x_0, x_1, \ldots, x_r, x, y) = 0$, where the matrices are derived from the distance matrix $D(\rho'')$.

We next guess $\hat{X}$. It remains to verify that there is a modification of the distances between each pair in Z such that the resulting space (X'', ρ'') is r-embeddable, i.e., satisfies the above two properties. To test this, we construct the formula F obtained by taking the conjunction of the atoms below and invoke Proposition 17 on the sentence ϕ obtained by prepending F with existential quantifications for every indeterminate $\{z_{i,j} \mid i < j,\ \{x_i, x_j\} \in Z\}$:

1. $(-1)^{j+1} CM_{Z_M}(x_0, x_1, \ldots, x_j) > 0$, where $1 \leq j \leq r$.
2. $CM_{Z_M}(x_0, x_1, \ldots, x_r, x) = 0$ for every $x \in X'' \setminus \hat{X}$.
3. $CM_{Z_M}(x_0, x_1, \ldots, x_r, x, y) = 0$ for every $x, y \in X'' \setminus \hat{X}$.
4. $z_{i,j} \geq 0$ for each indeterminate defined by Z_M. ◀

In the conclusion of the section, we remark that the parameter dependence can be improved for Euclidean Embedding with Outliers.

▶ **Theorem 21** (⋆). *Euclidean Embedding with Outliers can be solved in time* $n^{O(1)} \cdot min\{(d+3)^k, 2^{d+k}\}$.

5 FPT-Approximation for Unweighted Euclidean Embedding with Outliers

In this section, we give an FPT-time 2-approximation for Unweighted Euclidean Embedding with Outliers (UEEO) parameterized by the dimension d. Recall that in the optmization version of this problem, the input is $(\mathcal{X} = (X, \rho), d)$ where (X, ρ) is a distance space and the goal is to output a d-outlier set of minimum size.

Our main idea is to give a randomized algorithm generating sequences $U_0, \ldots, U_{d+1}$ and $A_1, \ldots, A_{d+2}$, where each A_i is a d-outlier set that is associated with the independent set U_{i-1} of size $i-1$. The construction of these sequences is roughly as follows, starting from $U_0 = \emptyset$ and proceeding iteratively. For each U_{i-1}:

1. Any element that cannot be d-embedded along with U_{i-1} gets added to A_i.
2. We compute those elements that can be added to U_{i-1} without increasing its embedding dimension, and associate a graph with these such that every d-outlier set disjoint from U_{i-1} includes a vertex cover of this graph. We then 2-approximate the minimum vertex cover of this graph and add it to A_i.
3. For the remaining elements (call this set W):
   a. Take all of W in A_i (which makes A_i a 2-approximation if at least half of W is in the solution),
   b. Uniformly at random add an element of W to U_{i-1}, resulting in U_i (which gives an independent set with sufficiently high probability if at least half of W is *not* in some fixed minimum d-outlier set).

▶ **Theorem 22.** Unweighted *Euclidean Embedding with Outliers can be 2-approximated in* $2^d \cdot n^{\mathcal{O}(1)}$ *time by a randomized algorithm with one-sided constant probability of error.*

Proof. Let S_{opt} be a hypothetical minimum sized d-outlier set. Let $X^\star = X \setminus S_{\mathsf{opt}}$. Let us assume that $(X^\star, \rho)$ is strongly d-embeddable. Else, it is strongly d'-embeddable for some $d' \leq d$, in which case we guess d' and set $d := d'$. In what follows, we give a randomized algorithm that aims to obtain a metric basis of $(X^\star, \rho)$ if certain pre-conditions are satisfied.

Alg-3(X, ρ, d):

1. Set $U_0 := \emptyset$.
2. For $i = 1$ to $d + 2$ do as follows.
   a. Construct $\mathcal{C}^i_{\mathsf{comp}}$, the set of elements $y \in X \setminus U_{i-1}$ such that $(U_{(i-1)} \cup \{y\}, \rho)$ is $(i-2)$-embeddable.
   b. Construct $\mathcal{C}^i_{\mathsf{def}}$, the set of elements $y \in X \setminus U_{i-1}$ such that $(U_{(i-1)} \cup \{y\}, \rho)$ is not d-embeddable.
   c. Construct $\mathcal{C}^i_{\mathsf{incomp}}$, the set of elements $y \in X \setminus (U_{i-1} \cup \mathcal{C}^i_{\mathsf{comp}} \cup \mathcal{C}^i_{\mathsf{def}})$ such that $(U_{(i-1)} \cup \{y\}, \rho)$ is not $(i-2)$-embeddable.
   d. Select an element u_i uniformly at random from $\mathcal{C}^i_{\mathsf{incomp}}$.
   e. Set $U_i := U_{(i-1)} \cup \{u_i\}$.

▷ Claim 23 (⋆). For every $i \in \{0, \ldots, d+1\}$, the following holds:

1. $\mathcal{C}^i_{\mathsf{def}}$ and $\mathcal{C}^i_{\mathsf{comp}}$ are disjoint.
2. Every d-outlier set of (X, ρ) disjoint from U_i contains $\mathcal{C}^{i+1}_{\mathsf{def}}$.
3. If for all $j \leq i$, $\mathcal{C}^j_{\mathsf{incomp}} \neq \emptyset$, then U_i is an independent set of size i.

We next argue that for each $i \in [d+1]$, the independent set U_i is preserved in $X^\star = X \setminus S_{\mathsf{opt}}$, with sufficiently high probability if certain conditions are met.

▷ Claim 24 (⋆). Let $i \in [d+1]$ such that for all $j \leq i$, $|\mathcal{C}^j_{\mathsf{incomp}} \cap S_{\mathsf{opt}}| < \frac{|\mathcal{C}^j_{\mathsf{incomp}}|}{2}$. Then, $\Pr[U_i \subseteq X^\star] \geq \frac{1}{2^i}$ and U_i is an independent set of size i.

Next, we build a series of d-outlier sets $A_1, \ldots, A_{d+2}$ as follows. For each $i \in [d+2]$, create an instance of the Vertex Cover problem on the graph G^i_{comp}, where $V(G^i_{\mathsf{comp}}) = \mathcal{C}^i_{\mathsf{comp}}$, and there is an edge between two elements $x, y \in \mathcal{C}^i_{\mathsf{comp}}$ if they $U_{i-1} \cup x, y$ is not $(i-2)$-embeddable. Using a factor-2 approximation algorithm for Vertex Cover, obtain a set Q_i such that $|Q_i| \leq 2|O_i|$, where O_i is a minimum vertex cover of G^i_{comp}. Finally, set $A_i := Q_i \cup \mathcal{C}^i_{\mathsf{incomp}} \cup \mathcal{C}^i_{\mathsf{def}}$. Once every A_i is computed, we compute $i^\star = \arg\min_{i \in [d+2]} |A_i|$ and return $A_{i^\star}$.

▷ Claim 25 (⋆). Let $\ell \in [d+1]$ be the least integer such that for all $j < \ell$, it holds that

$$|\mathcal{C}^j_{\mathsf{incomp}} \cap S_{\mathsf{opt}}| < \frac{|\mathcal{C}^j_{\mathsf{incomp}}|}{2}.$$

Then, A_ℓ is a factor-2 approximation with probability at least $\frac{1}{2^\ell}$.

Since we compute $i^\star = \arg\min_{i \in [d+2]} |A_i|$ and return $A_{i^\star}$, by Claim 25 we are guaranteed that $|A_{i^\star}| \leq 2|S_{\mathsf{opt}}|$ with probability at least $\frac{1}{2^{d+1}}$.

Running Time Analysis. We have already argued that we succeed with probability $\frac{1}{2^{d+1}}$. Thus, we can boost the success probability by independently running our polynomial-time algorithm 2^{d+1} times and returning the minimum size solution among these runs. Thus, the probability that the algorithm fails in all of the independent runs is upper bounded by $(1 - \frac{1}{2^{d+1}})^{2^{d+1}} \leq 1 - \frac{1}{e}$. Moreover, the total running time is upper bounded by $2^d n^{\mathcal{O}(1)}$. ◀

6 Lower bounds

In this section, we complement our algorithmic results by proving computational lower bounds. These lower bounds also motivate our choice of parameterizations. First, we show that Euclidean Embedding Editing is Para-NP-hard when parameterized by d. More precisely, we show that even the *unweighted* versions of both Euclidean Embedding with Outliers and Euclidean Metric Violation Distance are NP-hard even for $d = 1$.

▶ **Theorem 26** ($\star$). *Euclidean Embedding with Outliers is strongly* NP*-hard even for instances with unit weights, integer distances, and* $d = 1$.

▶ **Theorem 27** ($\star$). *Euclidean Metric Violation Distance is strongly* NP*-hard for the instances with unit weights, integer distances, and* $d = 1$.

Finally, we prove that Euclidean Embedding Editing is W[1]-hard when parameterized by $k_O + k_M$ only. Moreover, the hardness holds even for the unweighted variant of Euclidean Embedding with Outliers.

▶ **Theorem 28** ($\star$). *Euclidean Embedding with Outliers parameterized by* k *is* W[1]*-hard for* n*-point instances with unit weights and integer distance matrices with entries in* $\mathcal{O}(n)$.

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 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.

3 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5, 2009.

4 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

5 Leonard Mascot Blumenthal. *Theory and applications of distance geometry*. Clarendon Press, Oxford, 1953.

6 Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, New York, 2005.

7 Arthur Cayley. On the theory of determinants. *Philosophical Magazine*, 19:1–16, 1841.

8 Moses Charikar and Ruiquan Gao. Improved approximations for ultrametric violation distance. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1704–1737. SIAM, 2024. doi:10.1137/1.9781611977912.68.

9 Vincent Cohen-Addad, Chenglin Fan, Euiwoong Lee, and Arnaud De Mesmay. Fitting metrics and ultrametrics with minimum disagreements. In *Proceedings of the 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 301–311. IEEE, 2022. doi:10.1109/FOCS54457.2022.00035.

10 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.

11 J. Dattorro. *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2008.

12 E. D. Demaine, F. Gomez-Martin, H. Meijer, D. Rappaport, P. Taslakian, G. T. Toussaint, T. Winograd, and D. R. Wood. The distance geometry of music. *Computational Geometry*, 42(5):429–454, July 2009. doi:10.1016/J.COMGEO.2008.04.005.

13 Michel Deza, Monique Laurent, and Robert Weismantel. *Geometry of cuts and metrics*, volume 2. Springer, 1997.

14 L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 1655–1663. IEEE, 2001.

15 Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. Euclidean distance matrices: essential theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 32(6):12–30, 2015. doi:10.1109/MSP.2015.2398954.

16 Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer System Sciences*, 84:1–10, 2017. doi:10.1016/J.JCSS.2016.06.004.

17 B. Everitt and S. Rabe-Hesketh. *The Analysis of Proximity Data*. Arnold, London, 1997.

18 Chenglin Fan, Benjamin Raichel, and Gregory Van Buskirk. Metric violation distance: Hardness and approximation. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 196–209. SIAM, 2018. doi:10.1137/1.9781611975031.14.

19 András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

20 Anna C. Gilbert and Lalit Jain. If it ain't broke, don't fix it: Sparse metric repair. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 612–619. IEEE, 2017. doi:10.1109/ALLERTON.2017.8262793.

21 T. F. Havel and K. Wüthrich. An evaluation of the combined use of nuclear magnetic resonance and distance geometry for the determination of protein conformations in solution. *Journal of Molecular Biology*, 182(2):281–294, 1985.

22 Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–33. IEEE, 2001. doi:10.1109/SFCS.2001.959878.

23 Piotr Indyk and Jiri Matousek. Low-distortion embeddings of finite metric spaces. In *in Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

24 Viren Jain and Lawrence K. Saul. Exploratory analysis and visualization of speech and music by locally linear embedding. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 984–987. IEEE, 2004. doi:10.1109/ICASSP.2004.1326712.

25 Leo Liberti and Carlile Lavor. *Euclidean distance geometry*, volume 3. Springer, 2017.

26 Nathan Linial. Finite metric-spaces – combinatorics, geometry and algorithms. In *Proceedings of the International Congress of Mathematicians, Vol. III*, pages 573–586, Beijing, 2002. Higher Ed. Press.

27 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.

28 Karl Menger. Untersuchungen Ã¼ber allgemeine metrik. *Mathematische Annalen*, 100:75–163, 1928.

29 N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal. Locating the nodes: Cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine*, 22(4):54–69, July 2005. doi:10.1109/MSP.2005.1458287.

30 Isaac J. Schoenberg. Remarks to maurice frechet's article "sur la definition axiomatique d'une classe d'espace distances vectoriellement applicable sur l'espace de hilbert". *Annals of Mathematics*, 36:724–732, 1935.

31 R. Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function, part i. *Psychometrika*, 27:125–140, 1962.

32 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.

33 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.

34 W. Torgerson. *Theory and Methods of Scaling*. Wiley, New York, 1958.

35 Warren S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

36 Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 988–995. IEEE, 2004. doi:10.1109/CVPR.2004.256.

Signotopes with Few Plus Signs

Helena Bergold
Institut für Informatik, Freie Universität Berlin, Germany

Lukas Egeling
Department of Computer Science, ETH Zurich, Switzerland

Hung P. Hoang
Algorithm and Complexity Group, Faculty of Informatics, TU Wien, Austria

Abstract

Arrangements of pseudohyperplanes are widely studied in computational geometry. A rich subclass of pseudohyerplane arrangements, which has gained more attention in recent years, is the so-called signotopes. Introduced by Manin and Schechtman (1989), the higher Bruhat order is a natural order of r-signotopes on n elements, with the signotope corresponding to the cyclic arrangement as the minimal element. In this paper, we show that the lower (and by symmetry upper) levels of this higher Bruhat order contain the same number of elements for a fixed difference $n - r$. This result implies that given the difference $d = n - r$ and p, the number of one-element extensions of the cyclic arrangement of n hyperplanes in $\mathbb{R}^d$ with at most p points on one side of the extending pseudohyperplane does not depend on n, as long as $n \geq d + p$.

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

Keywords and phrases flip graph, higher Bruhat order, signotope, counting, Ferrers diagram, one-element extension

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.16

Related Version *arXiv link*: `https://arxiv.org/abs/2411.19208` [3]

Funding *Hung P. Hoang*: Austrian Science Foundation (FWF, project Y1329 START-Programm).

Acknowledgements We would like to thank Bernd Gärtner for stimulating discussions on the paper.

1 Introduction

Arrangements of geometrical and topological objects play an important role in computational geometry. A classical example is line arrangements, which consist of a family of pairwise intersecting lines. For various occasions, it is useful to study the combinatorial structure instead of the actual geometry. For this reason, pseudoline arrangements have gotten more and more attention after Levi's introductory paper [12]. In contrast to the geometric setting, it is possible to encode the structure of pseudolines efficiently. To determine the size of an optimal encoding for point sets, Knuth [11] studied the number of the so-called CC-Systems. By duality, there are the same number of pseudoline arrangements, which is $2^{\Theta(n^2)}$ [11, 7]. However, the precise leading constant in the exponent is still unknown, which leaves Knuth's question about the optimal encoding size open [11]. Recently, Cortés Kühnast, Dallant, Felsner, and Scheucher [6] determined the best-known lower bound of $2^{0.2721n^2}$ pseudoline arrangements. To tackle this challenging task, they used 3-signotopes for an encoding. Since the order on the elements itself is fixed, the set of 3-signotopes has less symmetric copies, which makes the encoding easier. As shown by Felsner and Weil [9] there is a bijection between pseudoline arrangements and 3-signotopes.

© Helena Bergold, Lukas Egeling, and Hung P. Hoang;
licensed under Creative Commons License CC-BY 4.0

In this paper, we consider higher dimensions. Pseudohyperplane arrangements in $\mathbb{R}^d$ are combinatorially encoded as uniform acyclic oriented matroids of rank $r = d + 1$. For more about oriented matroids, see [5]. The asymptotic number of oriented matroids of rank r is $2^{\Theta(n^{r-1})}$. However, little is known about the constant in the exponent. Instead of the general case, we consider signotopes of rank r, a rich subclass of pseudohyperplane arrangements, of which there are also $2^{\Theta(n^{r-1})}$, see [2, 4]. The leading constant is also unknown. In this paper, we determine the number of signotopes with few plus (or symmetrically, few minus) signs. This may be a step towards determining the precise number of all signotopes in the future.

To formally define signotopes, let $[a, b]$ denote the set $\{x \in \mathbb{Z} \mid x \geq a \wedge x \leq b\}$ for two integers a and b. Moreover, let $[n]$ be $[1, n]$ and $\binom{[a,b]}{r}$ denote the set of all subsets of size r of $[a, b]$. We call such a subset an *r-subset* of $[a, b]$.

▶ **Definition 1** (Signotope). *For $r \geq 1$, a* signotope of rank r *(or an* r-signotope*) on n elements is a sign function $\sigma : \binom{[n]}{r} \to \{+, -\}$, such that for every $(r + 1)$-subset $X = \{x_1, \ldots, x_{r+1}\}$ of $[n]$ with $x_1 < \cdots < x_{r+1}$, there is at most one sign change in the sequence*

$$\Big(\sigma(X \setminus \{x_1\}), \ldots, \sigma(X \setminus \{x_{r+1}\})\Big).$$

The set of all r-signotopes on n elements is denoted by $\mathcal{S}(n, r)$.

For fixed n and r, we can define an order on the set $\mathcal{S}(n, r)$ of all r-signotopes on n elements based on the number of $+$-signs. Specifically, two sign functions σ and $\tilde{\sigma}$ differ in a *single step*, if $\sigma^{-1}(+) \subset \tilde{\sigma}^{-1}(+)$ and $|\tilde{\sigma}^{-1}(+)| = |\sigma^{-1}(+)| + 1$. The transitive closure of the single step relation gives the higher Bruhat order.

▶ **Definition 2** (Higher Bruhat Order). *For $r \geq 1$, the* higher Bruhat order $B(n, r)$ *is the partial order $\preceq$ on the elements of $\mathcal{S}(n, r)$ with the transitive closure of the single step.*

Signotopes are the elements of the higher Bruhat order, which were introduced by Manin and Schechtman [13] as a higher dimensional equivalence of the weak Bruhat order for permutations. They were further studied by Kapranov and Voevodsky [10], Ziegler [15], and later Felsner and Weil [8]. In these works they give several geometric interpretations. The most important one for our purposes is the representation of an r-signotope on $[n]$ as a one-element extension of the cyclic arrangement of n hyperplanes in $\mathbb{R}^{n-r}$ [15].

▶ **Definition 3** (Cyclic arrangement). *The* cyclic arrangement $\mathbf{X}_c^{n,d}$ *is the set $\{H_1, \ldots, H_n\}$ in $\mathbb{R}^d$ given by*

$$H_i = \{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_1 + t_i x_2 + \cdots + t_i^{d-1} x_d + t_i^d = 0\}$$

for $i \in [d]$, with arbitrary real parameters $t_1 < \cdots < t_n$.

A *point* of the cyclic arrangement $\mathbf{X}_c^{n,n-r}$ is the intersection of $n - r$ hyperplanes and can be identified with the $n - (n - r) = r$ hyperplanes which do not contain this point. In $\mathbf{X}_c^{n,n-r}$, along each *line* (i.e., the intersection of $n - r - 1$ hyperplanes), the corresponding r-subsets of the crossing points are ordered lexicographically. A *one-element extension* of $\mathbf{X}_c^{n,n-r}$ is obtained by adding a pseudohyperplane to $\mathbf{X}_c^{n,n-r}$ in general position (i.e., the pseudohyperplane does not contain any point of $\mathbf{X}_c^{n,n-r}$). Additionally, the *extending pseudohyperplane* (i.e., the new pseudophyperplane) is oriented; that is, it divides $\mathbb{R}^d$ into two sides, a positive and a negative side. If a point lies in the positive side, the corresponding r-subset is mapped to $+$, and that correpsonding to a point in the negative side is mapped to $-$. Monotonicity of this sign mapping on r-subsets follows from the lexicographical order of the r-subsets along the lines. For an illustration see Figure 1.

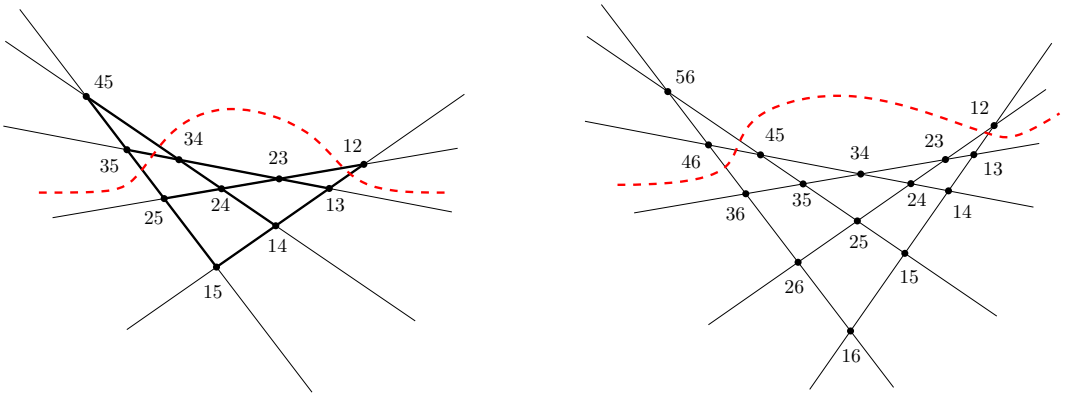


Figure 1 Two examples of one-element extensions of the cyclic arrangements $\mathbf{X}_c^{5,2}$ (left) and $\mathbf{X}_c^{6,2}$ (right). The red dashed lines are the extending pseudolines. To avoid clusters, we label each point of the arrangement by the lines that contain it (e.g., point 12 is the intersection of lines H_1 and H_2). The positive sides are the sides above the extending pseudolines. The corresponding two signotopes are mapped to each other in our bijection for Theorem 1.

In this paper, we count the number of signtopes in a given level of the higher Bruhat order (i.e., the number of signotopes with a given number of +-signs). We show that the bottom (and by symmetry the top) levels in the Bruhat order are the same whenever $n-r$ is the same. More precisely, we denote by $\mathcal{S}_{\leq p}(n,r)$ the set of all r-signotopes σ on n elements with at most p +-signs (i.e., $|\sigma^{-1}(+)| \leq p$). Our main result is then stated as follows.

▶ **Theorem 1.** *Let $n, \tilde{n}, r, \tilde{r}, d, p$ be natural numbers such that $n - r = \tilde{n} - \tilde{r} = d$ and $p \leq \min\{r, \tilde{r}\}$. Then there exists a bijection between $\mathcal{S}_{\leq p}(n,r)$ and $\mathcal{S}_{\leq p}(\tilde{n},\tilde{r})$ that preserves the relation $\preceq$.*

In other words the theorem states that the lower levels of the higher Bruhat order contain the same number of elements for a fixed difference $n-r$. Note that by symmetry the same holds for the upper levels, i.e., if there are $\leq r$ minus signs. Using the representation of signotopes as one-element extensions of the cyclic arrangement, the geometric interpretation of the theorem above is as follows.

▶ **Corollary 4.** *Given $d, p \geq 0$, for any $n \geq d+p$, the number of one-element extensions of the cyclic arrangement $\mathbf{X}_c^{n,d}$ with at most p points on one side of the extending pseudohyperplane is a constant (which depends only on d and p).*

The representation also allows us to prove the theorem. It turns out that whenever there are $\leq r$ plus signs (respectively minus signs), we only need to consider the local structure of the one-element extension. In particular, this local structure only depends on p and $n-r$, and not the exact values of n and r themselves; this insight gives rise to the bijection in Theorem 1. In Section 2 we give a more detailed outline and in Sections 3–6 we give the technical proof. In order to show this dependency, we use a generalization of the Ferrers diagrams. These diagrams are geometric illustrations of integer partitions [1], which are classical objects in number theory and combinatorics. This connection allows us not only to prove the bijection but also to quantify the size of $\mathcal{S}_{\leq p}(n,r)$ in Section 7.

2 Proof overview for Theorem 1

In this paper, since we discuss r-signotopes on $[n]$ where $n-r$ is fixed, it is more convenient to work with the complements of the r-subsets. Hence in Section 2.1, we introduce the notion of co-signotopes with related definitions analogous to the signotopes and rephrase Theorem 1 in terms of this new notion. Afterwards, we provide an overview of the proof.

2.1 Co-signotopes

For convenience, we write each subset as a sequence of its elements in the increasing order; that is we consider an r-subset $\{a_1, \ldots, a_r\}$ with $a_1 < \cdots < a_r$ as the tuple $(a_1, \ldots, a_r)$. This way, we can order the subsets in the lexicographical order of the tuple representations.

- For a function f, a d-subset $B = (b_1, \ldots, b_d)$ of $[n]$, and an index $i \in [d]$, let $B_i := B \setminus \{b_i\}$. Then the (f, B, i)-*series* is the sequence of $f(B_i \cup \{x\})$ for $x \in [n] \setminus B_i$ in the increasing order. In other words, if $[n] \setminus B_i = \{x_1, \ldots, x_{n-d+1}\}$ with $x_1 < \cdots < x_{n-d+1}$, then the (f, B, i)-series is exactly the sequence

  $$(f(B_i \cup \{x_1\}), \ldots, f(B_i \cup \{x_{n-d+1}\})).$$

  When f is the identity function, then this corresponds to the sequence $(B_i \cup \{x_1\}, \ldots, B_i \cup \{x_{n-d+1}\})$, and we call it simply the (B, i)-series. Note that the (B, i)-series contains B.
- For $0 \leq d < n$, a *co-signotope of rank* d (or a d*-co-signotope*) on n elements is a sign function $\tau : \binom{[n]}{d} \to \{+, -\}$, such that for every d-subset B of $[n]$ and index $i \in [d]$, there is at most one sign change in the (τ, B, i)-series.
- The set of all d-co-signotopes on n elements is denoted by $\bar{\mathcal{S}}(n, d)$.
- Given a sign function τ, a subset R is a $+$*-subset* of τ if $\tau(R) = +$, and it is a $-$*-subset* of τ if $\tau(R) = -$. The set of all d-co-signotopes on n elements with exactly p $+$-subsets is denoted by $\bar{\mathcal{S}}_p(n, d)$. The set of all d-co-signotopes on n elements with at most p $+$-subsets is denoted by $\bar{\mathcal{S}}_{\leq p}(n, d)$.
- The *complementary higher Bruhat order* $\bar{B}(n, d)$ is the partial order $\preceq_{\bar{\mathcal{S}}}$ on the elements of $\bar{\mathcal{S}}(n, d)$ with the transitive closure of the single step.

The following observation follows immediately from the definitions of signotopes and co-signotopes.

▶ **Observation 5.** *For non-negative integers n, d, p such that $d \leq n$, there is an isomorphism between $S_p(n, n-d)$ and $\bar{S}_p(n, d)$ that preserves the single step relation.*

Then we can rephrase Theorem 1 in terms of co-signotopes as follows.

▶ **Theorem 1** (Rephrased). *Let $n, \tilde{n}, d, p$ be integers such that $0 \leq p \leq \min\{n, \tilde{n}\} - d$. Then there exists a bijection between $\bar{\mathcal{S}}_{\leq p}(n, d)$ and $\bar{\mathcal{S}}_{\leq p}(\tilde{n}, d)$ that preserves the relation $\preceq_{\bar{\mathcal{S}}}$.*

2.2 Proof strategy of Theorem 1

Let $G_{n,d}$ be the underlying graph of the cyclic arrangement $\mathbf{X}_c^{n,d}$; that is, its vertices are the points of $\mathbf{X}_c^{n,d}$, and its edges are the segments of the lines of $\mathbf{X}_c^{n,d}$ such that the segments have two points of $\mathbf{X}_c^{n,d}$ as endpoints and contain no other points in the interior. In other words, the vertices of $G_{n,d}$ are the d-subsets, and two d-subsets B and B' are connected by an edge, if they are adjacent in the (B, i)-series for some $i \in [d]$. For illustration, the bold line segments in Figure 1(left) form the edges of the graph $G_{n,d}$.

The proof of Theorem 1 hinges on the following main idea. For a co-signotope $\tau \in \bar{\mathcal{S}}_{\leq p}(n,d)$, we consider the induced subgraph of $G_{n,d}$ on the $+$-subsets of τ. Then each connected component in this induced subgraph (which we also call the $+$-component of τ) can be characterized by a structure around a "source" subset (defined in the next section). In particular, locally, the $+$-component appears like a generalized version of the Ferrers diagrams. Further, this characterization is independent from that of other $+$-components and depends only on the source subset, d, and p. With this observation, we can independently map each $+$-component of τ to that of a co-signotope in $\bar{\mathcal{S}}_{\leq p}(\tilde{n},d)$.

In order to rigorously execute the idea above, we first provide some preliminary structural insights on the $+$-subsets of a co-signotope $\tau \in \bar{\mathcal{S}}_{\leq p}(n,d)$ in Section 3. Next, in Section 4, we show how we can map one $+$-component of a co-signotope $\tau \in \bar{\mathcal{S}}_{\leq p}(n,d)$ to that of a co-signotope $\tilde{\tau} \in \bar{\mathcal{S}}_{\leq p}(\tilde{n},d)$. To ease the explanation, in this section, we assume that there is exactly one $+$-component for τ and for $\tilde{\tau}$. Here is also where we explain the connection with the Ferrers diagrams. Then in Section 5, we show how we can map one signotope in $\bar{\mathcal{S}}_{\leq p}(n,d)$ to a signotope in $\bar{\mathcal{S}}_{\leq p}(\tilde{n},d)$ based on their sets of $+$-components. Finally, Section 6 wraps up the proof and provides a simple bijection as required by Theorem 1.

3 Structure of $+$-subsets of a co-signotope in $\bar{\mathcal{S}}_p(n,d)$

Consider a co-signotope in $\bar{\mathcal{S}}_p(n,d)$ that has only one $+$-subset. This unique $+$-subset needs to be a subset which appears at the start or end of every sequence it is contained in. The only d-subsets which fulfill this requirement are $S_{n,d,i}$ where $S_{n,d,i} = (1,\dots,i,n-d+i+1,\dots,n)$. We call such an $S_{n,d,i}$ a *source subset*. For convenience, we drop the subscripts n and d, when they are clear.

Recall that $G_{n,d}$ is the underlying graph of the cyclic arrangement $\mathbf{X}_c^{n,d}$, as defined in Section 2.2. We have the following lemma on the distance from a subset to a source subset.

▶ **Lemma 6.** *Let n,d,i be integers such that $1 \leq i \leq d \leq n$. Let $B = (b_1,\dots,b_d)$ be a d-subset. Then every path between B and $S_{n,d,i}$ in $G_{n,d}$ has length at least $b_i - b_{i+1} + n - d + 1$, where we define $b_0 := 0$ and $b_{d+1} := n+1$.*

Proof. We represent each d-subset as a set of d tokens, one placed on the real line at each element of the subset. Then the act of going from one d-subset to an adjacent d-subset in $G_{n,d}$ can be seen as moving a token along the real line (in an arbitrary direction) to the next free integer. Suppose that $b_i > i$. Note that $[b_i - 1]$ contains at least i elements of S_i and exactly $i-1$ elements of B. Hence, in the process of going along a path from S_i to B in $G_{n,d}$, there must be at least one token from the interval $[b_i - 1]$ going to the interval $[b_i, n]$. Therefore, every integer in $[i+1, b_i]$ must be visited by a token at some point in the process (these integers may or may not be visited by the same token). Note that this statement is vacuously true when $b_i = i$ or when $i = 0$.

Using a symmetric argument, we also deduce that every integer in $[b_{i+1}, n-d+i]$ must also be visited by a token at some point in the process above. Further, at any step, we only move one token, and hence, at most one integer in $[i+1, b_i] \cup [b_{i+1}, n-d+i]$ is visited at any step. Hence, we conclude that the path from S_i to B must have length at least $(b_i - i) + (n - d + i - b_{i+1} + 1)$. The lemma then follows. ◀

For a d-co-signotope τ, let $G_{n,d}[\tau]$ be the induced subgraph of $G_{n,d}$ on $\tau^{-1}(+)$ (i.e., the $+$-subsets of τ). A maximal connected induced subgraph of $G_{n,d}[\tau]$ is called a *$+$-component* of τ. Then it is easy to see that each $+$-component of τ contains at least one source subset. This is actually tight for all components when $\tau \in \bar{\mathcal{S}}_p(n,d)$ with $p \leq n-d$, as stated in the following lemma.

▶ **Lemma 7.** *Let n, d, p be non-negative integers such that $p \leq n - d$. Then for $\tau \in \bar{\mathcal{S}}_p(n, d)$, each $+$-component of τ has exactly one source subset.*

Proof. Suppose we have two source subsets S_i and S_j with $i < j$. Applying Lemma 6 with $B = S_j$, a path between two source subsets S_i and S_j with $i < j$ has length at least $i - (i + 1) + n - d + 1 = n - d$. This shows that every $+$-component containing S_i and S_j has at least $n - d + 1 > p$ $+$-subsets. Since τ has p $+$-subsets, it follows that no component of $G_{n,d}[\tau]$ can contain more than one source subset. Combined with the fact that each component needs to have at least one source subset, the lemma then follows. ◀

Motivated by the lemma above, for a co-signotope $\tau \in \bar{\mathcal{S}}(n, d)$ and for $i \in [0, d]$, we denote by $\mathcal{C}_i^\tau$ the $+$-component that contains S_i ($\mathcal{C}_i^\tau$ may be empty.)

For a co-signotope $\tau \in \bar{\mathcal{S}}(n, d)$, a d-subset B, and an index $j \in [d]$, we say the (τ, B, j)-series is *left-aligned* if it starts with a $+$-subset and ends with a $-$-subset, *right-aligned* if it starts with a $-$-subset and ends with a $+$-subset, and *flat* if there is no sign change. Since every (τ, B, j)-series has $n - d + 1$ subsets, if τ has at most $(n - d)$ $+$-subsets, then no flat series has only $+$-subsets. Hence, we have the following simple observation.

▶ **Observation 8.** *Let n, d, p be non-negative integers such that $p \leq n - d$. Then for every $+$-subset B of a co-signotope $\tau \in \bar{\mathcal{S}}_p(n, d)$ and for every index $j \in [d]$, the (τ, B, j)-series is not flat.*

The following lemma characterizes which d-subsets can be in the same $+$-component as some source subset S_i.

▶ **Lemma 9.** *Let n, d, c, p, i be integers such that $1 \leq c \leq p \leq n - d$ and $0 \leq i \leq d$. Further, let τ be a co-signotope in $\bar{\mathcal{S}}_p(n, d)$. Then if the $+$-component $\mathcal{C}_i^\tau$ has size c and contains a $+$-subset $B = (b_1, \ldots, b_d)$, then we have*

(a) *For all $j \in [i]$, $b_j \leq c + j - 1$ and the (τ, B, j)-series is left-aligned; and*

(b) *For all $j \in [i + 1, d]$, $b_j \geq n - (c + d - j) + 1$ and the (τ, B, j)-series is right-aligned.*

Proof. We first consider the case when $j \in [i]$. By Lemma 6, the shortest path between B and S_i is at least $b_i - b_{i+1} + n - d + 1$. Since B and S_i are in the same $+$-component $\mathcal{C}_i^\tau$ of size c, this implies $b_i - b_{i+1} + n - d + 1 \leq c - 1$. Moreover since $b_{i+1} < b_{i+2} < \cdots < b_d \leq n$, we obtain $b_{i+1} \leq n - d + i + 1$. Together this implies

$$b_i \leq (c - 1) - (-b_{i+1} + n - d + 1) = c + i - 1 + b_{i+1} - (n - d + i + 1) \leq c + i - 1.$$

Combining this with the fact that $b_j \leq b_i - (i - j)$, we have $b_j \leq c + j - 1$ as required.

We now prove by contradiction that the (τ, B, j)-series is left-aligned. Suppose this is not the case. Combined with Observation 8, (τ, B, j)-series must then be right-aligned, i.e., all of the subsets after B in the (B, j)-series then have to be $+$-subsets. That means each element in $[b_j, n]$ appears in some $+$-subset. If $b_j = j$, then also all elements in $[b_j]$ appear in S_i, a $+$-subset. Otherwise, using the same token argument as in the proof of Lemma 6, each element in $[i + 1, b_i]$ is contained in some $+$-subset along the path from S_i to B in $\mathcal{C}_i^\tau$. Combining with the fact that $[i] \subseteq S_i$, we obtain that each element in $[b_i]$ appears in some $+$-subset. Overall, each element in $[n]$ appears in some $+$-subset.

Next, observe that going from a subset to an adjacent subset in $G_{n,d}$, we remove one element and add one element. Hence, since $\mathcal{C}_i^\tau$ has at most p elements, the number of elements appearing in at least one $+$-subset is at most $d + p - 1 \leq n - 1$. This contradicts with the conclusion above that each element in $[n]$ appears in some $+$-subset.

Hence, (a) holds. With a symmetric argument, we also obtain (b), completing the proof of the lemma. ◀

A direct consequence of the lemma above is that for a given $\tau \in \bar{\mathcal{S}}_p(n,d)$, we can identify to which $+$-component a $+$-subset belongs.

▶ **Corollary 10.** *Let n, d, p be integers such that $1 \leq p \leq n-d$. Further, let τ be a co-signotope in $\bar{\mathcal{S}}_p(n,d)$ and $B = (b_1, \ldots, b_d)$ be a $+$-subset of τ. Then B is contained in the $+$-component $\mathcal{C}_i^\tau$, where i is either the largest index in $[d]$ such that the (τ, B, i)-series is left-aligned or zero if such an index does not exist.*

4 Co-signotopes with only one $+$-component

As a building block for the bijection, in this section, we consider $\tau \in \bar{\mathcal{S}}_p(n,d)$ such that $n \geq d+p$ and τ has exactly one $+$-component. We give a characterization of such a co-signotope via a variant of high-dimensional Ferrers diagrams. For convenience, we denote by $\bar{\mathcal{S}}_{p,i}(n,d)$ the set of all co-signotopes $\tau \in \bar{\mathcal{S}}_p(n,d)$ such that $\mathcal{C}_i^\tau$ is the only non-empty $+$-component of τ.

To facilitate the proof of the bijection we define a function $g_{n,d,i}$ for integers n, d, i such that $0 \leq i \leq d < n$. This function allows us to consider the $+$-component locally with new coordinates, where S_i has coordinate $(1, \ldots, 1)$, and to allude to the Ferrers diagrams. In the first step, we consider the following auxiliary function:

$$f_{n,d,i,j}(x) = \begin{cases} x - j + 1 & \text{if } 1 \leq j \leq i \\ (n-x) - (d-j) + 1 & \text{if } i < j \leq d. \end{cases}$$

Then we can now define $g_{n,d,i}$ that maps a d-subset to a point in the d-dimenional space.

▶ **Definition 11.** *Let n, d, i be integers such that $0 \leq i \leq d < n$. Then the function $g_{n,d,i}$ maps each d-subset $B = (b_1, \ldots, b_d)$ to the tuple*

$$\big(f_{n,d,i,1}(b_1), f_{n,d,i,2}(b_2) \ldots, f_{n,d,i,d}(b_d)\big).$$

We have the following observation on this function $g_{n,d,i}$.

▶ **Observation 12.** *The function $f_{n,d,i,j}$ is injective and has an inverse, and hence, the same holds for $g_{n,d,i}$. Further, if B is a $+$-subset of a co-signotope $\tau \in \bar{\mathcal{S}}_p(n,d)$, then by Lemma 9 and the fact that $b_j \in [j, n-(d-j)]$, each index of $g_{n,d,i}(B)$ is in $[p]$. In particular, $g_{n,d,i}(S_i) = (1, \ldots, 1)$.*

As mentioned before, $g_{n,d,i}$ allows us to make a connection with the Ferrers diagram of an integer partition [1]. An *integer partition* of a non-negative integer n is a way of writing n as a sum of positive integers, where the summands are sorted in the nonincreasing order. The *graphical representation* or *Ferrers diagram* of an integer partition of n is a set of points with positive integral indices in the plane such that (i,j) is in the set only if (i',j') is also in the set for all $i' \leq i$, $j' \leq i$. It is easy to see the one-to-one correspondence between an integer partition and its graphical representation. See Figure 2(left) for an illustration.

We define the following generalization of the Ferrers diagram.

▶ **Definition 13** ((d,i)-Ferrers diagram)**.** *Let $Z(d,i)$ be the set of all points $(a_1, \ldots, a_d)$ in $\mathbb{Z}^d_{>0}$ that satisfy $a_j \geq a_{j-1}$ for all $j \in [2,i]$, and $a_j \geq a_{j+1}$ for all $j \in [i+1, d-1]$. A* (d,i)-Ferrers diagram *with respect to p is a set $\mathcal{F}$ of p points in $Z(d,i)$ such that $(a_1, \ldots, a_d)$ is in $\mathcal{F}$ only if $(a'_1, \ldots, a'_d)$ is also in $\mathcal{F}$, for all $(a'_1, \ldots, a'_d) \in Z(d,i)$ that satisfies $a'_j \leq a_j$, $j \in [d]$.*

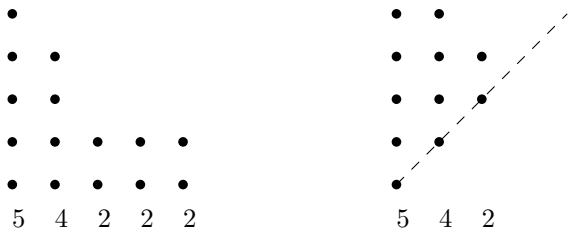


Figure 2 Left: A Ferrers diagram with the corresponding integer partition of 15, where the bottom-leftmost point corresponds to (1,1). Note that this is also a $(2,1)$-Ferrers diagram with respect to 15. **Right**: A $(2,2)$-Ferrers diagram with respect to 11. Since $a_2 \geq a_1$, there cannot be any point below the dashed line. The numbers are the count of points in each vertical line. Observe that these numbers must be strictly decreasing.

See Figure 2(right) for an illustration of a (d,i)-Ferrers diagram. Note that (2,1)-Ferrers diagrams are exactly the Ferrers diagrams.

We are now ready to characterize the co-signotopes in $\bar{\mathcal{S}}_{p,i}(n,d)$.

▶ **Lemma 14.** *Let n,d,p,i be integers such that $1 \leq p \leq n-d$ and $0 \leq i \leq d$. There is a one-to-one correspondence between a co-signotope τ in $\bar{\mathcal{S}}_{p,i}(n,d)$ and a (d,i)-Ferrers diagram $\mathcal{F}_\tau$ with respect to p. Further, this correspondence is described by $\mathcal{F}_\tau = \{g_{n,d,i}(B) \mid B \in \tau^{-1}(+)\}$.*

Proof. Suppose we have a co-signotope $\tau \in \bar{\mathcal{S}}_{p,i}(n,d)$, we show that for $\mathcal{F}_\tau := \{g_{n,d,i}(B) \mid B \in \tau^{-1}(+)\}$, the set $\mathcal{F}_\tau$ is a (d,i)-Ferrers diagram with respect to p. By Observation 12, the elements of $\mathcal{F}_\tau$ have positive integral indices, and $g_{n,d,i}$ is injective. Combined with the fact that τ has p $+$-subsets, this implies that $\mathcal{F}_\tau$ has p elements.

We start by showing that $\mathcal{F}_\tau \subset Z(d,i)$, for $Z(d,i)$ as defined in Definition 13. Consider a $+$-subset $B = (b_1, \ldots, b_d)$. For $j \in [2,i]$, we have $b_j > b_{j-1}$ by our convention of a subset. Hence, $f_{n,d,i,j}(b_j) + j - 1 > f_{n,d,i,j-1}(b_{j-1}) + (j-1) - 1$, which is equivalent to $f_{n,d,i,j}(b_j) \geq f_{n,d,i,j-1}(b_{j-1})$. Similarly, for $j \in [i+1, n-1]$, we obtain $f_{n,d,i,j}(b_j) \geq f_{n,d,i,j+1}(b_{j+1})$. It follows that $\mathcal{F}_\tau \subset Z(d,i)$. Then Lemma 9 implies that $\mathcal{F}_\tau$ satisfies the condition of a (d,i)-Ferrers diagram with respect to p in Definition 13.

For the other direction, suppose we have a (d,i)-Ferrers diagram $\mathcal{F}$ with respect to p. We show that the sign function τ such that $\tau^{-1}(+) = \{g^{-1}_{n,d,i}(X) \mid X \in \mathcal{F}\}$ is a co-signotope in $\bar{\mathcal{S}}_{p,i}(n,d)$. We first show that for every $X = (a_1, \ldots, a_d) \in \mathcal{F}$, $g^{-1}_{n,d,i}(X)$ is a d-subset. Indeed, since $\mathcal{F} \subset Z(d,i)$, it is easy to see that for $j \in [2,i] \cup [i+2,n]$, $f^{-1}_{n,d,i,j}(a_j) > f^{-1}_{n,d,i,j-1}(a_{j-1})$. For $j = i+1$, note that if the sum of the indices of X is s, then by Definition 13, it is easy to see that for $k \in [d,s]$, there exists an element Y in $\mathcal{F}$ whose sum of indices is exactly k. Combining this with the fact that there are p elements in $\mathcal{F}$, we obtain that $s \leq p+d-1$, and consequently, $a_i + a_{i+1} \leq p+1$. This implies

$$f^{-1}_{n,d,i,i+1}(a_{i+1}) - f^{-1}_{n,d,i,i}(a_i) = (n - a_{i+1} - d + i + 2) - (a_i + i - 1) = (n - d + 3) - (a_{i+1} + a_i) \geq 2.$$

This completes the proof that $g^{-1}_{n,d,i}(X)$ is a d-subset.

Next, we show that τ is a co-signotope, i.e., for every d-subset B and index j, the (τ, B, j)-series has at most one sign change. This is implied by the condition of a (d,i)-Ferrers diagram in Definition 13.

Lastly, observe that for any element $X = (a_1, \ldots, a_d)$ in $\mathcal{F}$, if it is not $(1, \ldots, 1)$, then we can find an index j such that the element X' obtained from X by replacing a_j with $a_j - 1$ is in $Z(d,i)$. Hence, X' is in $\mathcal{F}$. Repeating this process, we obtain a sequence of elements in $\mathcal{F}$ starting from X and ending at $(1, \ldots, 1)$. This corresponds to a walk in the graph $G_{n,d}$ from

a $+$-subset of τ to S_i. (Recall from Observation 12 that $g^{-1}_{n,d,i}((1,\dots,1)) = S_i$.) This implies that all $+$-subsets of τ are connected to S_i in $G_n[\tau]$. Hence, there is only one $+$-component, and this $+$-component contains S_i. Since $g^{-1}_{n,d,i}$ is injective, the $+$-component has p subsets. This completes the proof of the lemma. ◀

Observe that the set $\mathcal{F}_\tau$ in Lemma 14 depends only on d, i, and p, but not on n. Hence, this lemma straighforwardly implies a bijection between $\bar{\mathcal{S}}_{p,i}(n,d)$ and $\bar{\mathcal{S}}_{p,i}(\tilde{n},d)$ such that $\min\{n,\tilde{n}\} \geq p+d$, as follows.

▶ **Corollary 15.** *Let $n,\tilde{n},d,p,i$ be integers such that $1 \leq p \leq \min\{n,\tilde{n}\} - d$ and $0 \leq i \leq d$. Let $h_{n,\tilde{n},d,p,i}$ map a co-signotope τ in $\bar{\mathcal{S}}_{p,i}(n,d)$ to a sign function $\tilde{\tau} : \binom{[\tilde{n}]}{d} \to \{+,-\}$ such that*

$$\tilde{\tau}^{-1}(+) = \left\{ g^{-1}_{\tilde{n},d,i}\big(g_{n,d,i}(B)\big) \mid B \in \tau^{-1}(+) \right\}.$$

Then $h_{n,\tilde{n},d,p,i}$ is a bijection between $\bar{\mathcal{S}}_{p,i}(n,d)$ and $\bar{\mathcal{S}}_{p,i}(\tilde{n},d)$ that preserves the single step relation.

5 The $+$-component decomposition of a co-signotope

In the previous section, we studied co-signotopes with only one $+$-component. In this section, we go a step further and investigate how the $+$-subsets of a general co-signotope in $\bar{\mathcal{S}}_p(n,d)$ can be decomposed into several $+$-components. The *p-sequence* of a co-signotope τ in $\bar{\mathcal{S}}_p(n,d)$ is the $(d+1)$-tuple $(p_0,\dots,p_d)$, where p_i is the size of the $+$-component that contains S_i for $i \in [0,d]$ (i.e., $p_i = |\mathcal{C}^\tau_i|$). If we restrict the co-signotope τ such that we keep only one of its $+$-components as a $+$-component and set all remaining $+$-components to $-$, we still get a co-signotope. More precisely, we define the *$+$-component decomposition* of a co-signotope $\tau \in \bar{\mathcal{S}}_p(n,d)$, denoted by $\Delta(\tau)$, as the $(d+1)$-tuple $(\tau_0,\tau_1,\dots,\tau_d)$ such that for $i \in [0,d]$, each τ_i is a sign function with domain $\binom{[n]}{d}$, where $\tau_i^{-1}(+)$ is exactly $\mathcal{C}^\tau_i$. The following lemma shows that each τ_i is still a co-signotope.

▶ **Lemma 16.** *Let n,d,p be integers such that $0 \leq p \leq n-d$. For every co-signotope τ in $\bar{\mathcal{S}}_p(n,d)$ with $\Delta(\tau) = (\tau_0,\tau_1,\dots,\tau_d)$, τ_i is a co-signotope in $\bar{\mathcal{S}}_{|\mathcal{C}^\tau_i|,i}(n,d)$ for $i \in [0,d]$.*

Proof. For every d-subset B of $[n]$ and every index j, the $+$-subsets in the (τ,B,j)-series form a path in $G_{n,d}$. Hence τ_i is a co-signotope in $\bar{\mathcal{S}}_{|\mathcal{C}^\tau_i|,i}(n,d)$ for $i \in [0,d]$. ◀

Note that not every tuple $(p_0,\dots,p_d)$ such that $\sum_{i=0}^d p_i = p$ is the p-sequence of a co-signotope. There are some necessary conditions which turn out to be sufficient as well. For this we define a *sparse composition* of an integer p as a $(d+1)$-tuple $(c_0,\dots,c_d)$ of non-negative integers such that $\sum_{i=0}^n c_i = p$ and either $c_i = 0$ or $c_{i-1} = 0$ for all $i \in [d]$.

▶ **Lemma 17.** *Let n,d,p be non-negative integers such that $p \leq n-d$. For every $\tau \in \bar{\mathcal{S}}_p(n,d)$ its p-sequence $(p_0,\dots,p_d)$ is a sparse composition of p. Moreover, for every sparse composition $(c_0,\dots,c_d)$ of p and every co-signotopes $\tau_0,\dots,\tau_d$ such that $\tau_i \in \bar{\mathcal{S}}_{c_i,i}(n,d)$, there exists a unique co-signotope τ in $\bar{\mathcal{S}}_p(n,d)$ such that its p-sequence is $(c_0,\dots,c_d)$ and $\Delta(\tau) = (\tau_0,\tau_1,\dots,\tau_d)$.*

Proof. To show the first part, note that by Lemma 7, we obtain $\sum_{i=0}^d p_i = p$. We argue that for all $i \in [p]$, either $p_i = 0$ or $p_{i-1} = 0$. Indeed, if this is not the case, S_i and S_{i-1} are both $+$-subsets by definition. Then the (τ,S_i,i)-series starts with a $+$-subset S_i and

ends with a +-subset S_{i-1}. This implies that all subsets in (τ, B, i)-series are +-subsets, and hence this series is flat, a contradiction to Observation 8. Therefore, $(p_0, \dots, p_d)$ is a sparse composition of p.

Now let τ be the sign function over $\binom{[n]}{d}$ such that $\tau^{-1}(+) = \bigcup_{i=0}^{d} \tau_i^{-1}(+)$. By construction, τ has at most p +-subsets.

We start by arguing that τ is a co-signotope. Suppose that this is not the case. Then for some d-subset $B = (b_1, \dots, b_d)$ and an index j, the (τ, B, j)-series has more than one sign change. By definition of τ and by the fact that τ_i is a co-signotope in $\bar{\mathcal{S}}_{c_i,i}(n, d)$ for $i \in [0, d]$, we must have that for some $i, i' \in [0, d]$, the (τ_i, B, j)-series is left-aligned and $(\tau_{i'}, B, j)$-series is right-aligned. Because $(c_0, \dots, c_d)$ is a sparse composition, we have $|i - i'| \geq 2$. Let $B^\alpha = (b_1^\alpha, \dots, b_d^\alpha)$ and $B^\omega = (b_1^\omega, \dots, b_d^\omega)$ be the first and the last subsets of the (B, j)-series, respectively. We use the convention that $b_k = 0$ for $k < 1$ and $b_k = d + 1$ for $k > d$. It is easy to see the following:

(a) If $j \geq i$, then $[b_i^\alpha, b_{i+1}^\alpha] \subseteq [b_{i-1}, b_{i+1}]$;
(b) If $j < i$, then $[b_i^\alpha, b_{i+1}^\alpha] \subseteq [b_i, b_{i+1}]$;
(c) If $j \leq i' + 1$, then $[b_{i'}^\omega, b_{i'+1}^\omega] \subseteq [b_{i'}, b_{i'+2}]$;
(d) If $j > i' + 1$, then $[b_{i'}^\omega, b_{i'+1}^\omega] \subseteq [b_{i'}, b_{i'+1}]$.

From the above case distinction, we observe that when $j > i' + 1$, the interiors of $[b_i^\alpha, b_{i+1}^\alpha]$ and $[b_{i'}^\omega, b_{i'+1}^\omega]$ are disjoint, since $|i' - i| \geq 2$. When $j \leq i' + 1$ and $i' \geq i + 2$, these interiors are also disjoint. When $j \leq i' + 1$ and $i' \leq i - 2$, then (b) and (c) are applicable, and the two interiors are also disjoint.

Hence, we always have that the two interiors above are disjoint. Further, from (a)–(d) above, these interiors contain only integers from $[n] \setminus B \cup \{b_i, b_{i'+1}\}$. These two statements imply that $b_{i+1}^\alpha - b_i^\alpha + b_{i'+1}^\omega - b_{i'}^\omega \leq n - d + 4$.

By Lemma 6, the shortest path between S_i and B^α is $b_i^\alpha - b_{i+1}^\alpha + n - d + 1$, whereas the shortest path between $S_{i'}$ and B^ω is $b_{i'}^\omega - b_{i'+1}^\omega + n - d + 1$. Then the sum of the distances is

$$b_i^\alpha - b_{i+1}^\alpha + n - d + 1 + b_{i'}^\omega - b_{i'+1}^\omega + n - d + 1 \geq 2(n - d + 1) - (n - d + 4) \geq n - d - 2 \geq p - 2.$$

This implies that $c_i + c_{i'} \geq p + 1$, a contradiction to the fact that $(c_0, \dots, c_d)$ is a sparse composition of p.

Next, by Lemma 7, each +-component of τ contains exactly one source subset. Hence, no subset is a +-subset of more than one co-signotope among $\tau_0, \dots, \tau_d$. Therefore, τ has exactly p +-subsets, and each +-component of τ that contains S_i corresponds exactly to the +-component of τ_i. This also implies that τ is unique and $\Delta(\tau) = (\tau_0, \tau_1, \dots, \tau_d)$. It follows that $(c_0, \dots, c_d)$ is the p-sequence of τ, completing the proof of the lemma. ◀

6 Proof of Theorem 1

Equipped with Corollary 15, Lemma 16, and Lemma 17, we are now ready to prove Theorem 1.

▶ **Theorem 1** (Restated from Section 2.1)**.** *Let $n, \tilde{n}, d, p$ be integers such that $0 \leq p \leq \min\{n, \tilde{n}\} - d$. Then there exists a bijection between $\bar{\mathcal{S}}_{\leq p}(n, d)$ and $\bar{\mathcal{S}}_{\leq p}(\tilde{n}, d)$ that preserves the relation $\preceq_{\bar{S}}$.*

Proof of Theorem 1. Consider the following mapping $\zeta_{n,\tilde{n},d,p}$ from a co-signotope $\tau \in \bar{\mathcal{S}}_{\leq p}(n, d)$ to a co-signotope $\tilde{\tau} \in \bar{\mathcal{S}}_{\leq p}(\tilde{n}, d)$. Let $(p_0, \dots, p_d)$ be the p-sequence of τ. We first obtain the +-component decomposition $(\tau_0, \tau_1, \dots, \tau_d)$ of τ. Next, we use the bijection

$h_{n,\tilde{n},d,p_i,i}$ in Corollary 15 to map each of $\tau_i \in \bar{\mathcal{S}}_{p_i,i}(n,d)$ to $\tilde{\tau}_i \in \bar{\mathcal{S}}_{p_i,i}(\tilde{n},d)$. Then from the tuple $(\tilde{\tau}_0, \tilde{\tau}_1, \ldots, \tilde{\tau}_d)$, we can combine their $+$-subsets to construct $\tilde{\tau}$. By Lemma 17, $\tilde{\tau} \in \bar{\mathcal{S}}_p(\tilde{n},d)$.

By Corollary 15, Lemma 16, and Lemma 17, the mapping $\zeta_{n,\tilde{n},d,p}$ is injective. Also by these corollary and lemmas, the mapping $\zeta_{\tilde{n},n,d,p}$ maps $\tilde{\tau}$ back to τ and generally this mapping maps a signotope in $\bar{S}_p(\tilde{n},d)$ to $\bar{S}_p(n,d)$. This implies that $\zeta_{n,\tilde{n},d,p}$ is a bijection between $\bar{S}_p(n,d)$ and $\bar{S}_p(\tilde{n},d)$.

It remains to prove that $\zeta_{n,\tilde{n},d,p}$ preserves the relation $\preceq_{\bar{S}}$. For this, it is sufficient to prove that $\zeta_{n,\tilde{n},d,p}$ preserves the single step relation. Let τ and ρ be two d-co-signotopes in $\bar{\mathcal{S}}_{\leq p}(n,d)$ such that $\tau^{-1}(+) \subset \rho^{-1}(+)$ and $|\rho^{-1}(+)| = |\tau^{-1}(+)| + 1 = k+1$ for some k. Let $(\tau_0, \ldots, \tau_d) = \Delta(\tau)$ and $(\rho_0, \ldots, \rho_d) = \Delta(\rho)$. This implies that $\Delta(\tau)$ and $\Delta(\rho)$ agree in all coordinates, except for some i-th coordinate, where τ_i and ρ_i differ by a single step. Let $\tilde{\tau}$ and $\tilde{\rho}$ be the images of τ and ρ, respectively, in the bijection above. By Corollary 15, it follows that $\Delta(\tilde{\tau})$ and $\Delta(\tilde{\rho})$ also agree in all coordinates, except for some i-th coordinate, where the two corresponding co-signotopes differ by a single step. This in turns implies that $\tilde{\tau}$ and $\tilde{\rho}$ differ by a single step. This completes the proof of the theorem. ◀

A simple bijection

Although the mapping $\zeta_{n,\tilde{n},d,p}$ in the proof above satisfies the conditions of Theorem 1, its description is rather complicated. We now provide a simpler description of the bijection. In other words, we will prove that the mapping $\zeta_{n,\tilde{n},d,p}$ is identical to the mapping $\phi_{n,\tilde{n},d,p}$ as described below.

For a co-signotope $\tau \in \bar{\mathcal{S}}_p(n,d)$, a d-subset $B = (b_1, \ldots, b_d)$, we define $\gamma_{\tau,\tilde{n}}(B)$ as the tuple obtained from B by replacing b_j by $b_j + \tilde{n} - n$ if the (τ, B, j)-series is right-aligned.

▶ **Definition 18.** *Let $n, \tilde{n}, d, p$ be positive integers. Let $\phi_{n,\tilde{n},d,p}$ map a co-signotope $\tau \in \bar{\mathcal{S}}_{\leq p}(n,d)$ to a sign function $\tilde{\tau} : \binom{\tilde{n}}{d} \to \{+,-\}$ such that*

$$\tilde{\tau}^{-1}(+) = \Big\{\gamma_{\tau,\tilde{n}}(B) \mid B \in \tau^{-1}(+)\Big\}.$$

▶ **Lemma 19.** *$\phi_{n,\tilde{n},d,p}$ is a bijection as guaranteed by Theorem 1.*

Proof. Consider the mapping $\zeta_{n,\tilde{n},d,p}$ as described in the proof of Theorem 1. We prove that it is identical to $\phi_{n,\tilde{n},d,p}$. Note that the mapping $\zeta_{n,\tilde{n},d,p}$ essentially maps each $+$-subset of τ to a $+$-subset of $\tilde{\tau}$. In the step of computing the $+$-component decomposition of τ and the step of combining $(\tilde{\tau}_0, \tilde{\tau}_1, \ldots, \tilde{\tau}_d)$ into $\tilde{\tau}$, the $+$-subsets do not change, or rather, each $+$-subset is mapped to itself. In the middle step when we apply $h_{n,\tilde{n},d,s_i,i}$, the mapping of the $+$-subsets is done through $g^{-1}_{\tilde{n},d,i} \circ g_{n,d,i}$, by the definition of $h_{n,\tilde{n},d,i}$ in Corollary 15. By Definition 11, we can describe $g^{-1}_{\tilde{n},d,i} \circ g_{n,d,i}$ as follows: For a d-subset $B = (b_1, \ldots, b_d)$ in the same $+$-component of τ as S_i, $g^{-1}_{\tilde{n},d,i}(g_{n,d,i}(B))$ is obtained by replacing b_j with b_j for $j \in [i]$ and with $b_j + \tilde{n} - n$ for $j \in [i+1, d]$. Combining this description with Lemma 9, we obtain that $g^{-1}_{\tilde{n},d,i}(g_{n,d,i}(B)) = \gamma_{\tau,\tilde{n}}(B)$. It follows that $\zeta_{n,\tilde{n},d,p} \equiv \phi_{n,\tilde{n},d,p}$, as claimed. ◀

7 Plus count

Theorem 1 implies that given d and p, $\bar{S}_p(n,d)$ is a fixed number for all $n \geq p+d$. This motivates the following definition.

▶ **Definition 20.** *For two integers $d \geq 1$ and $p \geq 0$, the* plus count *with respect to d and p, denoted by $P_{d,p}$, is the size of $\bar{S}_p(n,d)$ for any $n \geq p+d$.*

Table 1 The values of $P_{d,p}$ for small d (vertical) and p (horizontal).

d / p	0	1	2	3	4	5	6	7	8	9	10
1	1	2	2	2	2	2	2	2	2	2	2
2	1	3	5	9	14	21	33	47	68	96	135
3	1	4	9	20	41	78	146	264	465	804	1368
4	1	5	14	36	86	192	413	857	1732	3422	6633
5	1	6	20	58	155	386	920	2110	4691	10176	21604
6	1	7	27	87	255	693	1790	4438	10636	24799	56485
7	1	8	35	124	394	1154	3192	8444	21534	53292	128571

From computer experiment, we obtain the values of $P_{d,p}$ for small d and p, as presented in Table 1.

We can provide a characterization of the plus counts based on the correspondences in Lemmas 14, 16, and 17. Let $F_{d,i}(p)$ be the number of (d,i)-Ferrers diagrams with respect to p, and let $\mathcal{D}_{d+1,p}$ be the set of all sparse compositions of p. Then the lemmas above give the following formula

$$P_{d,p} = \sum_{(c_0,\ldots,c_d)\in\mathcal{D}_{d+1,p}} \prod_{i=0}^{d} F_{d,i}(c_i). \tag{1}$$

For the remainder of this section, we discuss a few values of d and p. Due to space constraints, all the proofs in this section are presented in the full version [3]. We start with a few simple observations.

▶ **Lemma 21.** *It holds:*
(a) *For all $d \geq 1$: $P_{d,0} = 1$.*
(b) *For all $p > 0$: $P_{1,p} = 2$/*
(c) *For all $d \geq 1$: $P_{d,1} = d + 1$.*

Next, to prove the formulae for higher values of p, we show a few observations on $F_{d,i}$.

▶ **Lemma 22.** *For non-negative integers d, i, p with $d \geq i$, it holds:*
(a) $F_{d,i}(p) = F_{d,d-i}(p)$.
(b) $F_{d,i}(p) = \sum_{c=1}^{p} F_{i,0}(c)F_{d-i,0}(p+1-c)$.
(c) *For $d = 1$, $F_{d,i}(p) = 1$.*
(d) *For $d > 1$, $F_{d,i}(0) = 1$, $F_{d,i}(1) = 1$, $F_{d,0}(2) = 1$, and $F_{d,0}(3) = 2$.*
(e) *For $0 < i < d$, $F_{d,i}(2) = 2$.*
(f) $F_{d,1}(3) = F_{d,d-1}(3) = 4$. *For $1 < i < d-1$, $F_{d,i}(3) = 5$.*

Equipped with the lemma above, we can show the following.

▶ **Lemma 23.** *For $d \geq 1$, $P_{d,2} = \frac{1}{2}d^2 + \frac{3}{2}d$ and $P_{d,3} = \frac{1}{6}d^3 + d^2 + \frac{17}{6}d - 2$.*

Lastly, recall that an integer partition of n is the number of ways of writing n as the (unordered) sum of positive integers. Let $\pi(n)$ be the number of integer partitions of n. Further, let $\delta(n)$ be the number of ways of writing n as the (unordered) sum of distinct positive integers.

▶ **Lemma 24.** $P_{2,p} = \pi(p) + \sum_{i=0}^{p} \delta(i) \cdot \delta(p-i)$.

Note that $\pi(n)$ and $\delta(n)$ are counted by the OEIS sequences A000041 and A000009 [14], respectively. Further, the sum $\sum_{i=0}^{p} \delta(i) \cdot \delta(p-i)$ is counted by the OEIS sequence A022567. Hence, $P_{2,p}$ is counted by the sum of the two sequences A000041 and A022567.

8 Concluding remarks

Firstly, we note that Theorem 1 is tight in the sense that we cannot relax the condition $p \leq \min\{n, \tilde{n}\} - d$. As an example, consider the case when $d = n - 1 = \tilde{n} - 2$ and $p = 2$. In other words, we consider the possibility of a bijection between $\bar{S}_{\leq 2}(n, n-1)$ and $\bar{S}_{\leq 2}(n+1, n-1)$ or equivalently a bijection between $S_{\leq 2}(n, 1)$ and $S_{\leq 2}(n+1, 2)$. As shown in [3], we have

$$|S_{\leq 2}(n,1)| = 1 + n + \binom{n}{2}, \qquad |S_{\leq 2}(n+1,2)| = 1 + n + \binom{n}{2} + n - 1.$$

Hence, there cannot be any bijection between $S_{\leq 2}(n, 1)$ and $S_{\leq 2}(n+1, 2)$.

An intuitive reason is that when $p > n - d$, Observation 8 no longer holds. That is, there can be a flat series with only $+$-subsets. For example, when $p = n - d + 1$, there exists a co-signotope $\tau \in \bar{S}_p(n, d)$ such that its $+$-subsets form exactly the whole (τ, S_i, i)-series for some $i \in [0, d]$. In other words, its only $+$-component contains both S_i and S_{i+1}. However, for $\tilde{n} > n$ (i.e., $p \leq \tilde{n} - d$), Observation 8 holds again, and it is unclear if τ should be mapped to a co-signotope $\tilde{\tau} \in \bar{S}_p(\tilde{n}, d)$ with a nonempty $C_i^{\tilde{\tau}}$ or a nonempty $C_{i+1}^{\tilde{\tau}}$.

Secondly, from Figure 1, it seems that the bijection of Theorem 1 could be proved by the following simple geometric argument: Suppose $n < \tilde{n}$; we remove $\tilde{n} - n$ hyperplanes that are completely in the negative side of the extending hyperplane of the one-element extension of $\mathbf{X}_c^{\tilde{n},d}$. However, this argument does not work in a high dimension. For example, the positive side may consist of these two points: the intersection of the hyperplanes $H_1, \ldots, H_d$ and that of $H_{\tilde{n}-d+1}, \ldots, H_{\tilde{n}}$. For d and $\tilde{n}$ such that $d > \tilde{n} - d + 1$, there are then no hyperplanes completely on the negative side.

Lastly, it would be interesting to obtain understanding on more values of the plus counts, for example, their exact counts, generating function, or more relations. As a starter, we conjecture that $P_{d,3} = P_{d-1,3} + P_{d-1,2} + P_{d-1,1} + 3$.

References

1 George E. Andrews and Kimmo Eriksson. *Integer partitions*. Cambridge University Press, Cambridge, 2004. doi:10.1017/CBO9781139167239.

2 Martin Balko. Ramsey numbers and monotone colorings. *Journal of Combinatorial Theory, Series A*, 163:34–58, 2019. doi:10.1016/j.jcta.2018.11.013.

3 Helena Bergold, Lukas Egeling, and Hung P. Hoang. Signotopes with few plus signs, 2025. arXiv:2411.19208.

4 Helena Bergold, Stefan Felsner, and Manfred Scheucher. An Extension Theorem for Signotopes. In *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SoCG.2023.17.

5 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2 edition, 1999. doi:10.1017/CBO9780511586507.

6 Fernando Cortés Kühnast, Justin Dallant, Stefan Felsner, and Manfred Scheucher. An improved lower bound on the number of pseudoline arrangements. In *40th International Symposium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece*, volume 293 of *LIPIcs*, pages 43:1–43:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.43.

7 Stefan Felsner and Jacob E. Goodman. Pseudoline arrangements. In Toth, O'Rourke, and Goodman, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, third edition, 2017. doi:10.1201/9781315119601.

8 Stefan Felsner and Helmut Weil. A theorem on higher Bruhat orders. *Discrete & Computational Geometry*, 23(1):121–127, 2000. doi:10.1007/PL00009485.

9 Stefan Felsner and Helmut Weil. Sweeps, arrangements and signotopes. *Discrete Applied Mathematics*, 109(1):67–94, 2001. doi:10.1016/S0166-218X(00)00232-8.

10 Mikhail M. Kapranov and Vladimir A. Voevodsky. Combinatorial-geometric aspects of polycategory theory: Pasting schemes and higher Bruhat orders (list of results). *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 32:11–27, 1991. URL: http://www.numdam.org/item/?id=CTGDC_1991__32_1_11_0.

11 Donald E. Knuth. *Axioms and Hulls*, volume 606 of *LNCS*. Springer, 1992. doi:10.1007/3-540-55611-7_1.

12 F. Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*, 78:256–267, 1926.

13 Yu. I. Manin and V. V. Schechtman. Arrangements of hyperplanes, higher braid groups and higher Bruhat orders. *Advanced Studies in Pure Mathematics*, 17:289–308, 1989. Algebraic Number Theory – in honor of K. Iwasawa. doi:10.2969/aspm/01710289.

14 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. Published electronically at http://oeis.org.

15 Günter M. Ziegler. Higher Bruhat orders and cyclic hyperplane arrangements. *Topology*, 32(2):259–279, 1993. doi:10.1016/0040-9383(93)90019-R.

Sparse Bounded Hop-Spanners for Geometric Intersection Graphs

Sujoy Bhore
Department of Computer Science and Engineering, Indian Institute of Technology Bombay, India

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Zhengcheng Huang
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Shakhar Smorodinsky
Department of Computer Science, Ben-Gurion University of the Negev, Be'er Sheva, Israel

Csaba D. Tóth
Department of Mathematics, California State University Northridge, Los Angeles, CA, USA
Department of Computer Science, Tufts University, Medford, MA, USA

Abstract

We present new results on 2- and 3-hop spanners for geometric intersection graphs. These include improved upper and lower bounds for 2- and 3-hop spanners for many geometric intersection graphs in $\mathbb{R}^d$. For example, we show that the intersection graph of n balls in $\mathbb{R}^d$ admits a 2-hop spanner of size $O^*\left(n^{\frac{3}{2}-\frac{1}{2(2\lfloor d/2\rfloor+1)}}\right)$ and the intersection graph of n fat axis-parallel boxes in $\mathbb{R}^d$ admits a 2-hop spanner of size $O(n\log^{d+1} n)$.

Furthermore, we show that the intersection graph of general semi-algebraic objects in $\mathbb{R}^d$ admits a 3-hop spanner of size $O^*\left(n^{\frac{3}{2}-\frac{1}{2(2D-1)}}\right)$, where D is a parameter associated with the description complexity of the objects. For such families (or more specifically, for tetrahedra in $\mathbb{R}^3$), we provide a lower bound of $\Omega(n^{\frac{4}{3}})$. For 3-hop and axis-parallel boxes in $\mathbb{R}^d$, we provide the upper bound $O(n\log^{d-1} n)$ and lower bound $\Omega\left(n(\frac{\log n}{\log\log n})^{d-2}\right)$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric Spanners, Geometric Intersection Graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.17

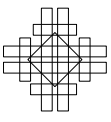
Related Version *Full Version*: https://doi.org/10.48550/arXiv.2504.05861 [9]

Funding *Timothy M. Chan*: Work supported in part by NSF Grant CCF-2224271.
Shakhar Smorodinsky: Partially supported by Grant 1065/20 from the Israel Science Foundation, by the United States – Israel Binational Science Foundation (NSF-BSF grant no. 2022792) and by ERC grant no. 882971 "GeoScape" and by the Erdős Center.
Csaba D. Tóth: Research was partially supported by the NSF award DMS-2154347.

1 Introduction

A *spanner* of a graph G is a subgraph G' that approximately preserves the distances between pairs of vertices in G. More formally, let $G=(V,E)$ be a simple graph. A spanning subgraph $G'=(V,E')$ is called a *t-spanner* for G if for every two vertices $x,y\in V$ $d_{G'}(x,y)\leq t\cdot d_G(x,y)$ where $d_G(x,y)$ denotes the shortest path distance between x and y in G. Here, $t\geq 1$ is a constant called the *stretch factor* (or *stretch* for short). Spanners are an important concept in graph theory and computer science, particularly in the context of geometric intersection graphs, where vertices represent geometric objects and edges represent intersections between

© Sujoy Bhore, Timothy M. Chan, Zhengcheng Huang, Shakhar Smorodinsky, and Csaba D. Tóth;
licensed under Creative Commons License CC-BY 4.0



these objects. The ability to construct spanners for geometric intersection graphs with fewer edges is crucial because it enables efficient algorithms for problems such as shortest path computation, network design, and routing while maintaining proximity to the original geometric distances. By using spanners, one can often reduce the complexity of geometric problems without sacrificing the quality of the solution (see, e.g., [31, 35, 38, 39, 40]).

The following theorem is a cornerstone in the theory of spanners for general graphs:

▶ **Theorem 1.1** ([3]). *Let $G = (V, E)$ be a simple undirected graph on n vertices. Let $t \geq 1$ be a fixed integer parameter. Then G contains a $(2t-1)$-spanner G' with $O(n^{1+\frac{1}{t}})$ edges.*

Note that the above theorem is non-trivial only for $t \geq 2$ or stretch factor at least 3. Indeed, one cannot hope to obtain a 2-spanner with $o(n^2)$ edges in general as is exemplified by the complete bi-partite graph $K_{\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil}$. The general bound $O(n^{1+\frac{1}{t}})$ is conjectured to be optimal due to the *Erdős Girth Conjecture* [25], and the spanner G' of this size can be computed by a greedy algorithm. It would be interesting to ask what families of graphs admit 2-spanners with $o(n^2)$ edges or 3-spanners with $o(n^{3/2})$ edges.

The **intersection graph** of a set system $(X, \mathcal{F})$ is a graph $G = (\mathcal{F}, E)$, where the vertices correspond to the sets in $\mathcal{F}$, and there is an edge between sets $A, B \in \mathcal{F}$ if and only if $A \cap B \neq \emptyset$. Every edge has unit weight. For intersection graphs, t-spanners are also called *t-hop spanners* to emphasize the use of hop distance. In this paper, our goal is to obtain hop spanners for geometric intersection graphs that are smaller than guaranteed by Theorem 1.1.

Previous work on spanners in intersection graphs. Initial work on spanners for geometric intersection graphs was motivated by wireless communication, and was limited to unit disk graphs in the plane. In this setting, spanners for weighted graphs were also considered, where the weight of an edge is the distance between the disk centers [28, 29, 30]. Recent work focused on hop spanners for unit disk graphs [10, 11, 22], and intersection graphs of other geometric objects (including string graphs) [16, 20]. Most previous bounds are constrained to objects in the plane. Chan and Huang [16] use shallow cuttings to show that any family of n objects, with union complexity bounded by $\mathcal{U}(n)$, admits a 2-hop spanner of size $O(\mathcal{U}(n) \log n)$. This yields an $O(n \log n)$ bound for disks or pseudodisks in the plane [33]; $n \log n \cdot 2^{O(\log^* n)}$ for fat objects of constant algebraic description complexity [4], and in particular $O(n \log n \log^* n)$ for fat triangles in the plane [4]. Lower bounds can be derived from constructions for geometric intersection graphs with girth $2t+1$, as a t-spanner contains the entire graph [7, 21, 41]. In particular, Davies [21] constructed families of axis-aligned boxes in $\mathbb{R}^3$ whose intersection graph has unbounded girth and $\omega(n)$ edges. The boxes in Davies' construction project to squares in the xy-plane, so they can be lifted to hypercubes in $\mathbb{R}^4$ with the same intersection pattern. See Table 1 for a list of known results.

1.1 New results

In this paper, we focus on 2- and 3-spanners. Recall that in general there is no hope to obtain a 2-spanner of sub-quadratic size (in n) nor a 3-spanner of $o(n^{3/2})$ size for general graphs with n vertices. No sub-quadratic bounds were previously known for 2-spanners for intersection graphs in dimension 3 or more even for unit balls. And no sub-$n^{3/2}$ bounds were previously known for 3-spanners for intersection graphs of non-fat objects in dimension above 2, even for simplices. We give the first 2-spanner constructions of sub-quadratic size (in fact, $O(n^{3/2})$ or better) for many types of fat objects, including balls, and we give the first 3-spanner constructions of sub-$n^{3/2}$ size for many types of non-fat objects, including

Table 1 Summary of previously known results. The trivial lower bound in each case is $\Omega(n)$. The upper bound in $(*)$ generalizes to $O(\mathcal{U}(n)\log n)$ for objects with union complexity $\mathcal{U}(.)$; and the lower bound holds for homothets of any convex body in the plane. The upper bound in $(**)$ holds for fat objects of constant algebraic description complexity. The function $\alpha_k(.)$ denotes the k-th function in the inverse Ackermann hierarchy: $\alpha_0(n) = n/2$, $\alpha_1(n) = \log n$, $\alpha_2(n) = \log^* n$ (the iterated logarithm), $\alpha_3(n) = \log^{**} n$ (the iterated iterated logarithm), etc.

Hops	Objects	Dim.	Upper Bounds	Lower Bounds	Ref.
2	unit disks	2	$O(n)$		[20]
	translates of a convex body	2	$O(n)$		[20]
	axis-aligned fat rectangles	2	$O(n \log n)$	$\Omega(n\frac{\log n}{\log\log n})$	[20]
	disks of arbitrary radii $(*)$	2	$O(n \log n)$	$\Omega(n\frac{\log n}{\log\log n})$	[16, 20]
	general fat objects $(**)$	2	$n \log n \cdot 2^{O(\log^* n)}$	$\Omega(n\frac{\log n}{\log\log n})$	[16, 20]
3	general fat objects	$d \geq 2$	$O(n \log n)$		[16]
	axis-aligned rectangles	2	$O(n \log n)$		[16]
	strings	2	$O(n \log^3 n)$		[16]
$O_k(1)$	strings	2	$O(n\alpha_k(n))$		[16]
$O(1)$	axis-aligned boxes	$d \geq 3$		$\omega(n)$	[21]
$O(1)$	axis-aligned hypercubes	$d \geq 4$	$O(n\alpha_k(n))$	$\omega(n)$	[16, 21]
$O_{d,k}(1)$	general fat objects	d	$O(n\alpha_k(n))$		[16]

Table 2 Summary of new results. Objects for $(*)$ are assumed to be semi-algebraic for some parameter D associated with their description complexity. The upper bounds for $(**)$ hold more generally for fat polyhedra of constant complexity, while the lower bounds for $(**)$ hold more specifically for unit regular tetrahedra or unit non-axis-aligned cubes in $\mathbb{R}^3$. The lower bounds for $(\dagger)$ hold more specifically for unit balls. The lower bound for $(\ddagger)$ holds specifically for unit axis-aligned hypercubes.

Hops	Objects	Dimension	Upper Bounds	Lower Bounds
2	general fat objects	$d \geq 3$	$O(n^{3/2})$	$\Omega(n^{4/3})$
	general fat objects $(*)$	$d \geq 3$	$O^*(n^{\frac{3}{2}-\frac{1}{2(2D-1)}})$	$\Omega(n^{4/3})$
	fat simplices $(**)$	3	$O^*(n^{10/7})$	$\Omega(n^{4/3})$
		$d \geq 4$	$O(n^{\frac{3}{2}-\frac{1}{O(d^2)}})$	$\Omega(n^{4/3})$
	balls $(\dagger)$	3	$O^*(n^{4/3})$	
		4	$O^*(n^{7/5})$	
		5	$O^*(n^{7/5})$	$\Omega(n^{4/3})$
		$d \geq 6$	$O^*(n^{\frac{3}{2}-\frac{1}{2(2\lfloor d/2\rfloor+1)}})$	$\Omega(n^{4/3})$
	fat axis-aligned boxes $(\ddagger)$	$d \geq 3$	$O(n \log^{d+1} n)$	$\Omega(n(\frac{\log n}{\log\log n})^{\lfloor d/2\rfloor-1})$
3	general objects $(*)$	$d \geq 3$	$O^*(n^{\frac{3}{2}-\frac{1}{2(2D-1)}})$	$\Omega(n^{4/3})$
	simplices	3	$O^*(n^{10/7})$	$\Omega(n^{4/3})$
		$d \geq 4$	$O(n^{\frac{3}{2}-\frac{1}{O(d^2)}})$	$\Omega(n^{4/3})$
	axis-aligned boxes	$d \geq 3$	$O(n \log^{d-1} n)$	$\Omega(n(\frac{\log n}{\log\log n})^{d-2})$

simplices. Furthermore, we complement our upper bounds with a number of lower bounds ruling out near-linear-size spanners for many types of objects. See Table 2 for a detailed list of all results.[1] Our results answer open questions by Chan and Huang [16].[2]

One of our main results deals with 2- and 3-spanners for semi-algebraic graphs. The notion of semi-algebraic graphs was the focus of several recent papers. Most notably, Fox et al. [27] provided improved bounds on Zarankiewicz's problem for the class of semi-algebraic graphs. Informally, a semi-algebraic graph $G = (V, E)$ is a bipartite graph where the vertices can be represented as points in $\mathbb{R}^d$ for some fixed small constant dimension d and for every pair of vertices u, v we have that $\{u, v\} \in E$ if some boolean combination of a constant number of inequalities involving $2d$-variate polynomials of constant degree is satisfied, where the variables in the polynomials are the coordinates of the two points representing u and v (see below for a formal definition). Almost all intersection graphs in the literature are, in fact, semi-algebraic. One of our main results is a bound of the form $O(n^{\frac{3}{2}-\delta_d})$ for 3-hop spanner for arbitrary intersection graphs of semi-algebraic sets in $\mathbb{R}^d$ and for 2-hop for semi-algebraic sets that are also "fat" where δ_d is a constant that depends on the description complexity of the objects. In this regard, it is important to note that without the fatness assumption, no sub-quadratic 2-hop spanner can be obtained as the graph $K_{n,n}$, which has n^2 edges, can be realized as the intersection of n horizontal and n vertical narrow rectangles, and any 2-hop spanner must contain all n^2 edges. Also, a common property for semi-algebraic intersection graphs is that they all have bounded VC-dimension. However, bounded VC-dimension by itself does not guarantee $o(n^{\frac{3}{2}})$ edges as demonstrated later in Theorem 4.6. Thus, the extra geometric properties that these graphs possess are crucial in order to obtain such bounds. Perhaps not surprisingly, one of our main tools for tackling this problem is an application of the divide-and-conquer, specifically the **polynomial partitioning** technique. This technique was introduced by Guth and Katz in their groundbreaking work on the *Erdős distinct distances* problem [32]. This method involves partitioning the space into cells defined by low-degree polynomials. A key result states that for a set of n points in $\mathbb{R}^d$, and for any $D > 0$, there exists a d-variate polynomial f of degree D such that the zero set $Z(f)$ of f partitions $\mathbb{R}^d$ into $O(D^d)$ cells (i.e., connected components of $\mathbb{R}^d \setminus Z(f)$), each containing in its interior at most $\frac{n}{D^d}$ points.

2 Upper Bounds for 3 Hops

In this section, we prove new upper bounds for 3-hop spanners for geometric intersection graphs. We begin with an easy lemma about general bipartite graphs in a lop-sided case:

▶ **Lemma 2.1.** *Let $G = (U \cup V, E)$ be a graph where $|U| = m$, $|V| = n$, and V is an independent set. Then G has a 3-hop spanner of size $O(m^2 + n)$.*

Proof. We construct a subgraph H of G as follows. For each $v \in V$ with $\deg(v) \geq 1$, let e_v be an arbitrary edge incident to v in G, and add e_v to H. Add $G[U]$ to H. Furthermore, for each pair of vertices $u, u' \in U$, let $\pi_{u,u'}$ be a 2-hop path between u and u' in G, if such

[1] In this paper, we use the O^* notation to hide n^ε factors for an arbitrarily small constant $\varepsilon > 0$.

[2] At the very end of their paper, Chan and Huang [16] specifically wrote: "Another question is whether near-linear bounds are possible for other intersection graphs not addressed here, e.g., for simplices in $\mathbb{R}^3$. Here, one might want to start more modestly with any upper bound better than for general graphs." Our $\Omega(n^{4/3})$ lower bound answers the question in the first sentence in the negative for simplices in $\mathbb{R}^3$, while our upper bounds show that the goal in the second sentence is attainable at least in the case of 3 hops.

a path exists, and add $\pi_{u,u'}$ to H. Clearly, H has $O(m^2+n)$ edges. We show that H is a 3-spanner of G. By construction, for each edge $e = uu'$ in $G[U]$, the distance between u and u' in H is one. Consider an edge uv in G with $u \in U$ and $v \in V$. If $e_v = uv$, then we can go from u to v in 1 hop. Otherwise, $e_v = u'v$, for some $u' \neq u$. Notice that in that case a 2-hop path $\pi_{u,u'}$ indeed exists and we can go from u to v in 3 hops by concatenating $\pi_{u,u'}$ with $u'v$. ◀

The above lemma allows us to almost immediately recover the standard $O(n^{3/2})$ upper bound on 3-hop spanners for general graphs (greedy algorithm yields same bound [3]):

▶ **Corollary 2.2.** *Every graph $G = (V, E)$ with n vertices has a 3-hop spanner of size $O(n^{3/2})$.*

Proof. Partition the vertices $V = \bigcup_{i=1}^{k} V_i$ into $k = \sqrt{n}$ groups of size $\sqrt{n}$ each, and apply Lemma 2.1 to each of the graphs $G_i = (V, E_i)$ where $E_i = E(G[V_i]) \cup E[V_i, V \setminus V_i]$. Namely, the edges in each graph comprise the inter-group edges and the bipartite edges between the group and its complement. Take the union of the spanners, with combined size $O(\sqrt{n} \cdot ((\sqrt{n})^2 + n)) = O(n^{3/2})$. It is easy to check that the resulting graph is indeed a 3-hop spanner as any original edge belongs to at least one of the graphs G_i as it is either induced by the U part of one of the graphs or a bipartite edge in one of the graphs. ◀

2.1 Semi-algebraic graphs

Here we improve the general bound $O(n^{3/2})$ for 3-hop spanners for semi-algebraic graphs. A natural approach is to apply known geometric divide-and-conquer and partitioning techniques to obtain biclique covers of sub-quadratic size, which then yields sub-quadratic upper bounds for 3-hop spanners. Unfortunately, for many higher-dimensional families of objects, the biclique cover bound can be much worse than $O(n^{3/2})$. Nevertheless, we show that these divide-and-conquer techniques can work synergistically with Lemma 2.1 to beat $O(n^{3/2})$: Specifically, we stop the recursion early when subproblems become sufficiently lop-sided and it is advantageous to switch to the $O(m^2+n)$ bound. In order to apply a divide-and-conquer we apply the polynomial partitioning technique as follows.

▶ **Theorem 2.3** (Guth and Katz [32, Theorem 4.1])**.** *Let P be a set of m points in $\mathbb{R}^D$ for a constant $D \in \mathbb{N}$. For every parameter $1 < k < m$, there exists a polynomial $f \in \mathbb{R}[x_1, \ldots, x_D]$ of degree $\deg(f) \in O(k^{1/D})$ such that every connected component of $\mathbb{R}^d \setminus Z(f)$ contains at most m/k points.*

This result generalizes to points lying in an irreducible variety.

▶ **Theorem 2.4** (Fox et al. [27, Theorem 4.3])**.** *Let P be a set of m points in a d-dimensional irreducible variety $V \subseteq \mathbb{R}^D$ for constant $1 \leq d \leq D \in \mathbb{N}$. For every parameter $1 < k < m$, there exists a polynomial $f \in \mathbb{R}[x_1, \ldots, x_D]$ of degree $\deg(f) \in O(k^{1/d})$ such that f is not in the (prime) ideal $I(V)$ and every connected component of $\mathbb{R}^d \setminus Z(f)$ contains at most m/k points.*

▶ **Corollary 2.5.** *Let P be a set of m points in a D-dimensional variety $V \subset \mathbb{R}^D$ for constants $1 \leq d \leq D \in \mathbb{N}$. For every parameter $1 < k < m$, there exists a partition $P = \bigcup_{i=1}^{q} P_i$ into $q = O(k)$ sets such that $|P_i| \leq m/k$; and for every $i = 1, \ldots q$, there is a connected set $\Delta_i \subset \mathbb{R}^d$, called a* cell*, such that $P_i \subset \Delta_i$ and the zero set $Z(h)$ of any polynomial of bounded degree crosses[3] $O(k^{1-1/D})$ cells (consequently, any semi-algebraic set crosses $O(k^{1-1/D})$ cells).*

[3] A set S *crosses* a cell Δ if both $\Delta \cap S$ and $\Delta \setminus S$ are nonempty.

Proof of Corollary 2.5 is deferred to arXiv [9].

▶ Remark 2.6. Corollary 2.5 uses a decomposition of the zero set of a partitioning polynomial into a union of irreducible varieties. For the algorithmic problem of constructing a partition $P = \bigcup_{i=1}^{q} P_i$ and corresponding cells Δ_i, one can avoid computing irreducible components with a multilevel partition; see [36] or [1].

▶ **Theorem 2.7.** *Let U and V be two sets of n elements. We are given points $p_1(u), \ldots, p_\ell(u) \in \mathbb{R}^D$ for each $u \in U$ and semi-algebraic sets $S_1(v), \ldots, S_\ell(v) \subset \mathbb{R}^D$ of constant description complexity for each $v \in V$, where D and ℓ are constants. For any ℓ-variate Boolean formula Φ, the graph*

$$G_\Phi[U,V] \;:=\; (U \cup V,\ \{(u,v) \in U \times V :\ \Phi([p_1(u) \in S_1(v)], \ldots, [p_\ell(u) \in S_\ell(v)])\})$$

has a 3-hop spanner with $O^*(n^{\frac{3D-2}{2D-1}}) = O^*(n^{\frac{3}{2} - \frac{1}{2(2D-1)}})$ edges.

Proof. We consider the more general setting, where $|U| = m$ and $|V| = n$. Let $T_\ell(m,n)$ denote the worst-case optimal size of a 3-hop spanner.

Let r be a parameter. By Corollary 2.5[4], we can partition $\mathbb{R}^D$ into $O(r^D)$ cells such that each cell contains at most m/r^D points of $\{p_\ell(u) : u \in U\}$, and each semi-algebraic set $S_\ell(v)$ crosses $O(r^{D-1})$ cells. For each cell Δ, let $U_\Delta = \{u \in U : p_\ell(u) \in \Delta\}$, $V_\Delta = \{v \in V : \partial S_\ell(v) \text{ crosses } \Delta\}$, $V'_\Delta = \{v \in V : S_\ell(v) \text{ contains } \Delta\}$, and $V''_\Delta = \{v \in V : S_\ell(v) \text{ does not intersect } \Delta\}$. Arbitrarily partition V_Δ into $\left\lceil \frac{|V_\Delta|}{n/r} \right\rceil$ subsets $V_\Delta^{(j)}$ of size at most n/r each, and recursively construct a 3-hop spanner for $G_\Phi[U_\Delta, V_\Delta^{(i)}]$ for each $V_\Delta^{(i)}$. Since $\sum_\Delta |V_\Delta| = O(nr^{D-1})$, the number of such recursive calls is $O(r^D)$. Furthermore, for each cell Δ, recursively construct a 3-hop spanner for $G_{\Phi'}[U_\Delta, V'_\Delta]$ and a 3-hop spanner for $G_{\Phi''}[U_\Delta, V''_\Delta]$, where Φ' (resp., Φ'') is the $(\ell-1)$-variate Boolean formula obtained by setting the ℓ-th variable to true (resp., false). The union of these spanners yields a 3-hop spanner for $G_\Phi[U,V]$. We thus obtain the following recurrence for $\ell \geq 1$:

$$T_\ell(m,n) \;\leq\; O(r^D)\left(T_\ell\left(\frac{m}{r^D}, \frac{n}{r}\right) + T_{\ell-1}(m,n)\right). \tag{1}$$

For base cases, we have $T_0(m,n) = O(m+n)$ (since a biclique $U \times V$ has a 3-hop spanner of linear size, consisting of two stars, centered at a fixed vertex of U and a fixed vertex of V), and we can use the upper bound $T_\ell(m,n) \leq O(m^2+n)$ for every ℓ by Lemma 2.1.

Assume inductively that $T_{\ell-1}(m,n) = O^*(m^{\frac{2D-2}{2D-1}} n^{\frac{D}{2D-1}})$ for $\sqrt{n} \leq m \leq n^D$. Choose r to be a large constant. Expand the recurrence for k levels so that $(m/r^{Dk})^2 \approx n/r^k$, i.e., $r^k \approx (m^2/n)^{\frac{1}{2D-1}}$. Then the recurrence (1) yields

$$\begin{aligned} T_\ell(m,n) \;&\leq\; \sum_{i=1}^{k} O(1)^i O(r^D) O(r^{Di}) \left(\frac{m}{r^{Di}}\right)^{\frac{2D-2}{2D-1}} \left(\frac{n}{r^i}\right)^{\frac{D}{2D-1}} \;+\; O(1)^k O(r^{Dk}) \left(\left(\frac{m}{r^{Dk}}\right)^2 + \frac{n}{r^k}\right) \\ &\leq\; O(1)^k r^D \cdot \left(m^{\frac{2D-2}{2D-1}} n^{\frac{D}{2D-1}} \;+\; n r^{k(D-1)}\right) \\ &=\; O(1)^k r^D \cdot \left(m^{\frac{2D-2}{2D-1}} n^{\frac{D}{2D-1}} \;+\; n(m^2/n)^{\frac{D-1}{2D-1}}\right) \;=\; O(1)^{\log_r m} r^D \cdot m^{\frac{2D-2}{2D-1}} n^{\frac{D}{2D-1}}. \end{aligned}$$

Making r an arbitrarily large constant, this yields $T_\ell(m,n) = O^*(m^{\frac{2D-2}{2D-1}} n^{\frac{D}{2D-1}})$ for $\sqrt{n} \leq m \leq n^D$. ◀

[4] For $D \leq 4$, we may alternatively use traditional $(1/r)$-*cuttings* [2, 18, 34] (in fact, later in Theorem 2.9, we will switch back to using cuttings), but for $D \geq 5$, optimal bounds on *vertical decompositions* are open and polynomial partitioning gives better results.

Theorem 2.7 immediately implies an $o(n^{3/2})$ bounds on 3-hop spanners for intersection graphs of any family of objects with constant description complexity.

▶ **Corollary 2.8.** *The intersection graph of every set of n simplices (or polyhedra each of constant complexity) in a constant dimension d has a 3-hop spanner with $O\left(n^{\frac{3}{2}-\frac{1}{O(d^2)}}\right)$ edges. Specifically, for $d=3$, the bound is $O^*(n^{10/7})$.*

Proof. Polyhedra of constant complexity can be decomposed into $O(1)$ simplices. A simplex in $\mathbb{R}^d$ has $d+1$ vertices, so it can be encoded as a point in $\mathbb{R}^{d(d+1)}$, and the set of all simplices intersecting a given simplex maps to a semi-algebraic set in $\mathbb{R}^{d(d+1)}$. We can thus apply Theorem 2.7 with $D=d^2+d$.

We can decrease D with more care. For example, for $d=3$, two simplices s and s' intersect if and only if (a) a vertex of s is inside s', or (b) an edge pq of s intersects a facet τ of s', or vice versa. The toughest case concerns (b), which is equivalent to: (i) the line through pq intersects τ, and (ii) p lies above the plane through τ, and (iii) q lies below the plane through τ, or vice versa. In (i), the line through pq can be encoded as a point in $\mathbb{R}^4$, whereas in (ii) or (iii), p or q is already a point in $\mathbb{R}^3$. We can thus apply Theorem 2.7 with $D=4$ (rather than $d(d+1)=12$). ◀

2.2 Improvement for point-halfspace incidence graphs

For incidence graphs between points and halfspaces, we can improve the exponent by switching to shallow cuttings [37] for the geometric divide-and-conquer:

▶ **Theorem 2.9.** *Let P be a set of n points and S be a set of n halfspaces in a constant dimension D. Then their incidence graph $G[P,S] := (P \cup S,\ \{(p,s) \in P \times S : p \in s\})$ has a 3-hop spanner with $O^*\left(n^{\frac{3\lfloor D/2\rfloor-2}{2\lfloor D/2\rfloor-1}}\right) = O^*\left(n^{\frac{3}{2}-\frac{1}{2(2\lfloor D/2\rfloor-1)}}\right)$ edges.*

Proof of Theorem 2.9 is deferred to arXiv version [9]. As one application, we obtain improved bounds for the case of balls, by a standard lifting mapping from balls to halfspaces:

▶ **Corollary 2.10.** *Every bipartite intersection graph between n red balls and n blue balls in a constant dimension d has a 3-hop spanner with $O^*(n^{\frac{3\lfloor d/2\rfloor+1}{2\lfloor d/2\rfloor+1}}) = O^*(n^{\frac{3}{2}-\frac{1}{2(2\lfloor d/2\rfloor+1)}})$ edges. For example, for $d=3$, the bound is $O^*(n^{4/3})$; for $d \in \{4,5\}$, it is $O^*(n^{7/5})$.*

Proof. A red ball with center $(x_1,\ldots,x_d)$ and radius y intersects a blue ball with center $(a_1,\ldots,a_d)$ and radius b iff $(x_1-a_1)^2+\cdots+(x_d-a_d)^2 \le (y+b)^2$, i.e., the point $(x_1,\ldots,x_d,y,x_1^2+\cdots+x_d^2-y^2) \in \mathbb{R}^{d+2}$ lies inside the halfspace $\{(x_1,\ldots,x_d,y,z) \in \mathbb{R}^{d+2} : z-2a_1x_1-\cdots-2a_dx_d-2by+a_1^2+\cdots+a_d^2-b^2 \le 0\}$. Thus, we can apply Theorem 2.9 with $D=d+2$. ◀

Note that in the non-bipartite case, an $O(n \log n)$ bound is known for 3-hop spanners [16], by exploiting the fatness of balls. In the next section, we show that the bound in Corollary 2.10 actually holds for *2-hop* spanners for balls in the non-bipartite setting.

3 Upper Bounds for 2 Hops

As mentioned in the introduction, for many families of objects, e.g., rectangles in $\mathbb{R}^2$, subquadratic bounds for 2-hop spanners of their (non-bipartite) intersection graphs are not possible. In this section, we show that nontrivial upper bounds for 2-hop spanners are possible for intersection graphs of *fat* objects (including balls).

3.1 When one side forms a clique

We begin by observing that most of the results in Section 2 works for 2 hops if one side of the bipartite graph forms a clique. This very simple observation will be the key to deriving our fat-object results later:

▶ **Lemma 3.1.** *Let $G = (U \cup V, E)$ be a bipartite graph where $|U| = m$ and $|V| = n$. Let G^+ be the union of G and a clique on U. Then G^+ has a 2-hop spanner of size $O(m^2 + n)$.*

Proof. We construct a spanner subgraph H^+ of G^+ as follows. For each vertex $v \in V$, $\deg(v) \geq 1$, let e_v be an arbitrary edge incident to v, and add e_v to H^+. Furthermore, add the clique on U to H^+. We show that H^+ is a 2-hop spanner of G^+. Consider an edge uv in G. If $e_v = uv$, then we can go from u to v in 1 hop. Otherwise, say e_v is $u'v$. We can go from u to v in 2 hops via the edges uu' and $u'v$. ◀

The above lemma implies an $O(n^{3/2})$ upper bound on 2-hop spanners in this setting, by the same grouping trick from the proof of Corollary 2.2:

▶ **Corollary 3.2.** *Let $G = (U \cup V, E)$ be a bipartite graph where $|U| + |V| = n$. Let G^+ be the union of G and a clique on U. Then G^+ has a 2-hop spanner of size $O(n^{3/2})$.*

We also immediately obtain the following analogs to Theorems 2.7 and 2.9:

▶ **Theorem 3.3.** *Let U and V be two sets of n elements. We are given points $p_1(u), \ldots, p_\ell(u) \in \mathbb{R}^D$ for each $u \in U$ and semi-algebraic sets $S_1(v), \ldots, S_\ell(v) \subset \mathbb{R}^D$ of constant description complexity for each $v \in V$, where D and ℓ are constants. For any ℓ-variate Boolean formula Φ, the graph*

$$G \;=\; (U \cup V,\; \{(u,v) \in U \times V :\; \Phi([p_1(u) \in S_1(v)], \ldots, [p_\ell(u) \in S_\ell(v)])\} \;\cup\; (U \times U))$$

has a 2-hop spanner with $O^*(n^{\frac{3D-2}{2D-1}}) = O^*(n^{\frac{3}{2} - \frac{1}{2(2D-1)}})$ edges.

Proof. We follow the proof of Theorem 2.7, but using Lemma 3.1 in place of Lemma 2.1. We also add a 2-hop spanner for $U \times U$ of $O(m)$ size (namely, a star). ◀

▶ **Theorem 3.4.** *Let P be a set of n points and S be a set of n halfspaces in a constant dimension D. Then the graph $G = (P \cup S,\; \{(p,s) \in P \times S : p \in s\} \cup (P \times P))$ has a 2-hop spanner with $O^*(n^{\frac{3\lfloor D/2 \rfloor - 2}{2\lfloor D/2 \rfloor - 1}}) = O^*(n^{\frac{3}{2} - \frac{1}{2(2\lfloor D/2 \rfloor - 1)}})$ edges.*

Proof. We follow the proof of Theorem 2.7, but using Lemma 3.1 in place of Lemma 2.1, and just replacing the path $\pi_{p,p'}$ with an edge pp'. ◀

3.2 Intersection graphs of fat objects

To obtain our results for fat objects, our idea is to adapt one of the methods by Chan and Huang [16, Section 3.1], originally developed for constructing 3-hop spanners of $O(n \log n)$ size. In adapting their method to construct 2-hop spanners, we realize that the main subproblems arising from the recursion correspond to the case when one side forms a clique – a case which fortunately has already been addressed by the previous subsection! If the fat objects are all of similar size, then a simple grid approach will easily reduce to the one-sided clique case. For fat objects of arbitrary sizes, Chan and Huang [16] used a divide-and-conquer approach based on shifted quadtrees.

We use the following definition of fatness [13]: A family of objects is *fat* if for every axis-aligned hypercube γ with side length ℓ, there exist $O(1)$ points hitting all objects that intersect γ and have diameter at least ℓ. There exists a number of different definitions

of fatness in the geometry literature (e.g., see [8, 23]). We find the above definition most appropriate for our purpose. It is not difficult to see that arrangements of Euclidean balls or hyperrectangles of bounded aspect ratios in $\mathbb{R}^d$ are fat according to this definition.

▶ **Theorem 3.5.** *Let V be a set of n fat objects in $\mathbb{R}^d$ for constant d. Then the intersection graph G of V has a 2-hop spanner with $O(n^{3/2})$ edges.*

Proof. Recall that a *quadtree cell* is an axis-aligned box of the form $[i_1/2^k, (i_1+1)/2^k) \times \cdots \times [i_d/2^k, (i_d+1)/2^k)$ for some integers $i_1, \ldots, i_d, k$. An object u of diameter ℓ is *C-aligned* if it is contained in a quadtree cell of side length $C\ell$. As argued in [16, Section 3.1], as a consequence of a known shifting lemma [13], one can find $O(d)$ vectors $\tau_1, \ldots, \tau_{O(d)}$ with the property that for every pair of objects u and v, there exists at least one vector τ_j such that $u+\tau_j$ and $v+\tau_j$ are both C-aligned for some $C = O(d)$. For each τ_j, it suffices to construct a 2-hop spanner for the subset of all objects u such that $u+\tau_j$ is C-aligned; we can then output the union of these $O(d)$ spanners. From now on, we may thus assume that all objects are C-aligned. Let $T(n)$ be the worst-case optimal 2-hop spanner size under this assumption.

1. First find a quadtree cell γ such that there are at most αn objects completely inside γ and at most αn objects completely outside γ, with $\alpha := \frac{2^d}{2^d+1}$; see [16, Lemma 10] (based on earlier work in [5, 12]). This cell corresponds to a "centroid" of the quadtree. Recursively construct a 2-hop spanner for the objects completely inside γ and a 2-hop spanner for the objects completely outside γ.
2. Let Q_γ be a set of points hitting all objects that intersect $\partial\gamma$. By alignedness, these objects have diameter at least ℓ_γ/C, where ℓ_γ denotes the side length of γ. By fatness, a hitting set of size $|Q_\gamma| = O(1)$ exists (since $\partial\gamma$ can be covered by C^d hypercubes of side length ℓ_γ/C).
3. For each point $q \in Q_\gamma$, let U_q be the subset of all objects hit by q (this subset induces a clique). Let G_q be the bipartite intersection graph between U_q and V. Construct a 2-hop spanner for $G_q \cup (U_q \times U_q)$ by Corollary 3.2.

We claim that the union of the spanners found is a 2-hop spanner of G. To see this, consider an edge uv in G. If u and v are both inside γ or both outside γ, then u and v are reachable by 2 hops by induction. Otherwise, one of the objects, say u, intersects $\partial\gamma$. Then u is hit by some point $q \in Q_\gamma$, and so $uv \in G_q$ and u and v are reachable by 2 hops.

We thus obtain the following recurrence:

$$T(n) \;\le\; \max_{n_1,n_2 \le \alpha n:\; n_1+n_2 \le n} \Big(T(n_1) + T(n_2)\Big) + O(n^{3/2}).$$

This solves to $T(n) = O(n^{3/2})$. ◀

We similarly obtain improvements when the fat objects are semialgebraic with constant description complexity:

▶ **Theorem 3.6.** *Let V be a set of n fat objects in $\mathbb{R}^d$ for constant d. We are given points $p_1(v), \ldots, p_\ell(v) \in \mathbb{R}^D$ and semi-algebraic set $S_1(v), \ldots, S_\ell(v) \subset \mathbb{R}^D$ of constant description complexity for each $v \in V$, where D and ℓ are constants, satisfying the following property: two objects $u, v \in V$ intersect iff $\Phi([p_1(u) \in S_1(v)], \ldots, [p_\ell(u) \in S_\ell(v)])$, where Φ is an ℓ-variate Boolean formula. Then the intersection graph G of V has a 2-hop spanner with $O^*(n^{\frac{3D-2}{2D-1}}) = O^*(n^{\frac{3}{2}-\frac{1}{2(2D-1)}})$ edges.*

Proof. Same as the proof of Theorem 3.5, but using Theorem 3.3 instead of Corollary 3.2. ◀

▶ **Corollary 3.7.** *The intersection graph of every set of n fat simplices (or fat polyhedra each of constant complexity, e.g., non-axis-aligned hypercubes) in a constant dimension d has a 2-hop spanner with $O(n^{\frac{3}{2}-\frac{1}{O(d^2)}})$ edges. Specifically, for $d = 3$, the bound is $O^*(n^{10/7})$.*

Proof. As in the proof of Corollary 2.8, this follows by setting $D = d^2$, or in the $d = 3$ case, $D = 4$. ◀

▶ **Theorem 3.8.** *Let V be a set of n fat objects in a constant dimension d. We are given a point $p(v) \in \mathbb{R}^D$ and a halfspace $S(v) \subset \mathbb{R}^D$ for each $v \in V$, for a constant D, satisfying the following property: two objects u and v intersect iff $p(u) \in S(v)$. Then the intersection graph G of V has a 2-hop spanner with $O^*(n^{\frac{3\lfloor D/2\rfloor-2}{2\lfloor D/2\rfloor-1}}) = O^*(n^{\frac{3}{2}-\frac{1}{2(2\lfloor D/2\rfloor-1)}})$ edges.*

Proof. Same as the proof of Theorem 3.5, but using Theorem 3.4 instead of Corollary 3.2. ◀

▶ **Corollary 3.9.** *The intersection graph of n balls in a constant dimension d has a 2-hop spanner with $O^*(n^{\frac{3\lfloor d/2\rfloor+1}{2\lfloor d/2\rfloor+1}}) = O^*(n^{\frac{3}{2}-\frac{1}{2(2\lfloor d/2\rfloor+1)}})$ edges. For example, for $d = 3$, the bound is $O^*(n^{4/3})$; for $d \in \{4, 5\}$, the bound is $O^*(n^{7/5})$.*

Proof. As in the proof of Corollary 2.10, this follows from Theorem 3.8 by setting $D = d + 2$. ◀

For fat axis-aligned boxes, we have the following theorem, which generalizes a result by Conroy and Tóth [20] from $d = 2$ to higher dimensions:

▶ **Theorem 3.10.** *For every $d \geq 2$, the intersection graph of n fat axis-aligned boxes (e.g., axis-aligned hypercubes) in a constant dimension d has a 2-hop spanner with $O(n \log^{d+1} n)$ edges.*

Proof. Same as the proof of Theorem 3.6, but to construct a 2-hop spanner for $G_p \cup (U_q \times U_q)$, we use a known *biclique cover* construction for the bipartite intersection graph G_p of boxes: this gives a collection of bicliques of the form $U_i \times V_i$, covering all the edges of G_p, with total size $\sum_i (|U_i| + |V_i|) = O(n \log^d n)$. (The construction is obtained by range-tree-style divide-and-conquer; e.g., see [14].) When the clique $U_i \times U_i$ is added to the biclique $U_i \times V_i$, there is a trivial 2-hop spanner of size $O(|U_i| + |V_i|)$ (namely, take a star centered at an arbitrary point in U_i). We just take the union of these spanners (together with a linear-size 2-hop spanner for $U_q \times U_q$) and obtain a spanner of size $O(n \log^d n)$ for $G_p \cup (U_q \times U_q)$. The recursion in the proof of Theorem 3.6 causes one more logarithmic factor. ◀

We note that the known $O(n \log n)$-size 3-hop spanner construction for axis-aligned rectangles in $\mathbb{R}^2$ by Chan and Huang [16] can be extended to give an $O(n \log^{d-1} n)$-size 3-hop spanner for axis-aligned boxes in $\mathbb{R}^d$. We defer this result to arXiv [9].

4 Lower Bounds

In this section, we prove lower bounds for 3-hop and 2-hop spanners, by relating the problem to the combinatorial question of bounding the size of geometric graphs avoiding $K_{2,2}$. Our lower bounds for 3-hop spanners are obtained from the following simple observation:

▶ **Observation 4.1.** *If a bipartite graph G does not contain any 4-cycle, i.e., $K_{2,2}$, then any 3-hop spanner of G must include all its edges.*

For 2-hop spanners, one could similarly derive lower bounds from constructions of graphs without 3-cycles (i.e., triangle-free graphs). However, for fat objects, triangle-free intersection graphs actually are known to have linear size. Instead, we propose the following lemma, showing that constructions of large bipartite intersection graphs between two sets of objects that avoid $K_{2,2}$ (where objects within each set are allowed to intersect) can also yield lower bounds for 2-hop spanners for (not-necessarily bipartite) intersection graphs:

▶ **Lemma 4.2.** *If $G = (U \cup V, E)$ is a graph (with $U \cap V = \emptyset$), and the bipartite subgraph $E' = E \cap (U \times V)$ does not contain $K_{2,2}$, then any 2-hop spanner of G must have at least $|E'|$ edges.*

Proof. Basically, we want to show that extra edges in $U \times U$ and $V \times V$ do not help. We use a charging argument. Let H be a 2-hop spanner of G. For each edge $uv \in E'$ with $u \in U$ and $v \in V$, if $uv \in H$, we charge the edge uv to itself. Otherwise, H must contain edges ux and xv for some x. Without loss of generality, say $x \in U$. We charge the edge uv to the edge ux.

Since the total charge is $|E'|$, it suffices to show that no edge in H receives more than one charge. Suppose that some edge e in H receives charges from two different edges $f_1, f_2 \in E'$. For f_1 to be charged to e, there must exist an edge $g_1 \in E'$ such that e, f_1, g_1 form a triangle. Similarly, there must exist an edge $g_2 \in E'$ such that e, f_2, g_2 form a triangle. Since $f_1, f_2 \notin H$ and $g_1, g_2 \in H$, the edges f_1, g_1, f_2, g_2 are distinct and form a $K_{2,2}$ in E': a contradiction. ◀

We can use known combinatorial results to construct $K_{2,2}$-free intersection graphs:

▶ **Lemma 4.3.** *For every positive integer n, the following hold:*

(a) *There exist n points and n lines in $\mathbb{R}^2$ whose incidence graph does not contain $K_{2,2}$ and has $\Omega(n^{4/3})$ edges.*

(b) *There exist n tetrahedra in $\mathbb{R}^3$ whose intersection graph is bipartite, does not contain $K_{2,2}$, and has $\Omega(n^{4/3})$ edges.*

(c) *There exist n red/blue congruent regular tetrahedra (or congruent non-axis-aligned cubes) in $\mathbb{R}^3$ such that the bipartite intersection graph between the red tetrahedra and the blue tetahedra does not contain $K_{2,2}$ and has $\Omega(n^{4/3})$ edges.*

(d) *There exist n points and n halfspaces in $\mathbb{R}^5$ whose incidence graph does not contain $K_{2,2}$ and has $\Omega(n^{4/3})$ edges.*

(e) *There exist n red/blue congruent balls in $\mathbb{R}^5$ such that the bipartite intersection graph between the red balls and the blue balls does not contain $K_{2,2}$ and has $\Omega(n^{4/3})$ edges.*

(f) *There exist n points and n axis-aligned boxes in $\mathbb{R}^d$ whose incidence graph does not contain $K_{2,2}$ and has $\Omega(n(\log n/\log\log n)^{d-1})$ edges.*

(g) *There exist n axis-aligned boxes in $\mathbb{R}^d$ whose intersection graph is bipartite, does not contain $K_{2,2}$, and has $\Omega(n(\log n/\log\log n)^{d-2})$ edges.*

(h) *There exist n red/blue congruent axis-aligned hypercubes in $\mathbb{R}^d$ such that the bipartite intersection graph between the red hypercubes and blue hypercubes does not contain $K_{2,2}$ and has $\Omega(n(\log n/\log\log n)^{\lfloor d/2\rfloor-1})$ edges.*

Proof.

(a) This is well-known, by a construction of Erdős [26] (see also [6, 24]) and incidence graphs between points and lines in $\mathbb{R}^2$ automatically do not contain $K_{2,2}$.

(b) Let P be the set of points and L be the set of lines in $\mathbb{R}^2$ from (a). Map each point $p \in P$ to the red vertical line $\varphi(p) = p \times (-\infty, \infty)$ in $\mathbb{R}^3$. Map the i-th line $\ell \in L$ to a blue line $\psi(\ell) = \ell \times \{i\}$ in $\mathbb{R}^3$ (which lies on the horizontal plane $z = i$). Then $\varphi(p)$ intersects $\psi(\ell)$ iff p lies on ℓ. Moreover, there are no red-red or blue-blue intersections. The result then follows by viewing these red/blue lines as thin tetrahedra.

(c) Let P be the set of points and L be the set of lines in $\mathbb{R}^2$ from (a). We may clip each line $\ell \in L$ to a line segment of a sufficiently large length M. Map each point $p \in P$ to some red regular tetrahedron $\varphi(p)$ in $\mathbb{R}^3$ that has side length M, lies inside the halfspace $z \geq 0$, and touches the plane $z = 0$ precisely at the point $p \times \{0\}$. Map each line segment $\ell \in P$ to some blue regular tetrahedron $\psi(\ell)$ in $\mathbb{R}^3$ that has side length M, lies inside the halfspace $z \leq 0$, and touches the plane $z = 0$ precisely at the line segment $\ell \times \{0\}$. Then $\varphi(p)$ intersects $\psi(\ell)$ iff p lies on ℓ. The construction is similar for non-axis-aligned cubes.

(d) This was noted by Chan and Har-Peled [15]. Specifically, let P be the set of points and L be the set of lines in $\mathbb{R}^2$ from (a). A point $(x, y) \in P$ is incident on a line in L with equation $ax + by = 1$ iff $(ax + by - 1)^2 \leq \delta$, i.e., $a^2x^2 + 2abxy + b^2y^2 - 2ax - 2by + 1 \leq \delta$, for a sufficiently small $\delta > 0$. Map each point $p = (x, y) \in P$ to a point $\varphi(p) = (x^2, xy, y^2, x, y) \in \mathbb{R}^5$. Map each line $\ell \in L$ with equation $ax + by = 1$ to a halfspace $\psi(\ell) = \{(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5) \in \mathbb{R}^5 : a^2\xi_1 + 2ab\xi_2 + b^2\xi_3 - 2a\xi_4 - 2b\xi_5 + 1 \leq \delta\}$. Then $\varphi(p)$ lies inside $\psi(\ell)$ iff p lies on ℓ.

(e) Let P be the set of points and S be the set of halfspaces in $\mathbb{R}^5$ from (d). We can view each halfspace of S as a giant ball, of a fixed and sufficiently large radius M. Now map each point $p \in P$ to a red ball $\varphi(p)$ centered at p of radius $M/2$. Map each ball $s \in S$ to a blue ball $\psi(s)$ with the same center of radius $M/2$. Then $\varphi(p)$ intersects $\psi(s)$ iff p lies inside s.

(f) This follows from a construction by Chazelle [17], as noted by Chan and Har-Peled [15].

(g) Let P be the set of points and S the set of boxes in $\mathbb{R}^{d-1}$ from (f). Map each point $p \in P$ to a red vertical line $\varphi(p) = p \times (-\infty, \infty)$ in $\mathbb{R}^d$. Map the i-th box $s \in S$ to a blue box $\psi(s) = s \times \{i\}$ in $\mathbb{R}^d$. Then $\varphi(p)$ intersects $\psi(s)$ iff p is inside s. Moreover, there are no red-red or blue-blue intersections. The result then follows by viewing the red lines as thin boxes.

(h) Say d is even. Let P be the set of points and S be the set of boxes in $\mathbb{R}^{d/2}$ from (f). Map each point $p = (x_1, \ldots, x_{d/2}) \in P$ to a red hypercube $\varphi(p) = [-x_1, -x_1 + M] \times [x_1, x_1 + M] \times \cdots \times [-x_{d/2}, -x_{d/2} + M] \times [x_{d/2}, x_{d/2} + M]$ in $\mathbb{R}^d$ for a sufficiently large side length M. Map each box $s = [a_1, b_1] \times \cdots \times [a_{d/2}, b_{d/2}] \in S$ to a blue hypercube $\psi(s) = [-a_1 - M, -a_1] \times [b_1 - M, b_1] \times \cdots \times [a_{d/2} - M, a_{d/2}] \times [b_{d/2} - M, b_{d/2}]$ in $\mathbb{R}^d$. Then $\varphi(p)$ intersects $\psi(s)$ iff p lies inside s. ◀

Lower bounds for 3-& 2-hop spanners for various intersection graphs now follows.

▶ **Theorem 4.4.** *For every positive integer n, the following holds:*

(i) *There exist n tetrahedra in $\mathbb{R}^3$ such that any 3-hop spanner of their intersection graph requires $\Omega(n^{4/3})$ edges.*

(ii) *There exist n congruent regular tetrahedra (or congruent non-axis-aligned cubes) in $\mathbb{R}^3$ such that any 2-hop spanner of their intersection graph requires $\Omega(n^{4/3})$ edges.*

(iii) *There exist n congruent balls in $\mathbb{R}^5$ such that any 2-hop spanner of their intersection graph requires $\Omega(n^{4/3})$ edges.*

(iv) *There exist n axis-aligned boxes in $\mathbb{R}^d$ such that any 3-hop spanner of their intersection graph requires $\Omega(n(\log n/\log\log n)^{d-2})$ edges.*

(v) *There exist n congruent axis-aligned hypercubes in $\mathbb{R}^d$ such that any 2-hop spanner of their intersection graph requires $\Omega(n(\log n/\log\log n)^{\lfloor d/2 \rfloor - 1})$ edges.*

Proof.

(i) By Lemma 4.3(b) and Observation 4.1.

(ii) By Lemma 4.3(c) and Lemma 4.2.

(iii) By Lemma 4.3(e) and Lemma 4.2.

(iv) By Lemma 4.3(g) and Observation 4.1.

(v) By Lemma 4.3(h) and Lemma 4.2. ◀

▶ Remark 4.5. It remains open to prove better lower bounds for simplices in dimensions $d > 3$ or balls in dimensions $d > 5$, ideally with exponent converging to $3/2$ as d increases. It is possible to adapt Observation 4.1 and Lemma 4.2 to avoid $K_{2,c}$ instead of $K_{2,2}$ for any constant c (while losing some $c^{O(1)}$ factors), but we are not aware of any stronger lower bounds on geometric graphs avoiding $K_{2,c}$ that could be exploited here (except in some lop-sided bipartite cases).

One common property to all the geometric intersection graphs mentioned in this paper is that they all have bounded VC-dimension[5]. As already noted in the introduction, bounded VC-dimension alone does not guarantee a 3-spanner with $o(n^{3/2})$ edges, as demonstrated in the following theorem:

▶ **Theorem 4.6.** *For any $n \in \mathbb{N}$, there exists a graph $G = (V_1 \cup V_2, E)$ with VC-dimension at most 2 so that any 3-spanner for G has $\Omega(n^{3/2})$ edges.*

Proof. The construction consists of the known constructions of bipartite graphs without C_4 with $\Omega(n^{3/2})$ edges (e.g., the construction with projective planes over finite fields $(F_p)^2$). Its easy to see that the VC-dimension of such graphs is at most 2. By Observation 4.1, any 3-spanner must use all bipartite edges. ◀

▶ Remark 4.7. Bounded VC-dimension is known to imply the existence of spanning trees with sublinear crossing number [19], biclique covers with subquadratic size, and various other properties often used in geometric range searching. Along the same lines, one may wonder whether general set systems with bounded VC-dimension admit some analog to $(1/r)$-cuttings [18] or polynomial partitioning. Interestingly, our work provides a negative answer to this question: An analog to cuttings would imply an $o(n^{3/2})$ upper bound for 3-hop spanners, by following the method in the proof of Theorems 2.7 and 2.9, and this would contradict Theorem 4.6.

References

1 Pankaj K. Agarwal, Boris Aronov, Esther Ezra, Matthew J. Katz, and Micha Sharir. Intersection queries for flat semi-algebraic objects in three dimensions and related problems. *ACM Trans. Algorithms*, 2025. Just Accepted. doi:10.1145/3721290.

2 Pankaj K. Agarwal and Jirí Matoušek. On range searching with semialgebraic sets. *Discret. Comput. Geom.*, 11:393–418, 1994. doi:10.1007/BF02574015.

3 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.

4 Boris Aronov, Mark de Berg, Esther Ezra, and Micha Sharir. Improved bounds for the union of locally fat objects in the plane. *SIAM J. Comput.*, 43(2):543–572, 2014. doi:10.1137/120891241.

5 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.

6 Martin Balko, Adam Sheffer, and Ruiwen Tang. The constant of point-line incidence constructions. *Comput. Geom.*, 114:102009, 2023. doi:10.1016/J.COMGEO.2023.102009.

[5] The *VC-dimension* of a graph is defined as the VC-dimension of the set system induced by the neighborhoods of its vertices.

7 Abdul Basit, Artem Chernikov, Sergei Starchenko, Terence Tao, and Chieu-Minh Tran. Zarankiewicz's problem for semilinear hypergraphs. *Forum of Mathematics, Sigma*, 9:e59, 2021. doi:10.1017/fms.2021.52.

8 Mark de Berg, A. Frank van der Stappen, Jules Vleugels, and Matthew J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002. doi:10.1007/S00453-002-0961-X.

9 Sujoy Bhore, Timothy M. Chan, Zhengcheng Huang, and Shakhar Smorodinsky Csaba D. Tóth. Sparse bounded hop-spanners for geometric intersection graphs. *CoRR*, abs/2504.05861, 2025. doi:10.48550/arXiv.2504.05861.

10 Ahmad Biniaz. Plane hop spanners for unit disk graphs: Simpler and better. *Comput. Geom.*, 89:101622, 2020. doi:10.1016/j.comgeo.2020.101622.

11 Nicolas Catusse, Victor Chepoi, and Yann Vaxès. Planar hop spanners for unit disk graphs. In *Proc. 6th ALGOSENSORS*, volume 6451 of *LNCS*, pages 16–30. Springer, 2010. doi:10.1007/978-3-642-16988-5_2.

12 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discret. Comput. Geom.*, 20(3):359–373, 1998. doi:10.1007/PL00009390.

13 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.

14 Timothy M. Chan. Dynamic subgraph connectivity with geometric applications. *SIAM J. Comput.*, 36(3):681–694, 2006. doi:10.1137/S009753970343912X.

15 Timothy M. Chan and Sariel Har-Peled. On the number of incidences when avoiding an induced biclique in geometric settings. In *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1398–1413, 2023. doi:10.1137/1.9781611977554.ch50.

16 Timothy M. Chan and Zhengcheng Huang. Constant-hop spanners for more geometric intersection graphs, with even smaller size. In *Proc. 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 23:1–23:16. Schloss Dagstuhl, 2023. doi:10.4230/LIPICS.SOCG.2023.23.

17 Bernard Chazelle. Lower bounds for orthogonal range searching: I. The reporting case. *J. ACM*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

18 Bernard Chazelle. Cuttings. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035179.CH25.

19 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.

20 Jonathan B. Conroy and Csaba D. Tóth. Hop-spanners for geometric intersection graphs. *Journal of Computational Geometry*, 14(2):26–64, 2023. doi:10.20382/jocg.v14i2a3.

21 James Davies. Box and segment intersection graphs with large girth and chromatic number. *Advances in Combinatorics*, 2021. doi:10.19086/aic.25431.

22 Adrian Dumitrescu, Anirban Ghosh, and Csaba D. Tóth. Sparse hop spanners for unit disk graphs. *Comput. Geom.*, 100:101808, 2022. doi:10.1016/J.COMGEO.2021.101808.

23 Alon Efrat, Matthew J. Katz, Frank Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. *Comput. Geom.*, 15(4):215–227, 2000. doi:10.1016/S0925-7721(99)00059-0.

24 György Elekes. Sums versus products in number theory, algebra and Erdős geometry. In Gábor Halász, László Lovász, Miklós Simonovits, and Vera T. Sós, editors, *Paul Erdős and his Mathematics II*, Bolyai Society Mathematical Studies, pages 241–290. Springer, Berlin, 2001. URL: https://link.springer.com/book/9783540422365.

25 Paul Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, pages 29–36, Prague, 1964. Publishing House of the Czechoslovak Academy of Sciences. URL: https://old.renyi.hu/~p_erdos/1964-06.pdf.

26 Paul Erdős. Problems and results in combinatorial geometry. In *Discrete Geometry and Convexity*, volume 440 of *Ann. New York Acad. Sci.*, pages 1–11. Wiley Online Library, 1985. URL: https://old.renyi.hu/~p_erdos/1985-23.pdf.

27 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Euro. Math. Soc.*, 19(6):1785–1810, 2017. doi:10.4171/JEMS/705.

28 Martin Fürer and Shiva Prasad Kasiviswanathan. Spanners for geometric intersection graphs with applications. *J. Comput. Geom.*, 3(1):31–64, 2012. doi:10.20382/jocg.v3i1a3.

29 Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanners for routing in mobile networks. *IEEE J. Sel. Areas Commun.*, 23(1):174–185, 2005. doi:10.1109/JSAC.2004.837364.

30 Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM J. Comput.*, 35(1):151–169, 2005. doi:10.1137/S0097539703436357.

31 Joachim Gudmundsson, Giri Narasimhan, and Michiel H. M. Smid. Applications of geometric spanner networks. In *Encyclopedia of Algorithms*, pages 86–90. Springer, 2016. doi:10.1007/978-1-4939-2864-4_15.

32 Larry Guth and Nets Hawk Katz. On the Erdős distinct distances problem in the plane. *Annals of Mathematics*, 181(1):155–190, 2015. doi:10.4007/annals.2015.181.1.2.

33 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discret. Comput. Geom.*, 1:59–70, 1986. doi:10.1007/BF02187683.

34 Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51(5):699–730, 2004. doi:10.1145/1017460.1017461.

35 Xiang-Yang Li. Algorithmic, geometric and graphs issues in wireless networks. *Wirel. Commun. Mob. Comput.*, 3(2):119–140, 2003. doi:10.1002/WCM.107.

36 Jirí Matousek and Zuzana Patáková. Multilevel polynomial partitions and simplified range searching. *Discret. Comput. Geom.*, 54(1):22–41, 2015. doi:10.1007/S00454-015-9701-2.

37 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

38 Joseph S.B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In *Handbook of Discrete and Computational Geometry*, chapter 32, pages 849–874. Chapman and Hall/CRC, 2017.

39 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

40 Rajmohan Rajaraman. Topology control and routing in ad hoc networks: a survey. *SIGACT News*, 33(2):60–73, 2002. doi:10.1145/564585.564602.

41 István Tomon and Dmitriy Zakharov. Turán-type results for intersection graphs of boxes. *Combinatorics, Probability and Computing*, 30(6):982–987, 2021. doi:10.1017/S0963548321000171.

Finding a Shortest Curve That Separates Few Objects from Many

Therese Biedl
University of Waterloo, Canada

Éric Colin de Verdière
LIGM, CNRS, Univ Gustave Eiffel, F-77454 Marne-la-Vallée, France

Fabrizio Frati
Roma Tre University, Rome, Italy

Anna Lubiw
University of Waterloo, Canada

Günter Rote
Institut für Informatik, Freie Universität Berlin, Germany

Abstract

We present a fixed-parameter tractable (FPT) algorithm to find a shortest curve that encloses a set of k required objects in the plane while paying a penalty for enclosing unwanted objects.

The input is a set of interior-disjoint simple polygons in the plane, where k of the polygons are *required* to be enclosed and the remaining *optional* polygons have non-negative penalties. The goal is to find a closed curve that is disjoint from the polygon interiors and encloses the k required polygons, while minimizing the length of the curve plus the penalties of the enclosed optional polygons. If the penalties are high, the output is a shortest curve that separates the required polygons from the others. The problem is NP-hard if k is not fixed, even in very special cases. The runtime of our algorithm is $O(3^k n^3)$, where n is the number of vertices of the input polygons.

We extend the result to a graph version of the problem where the input is a connected plane graph with positive edge weights. There are k required faces; the remaining faces are optional and have non-negative penalties. The goal is to find a closed walk in the graph that encloses the k required faces, while minimizing the weight of the walk plus the penalties of the enclosed optional faces. We also consider an inverted version of the problem where the required objects must lie outside the curve. Our algorithms solve some other well-studied problems, such as geometric knapsack.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Enclosure, curve, separation, weakly simple polygon, Euler tour

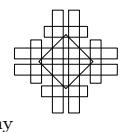
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.18

Related Version *Full Version*: `https://arxiv.org/abs/2504.03558v1` [5]

Funding *Therese Biedl*: Natural Science and Engineering Research Council of Canada.
Fabrizio Frati: Supported by the European Union, Next Generation EU, Mission 4, Component 1, CUP J53D23007130006 PRIN proj. 2022ME9Z78 "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization".
Anna Lubiw: Supported by the Natural Sciences and Engineering Research Council of Canada.

Acknowledgements This work was initiated at the Dagstuhl seminar 24072 "Triangulations in Geometry and Topology" in February 2024. We thank the organizers, Maike Buchin, Jean Cardinal, Arnaud de Mesmay, and Jonathan Spreer, as well as all participants, for the stimulating atmosphere. We also thank Alexander Wolff for significant contributions, Saul Schleimer for useful discussions, and the anonymous reviewers for their constructive feedback.

© Therese Biedl, Éric Colin de Verdière, Fabrizio Frati, Anna Lubiw, and Günter Rote;
licensed under Creative Commons License CC-BY 4.0



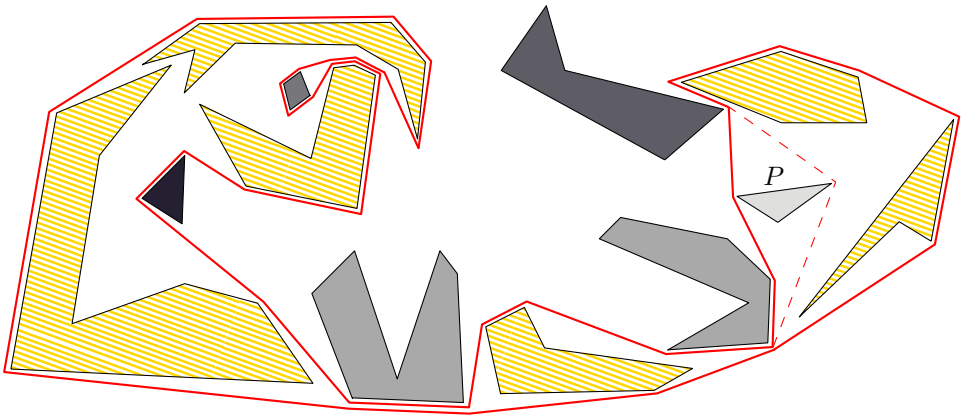


Figure 1 The Geometric-Enclosure-with-Penalties problem. The weakly simple polygon W (in red) encloses the required polygons R (yellow hatched) while paying a penalty for enclosing any of the optional polygons O (in gray, darker for larger penalties). Overlapping paths are slightly displaced for visibility. Changing W via the dashed path to put polygon P outside would be preferable if the increase in length is less than the penalty of P.

1 Introduction

We investigate the separation problem of finding a shortest curve that encloses a subset of objects while excluding other objects. A very basic setting is for points in the plane: given n points in the plane and a subset of size k, find a minimum-perimeter polygon containing the specified k points and excluding the other $n - k$ points. This problem is NP-hard when k may be large, as proved by Eades and Rappaport [11] for the case $k = n/2$ via a simple reduction from the Travelling Salesman Problem.

As a special case of our main result, we give the first algorithm for this problem that is fixed-parameter tractable (FPT) in k. Our result is far more general and applies in two settings, a geometric setting and a graph-theoretic setting.

Geometric-Enclosure-with-Penalties. We generalize from point objects to objects that are interior-disjoint simple polygons in the plane, and we consider a *weighted* form of exclusion.

Input. The input is a set of simple interior-disjoint polygons partitioned into a set R of k **required polygons** and the remaining set O of **optional polygons**. Each optional polygon $P \in O$ comes with a non-negative **penalty** π_P where we allow $\pi_P = +\infty$.

Output. The goal is to find a weakly simple polygon W that does not intersect the interior of any input polygon and encloses all polygons of R while minimizing the **cost** $c(W)$, which is defined to be the Euclidean length of W plus the penalties of the polygons of O that are inside W. See Figure 1 for an example. A polygon with penalty $+\infty$ must be excluded. A polygon with penalty 0 may be included or excluded without making a difference, so it only acts as an obstacle to the solution curve. As Figure 1 illustrates, the problem would be ill-defined if we required the solution curve W to be a simple polygon. The natural condition is that W should be a **weakly simple polygon**, whose boundary may touch or overlap itself but not cross itself. We give a precise definition in Section 2.1. An important property is that a weakly simple polygon encloses a well-defined region. Our first main result is:

▶ **Theorem 1.** Geometric-Enclosure-with-Penalties *for k required polygons can be solved in $O(3^k n^3)$ time and $O(2^k n^2)$ space, if the input polygons have n vertices in total.*

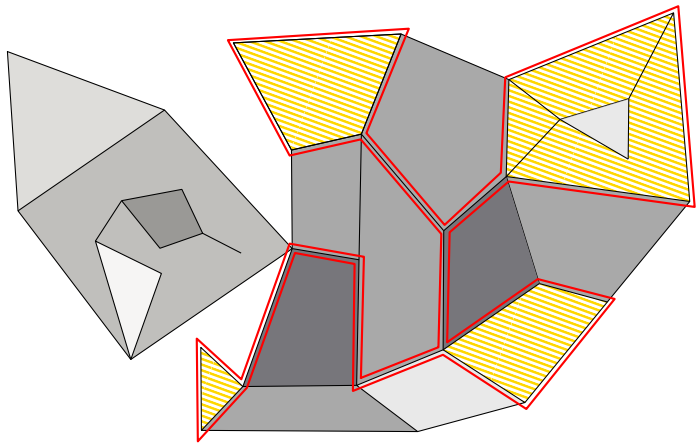


Figure 2 The Graph-Enclosure-with-Penalties problem. The colors have the same meaning as in Figure 1. A weakly simple closed walk W which is a solution for the instance is red (bold). The edge weights, which are not shown, are unrelated to the Euclidean lengths.

If all objects are points, this can be handled by approximating each point by a small triangle. An exact solution for an arbitrary mix of point and polygon objects appears in [5, Appendix I].

Graph-Enclosure-with-Penalties. In this setting, the objects are faces of a plane graph.

Input. The input is a simple connected plane graph G and positive edge weights. The bounded faces of G are partitioned into a set R of k **required faces** and the remaining set O of **optional faces**. Each optional face F has a penalty π_F from $\mathbb{R}_{\geq 0} \cup \{+\infty\}$.

Output. The goal is to find a weakly simple closed walk W in G such that faces of R are inside W while minimizing the cost $c(W)$ which is defined to be the sum of the weights of the edges of W plus the penalties of the faces of O that are inside W. See Figure 2 for an example. Intuitively, a **weakly simple** closed walk is one without crossings; we give a more precise definition in the full version of this paper [5, Appendix A]. For a weakly simple closed walk, the notions of inside and outside are well-defined. Our second main result is:

▶ **Theorem 2.** Graph-Enclosure-with-Penalties *can be solved in* $O(3^k n^3)$ *time and* $O(2^k n^2)$ *space, where* k *is the number of required faces and* n *is the number of vertices of* G.

A common framework. Although the two settings described above seem different, we resolve them into a common geometric framework, which we call Enclosure-with-Penalties. Our algorithm applies to this general problem. The basic idea is to transform the graph problem into a geometric problem by taking a straight-line embedding of the graph. The bounded faces of the graph become polygons slightly more general than simple polygons. We also consider the outer face as an unbounded polygon. Then the "free space" between the polygons consists only of the graph edges. This gives us a geometric problem, albeit with arbitrary positive edge weights defined on edges that have a polygon on each side. In Section 3 we define the Enclosure-with-Penalties problem by generalizing the Geometric-Enclosure-with-Penalties problem to include these instances.

We remark that the resulting algorithm for Theorem 2 makes essential use of the straight-line embedding of the input graph. In particular, the subproblems that we solve depend on the embedding. This imposition of geometry seems artificial, but oddly enough, we do not know how to formulate our algorithm in a purely combinatorial setting.

Our approach. We use dynamic programming (Section 4) to build a polygon W that is locally correct – we use segments that do not intersect the interior of any object and we account for required objects and tally the penalties as we add triangles to W. We will prove that the cost computed by the algorithm is correct, but this is tricky because W itself will not necessarily be weakly simple. "Inside" is no longer well-defined. Instead, we use winding numbers to give a measure of the cost of W that matches the cost computed by the algorithm.

In Section 5 we give an algorithm to *uncross* W to a weakly simple polygon without increasing its cost, which provides our final output. Correctness is proved in Section 6.

The run-time for the Uncrossing Algorithm is dominated by the run-time for the dynamic program. To obtain our claimed run-time we speed up the dynamic program in Section 7.

Lower bounds. To complement our algorithms we prove that, under the Exponential Time Hypothesis (ETH), the Geometric- and Graph-Enclosure-with-Penalties problems cannot be solved in $2^{o(k)} \cdot n^{O(1)}$ time, implying that the linear dependence on k in the exponent of the running time of our algorithms is the best possible assuming ETH. The proof is a reduction from unweighted Planar Steiner Tree, which admits a lower bound by a result by Marx, Pilipczuk, and Pilipczuk [15, Theorem 1.2]. See [5, Appendix K].

Swapping the inside with the outside. We extend our algorithm to an **inverted** version of the Enclosure-with-Penalties problem where the required objects have to be *outside* W, and the objective is to minimize the length of W plus the penalties of the polygons of O that are *outside* W. The runtime remains the same, see Section 8. This algorithm provides a new faster solution to the geometric knapsack problem discussed below.

Negative penalties. We can allow some number ℓ of objects with negative penalties (rewards); in this case, the runtime is increased by a factor of 3^ℓ. See [5, Appendix J].

1.1 Related work

Cut problems and separator problems in graphs have a long history, and separation problems in geometric settings are a natural and well studied counterpart.

Geometric knapsack problem. Geometric separation problems were first explored by Eades and Rappaport [11] (as discussed above) and by Arkin, Khuller, and Mitchell, who introduced the *geometric knapsack problem* [4], which corresponds to the *inverted* version of the Geometric-Enclosure-with-Penalties problem in the special case where there are no required objects. (In their equivalent formulation, each object has a finite nonnegative value, and the goal is to compute a curve that maximizes the total value of the enclosed objects minus its length.) They gave an algorithm with running time $O(n^4)$ [4, Theorem 6]. In the full version [5, Appendix M], we note some issues regarding weak simplicity and interior/exterior of their output. Since there are no required objects, our algorithm for the inverted problem solves the geometric knapsack problem in time $O(n^3)$.

Relation to homotopy and homology. Our problem has a topological flavor and is therefore, in principle, amenable to homotopy and homology techniques. However, these techniques are unlikely to lead to algorithms that are FPT in k, even assuming only infinite penalties. In particular, for the Graph-Enclosure-with-Penalties problem, enumerating a set of candidate homotopy classes, i.e., the possible ways how a solution winds around the objects to

enclose the required objects and avoid the most undesirable ones, is possible using a technique by Chambers, Colin de Verdière, Erickson, Lazarus, and Whittlesey [7] (specifically, because the exchange argument [7, Proposition 4.2] is also valid for our problem), but this results in an algorithm with runtime $K^{O(K)} \cdot n \log n$, where K is the number k of required objects *plus* the number of objects with nonzero penalty. Similarly, the technique of *homology covers*, by Chambers, Erickson, Fox, and Nayyeri [8], is applicable, but yields algorithms with runtime $K^{O(K)} \cdot n \log \log n$ or $2^{O(K)} \cdot n \log n$, thus again with an exponential dependence on K. If there are many objects with nonzero penalty, our algorithm with runtime $O(3^k n^3)$ is faster.

Specifying only the number of objects to be enclosed. If we are just given a set of n points in general position and the exact *number* $k \leq n$ of points to be enclosed, a minimum-perimeter polygon enclosing at least k points is convex, contains exactly k points, and can be found in polynomial time by an algorithm of Eppstein, Overmars, Rote, and Woeginger [12, Corollary 5.3, Case 3]. This algorithm could for example be used to identify an unusual cluster in an otherwise uniformly distributed point set. However, if the input consists of polygons instead of points, we are not aware of a better method than guessing the k polygons to be enclosed and applying our main result, resulting in an algorithm of running time $O(\binom{N}{k} 3^k n^3) = O(N^k n^3)$ if there are N objects.

More variations. Some related separation and enclosure problems have been recently studied. Abrahamsen, Giannopoulos, Löffler, and Rote [1] investigated the problem of separating two or more groups of polygonal objects by a shortest system of fences. These fences may form an arbitrary plane graph, not necessarily a cycle. For separating *two* groups of polygons by a shortest fence, there is an algorithm of running time $O(n^4 \log^3 n)$, based on minimum-cut techniques [1]. For separating more than two groups, only approximation algorithms are known. While we study a problem in the same spirit, a key difference is that we require a single (weakly) simple cycle, which makes the techniques of this article not applicable for us.

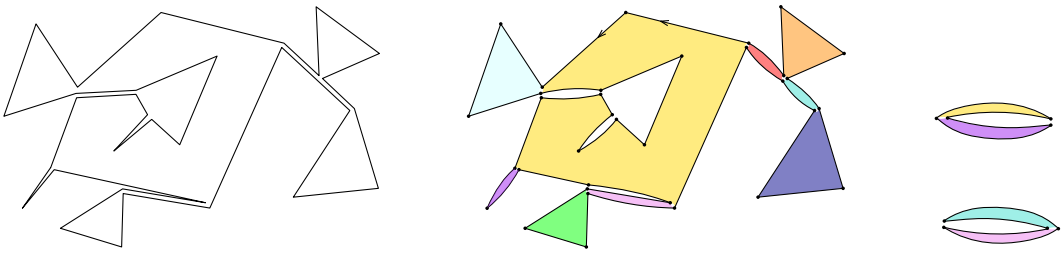
Chan, He, and Xue [9] studied the problem of choosing a smallest subset of objects from a given set whose union encloses a given point set. For this problem, there are only approximation algorithms. Klost, van Kreveld, Perz, Rote, and Tkadlec [13] have recently investigated optimum "blob-trees", where the objective is to *connect* a set of points by a combination of curves and edges.

2 Preliminaries

2.1 Weakly simple polygons

A polygon is **weakly simple** if it has fewer than three vertices, or it has at least three vertices and for any $\varepsilon > 0$, the vertices can be perturbed by at most ε to yield a simple polygon [2, 10]. We traverse a weakly simple polygon counterclockwise, i.e., with the interior to the left of each edge. Our proof uses a combinatorial characterization of a weakly simple polygon in terms of a non-crossing Euler tour in a plane multigraph [5, Lemma 16 in Appendix A]. This allows us to partition the edges of a weakly simple polygon into boundary walks of interior faces, see Figure 3. Note that we first subdivide an edge when a vertex lies in its interior.

A vertex of a weakly simple polygon with incoming edge e and outgoing edge f is a **transition vertex** if e and f belong to different interior faces. An interior face of two edges is a **corridor** and an interior face of more than two edges is a **chamber**. A chamber is not necessarily a simple polygon, but it is almost simple (see Figure 3(b)). More formally, a **bounded almost-simple polygon** is the boundary walk of an interior face of a connected straight-line



(a) (b) (c)

Figure 3 (a) A weakly simple polygon drawn via its ε-approximation. (b) The edges, after subdividing at interior vertices (forks), are partitioned into interior faces. Four faces are corridors and five are chambers. The largest chamber (in yellow) is almost-simple but not simple. (c) Non-uniqueness of the faces for a weakly simple polygon that traverses a line segment four times. In the top figure the two vertices on the left are transition vertices; this is reversed in the bottom figure.

graph drawing in the plane. We also allow an **unbounded almost-simple polygon** by traversing the boundary of the outer face clockwise. An almost-simple polygon has a connected interior and a bounded almost-simple polygon can be triangulated. Almost-simple polygons play two roles: the bounded ones arise as chambers; and our general Enclosure-with-Penalties problem allows almost-simple input polygons (including a single unbounded one).

All these concepts are made rigorous in [5, Appendix A]. We note that the partition of the edges of a weakly simple polygon into interior faces (and hence the definition of corridors and transition vertices) is not unique, see Figure 3(c). This non-uniqueness, which is inherent in the ε-approximation definition of weakly simple polygons, does not affect our proofs.

2.2 Winding number and winding parity

Our algorithm will construct intermediate polygons that are not necessarily weakly simple; so it will be useful to generalize "enclosed by" in terms of winding numbers. For a polygon W and a point x not lying on a vertex or edge of W, the **winding number** $\text{wind}(W, x)$ of x with respect to W is defined as follows. Take a ray ρ from x that avoids vertices of W. If an edge of W crosses ρ from right to left, we count this as $+1$; a crossing from left to right is counted as -1, and the total count gives the winding number. This is well-defined independent of the choice of ρ. The winding number is undefined for points x on W. Observe that, for a weakly simple polygon W traversed counterclockwise, point x lies in the interior of W if and only if $\text{wind}(W, x) = 1$. The **winding parity** of x with respect to W is $\text{wind}(W, x) \bmod 2$.

3 Our Common Framework: Enclosure-with-Penalties

In this section we formally define the Enclosure-with-Penalties problem that provides a common framework for both the geometric and graph settings.

Input:

- A set of interior-disjoint almost-simple polygons in the plane. We allow a single polygon to be unbounded. We subdivide polygon edges to ensure that no polygon vertex lies in the interior of an edge of another polygon. The **free space** is the plane minus the interiors of the polygons.
- A partition of the input polygons into a set R of k **required** polygons and the remaining set O of **optional** polygons. If there is an unbounded polygon, it must lie in O.

- For each polygon $P \in O$, a **penalty** $\pi_P \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$.
- The weight w_{ab} of a line segment ab in the free space is its Euclidean length, except for *squeezed edges*. A squeezed edge is a polygon edge that is incident to polygons on both sides. We may specify an *arbitrary* positive weight for a squeezed edge. Subsegments of a squeezed edge get proportional weight, and combinations of different squeezed or non-squeezed segments have their weights added.

Output: A weakly simple polygon W that lies in the free space and contains all polygons of R while minimizing the **cost** $c(W)$, which is defined as

$$c(W) := w(W) + \pi(W), \tag{1}$$

where $w(W)$ is the sum of the weights of the edges of W, and $\pi(W)$ is the sum of the penalties of the polygons of O that are inside W. Our main result is:

▶ **Theorem 3.** Enclosure-with-Penalties *for k required polygons can be solved in $O(3^k n^3)$ time and $O(2^k n^2)$ space, if the input polygons have n vertices in total.*

Theorem 1 is an immediate consequence of Theorem 3. Theorem 2 follows from Theorem 3 via a straight-line embedding of the graph, as outlined in Section 1 and detailed in the full version [5, Appendix B].

The Enclosure-with-Penalties problem as defined above does not allow point objects. (They are not almost-simple.) [5, Appendix I] shows how to deal with point objects.

4 Dynamic Programming Algorithm

The algorithm builds a polygon composed of free-space edges, where a **free-space edge** is an inclusionwise minimal line segment in the free space whose endpoints are vertices of the input polygons. We prove in Section 6 that this restriction to free-space edges is valid. We refer to a solution interchangeably as a polygon or as a closed walk in the graph of free-space edges.

The intuition for the algorithm is based on the decomposition of a weakly simple polygon W into corridors and chambers joined at "cutpoints", see Figure 3. A cutpoint separates W into subpolygons and partitions the set of enclosed objects. Our first type of subproblem finds polygons that enclose a specified subset of R and go through a specified vertex.

A corridor is a digon, and a chamber can be triangulated by adding chords, where a chord may cut through polygons. We therefore use digons and triangles as the basic building blocks to construct our solutions. A chord cuts off part of the solution. Our second type of subproblem finds polygons that use a walk of free-space edges between two given vertices p and q together with the chord pq (called the *mouth*) to enclose a specified subset of R.

Since a mouth may cut through polygons, we choose a **reference point** r_P in the interior of every input polygon P, and aim to enclose r_P for $P \in R$. Observe that a weakly simple polygon W in the free space encloses P if and only if r_P lies in the interior of W.

The dynamic program explicitly keeps track of the subset of required objects that are enclosed by partial solutions (2^k possibilities). However, when combining two partial solutions, the algorithm does not have enough information to check whether they cross. Thus, we allow self-crossing solutions. In particular, our use of the word "enclosing" is aspirational, and will only be made precise in terms of winding numbers, see Section 4.2. When we state the algorithm, we invite the reader to think of a weakly simple solution without crossings.

Types of subproblems. A subproblem of type C ("closed") is rooted at a vertex p, and we build a closed walk that goes through p and is composed of free-space edges. A subproblem of type M ("mouth") is rooted at a segment pq between vertices of input polygons, called the **mouth**, and we build an open walk of free-space edges from p to q; adding segment qp closes the walk. In addition to the root, each subproblem has two more parameters, B and t: The set $B \subseteq R$ specifies the precise subset of required objects that must be enclosed, and the integer $t \geq 0$ is an upper bound on the number of edges of the walk. In each case, the subproblem looks for a *minimum-cost* solution with the required properties.

4.1 Dynamic programming recursion

We now give recursive formulas for C and M, preceded in each case by an explanation of the formulas. The formulas with the respective partitions of the walk are illustrated in Figure 4.

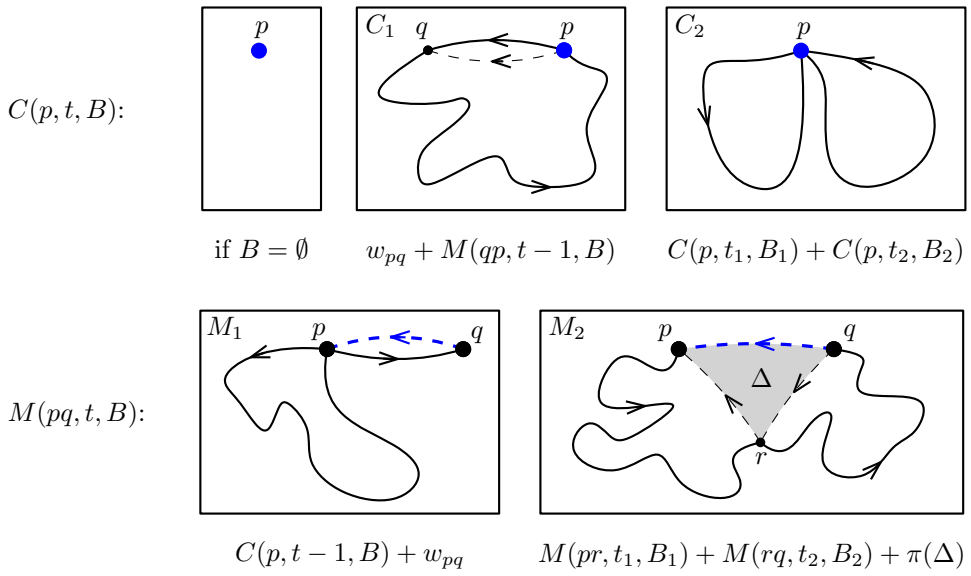


Figure 4 Cases of the recursion. Solid edges are free-space edges; dashed edges are mouths.

For $C(p,t,B)$ we have two base cases: if $B = \emptyset$, then the shortest closed walk is just the point p and its cost is 0 (Equation (2)); and if $B \neq \emptyset$ and $t \leq 1$, there is no solution, and we set the cost to ∞ (Equation (3)). Otherwise we have two general cases: the closed walk uses an edge pq of the free space (for some q) plus a solution $M(qp, t-1, B)$ (Equation (4)); or the closed walk is composed of two smaller closed walks that both go through p (Equation (5)). The notation $\sqcup$ means disjoint union: we partition the objects in B into two sets, each "enclosed" by exactly one of the two closed walks.

The base cases define $C(p,t,B)$ for $B = \emptyset$ and for $t \leq 1$:

$$C(p,t,\emptyset) := 0, \text{ for } t \geq 0 \tag{2}$$

$$C(p,t,B) := \infty, \text{ for } B \neq \emptyset \text{ and } t \leq 1 \tag{3}$$

In the general case, for $B \neq \emptyset$ and $t \geq 2$, we set

$$C(p,t,B) := \min\{C_1, C_2\}, \text{ where}$$

$$C_1 := \min\big\{ w_{pq} + M(qp, t-1, B) \;\big|\; pq \text{ is a free-space edge} \big\} \tag{4}$$

$$C_2 := \min\big\{ C(p,t_1,B_1) + C(p,t_2,B_2) \;\big|\; t = t_1 + t_2;\ B = B_1 \sqcup B_2;\ B_1, B_2 \neq \emptyset \big\} \tag{5}$$

For $M(pq, t, B)$ there are two possibilities: if pq is a free-space edge, we can use a closed walk at p plus the edge pq (Equation (6)); or we can attach a triangle $\Delta = prq$ to the mouth pq (Equation (7)). In the first case we add the weight of the edge pq. In the triangle case we take into account the polygons $R(\Delta)$ with reference points in Δ, where we consider Δ to be closed on pr and rq and open on pq, and we define $\pi(\Delta)$ to be the sum of the penalties of polygons of O with reference points in Δ.

$M(pq, t, B)$ is defined only for $t \geq 1$:

$$M(pq, t, B) := \min\{M_1, M_2\}, \text{ where}$$

$$M_1 := \begin{cases} C(p, t-1, B) + w_{pq} & \text{if } pq \text{ is a free-space edge} \\ \infty, & \text{otherwise} \end{cases} \tag{6}$$

$$M_2 := \min\Big\{ M(pr, t_1, B_1) + M(rq, t_2, B_2) + \pi(\Delta) \;\Big|\; \Delta = prq \text{ is a counterclockwise triangle}, \; t = t_1 + t_2,\ t_1 \geq 1, t_2 \geq 1,\ B = B_1 \sqcup B_2 \sqcup R(\Delta) \Big\} \tag{7}$$

As we shall see later in Lemma 13, the optimal walk has at most $6n$ edges. Thus, we define the solution to the whole problem as

$$c_{\mathrm{DP}} := \min\{\, C(p, 6n, R) \mid p \text{ a vertex} \,\}. \tag{8}$$

When we allow point objects, the algorithm needs a few refinements, see [5, Appendix I]. In the following sections we prove that c_{DP} is the correct value. Although not required by our proof, we note for completeness in [5, Appendix L] that the class of polygons over which the algorithm optimizes is the class of *immersed* or *self-overlapping* weakly simple polygons.

Runtime and Space. A routine analysis shows that the runtime of the dynamic program is $O(3^k n^5)$, see the full version [5, Appendix C.1]. A more efficient version of the dynamic program, given in Section 7, eliminates the parameter t and runs in time $O(3^k n^3)$.

4.2 Extracting the solution

With every finite value computed in the dynamic program we can naturally associate an open or closed walk of free-space edges that has this value as its cost (with "partially enclosed" objects taken into account according to their reference points): Figure 4 illustrates how each value is computed from the values of subproblems; the walks corresponding to those subproblems are composed accordingly. (For more details, see [5, Appendix C.2].) We will prove in Section 6 that c_{DP} is finite; so the associated closed walk exists:

▶ **Definition 4.** *W_{DP} is the polygon associated with the optimum solution value c_{DP} in (8).*

The polygon W_{DP} uses free-space edges, but it might not be weakly simple, and there is no notion of enclosed objects. Instead, we use winding numbers: we show that the reference point of any object in R has winding number 1 in W_{DP}, and we define a cost measure for W_{DP} in terms of winding numbers and prove equality with c_{DP}.

▶ **Definition 5.** *For any polygon W in the free space, define the **cost** to be*

$$c(W) := w(W) + \sum_{P \in O} \mathrm{wind}(W, r_P) \cdot \pi_P.$$

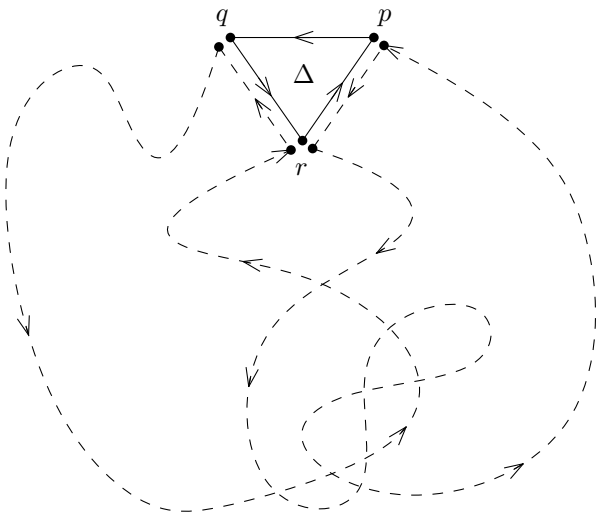


Figure 5 Gluing together closed walks, which may cross each other and are possibly self-crossing.

When W is a counterclockwise weakly simple polygon, this matches the previous definition $c(W) = w(W) + \pi(W)$, see Equation (1). The solution W_{DP} produced by the algorithm has the following properties:

▶ **Lemma 6.**
(A) $c_{\mathrm{DP}} = c(W_{\mathrm{DP}})$;
(B) *for all* $P \in R$, $\mathrm{wind}(W_{\mathrm{DP}}, r_P) = 1$;
(C) *for all points* x *that do not lie on* W_{DP}, $\mathrm{wind}(W_{\mathrm{DP}}, x) \geq 0$.

Lemma 6 is proved in [5, Appendix C.2] by induction as the dynamic program builds solutions by gluing together open/closed walks. The induction must apply also to open walks, and we use the fact that winding numbers add when gluing walks together, see Figure 5.

5 Uncrossing Algorithm and Final Output W_{ALG}

The final step of our algorithm "uncrosses" the closed walk W_{DP} produced by the dynamic program and turns it into a weakly simple polygon W_{ALG} without increasing the cost. To do so, we cut it into subpaths, eliminate some, and reorder the rest.

Our algorithm uses a known result about taking a plane multigraph (specified via its rotation system) and finding a **non-crossing Euler tour** in which successive visits to a vertex do not cross each other. (See [5, Appendix A] for more detailed definitions.) The existence of such a tour in an Eulerian plane multigraph is an easy exercise, see [17] or [18, Lemma 3.1], and a linear-time algorithm was given by Akitaya and Tóth [3]. We summarize it in the following proposition, and give a self-contained proof in [5, Appendix D].

▶ **Proposition 7** (Uncrossing Eulerian plane multigraphs)**.** *Given a plane connected Eulerian multigraph* H *with* m *edges, specified by its combinatorial map, we can, in* $O(m)$ *time, compute a non-crossing Euler tour of* H*.*

We now sketch our algorithm to uncross any polygon W to a weakly simple polygon W'. An **interior crossing** is a point that is in the relative interiors of two (or more) edges that are not collinear. A **fork** (or *interior vertex*) is a vertex in the relative interior of an edge.

▶ **Algorithm 8** (Uncrossing Algorithm).

1. *Subdivide every edge of W at every fork and interior crossing.*
2. *In the resulting multiset of edges (line segments in the plane) reduce multiplicities to* 1 *or* 2 *by repeatedly discarding pairs of equal line segments. The result is a plane connected Eulerian multigraph.*
3. *Apply Proposition 7 to find a non-crossing Euler tour. This corresponds to a weakly simple polygon W'.*

In the full version of our paper [5, Appendix D], we give further details of the algorithm and an implementation with runtime $O(t \log t + s)$ where t is the number of edges of W and $s \in O(t^2)$ is the number of interior crossings of W (with multiple crossings counted once). For input W_{DP} we show that there are no interior crossings, so the runtime is $O(n \log n)$.

We use the following important property of the Uncrossing Algorithm. Recall the definition of winding parity from Section 2.2.

▶ **Lemma 9.** *Every point x in the plane that does not lie on W has the same winding parity in W and in W'.*

Proof. For any ray r from x to infinity that avoids vertices of W, the parity of the number of edges it crosses is the same for W and W' since we have discarded pairs of equal line segments. The direction in which W' traverses an edge does not affect the winding parity. ($+1$ and -1 have the same parity.) ◀

▶ **Definition 10.** *W_{ALG} is the output of the Uncrossing Algorithm on input W_{DP}, oriented in counterclockwise direction.*

▶ **Lemma 11.** *W_{ALG} is a weakly simple polygon in the free space. W_{ALG} encloses R and $c(W_{\mathrm{ALG}}) \le c_{\mathrm{DP}}$.*

Proof. Consider a polygon $P \in R$. By Lemma 6(B), $\mathrm{wind}(W_{\mathrm{DP}}, r_P) = 1$. By Lemma 9, r_P has the same winding parity in W_{ALG}. Since W_{ALG} is weakly simple, every point has winding number 0 or 1. Thus $\mathrm{wind}(W_{\mathrm{ALG}}, r_P) = 1$ and W_{ALG} encloses P.

Next we consider costs. The definition of the costs in Equation (1) gives

$$c(W_{\mathrm{ALG}}) = w(W_{\mathrm{ALG}}) + \pi(W_{\mathrm{ALG}}),$$

where $\pi(W_{\mathrm{ALG}})$ is the sum of the penalties of objects of O enclosed by W_{ALG}.

By Lemma 6(A) and the definition of $c(W_{\mathrm{DP}})$,

$$c_{\mathrm{DP}} = c(W_{\mathrm{DP}}) = w(W_{\mathrm{DP}}) + \sum_{P \in O} \mathrm{wind}(W_{\mathrm{DP}}, r_P) \cdot \pi_P.$$

The Uncrossing Algorithm ensures that $w(W_{\mathrm{ALG}}) \le w(W_{\mathrm{DP}})$. It remains to compare the penalties. Let P be a polygon of O enclosed by W_{ALG}, i.e., with $\mathrm{wind}(W_{\mathrm{ALG}}, r_P) = 1$. By Lemma 9, the reference point r_P has the same winding parity in W_{DP}, and by Lemma 6(C), $\mathrm{wind}(W_{\mathrm{DP}}, r_P) \ge 0$. Thus $1 \le \mathrm{wind}(W_{\mathrm{DP}}, r_P)$ and

$$\pi(W_{\mathrm{ALG}}) = \sum_{P \in O} \mathrm{wind}(W_{\mathrm{ALG}}, r_P) \cdot \pi_P \le \sum_{P \in O} \mathrm{wind}(W_{\mathrm{DP}}, r_P) \cdot \pi_P$$

Therefore $c(W_{\mathrm{ALG}}) \le c_{\mathrm{DP}}$. (In fact, equality holds, as shown in the next section.) ◀

6 Correctness Proof

In defining W_{ALG}, we relied on the assumption that c_{DP} is finite. In this section we prove this fact, which implies that W_{ALG} exists, and we prove our main correctness result:

▶ **Theorem 12.** *W_{ALG} is an optimum solution to the* Enclosure-with-Penalties *problem.*

We defined the Enclosure-with-Penalties problem over the continuous space of all weakly simple polygons, but our algorithm only explores the discrete space of weakly simple polygons composed of at most $6n$ free-space edges. So we first prove that there is an optimum solution in this discrete space. A *feasible* solution is a weakly simple polygon that lies in the free space and encloses R.

▶ **Lemma 13.** *For the* Enclosure-with-Penalties *problem, there exists an optimum solution W_{OPT} of finite cost that consists of at most $6n$ free-space edges.*

Proof idea (Details in [5, Appendix E]). Let $\mathcal{S}$ be the discrete set of feasible solutions that consist of free-space edges each traversed at most twice. Because a planar graph on n vertices has at most $3n$ edges, any solution in $\mathcal{S}$ has at most $6n$ free-space edges.

We next prove that $\mathcal{S}$ contains a feasible solution that encloses R and excludes O, and thus has finite cost. The idea is to take the cycle boundaries of polygons in R and join them by paths traversed twice.

Since $\mathcal{S}$ is finite and nonempty, this implies that, among the solutions in $\mathcal{S}$, there is a solution W^* of minimum cost.

Finally, we prove that any feasible solution not in $\mathcal{S}$ can be homotopically shortened and then uncrossed to get a solution in $\mathcal{S}$ of no greater cost. Thus W^* is an optimum solution. ◀

We prove that the solution W_{OPT} from Lemma 13 is one of the candidate solutions over which the dynamic program optimizes. As a consequence:

▶ **Lemma 14.** $c_{\mathrm{DP}} \leq c(W_{\mathrm{OPT}})$.

Theorem 12 then follows: Lemmas 13 and 14 establish that c_{DP} is finite. Thus W_{ALG} exists. By Lemma 11, W_{ALG} is a feasible solution and $c(W_{\mathrm{ALG}}) \leq c_{\mathrm{DP}}$. Combining with Lemma 14 yields $c(W_{\mathrm{OPT}}) \leq c(W_{\mathrm{ALG}}) \leq c_{\mathrm{DP}} \leq c(W_{\mathrm{OPT}})$. Thus W_{ALG} is optimal.

We say a few words about the proof of Lemma 14. By the definition of c_{DP}, it suffices to show that $C(p, 6n, R) \leq c(W_{\mathrm{OPT}})$ for a vertex p on W_{OPT}. We give an inductive proof of the more general statement that $C(p, t, B)$ is at most the cost of any weakly simple polygon W with at most t free-space edges that encloses B and goes through p. Since W_{OPT} has at most $6n$ edges, this implies Lemma 14. The following lemma, which is proved in [5, Appendix E], includes an analogous inductive statement for $M(pq, t, B)$, with a suitable definition of the cost $c(W_0)$ of an open walk W_0. It refers to transition vertices, which were defined in Section 2.1.

▶ **Lemma 15.** (A) *Let W be a weakly simple polygon with ℓ free-space edges, going through vertex p, and let B be the objects of R that are enclosed by W. Then, for all $t \geq \ell$, $C(p, t, B) \leq c(W)$.*

(B) *Let W_0 be an open walk with ℓ free-space edges from p to q such that the polygon $W = W_0 + qp$ is weakly simple and q is not a transition vertex of W. Let B be the objects of R whose reference points lie inside W and not on pq. Then, for all $t \geq \ell$, $M(pq, t, B) \leq c(W_0)$.*

7 Reducing the Runtime with a Dijkstra-Style Algorithm

The runtime of our algorithm to solve the Enclosure-with-Penalties problem can be reduced by a $\Theta(n^2)$ factor, leading to the bound of Theorem 3.

The dynamic programming algorithm in Section 4 is guided by a parameter t, which limits the number of edges of the walk. We have proved (Lemma 13) that there is an optimal solution with at most $6n$ edges. Hence, the solution cannot be improved by allowing larger values of t; the iteration stabilizes, and the algorithm can stop when t reaches $6n$. The parameter t has been useful for ensuring that the quantities in the dynamic programming algorithm are well-defined, and it is essential as an induction variable for the proofs. We will now eliminate t and solve the recursion in the style of Dijkstra's algorithm for shortest paths. In particular, we use a generalization of Dijkstra's algorithm as proposed by Knuth [14].

More specifically, we define $C(p,B) := C(p,6n,B)$ and $M(pq,B) := M(pq,6n,B)$ in terms of the quantities from Section 4. By the above observations, $C(p,B) = C(p,t,B)$ and $M(pq,B) = M(pq,t,B)$ for all $t \geq 6n$. Therefore, the limit quantities $C(p,B)$ and $M(pq,B)$ fulfill a variation of the recursions (2–7) where the parameter t is eliminated:

$$C(p,\emptyset) = 0 \tag{9}$$

$$C(p,B) = \min\{C_1(p,B), C_2(p,B)\} \text{ for } B \neq \emptyset, \text{ where} \tag{10}$$

$$C_1(p,B) = \min\big\{\, w_{pq} + M(qp,B) \;\big|\; pq \text{ is a free-space edge} \,\big\} \tag{11}$$

$$C_2(p,B) = \min\big\{\, C(p,B') + C(p,B'') \;\big|\; B = B' \sqcup B'';\ B', B'' \neq \emptyset \,\big\} \tag{12}$$

$$M(pq,B) = \min\{M_1(pq,B), M_2(pq,B)\}, \text{ where} \tag{13}$$

$$M_1(pq,B) = \begin{cases} w_{pq} + C(q,B), & \text{if } pq \text{ is a free-space edge} \\ \infty, & \text{otherwise} \end{cases} \tag{14}$$

$$\begin{aligned} M_2(pq,B) = \min\big\{\, & M(pr,B') + M(rq,B'') + \pi(\Delta) \;\big|\; \\ & prq = \Delta \text{ is a counterclockwise triangle;} \\ & B = B' \sqcup B'' \sqcup \{P \in R \mid r_P \in \Delta\} \,\big\} \end{aligned} \tag{15}$$

As in Equation (8), we define the solution to the whole problem as

$$c_{\mathrm{DP}} := \min\{\, C(p,R) \mid p \text{ is a vertex} \,\}. \tag{16}$$

By Lemma 13 and by the definition of $C(p,B)$ and $M(pq,B)$, the value of c_{DP} resulting from (16) is the same as the one resulting from (8).

The system (9–15) is a *system of equations* that involves cyclic dependencies. Nevertheless, we can show that it has a unique solution [5, Appendix F.1, Lemma 22]. The reason is that on the right-hand side of the equations, the result of any expression combining some quantities of the form $C(p,B)$ and $M(pq,B)$ is always *larger* than these quantities.

Similar to Dijkstra's shortest-path algorithm, our algorithm maintains tentative values $C(p,B)$ and $M(pq,B)$. The smallest of the tentative values is made permanent, and all right-hand side expressions where this value appears are evaluated and used to update the corresponding tentative left-hand side values. The algorithm that carries out this idea is shown in the full version [5, Appendix F.2, Algorithm 1].

The most numerous quantities are the $O(2^k n^2)$ values $M(pq,B)$, and hence the space complexity is $O(2^k n^2)$. Compared to the running time for the dynamic programming algorithm in Section 4, we save a factor n^2: The elimination of t reduces the number of recursions by a factor $\Theta(n)$, and we save another factor $\Theta(n)$ because we need not go through all decompositions $t = t_1 + t_2$ on the right-hand side. The analysis of the full algorithm is given in [5, Appendix F.3]. The total running time is $O(3^k n^3)$, as claimed in Theorem 3.

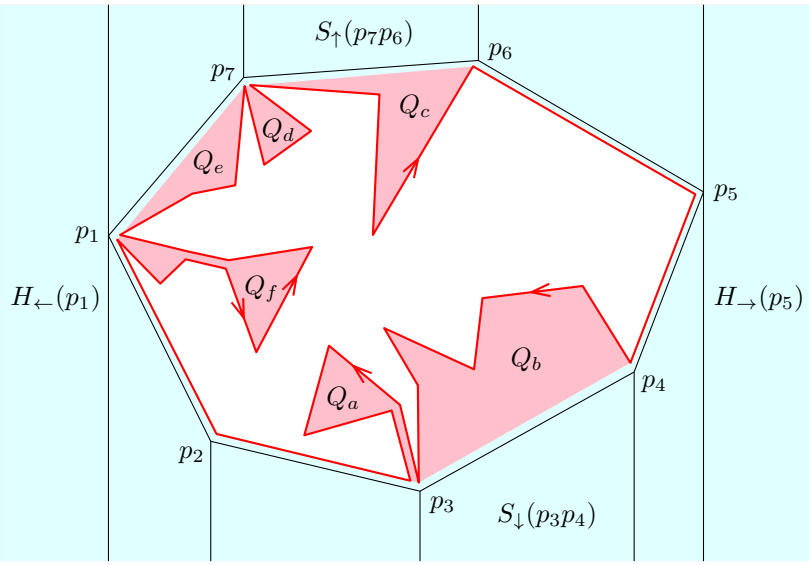


Figure 6 Partition of the outside into pockets $Q_a, \ldots, Q_f$, two half-planes, and seven planks.

8 The Inverted Problem

In the inverted problem, the required objects have to be outside the weakly simple polygon W, and the penalties of the polygons that are outside W are counted (see the introduction). The approach for the original problem has to be adapted accordingly. To derive a suitable dynamic programming formulation, we decompose the *outside* of W into elementary pieces, as shown in Figure 6: We form the convex hull of W and extend vertical rays upward and downward from the convex hull vertices. This leads to two additional types of regions:

- a left and a right half-plane, each bounded by a vertical line through an object vertex;
- vertical *planks*, that is, regions bounded by a line segment and two vertical upward rays or two vertical downward rays. We discuss such regions in [5, Appendix F.4].

We stick to the convention that the interior of the region of interest lies on the left side of W. Accordingly, the solution polygon W is now oriented clockwise.

In the algorithm, we build the region of interest outside-in, see Figure 7, starting from a left half-plane bounded by two vertical rays through an object vertex. We add planks from left to right along common rays, and, as in Section 4, we may also attach triangles along common edges and digons along common vertices. In addition to the usual bounded walks, we now also consider polygonal walks $W^{\updownarrow}$ that start from the endpoint of a vertical downward ray and end at a vertical upward ray. More precisely, for each pair of vertices p, q, we consider a subproblem of type U ("unbounded"), which considers regions bounded by a vertical ray down from p, a walk W from p to q, and a vertical ray up from q, see Figure 7 for an example; $p = q$ is allowed. Accordingly, the algorithm computes quantities $U(p, q, t, B)$ for all $B \subseteq R$ and $t \leq 6n$. The two unbounded rays jointly play the role of the mouth.

The full paper [5, Appendix G] describes how the dynamic programming recursion and the correctness proof of Section 6 for the non-inverted problem is adapted for the inverted problem.

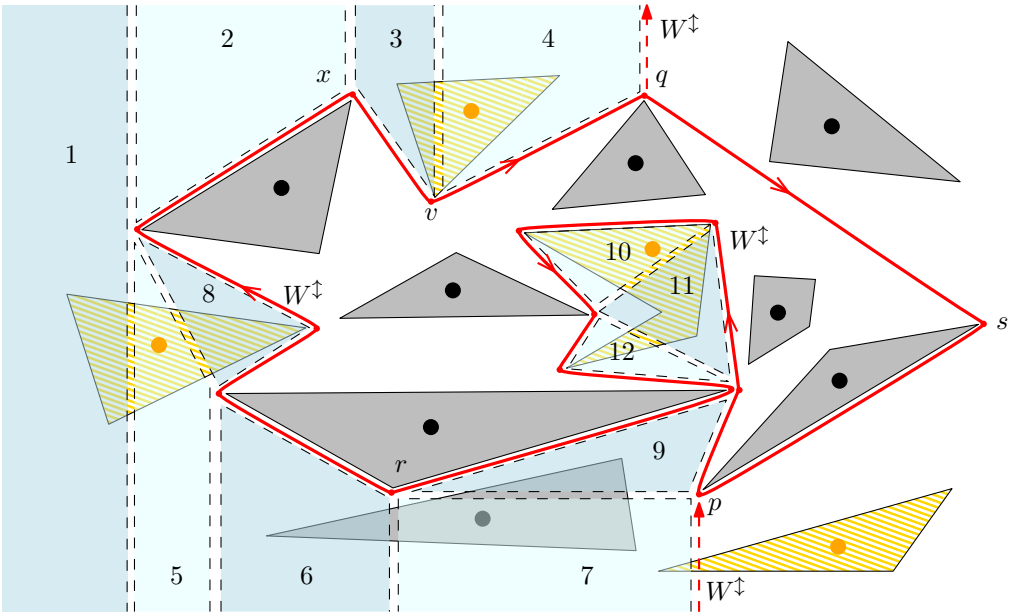


Figure 7 A weakly simple polygon W (red/bold) that has the four required objects (yellow/hatched) on the outside. The penalties of two optional (grey) objects is added to the length when the cost is computed. We grow the outer region triangle by triangle, considering also "triangles" that extend to $-\infty$ or $+\infty$ (vertical planks), or both, like the left half-plane (number 1). The union of the shaded blue regions is bounded by vertical rays from p downward and from q upward plus a weakly simple walk between p and q. The extended walk $W^{\updownarrow}$ is a candidate solution considered for the subproblem $U(p, q, B, t)$, when $t \geq 13$ and B consists of the three leftmost required objects. (The fourth required object has its reference point (thick dot) outside the region.) Two more planks, spanned by ps and qs, plus the right half-plane through s, would complete the outside region of W.

9 Applications, Extensions, and Open Problems

Splitting a surface. Our result may shed some light on the following open problem by Bulavka, Colin de Verdière, and Fuladi [6, Conclusion]: given an orientable combinatorial surface of genus g, and an integer g', $1 \leq g' < g$, is it FPT in g' to compute a shortest weakly simple closed curve that cuts off a surface of genus g'? The problem is FPT in g [7, Theorem 6.1]. Our algorithm for Graph-Enclosure-with-Penalties shows that the answer is yes when restricting to some (admittedly very special) instances, see [5, Appendix H], and thus provides some hope for a positive answer in general, although this remains open.

Curved objects and line segments. We believe that our approach carries over to more general objects. Curved objects that are sufficiently well behaved can be treated by considering all bitangents as free-space edges. We can handle point objects [5, Appendix I]; line segments without other vertices in their interior should also be doable. However, extending to weakly simple polygon objects seems difficult. Even for a path object consisting of two line segments joined at point p it is a challenge to prevent a solution from cutting through the path at p.

Recognizing weakly simple self-overlapping polygons. As mentioned, our dynamic program optimizes over the class of weakly simple self-overlapping polygons, see [5, Appendix L]. Weakly simple polygons can be recognized in $O(n \log n)$ time [2], and self-overlapping polygons in time $O(n^3)$ [16]. Can weakly simple self-overlapping polygons be recognized efficiently?

References

1 Mikkel Abrahamsen, Panos Giannopoulos, Maarten Löffler, and Günter Rote. Geometric multicut: shortest fences for separating groups of objects in the plane. *Discrete Comput. Geom.*, 64:575–607, 2020. doi:10.1007/s00454-020-00232-w.

2 Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. Recognizing weakly simple polygons. *Discrete & Computational Geometry*, 58(4):785–821, 2017. doi:10.1007/S00454-017-9918-3.

3 Hugo A. Akitaya and Csaba D. Tóth. Reconstruction of weakly simple polygons from their edges. *International Journal of Computational Geometry & Applications*, 28(02):161–180, 2018. doi:10.1142/S021819591860004X.

4 Esther M. Arkin, Samir Khuller, and Joseph S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993. doi:10.1007/BF01769706.

5 Therese Biedl, Éric Colin de Verdière, Fabrizio Frati, Anna Lubiw, and Günter Rote. Finding a shortest curve that separates few objects from many, April 2025. arXiv:2504.03558v1.

6 Denys Bulavka, Éric Colin de Verdière, and Niloufar Fuladi. Computing shortest closed curves on non-orientable surfaces. 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 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SoCG.2024.28.

7 Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom.*, 41(1–2):94–110, 2008. doi:10.1016/j.comgeo.2007.10.010.

8 Erin W. Chambers, Jeff Erickson, Kyle Fox, and Amir Nayyeri. Minimum cuts in surface graphs. *SIAM J. Comput.*, 52(1):156–195, 2023. doi:10.1137/19M1291820.

9 Timothy M. Chan, Qizheng He, and Jie Xue. Enclosing points with geometric objects. 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 35:1–35:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.35.

10 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1670, 2015. doi:10.1137/1.9781611973730.110.

11 Peter Eades and David Rappaport. The complexity of computing minimum separating polygons. *Patt. Recog. Lett.*, 14(9):715–718, 1993. doi:10.1016/0167-8655(93)90140-9.

12 David Eppstein, Mark Overmars, Günter Rote, and Gerhard Woeginger. Finding minimum area k-gons. *Discrete Comp. Geom.*, 7:45–58, 1992. doi:10.1007/BF02187823.

13 Katharina Klost, Marc van Kreveld, Daniel Perz, Günter Rote, and Josef Tkadlec. Minimum spanning blob-trees. In Jan Kratochvíl and Giuseppe Liotta, editors, *Abstracts of the 41st European Workshop on Computational Geometry (EuroCG 2025), April 9–11, Liblice, Czech Republic*, pages 23:1–23:8, 2025. doi:10.48550/arXiv.2503.02439.

14 Donald E. Knuth. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, 1977. doi:10.1016/0020-0190(77)90002-3.

15 Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. On subexponential parameterized algorithms for Steiner tree and directed subset TSP on planar graphs. In *Proc. 59th Ann. IEEE Symp. Foundat. Comput. Sci. (FOCS)*, pages 474–484, 2018. doi:10.1109/FOCS.2018.00052.

16 Peter W. Shor and Christopher J. Van Wyk. Detecting and decomposing self-overlapping curves. *Comput. Geom.: Theory and Applications*, 2(1):31–50, 1992. doi:10.1016/0925-7721(92)90019-O.

17 David Singmaster and Jerrold W. Grossman. Solution to problem E2897: An Eulerian circuit with no crossings. *The American Mathematical Monthly*, 90(4):287–288, 1983. doi:10.2307/2975767.

18 Mu-Tsun Tsai and Douglas B. West. A new proof of 3-colorability of Eulerian triangulations. *Ars Mathematica Contemporanea*, 4:73–77, 2011. doi:10.26493/1855-3974.193.8e7.

Polychromatic Coloring of Tuples in Hypergraphs

Ahmad Biniaz
School of Computer Science, University of Windsor, Canada

Jean-Lou De Carufel
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

Anil Maheshwari
School of Computer Science, Carleton University, Ottawa, Canada

Michiel Smid
School of Computer Science, Carleton University, Ottawa, Canada

Shakhar Smorodinsky
Department of Computer Science, Ben-Gurion University of the Negev, Be'er Sheva, Israel

Miloš Stojaković
Department of Mathematics and Informatics, University of Novi Sad, Serbia

Abstract

A hypergraph H consists of a set V of vertices and a set E of hyperedges that are subsets of V. A *t-tuple* of H is a subset of t vertices of V. A t-tuple *k-coloring* of H is a mapping of its t-tuples into k colors. A coloring is called *(t,k,f)-polychromatic* if each hyperedge of E that has at least f vertices contains tuples of all the k colors. Let $f_H(t,k)$ be the minimum f such that H has a (t,k,f)-polychromatic coloring. For a family of hypergraphs $\mathcal{H}$ let $f_{\mathcal{H}}(t,k)$ be the maximum $f_H(t,k)$ over all hypergraphs H in $\mathcal{H}$. Determining $f_{\mathcal{H}}(t,k)$ has been an active research direction in recent years. This is challenging even for $t=1$. We present several new results in this direction for $t \geq 2$.

- Let $\mathcal{H}$ be the family of hypergraphs H that is obtained by taking any set P of points in $\mathbb{R}^2$, setting $V := P$ and $E := \{d \cap P \colon d \text{ is a disk in } \mathbb{R}^2\}$. We prove that $f_{\mathcal{H}}(2,k) \leq 3.7^k$, that is, the pairs of points (2-tuples) can be k-colored such that any disk containing at least 3.7^k points has pairs of all colors. We generalize this result to points and balls in higher dimensions.
- For the family $\mathcal{H}$ of hypergraphs that are defined by grid vertices and axis-parallel rectangles in the plane, we show that $f_{\mathcal{H}}(2,k) \leq \sqrt{ck \ln k}$ for some constant c. We then generalize this to higher dimensions, to other shapes, and to tuples of larger size.
- For the family $\mathcal{H}$ of shrinkable hypergraphs of VC-dimension at most d we prove that $f_{\mathcal{H}}(d{+}1,k) \leq c^k$ for some constant $c = c(d)$. Towards this bound, we obtain a result of independent interest: Every hypergraph with n vertices and with VC-dimension at most d has a $(d{+}1)$-tuple T of depth at least $\frac{n}{c}$, i.e., any hyperedge that contains T also contains $\frac{n}{c}$ other vertices.
- For the relationship between t-tuple coloring and vertex coloring in any hypergraph H we establish the inequality $\frac{1}{e} \cdot tk^{\frac{1}{t}} \leq f_H(t,k) \leq f_H(1, tk^{\frac{1}{t}})$. For the special case of $k=2$, referred to as the bichromatic coloring, we prove that $t+1 \leq f_H(t,2) \leq \max\{f_H(1,2), t+1\}$; this improves upon the previous best known upper bound.
- We study the relationship between tuple coloring and epsilon nets. In particular we show that if $f_H(1,k) = O(k)$ for a hypergraph H with n vertices, then for any $0 < \epsilon < 1$ the t-tuples of H can be partitioned into $\Omega\left(\left(\frac{\epsilon n}{t}\right)^t\right)$ ϵ-t-nets. This bound is tight when t is a constant.

2012 ACM Subject Classification Theory of computation → Computational geometry

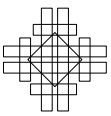
Keywords and phrases Hypergraph Coloring, Polychromatic Coloring, Geometric Hypergraphs, Cover Decomposable Hypergraphs, Epsilon Nets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.19

Related Version *Full Version*: https://arxiv.org/abs/2503.22449 [12]

Funding *Ahmad Biniaz*: Supported in part by NSERC.
Jean-Lou De Carufel: Supported in part by NSERC.

© Ahmad Biniaz, Jean-Lou De Carufel, Anil Maheshwari, Michiel Smid, Shakhar Smorodinsky, and Miloš Stojaković;
licensed under Creative Commons License CC-BY 4.0



Anil Maheshwari: Supported in part by NSERC.
Michiel Smid: Supported in part by NSERC.
Shakhar Smorodinsky: Partially supported by Grant 1065/20 from the Israel Science Foundation, by the United States – Israel Binational Science Foundation (NSF-BSF grant no. 2022792) and by ERC grant no. 882971 "GeoScape" and by the Erdős Center.
Miloš Stojaković: Partly supported by Ministry of Science, Technological Development and Innovation of Republic of Serbia (Grants 451-03-137/2025-03/200125 & 451-03-136/2025-03/200125). Partly supported by Provincial Secretariat for Higher Education and Scientific Research, Province of Vojvodina (Grant No. 142-451-2686/2021).

Acknowledgements This work was initiated at the 20th Gremo's Workshop on Open Problems, held in Wergenstein, Switzerland, in June 2023. It was then continued at the 11th Annual Workshop on Geometry and Graphs, held at the Bellairs Research Institute in Holetown, Barbados, in March 2024. We thank the organizers and participants of both workshops. We also thank the reviewers of SoCG 2025 who verified our proofs and provided valuable feedback.

1 Introduction

Polychromatic coloring and cover-decomposition of hypergraphs have rich backgrounds. They have been studied for both abstract and geometric hypergraphs as early as 1980 (see, e.g., [39, 40, 42, 43]) and are still active research areas (see, e.g., [1, 2, 14, 16, 18, 46]). Perhaps the study of these topics gained more attention after a question of Pach [39] in 1980:[1]

> *Does there exist a constant f such that every finite set of points in the plane can be colored with two colors in such a way that any unit disk that contains f points contains points of both colors?*

This question initiated the study of polychromatic coloring in geometric hypergraphs. Surprisingly enough, the answer to the question turned out to be negative, as it was proved in 2016 by Pach and Pálvölgyi [44]. In particular, they showed that for any natural number f, there is a finite set P of points in the plane such that for any 2-coloring of P, one can find a unit disk containing at least f points of one color and no point of the other color. The same question can be asked for other convex bodies and for more than two colors: Given a convex body C and any natural number $k \geq 2$, is there a function $f\colon \mathbb{N} \mapsto \mathbb{N}$ such that any finite point set in the plane can be k-colored such that any translate of C that contains at least $f(k)$ points, contains points of all the k colors? For instance, Pálvölgyi and Tóth [45] proved that $f(k)$ exists when C is a convex polygon – it was shown later [8, 26] that $f(k) = O(k)$. The negative answer of [44] to the original question of Pach implies that $f(2)$ does not exist if C is a disk. More results and details in this direction can be found in the survey article by Pach, Pálvölgyi and Tóth [41] and the webpage maintained by Keszegh and Pálvölgyi [30].

The containment of points in disks (and other convex bodies) can be represented by geometric hypergraphs, also known as range spaces. A *hypergraph* $H = (V, E)$ consists of a set V of vertices and a set E of hyperedges that are subsets of V. A (possibly infinite) set of hypergraphs is called a *family* of hypergraphs. Let $f_{\mathcal{H}}(k)$ be the smallest natural number such that the vertices of any hypergraph H in $\mathcal{H}$ can be k-colored such that any hyperedge of H that has at least $f_{\mathcal{H}}(k)$ vertices contains points of all colors. If no such number exists, then $f_{\mathcal{H}}(k) := \infty$. If $f_{\mathcal{H}}(2)$ exists, then $\mathcal{H}$ is called *cover-decomposable* – a term that Pach introduced for $k = 2$.

[1] This is a dual form of the original question that is stated in terms of f-folds and coverings; see [44].

Determining $f_{\mathcal{H}}(k)$ is an active research direction in both abstract and geometric hypergraphs; see e.g. [2, 14, 16, 17, 18, 20] and the references therein. Geometric hypergraphs, in particular, have received special attention. These hypergraphs are typically defined by points and convex bodies in some fixed constant dimension. For instance, let C be a convex body in $\mathbb{R}^2$ and let $\mathcal{H}$ be the family of all hypergraphs H that is obtained by taking any set P of points in $\mathbb{R}^2$, setting $V := P$ and $E := \{C' \cap P \colon C' \text{ is a translate of } C\}$. As mentioned above, if C is a disk then $f_{\mathcal{H}}(2)$ does not exist and hence $\mathcal{H}$ is not cover-decomposable [44], and if C is a convex polygon then $f_{\mathcal{H}}(2)$ exists and $\mathcal{H}$ is cover-decomposable [45]. Finding $f_{\mathcal{H}}(k)$ could be more challenging if we could scale or rotate the convex body. For example if C is the family of axis-parallel squares then $f_{\mathcal{H}}(2) \leq 215$ [2], and if C is the family of axis-parallel rectangles then $f_{\mathcal{H}}(2)$ does not exist [21].

A natural question thus arises: For hypergraphs that are not cover-decomposable with respect to the coloring of vertices, such as those defined by disks and axis-parallel rectangles, can we prove a positive result when we color subsets rather then the vertices? This leads to a natural generalization of the problem: color pairs, triples, or tuples of points. This generalization is referred to as *tuple coloring*. Let $H = (V, E)$ be a hypergraph. For a natural number t, a *t-tuple* of H is a subset of t vertices of V. The number of t-tuples of H is $\binom{|V|}{t}$. A t-tuple *k-coloring* of H is mapping of its t-tuples into k colors. For $t = 1$, this is equivalent to vertex coloring. A coloring is called *(t, k, f)-polychromatic* if each hyperedge of E that has at least f vertices contains tuples of all the k colors. Let $f_H(t, k)$ denote the least integer f such that H has a (t, k, f)-polychromatic coloring. For a hypergraph family $\mathcal{H}$ let $f_{\mathcal{H}}(t, k)$ denote the maximum $f_H(t, k)$ over all hypergraphs H in $\mathcal{H}$. In other words, $f_{\mathcal{H}}(t, k)$ is the minimum number for which every hypergraph in $\mathcal{H}$ has a $(t, k, f_{\mathcal{H}}(t, k))$-polychromatic coloring. For the standard vertex coloring, where $t = 1$, the value $f_{\mathcal{H}}(1, k)$ is equal to $f_{\mathcal{H}}(k)$ as defined above. With this definition, $\mathcal{H}$ is cover-decomposable if $f_{\mathcal{H}}(1, 2)$ exists. The family $\mathcal{H}$ is called *t-cover-decomposable*, for $t \geq 2$, if $f_{\mathcal{H}}(t, 2)$ exists.

1.1 Other related results

There is a rich literature for vertex-coloring (i.e. when $t = 1$) of geometric hypergraphs. In these hypergraphs, the vertices are usually defined by points and the hyperedges are defined by geometric shapes such as disks, rectangles, squares, bottomless rectangles, strips, quadrants, triangles, octants, and halfplanes; see, for example [2, 7, 8, 10, 16, 20, 21, 26, 31, 44, 45] and a recent article by Planken and Ueckerdt [46] that gives a summary of the results.

For $t \geq 2$, coloring t-tuples in hypergraphs is closely related to Ramsey-type problems (see [35] for Ramsey problems in hypergraphs). In particular, it follows from Ramsey's theorem that for any two natural numbers t and d, where $t \leq d$, the family of complete d-uniform hypergraphs is not t-cover-decomposable, i.e., $f_{\mathcal{H}}(t, 2)$ is unbounded.

A recent work of Ackerman, Keszegh, and Pálvölgyi [1] is devoted to determining $f_{\mathcal{H}}(t, k)$ for $t \geq 2$ in geometric hypergraphs that are defined by halfspaces, pseudo-disks, and boxes. In particular, for the family $\mathcal{H}$ of hypergraphs that are defined by points and axis-parallel boxes in dimension $d \geq 2$ they show that $f_{\mathcal{H}}(t, k) \leq k^{2^{d-1}} + t - 1$ when $t \geq 2$ (to better appreciate this bound we shall recall that $f_{\mathcal{H}}(t, k)$ is unbounded when $t = 1$, as shown in [21]). From this result it follows that for the family $\mathcal{H}$ of hypergraphs that are defined by points and homothets[2] of a fixed convex polytope with h facets in some dimension it holds that $f_{\mathcal{H}}(t, k) \leq k^{2^{h-1}} + t - 1$ [1].

[2] A homothet of an object is obtained by translating and scaling the object.

1.2 Preliminaries

When the family or hypergraphs or a particular hypergraph is clear from the context we may drop the subscript and simply write $f(t,k)$ for $f_{\mathcal{H}}(t,k)$ and $f_H(t,k)$. First we define the notion of shrinkability which is a common property of many geometric hypergraphs.

▶ **Definition 1.1** (Shrinkability). *A hypergraph H is shrinkable if for every hyperedge $e \in H$ and for every integer $1 \leq i \leq |e|$ there exists a hyperedge e' in H such that $e' \subseteq e$ and $|e'| = i$.*

The VC-dimension of a hypergraph is a measure of its complexity, and it plays a central role in statistical learning, computational geometry, and other areas of computer science and combinatorics (see, e.g., [4, 13, 33, 36]).

▶ **Definition 1.2** (VC-dimension). *The VC-dimension of a hypergraph $H = (V, E)$ is the size of the largest subset $S \subseteq V$ such that for every subset $B \subseteq S$ there is a hyperedge $e \in E$ where $e \cap S = B$. Such a subset S is said to be "shattered" by H.*

The following inequality, usually referred to as the Sauer-Shelah-Perles lemma [15, 48, 50], relates the number of vertices and hyperedges of a hypergraph of VC-dimension d:

$$|E| \leq \sum_{i=0}^{d} \binom{|V|}{i}.$$

For a hypergraph $H = (V, E)$ and a subset $X \subseteq V$ the *projection* of H to X is the hypergraph $H[X] = (X, \{e \cap X \colon e \in E\})$. The VC-dimension of $H[X]$ is, at most, the VC-dimension of H because any subset that is shattered by $H[X]$ is also shattered by H.

Let P be a set of points in the plane. The *depth* of a pair $\{x, y\}$ of points in P, denoted by $d_P(x, y)$, is defined as the maximum integer i such that any disk containing x and y contains at least i other points of P. Notice that $0 \leq d_P(x, y) \leq |P| - 2$. The existence of pairs of large depth (also known as deep pairs) is studied in [11, 24, 28, 29, 38, 47]. The notion of depth can be generalized to tuples of points and to higher dimensions as the maximum number i such that any ball containing a deep tuple also contains i other points. We extend the notion of depth further into hypergraphs.

▶ **Definition 1.3** (Depth). *Let $H = (V, E)$ be a hypergraph and S be a subset of V. The depth $d_H(S)$ of S is defined as the maximum integer such that for every hyperedge $e \in E$ that contains S, we have $|e \backslash S| \geq d_H(S)$. If no hyperedge contains S then $d_H(S) := |V \backslash S|$.*

Note that if S has depth i, then in particular every hyperedge that contains S contains at least i other vertices. The notion of depth is monotone with respect to projections, in the sense that for any $X \subseteq V$, we have $d_{H[X]}(S) \leq d_H(S)$.

Let H be a hypergraph and $0 < \epsilon < 1$ be a real number. A subset S of vertices is called an *ϵ-net* for H if any hyperedge e of size at least $\epsilon|V|$ contains a vertex from S. This notion was introduced by Haussler and Welzl [27]. It was then generalized by Mustafa and Ray [37] to tuples and appears under *ϵ-t-nets*; see also [5, 23].

▶ **Definition 1.4** (ϵ-t-Net). *Let $H = (V, E)$ be a hypergraph, $\epsilon \in (0, 1)$ be a real number, and t be a natural number. A subset N of t-tuples of V is called an ϵ-t-net if any hyperedge e in E of size at least $\epsilon|V|$ contains at least one of the t-tuples in N.*

1.3 Our contributions

We present upper bounds on $f(t,k)$ for several families of geometric hypergraphs and hypergraphs of bounded VC-dimension. Some of our bounds are new, and some improve over previously known bounds. These bounds are interesting because, for abstract hypergraphs, $f(t,k)$ is unbounded even if the hypergraph is uniform and $k=2$ (due to Ramsey's theorem; see [35]). We also study the relationship between vertex-coloring, tuple-coloring, cover-decomposability, and epsilon t-nets.

Recall that $f(t,2)$, with $t=1$, is unbounded for hypergraphs that are defined by points and disks in the plane [44] . We prove, however, that it is bounded for $t=2$. A point set P in dimension d is said to be in *general position* if no $(d+2)$ points of P lie on a sphere.

▶ **Theorem 1.5.** *Let $\mathcal{H}$ be the family that contains any hypergraph H that can be obtained by taking a finite set P of points in $\mathbb{R}^2$ in general position setting $V(H) := P$ and $E(H) := \{d \cap P : d \text{ is a disk in } \mathbb{R}^2\}$. Then for any natural number k it holds that $f_{\mathcal{H}}(2,k) \leq 3.7^k$.*

In other words, for any finite set P of points in the plane, the $\binom{|P|}{2}$ pairs of points can be k-colored such that any disk containing at least 3.7^k points must also contain a pair (of points) from each of the k color classes. To obtain this bound, we show some properties of containment of points in disks and also employ a result about the depth of pairs of points – a well-studied topic in discrete and computational geometry [24, 47]. We then generalize this result to tuples and balls in higher dimensions. The parameter e below denotes Euler's number.

▶ **Theorem 1.6.** *Let $d \geq 3$ be an integer and set $t := \lfloor \frac{d+3}{2} \rfloor$. Let $\mathcal{H}$ be the family of all hypergraphs H that can be obtained by taking a finite set P of points in $\mathbb{R}^d$ in general position setting $V(H) := P$ and $E(H) := \{b \cap P : b \text{ is a ball in } \mathbb{R}^d\}$. Then for any natural number k it holds that $f_{\mathcal{H}}(t,k) \leq \left(\frac{4}{5ed^3}\right)^k$.*

For points and axis-parallel rectangles in the plane, we know from [21] that $f(t,2)$ is unbounded when $t=1$. However for $t=2$ a result of [1] implies that $f(2,k) \leq k^2+1$. We study the same problem but for grid points.

▶ **Theorem 1.7.** *Let $\mathcal{H}$ be the family of all hypergraphs H that can be obtained by taking the vertex set P of a regular grid in $\mathbb{R}^2$ setting $V(H){:=}P$ and $E(H){:=}\{r \cap P : r \text{ is an axis-parallel rectangle in } \mathbb{R}^2\}$. Then for any natural number $k \geq 2$ it holds that $f_{\mathcal{H}}(2,k) \leq \sqrt{ck \ln k}$, for some constant $c \geq 126$.*

To better appreciate this bound, observe that $f(2,k) = \Omega(\sqrt{k})$ for any hypergraph (see also Section 5). Theorem 1.8 (proven in the full version of the paper [12]) generalizes this result.

▶ **Theorem 1.8.** *Let $\mathcal{H}$ be the family of all hypergraphs that can be obtained by grid points and boxes (or balls) in any fixed dimension $d \geq 2$. Then for any natural numbers $t \geq 1$ and $k \geq 2$ it holds that $f_{\mathcal{H}}(t,k) = O\left((k \ln k)^{1/t}\right)$.*

We also study hypergraphs of bounded VC-dimension.

▶ **Theorem 1.9.** *For every shrinkable hypergraph H with VC-dimension at least 2 and at most d and with $n \geq 2d+2$ vertices it holds that $f_H(d+1,k) \leq c^{k-1}$ for any $k \geq 1$ and any $c \geq 4(d+1)^{d+1}$.*

This result applies to many families of geometric hypergraphs. For example, hypergraphs defined by points and halfspaces, boxes, or balls in $\mathbb{R}^d$ are shrinkable and have bounded VC dimensions. Along with a proof of this result, we show the existence of a tuple of large depth in hypergraphs of bounded VC-dimension, which is of independent interest.

▶ **Theorem 1.10.** *For every hypergraph H with VC-dimension at least 2 and at most d and with $n \geq 2d+2$ vertices there exists a $(d+1)$-tuple T of vertices such that $d_H(T) \geq \frac{n}{c}$, for $c = 4(d+1)^{d+1}$.*

For hypergraphs where $f(1,x)$ is linear in x, we prove the following asymptotic tight bounds for their tuple-coloring and for decomposition of their tuples into epsilon t-nets.

▶ **Theorem 1.11.** *Let H be a hypergraph for which $f_H(1,x) = O(x)$, for any $x \geq 1$. Then for any natural numbers t and k with $tk^{1/t} \geq 1$ we have $f_H(t,k) = \Theta(tk^{\frac{1}{t}})$.*

▶ **Theorem 1.12.** *Let H be a hypergraph with n vertices such that $f_H(1,x) = O(x)$, for any $x \geq 1$. Then for any $0 < \epsilon < 1$ and any $t \geq 1$ the t-tuples of vertices of H can be decomposed into $\Omega\left((\frac{\epsilon n}{t})^t\right)$ pairwise disjoint ϵ-t-nets for H. This bound is asymptotically tight when t is a constant.*

For an implication of Theorem 1.11 consider the family $\mathcal{H}$ of hypergraphs that are defined by points and halfplanes in $\mathbb{R}^2$. From a result of Smorodinsky and Yuditsky [52] we know that $f_{\mathcal{H}}(1,x) \leq 2x-1$, i.e., any set of points in the plane can be colored by x colors such that any halfplane containing at least $2x-1$ points contains points of all colors. Therefore, Theorem 1.11 implies that $f_{\mathcal{H}}(t,k) = \Theta(tk^{\frac{1}{t}})$.

Finally, we study the relationship between $f(1,k)$ and $f(t,k)$ for $k = 2$, which is usually referred to as bichromatic coloring. It is implied from a result of Ackerman, Keszegh, and Pálvölgyi [1, Proposition 4] that $f_H = (t,2) \leq \max\{f_H(1,2), 2t-1\}$ for any family H and any $t \geq 2$. We present the following improved upper bound.

▶ **Theorem 1.13.** *For any hypergraph H and any integer $t \geq 2$ it holds that*

$$t+1 \leq f_H(t,2) \leq \max\{f_H(1,2), t+1\}.$$

The matching term $t+1$ in the lower and upper bounds of Theorem 1.13 determines the exact value of $f(t,2)$ for some classes of hypergraphs. For instance, when H is defined by points and halfplanes in $\mathbb{R}^2$, the result of Smorodinsky and Yuditsky [52] gives $f_H(1,2) = 3$, and thus Theorem 1.13 implies the following corollary.

▶ **Corollary 1.14.** *Let $t \geq 2$ be an integer. For the family $\mathcal{H}$ of hypergraphs that are defined by points and halfplanes in $\mathbb{R}^2$, it holds that $f_{\mathcal{H}}(t,2) = t+1$.*

2 Coloring Tuples of Points with respect to Disks and Balls

In this section, we prove Theorem 1.5 and Theorem 1.6. Our proofs relies on some simple structural properties of the containment of points in disks and balls. They also employ some results about the existence of deep tuples in point sets. Without loss of generality, all point sets that are considered in this section are assumed to be in general position.

2.1 Points and disks in $\mathbb{R}^2$

Let $\mathcal{H}$ be the family of hypergraphs defined by points and disks in the plane. That is, $\mathcal{H}$ contains any hypergraph H that can be obtained by taking a finite set P of points in $\mathbb{R}^2$ setting $V(H) := P$ and $E(H) := \{d \cap P : d \text{ is a disk in } \mathbb{R}^2\}$. In this section we prove that $f_{\mathcal{H}}(2,k) \leq 3.7^k$. We say that a disk d *contains* a point p if p is in the interior or on the boundary of d. Alternatively, we say that p is in d.

One part of our proof employs the following result of Ramos and Viaña [47]; this is a stronger version of a result that previously appeared in [24].

▶ **Lemma 2.1** (Ramos and Viaña [47]). *In any set of n points in the plane, there is a pair of points such that any circle through them has, both inside and outside, at least $\frac{n}{4.7}$ points (excluding the two points themselves).*

This result is obtained by using the duality (of circles in $\mathbb{R}^2$ and planes in $\mathbb{R}^3$) and some known results on facets of points in $\mathbb{R}^3$. The pair that is obtained by Lemma 2.1 has the property that any circle through them has at least $\frac{n}{4.7}$ and at most $n-\frac{n}{4.7}-2$ points inside. Combining this with the notion of depth, we get the following corollary.

▶ **Corollary 2.2.** *In any set of n points in the plane there is a pair $\{x, y\}$ such that*

$$\frac{n}{4.7} \leq d_P(x, y) < \frac{3.7n}{4.7}.$$

The following lemmas, though very simple, play important roles in our proof. We denote the boundary circle of a disk D by ∂D.

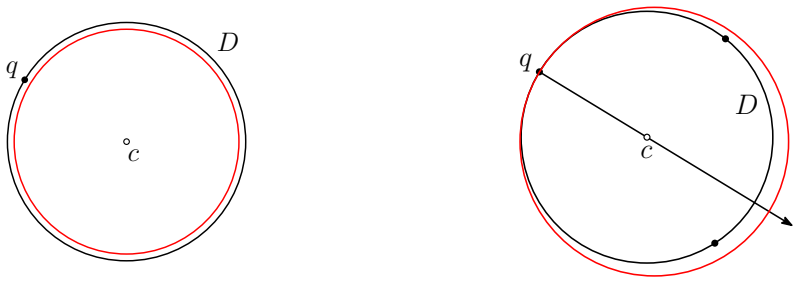


Figure 1 (a) Shrinking D to lose q. (b) Enlarging D along the ray from q to c.

▶ **Lemma 2.3.** *Let P be a finite set of points in the plane and let D be a disk with $|D \cap P| = j$ for some $j \geq 1$. There exists a disk D' such that $(D' \cap P) \subseteq D$ and $|D' \cap P| = j - 1$.*

Proof. Fix D at its center c and shrink it until a point $q \in P$ appears on ∂D for the first time. If q is the only point of P on ∂D, we shrink D slightly to lose q, as in Figure 1(a). Assume that there are other points of P on ∂D. Fix D at q and enlarge it slightly along the ray from q to c such that all other points on ∂D fall inside D and no new point appears in D, as in Figure 1(b). Now, q is the only point on ∂D. Again, we fix D at the center of the new disk and shrink it slightly to lose q. ◀

By repeatedly applying Lemma 2.3, one can shrink/enlarge a disk D to lose 1, 2, 3, or more points of P. Therefore, we get the following more general lemma.

▶ **Lemma 2.4.** *Let P be a finite set of points and D be a disk with $|D \cap P| = j$. For any integer $0 \leq i \leq j$ there exists a disk D' such that $(D' \cap P) \subseteq D$ and $|D' \cap P| = i$.*

▶ **Lemma 2.5.** *Let D be a disk and x, y be two points in the interior of D. There exists a disk that is contained in the interior of D such that its boundary passes through x and y.*

Proof. Fix D at its center c and shrink it until a point, say x, appears on ∂D for the first time. Then fix D at x and shrink it along the segment cx until y appears on ∂D. This disk satisfies the constraints of the lemma. ◀

▶ **Lemma 2.6.** *Let P be a finite set of points in the plane, and let S be a subset of P. Then for any pair $\{x, y\}$ in S it holds that $d_S(x, y) \leq d_P(x, y)$.*

Proof. This is true because for any disk D in the plane, the number of points of P in D is at least the number of points of S in D. ◀

We are ready for the proof of Theorem 1.5.

Proof of Theorem 1.5. Consider any hypergraph H in $\mathcal{H}$ with vertex set P. We color each pair $\{x, y\}$ of points in P with one of the k colors $\{0, 1, \ldots, k-1\}$ as follows:

$$\text{color of } \{x,y\} = \begin{cases} 0 & \text{if } d_P(x,y) = 0 \\ i+1 & \text{if } 3.7^i \leq d_P(x,y) < 3.7^{i+1} \text{ for some } i \in \{0,1,\ldots,k-3\} \\ k-1 & \text{if } d_P(x,y) \geq 3.7^{k-2}. \end{cases}$$

We prove that any disk D containing at least 3.7^k points has pairs of all the k colors.

By an application of Lemma 2.4 there is a disk D_0 such that $(D_0 \cap P) \subseteq D$ and $|D_0 \cap P| = 2$. The two points in D_0 have depth 0 and thus are colored 0.

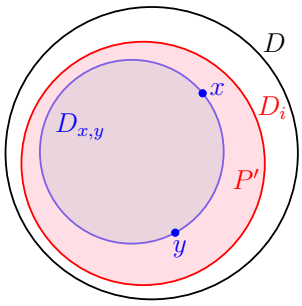


Figure 2 Illustration of the proof of Theorem 1.5.

For each $i \in \{0, \ldots, k-2\}$, we show that there is a pair of points with color $i{+}1$ in D. Let D_i be a disk such that $(D_i \cap P) \subseteq D$ and $|D_i \cap P| = 4.7 \cdot 3.7^i$ (such a disk exists by Lemma 2.4 and the fact that $4.7 \cdot 3.7^i < 3.7^k \leq |D \cap P|$). Let P' be the set of points in D_i. See Figure 2. By Corollary 2.2 the set P' contains a pair $\{x, y\}$ such that any disk through them has at least $\frac{|P'|}{4.7} = 3.7^i$ and at most $\frac{3.7|P'|}{4.7} - 2 = 3.7^{i+1} - 2$ points of P' inside. Thus $3.7^i \leq d_{P'}(x, y) < 3.7^{i+1}$. We argue that these bounds also hold with respect to P, that is $3.7^i \leq d_P(x, y) < 3.7^{i+1}$. The lower bound is implied from Lemma 2.6 because $d_{P'}(x, y) \leq d_P(x, y)$. To verify the upper bound, we take an arbitrary disk D_{xy} through x and y in the interior of D_i (such a disk exists by Lemma 2.5). The disk D_{xy} contains at most 3.7^{i+1} points of P'. Since D_{xy} does not contain any point outside D, it has at most 3.7^{i+1} points of P. This verifies the upper bound on $d_P(x, y)$. Therefore, the pair $\{x, y\}$ has color $i{+}1$. ◀

▶ Remark. To assist the readability of the proof of Theorem 1.5 we avoided rounding when applying Lemma 2.4; this has a negligible impact on the proof. The disk D_i would be chosen to have $\lceil 4.7 \cdot 3.7^i \rceil$ points, which is smaller than 3.7^k. The depth of the pair $\{x, y\}$ in P' would be at least $\frac{\lceil 4.7 \cdot 3.7^i \rceil}{4.7} \geq 3.7^i$ and at most $|P'| - \frac{|P'|}{4.7} - 2 \leq \frac{3.7\lceil 4.7 \cdot 3.7^i \rceil}{4.7} - 2 < 3.7^{i+1}$.

▶ Remark. Ramos and Viaña [47] conjectured that the correct bound in Lemma 2.1 should be $\lfloor \frac{n}{4} \rfloor - 1$. This bound, if true, would be the best achievable [29]. Moreover, it would immediately improve the bound of Theorem 1.5 to 3^k.

2.2 Points and balls in $\mathbb{R}^d$

Our previous result for pairs and disks can be generalized for tuples and balls in higher dimensions. We only need results analogous to Lemma 2.1 (or Corollary 2.2) and Lemma 2.4 in higher dimensions. The following lemma is analogous to Lemma 2.1.

▶ **Lemma 2.7** (Smorodinsky, Sulovský, and Wagner [51]). *In any set of n points in $\mathbb{R}^d$ there is a subset S of size $\lfloor \frac{d+3}{2} \rfloor$ such that any ball containing S contains at least $\frac{4n}{5ed^3}$ points.*

The coefficient $\frac{4}{5ed^3}$ in the lemma is an improvement over the previous (smaller) constant due to Bárány et al. [11]. It is surprising that the bound $\lfloor \frac{d+3}{2} \rfloor$ in the lemma is strongly sharp as proved in [11], in the sense that, there is a set P of points in $\mathbb{R}^d$ (obtained on the moment curve) such that for any subset S of P with $|S| < \lfloor \frac{d+3}{2} \rfloor$ there is a ball that contains S but no other point of P. An analogue to Lemma 2.4 can be obtained in a similar fashion by repeatedly shrinking the ball to lose one point in each iteration.

▶ **Lemma 2.8.** *Let P be a finite set of points in $\mathbb{R}^d$ and B be a ball with $|B \cap P| = j$ for some $j \geq 1$. There exists a ball B' such that $(B' \cap P) \subseteq B$ and $|B' \cap P| = j - 1$.*

Proof. Fix B at its center c and shrink it until, for the first time, a point $q \in P$ appears on ∂B. If ∂B contains other points of P, fix B at q and enlarge slightly along cq so that q becomes the only point on ∂B. Then fix B at c and shrink slightly to lose q. ◀

We sketch a proof of Theorem 1.6, which is somewhat analogous to that of Theorem 1.5.

Proof Sketch for Theorem 1.6. Our proof uses a coloring approach similar to that in the proof of Theorem 1.5. We color every tuple T of size t as follows. If the depth of T is 0 then color it 0 if the depth is at least $\left(\frac{4}{5ed^3}\right)^{k-2}$ then color it $k-1$, and if the depth is at least $\left(\frac{4}{5ed^3}\right)^{i}$ but smaller than $\left(\frac{4}{5ed^3}\right)^{i+1}$ then color it i+1. Then by applications of Lemma 2.8 and Lemma 2.7 one can show that any ball containing at least $\left(\frac{4}{5ed^3}\right)^{k}$ points contains t-tuples of all the k colors. ◀

▶ Remark. The proof of Theorem 1.6 is not completely analogous to that of Theorem 1.5. The proof of Theorem 1.5 relies on the existence of a pair whose depth is large but also not too large (Corollary 2.2). The latter part, however, is not given by Lemma 2.7, and thus the proof of Theorem 1.6 relies on the existence of a pair of large depth; later we will see a similar argument in the proof of Theorem 1.9. Such a proof would also work for Theorem 1.5, giving a bound of 4.7^k instead of 3.7^k.

▶ Remark. The bound of Theorem 1.6 also applies to hypergraphs defined by points and halfspaces in dimension d because a halfspace can be seen as a very large ball.

3 Coloring Tuples of Grid Points

Here we prove Theorem 1.7. Our main tool is the Lovász local lemma [25]. Recall that $\mathcal{H}$ is the family of all hypergraphs H that is obtained by taking the vertex set P of a regular grid in $\mathbb{R}^2$ setting $V(H)$:=P and $E(H)$:=$\{r \cap P : r$ is an axis-parallel rectangle in $\mathbb{R}^2\}$. We prove for any natural number $k \geq 2$ that $f_{\mathcal{H}}(2,k) \leq \sqrt{ck \ln k}$, for some constant $c \geq 126$.

Proof of Theorem 1.7. Set $m := \sqrt{ck \ln k}$. Consider any hypergraph in $\mathcal{H}$. It suffices to prove the statement for hyperedges (i.e. rectangles) that contain at least m and at most $2m$ points, because any rectangle with more than $2m$ points can be shrunk to have between m and $2m$ points. Any polychromatic coloring with respect to shrunk rectangles is also polychromatic coloring with respect to original rectangles.

We prove the existence of a k-coloring of pairs of P using the probabilistic method and the Lovász local lemma [25]; the coloring itself can be computed by the constructive proof of the local lemma that is given by Moser and Tardos [34]. We color the pairs in P by one of the colors from $\{0, \dots, k-1\}$ uniformly at random. Let R be any rectangle containing at least m and at most $2m$ points. The number of pairs in R is at least $\binom{m}{2} \geq \frac{1}{4}ck \ln k$. Consider any color $i \in \{0, \dots, k-1\}$. The probability that a pair $\{x, y\}$ is not colored i is

$$\Pr(\{x, y\} \text{ is not colored } i) = 1 - \frac{1}{k} \leq e^{\frac{-1}{k}}.$$

Since the pairs are colored independently, the probability that no pair in R is colored i is

$$\Pr(\text{no color } i \text{ in } R) \leq \left(e^{\frac{-1}{k}}\right)^{\frac{1}{4}ck \ln k} = \frac{1}{k^{c/4}}.$$

Summing over all the k colors, the union bound implies that the probability of a "bad" event E_R that the rectangle R is not polychromatic (not having at least one of the k colors) is

$$p = \Pr(E_R) \leq \frac{1}{k^{\frac{c}{4}-1}}.$$

Notice that for two rectangles R and R' that do not overlap on any grid point the events E_R and $E_{R'}$ are independent.

Fix a rectangle R. The length and width of R is at most $2m$. Thus, R lies in a square S of side length $2m$. The rectangle R can overlap only with rectangles that are at most $2m$ points away from R in each direction. Thus, the overlapping rectangles must lie in a (bigger) square S^+ of side length $6m$. The number of points in S^+ is $(6m)^2$. Thus, the number of overlapping rectangles is at most $(6m)^4$ because each such rectangle can be defined by two points (say bottom left and top right points). Therefore, any bad event E_R is independent of all but at most

$$d = 6^4 m^4 = 6^4 c^2 k^2 \ln^2 k \leq 6^4 c^2 k^4$$

other events. We claim that $4pd < 1$. Indeed, notice that

$$4pd \leq 4 \frac{1}{k^{\frac{c}{4}-1}} 6^4 c^2 k^4 = \frac{4 \cdot 6^4 c^2}{k^{\frac{c}{4}-5}} \leq \frac{4 \cdot 6^4 c^2}{2^{\frac{c}{4}-5}},$$

where the last inequality is valid because $k \geq 2$. The last expression is a decreasing function in c over the interval $[126, +\infty)$, and its value is less than 1 at $c = 126$. Since $4pd < 1$, the local lemma implies that with a positive probability none of the bad events occur, which in turn implies the existence of a k-coloring of pairs in P such that every rectangle containing at least m points is polychromatic. This completes the proof. ◀

▶ Remark 3.1. Due to our desire to have a short proof, we did not optimize the constant c in Theorem 1.7. It can be improved in several ways, for instance, (i) one can shrink rectangles to have between m and $m + \sqrt{m}$ points instead of m and $2m$ points, (ii) one can consider the neighborhood of R itself instead of the neighborhood of the enclosing square S^+ which is larger, (iii) to count rectangles intersecting R one can exclude those that contain less than m points, and (iv) by using the stronger threshold of $epd < 1$ for the local lemma [49].

4 Coloring Tuples in Hypergraphs of Bounded VC-Dimension

In this section we first prove Theorem 1.10 – the existence of a $(d+1)$-tuple of depth at least $\frac{n}{c}$, with $c = 4(d+1)^{d+1}$, in any hypergraph H of VC-dimension at least 2 and at most d and with $n \geq 2d+2$ vertices. This result, of independent interest, is analogous to Lemma 2.1 and Lemma 2.7. Our proof uses the probabilistic method, which is a common technique in combinatorics and discrete geometry [6, 22]. We then use this result to prove Theorem 1.9.

Proof of Theorem 1.10. Let V be the vertex set of H and $\mathcal{E}$ be the hyperedge set of H. Let k be the maximum depth of a $(d+1)$-tuple in V with respect to H.

If H has a $(d+1)$-tuple T that is not in any hyperedge of $\mathcal{E}$ then, by definition, $d_H = n - (d+1) \geq \frac{n}{2}$ and the lemma holds. Therefore, we assume that each $(d+1)$-tuple T belongs to some hyperedge of $\mathcal{E}$. Let $e_T \in \mathcal{E}$ be a hyperedge that identifies the depth of T, i.e, $T \subseteq e_T$ and $d_H(T) = |e_T| - (d+1)$.

Let $X \subseteq V$ be a random sample of the vertices from V where each vertex is chosen from X independently with a fixed probability $1/n \leq p \leq 1$ (to be determined). Let $\mathcal{E}'$ be the set of all hyperedges in $H[X]$ with cardinality equal to $d+1$. Notice that $|X|$ and $|\mathcal{E}'|$ are random variables. Also, notice that the VC-dimension d' of the projection $H[X]$ is at most d. Thus by the Sauer-Shelah-Perles lemma [48, 50, 15] we have $|\mathcal{E}'| \leq \sum_{i=0}^{d'} \binom{|X|}{i}$. This inequality also holds for the expectations, and we have

$$
\begin{aligned}
\mathbb{E}(|\mathcal{E}'|) &\leq \mathbb{E}\left(\sum_{i=0}^{d'} \binom{|X|}{i}\right) \\
&\leq \mathbb{E}\left(\sum_{i=0}^{d} \binom{|X|}{i}\right) \\
&\leq \mathbb{E}\Big(|X|(|X|-1)\cdots(|X|-d+1)\Big) \\
&= \sum_{i=0}^{n} i(i-1)\cdots(i-d+1)\binom{n}{i} p^i (1-p)^{n-i} \\
&= \sum_{i=d}^{n} \frac{n(n-1)\cdots(n-i+1)}{(i-d)!} p^i (1-p)^{n-i} \\
&= n(n-1)\cdots(n-d+1)p^d \sum_{i=d}^{n} \frac{(n-d)\cdots(n-i+1)}{(i-d)!} p^{i-d}(1-p)^{(n-d)-(i-d)} \\
&= n(n-1)\cdots(n-d+1)p^d \\
&\leq (np)^d.
\end{aligned}
$$

For each $(d+1)$-tuple T in V the probability that $T \in \mathcal{E}'$ is at least the probability that $e_T \cap X = T$ which is exactly $p^{d+1}(1-p)^{d_H(T)}$. Each such probability is at least $p^{d+1}(1-p)^k$ because $d_H(T) \leq k$ for all T. By interpreting $|\mathcal{E}'|$ as the sum of binary indicator variables (one for each $(d+1)$-tuple) and then applying the linearity of expectation, we get

$$
p^{d+1}(1-p)^k \binom{n}{d+1} \leq \mathbb{E}(|\mathcal{E}'|).
$$

Combining this with the upper bound $(np)^d$ and setting $p := \frac{1}{k}$ we get

$$
\binom{n}{d+1} \leq \frac{kn^d}{(1-\frac{1}{k})^k}.
$$

Since $(1-1/k)^k \geq \frac{1}{4}$, we have

$$\binom{n}{d+1} \leq 4kn^d.$$

So $k \geq \frac{\binom{n}{d+1}}{4n^d} \geq \frac{n}{c}$ for $c = 4(d+1)^{d+1}$. This completes the proof. ◀

Now we use Theorem 1.10 to prove Theorem 1.9 which states that for every shrinkable hypergraph H with VC-dimension at least 2 and at most d and with $n \geq 2d+2$ we have $f_H(d+1,k) \leq c^{k-1}$, for any $k \geq 1$ and any $c \geq 4(d+1)^{d+1}$.

Proof of Theorem 1.9. Our proof is somewhat similar to that of Theorem 1.5. We color each $(d+1)$-tuple of vertices of H with one of the k colors $\{0,1,\ldots,k-1\}$ as follows:

$$\text{color of } T = \begin{cases} 0 & \text{if } d_H(T) = 0 \\ i+1 & \text{if } c^i \leq d_H(T) < c^{i+1} \text{ for some } i \in \{0,1,\ldots,k-3\} \\ k-1 & \text{if } d_H(T) \geq c^{k-2}. \end{cases}$$

We prove that any hyperedge e of H that contains at least c^{k-1} vertices, has $(d+1)$-tuples of all the k colors. Since H is shrinkable there is a hyperedge e_0 such that $e_0 \subset e$ and $|e_0| = d+1$. The hyperedge e_0 itself is a $(d+1)$-tuple of depth 0, and hence is colored 0.

For each $i \in \{0,\ldots,k-2\}$ we show that there is a $(d+1)$-tuple with color $i+1$ in e. Let e_i be a hyperedge such that $e_i \subseteq e$ and $|e_i| = c^{i+1}$ (e_i exists by shrinkability of H and the fact that $c^{i+1} \leq c^{k-1} \leq |e|$). Let V be the set of vertices in e_i. The VC-dimension of $H[V]$ is at most the VC-dimension of H, which is at most d. The number of vertices of $H[V]$ is $|V| = c^{i+1} \geq c \geq 4(d+1)^{d+1} \geq 2d+2$. Hence by Theorem 1.10 the hypergraph $H[V]$ has a $(d+1)$-tuple T of depth at least $\frac{|V|}{c} = c^i$. On the other hand, the depth of T in $H[V]$ is at most $c^{i+1} - (d+1)$ because this is the most number of vertices (other than T) that an edge of $H[V]$ can have. Thus $c^i \leq d_{H[V]}(T) < c^{i+1}$. We argue that these bounds also hold with respect to H, that is $c^i \leq d_H(T) < c^{i+1}$. The lower bound is implied from the monotonicity of depth with respect to projections. The upper bound is verified by the edge e_i itself, which is an edge of H, it contains T, and it has at most $c^{i+1} - (d+1)$ other vertices of H. Therefore, T has color $i+1$. ◀

5 Relationships Between $f(1,k)$, $f(t,k)$, and ϵ-t-Nets

To have a $(t,k,f(t,k))$-*polychromatic* coloring, every edge with at least $f(t,k)$ vertices must have t-tuples of all the k colors. This implies that $\binom{f(t,k)}{t} \geq k$. By the standard upper bound for the binomial coefficient, we have $\binom{f(t,k)}{t} \leq \left(\frac{e \cdot f(t,k)}{t}\right)^t$. Combining the two inequalities yields the lower bound

$$f(t,k) \geq \frac{1}{e} \cdot tk^{\frac{1}{t}}. \tag{1}$$

The following folklore result, though very simple, gives an upper bound on $f(t,k)$ in terms of $f(1,tk^{\frac{1}{t}})$. We have not seen a proof of this result elsewhere; we give a short proof.

▶ **Lemma 5.1.** *For any hypergraph H and natural numbers t and k with $tk^{\frac{1}{t}} \geq 1$ it holds that $f_H(t,k) \leq f_H(1,tk^{\frac{1}{t}})$.*

Proof. Set $x := tk^{\frac{1}{t}}$ and color the vertices with x colors such that any edge with $f_H(1,x)$ vertices is polychromatic. Let C_1 be the set of these x colors. Now consider all combinations of t different colors in C_1. We have $\binom{x}{t}$ many such combinations. Consider each combination as a new color class, and let C_t be the set of the new colors (there is a one-to-one correspondence between the colors in C_t and the combinations of t colors in C_1). The standard lower bound for binomial coefficient yields

$$\binom{x}{t} = \binom{tk^{\frac{1}{t}}}{t} \geq \left(\frac{tk^{\frac{1}{t}}}{t}\right)^t = k.$$

Thus C_t has at least k colors. We use the colors in C_t to color each t-tuple T of vertices of H. If the vertices of T are colored by t different colors in C_1, we color T by the corresponding color in C_t. If the vertices of T are colored by less than t colors in C_1 (i.e., some vertices have the same color), we color T by an arbitrary color in C_t.

To prove that our coloring is polychromatic, consider any hyperedge e with at least $f_H(1,x)$ vertices. Since e is polychromatic with respect to the vertex coloring by C_1, it contains x vertices of distinct colors. Of these vertices, each combination of size t would be a t-tuple that is colored by a unique color in C_t. Based on this and the one-to-one correspondence introduced earlier, we conclude that all colors of C_t appear in t-tuples of e, and hence e is polychromatic with respect to t-tuple coloring by C_t. ◀

Proof of Theorem 1.11. A combination of (1) and Lemma 5.1 gives an asymptotically tight bound for $f_H(t,k)$ when $f_H(1,x)$ is linear in x. The statement of the theorem then follows by setting $x := tk^{1/t}$. ◀

5.1 Relationship to ϵ-t-nets

The existence of small-size ϵ-nets was first shown by Haussler and Welzl [27] who proved that any finite hypergraph with VC-dimension d has an ϵ-net of size $O((d/\epsilon)\log(d/\epsilon))$, a bound that was later improved to $O((d/\epsilon)\log(1/\epsilon))$ in [32]. The ϵ-nets are studied extensively in computer science and have found applications in areas such as computational geometry, algorithms, machine learning, and social choice theory; see, e.g., [3, 9, 13, 19]. Alon et al. [5] showed the existence of small-size ϵ-t-nets in various geometric hypergraphs and hypergraphs with bounded VC-dimension.

Here, we prove Theorem 1.12 that gives a decomposition of t-tuples into ϵ-t-nets. It shows the existence of $\Omega\left((\frac{\epsilon n}{t})^t\right)$ pairwise disjoint ϵ-t-nets in a hypergraph H with n vertices for which $f_H(1,x) = O(x)$.

Proof of Theorem 1.12. From Theorem 5.1 and linearity of $f_H(1,x)$ we have $f_H(t,k) = f_H(1,tk^{\frac{1}{t}}) = O(tk^{\frac{1}{t}})$, for any k where $tk^{\frac{1}{t}} \geq 1$. Let k', with $tk'^{\frac{1}{t}} \geq 1$, be the largest integer such that $f_H(t,k') \leq \epsilon n$ (notice that $f_H(t,k'+1) > \epsilon n$). Then, every hyperedge of size at least ϵn contains tuples of all color classes. This means that each of the k' color classes is an ϵ-t-net. To estimate k' we have $\epsilon n < f_H(t,k'+1) = O(t(k'+1)^{\frac{1}{t}})$. This would imply that $k' = \Omega\left((\frac{\epsilon n}{t})^t\right)$. When t is a constant this gives $\Omega(\epsilon^t n^t)$.

To verify the tightness of this bound for constant t, consider a hyperedge e with exactly ϵn vertices. The number of distinct t-tuples in e is $\binom{\epsilon n}{t} \leq \left(\frac{e\cdot\epsilon n}{t}\right)^t = O(\epsilon^t n^t)$. This means that we cannot decompose t-tuples into more than $O(\epsilon^t n^t)$ families of ϵ-t-nets because otherwise e would violate the ϵ-t-net property of some color class. ◀

5.2 Cover-decomposability

The case $k = 2$ is usually referred to as bichromatic coloring. The notion of cover-decomposability is also derived from this case. Recall that a hypergraph is cover-decomposable if $f(1,2)$ is bounded, and it is t-cover-decomposable if $f(t,2)$ is bounded.

Below we prove that $t+1 \leq f_H(t,2) \leq \max\{f_H(1,2), t+1\}$ for any hypergraph H. This improves upon the previous upper bound of $\max\{f_H(1,2), 2t-1\}$, given in [1, Proposition 4].

Proof of Theorem 1.13. The lower bound is obvious because in order for an edge to be polychromatic, it must contain at least two distinct t-tuples, and hence at least $t+1$ vertices ($t-1$ vertices could be shared between the two t-tuples).

Now we prove the upper bound. The upper bound holds if $f_H(1,2)$ is unbounded. Assume that $f_H(1,2)$ is bounded. Let V be the set of vertices of H, and let $\phi\colon V \to \{\text{red}, \text{blue}\}$ be a two-coloring of the vertices in V that achieves $f_H(1,2)$. We use ϕ to obtain a two-coloring ξ of the t-tuples of V such that $\xi\colon \binom{V}{t} \to \{\text{odd}, \text{even}\}$, where $\binom{V}{t}$ denotes the set of all t-element subsets of V. We color each t-tuple T by the parity of the number of its blue vertices. If the number of blue vertices in T is odd, then we color it odd, and if the number of blue vertices in T is even, then we color it even. We show that this is a valid two-coloring of t-tuples, in the sense that any hyperedge $e \in H$ of cardinality at least $\max\{f_H(1,2), t+1\}$ contains two t-tuples such that one is colored odd and the other is colored even.

Since $|e| \geq f_H(1,2)$, e is polychromatic under ϕ. Thus, there are two vertices in e, say r and b, colored red and blue, respectively. Since $|e| \geq t+1$, $e \backslash \{r,b\}$ has at least $t-1$ vertices. Pick an arbitrary subset t' of $e \backslash \{r,b\}$ of size exactly $t-1$. Then $t' \cup \{r\}$ and $t' \cup \{b\}$ are two t-tuples in e and the cardinality of their blue vertices differs by 1. Therefore, one of them is colored odd, and the other is colored even. ◀

6 Conclusions

We presented several new results on polychromatic coloring of tuples in hypergraphs. In particular we gave an exponential bound (in terms of the number of colors) for polychromatic coloring of pairs of points in the plane with respect to disks (Theorem 1.5) – such a bound does not exist for coloring each point individually. One natural question is whether a polynomial upper bound is achievable for coloring pairs. Another natural question is to close the gap between $\Omega(\sqrt{k})$ and $O(\sqrt{k \ln k})$ for polychromatic coloring of pairs of grid points with respect to axis-aligned rectangles. It would also be interesting to explore similar bounds for other point sets and other objects, for example, one may relate the bounds to the fatness of objects.

References

1 Eyal Ackerman, Balázs Keszegh, and Dömötör Pálvölgyi. Coloring Delaunay-edges and their generalizations. *Computational Geometry*, 96:101745, 2021. doi:10.1016/J.COMGEO.2021.101745.

2 Eyal Ackerman, Balázs Keszegh, and Máté Vizer. Coloring points with respect to squares. *Discret. Comput. Geom.*, 58(4):757–784, 2017. Also in *SoCG'16*. doi:10.1007/S00454-017-9902-Y.

3 Noga Alon, Graham R. Brightwell, Hal A. Kierstead, Alexandr V. Kostochka, and Peter Winkler. Dominating sets in k-majority tournaments. *J. Comb. Theory B*, 96(3):374–387, 2006. doi:10.1016/J.JCTB.2005.09.003.

4 Noga Alon, David Haussler, and Emo Welzl. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In *Proceedings of the 3rd annual symposium on Computational geometry (SoCG)*, pages 331–340. ACM, 1987. doi:10.1145/41958.41994.

5 Noga Alon, Bruno Jartoux, Chaya Keller, Shakhar Smorodinsky, and Yelena Yuditsky. The ϵ-t-net problem. *Discret. Comput. Geom.*, 68(2):618–644, 2022. Also in *SoCG'20*. doi:10.1007/S00454-022-00376-X.

6 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Third Edition*. Wiley, 2008.

7 Greg Aloupis, Jean Cardinal, Sébastien Collette, Shinji Imahori, Matias Korman, Stefan Langerman, Oded Schwartz, Shakhar Smorodinsky, and Perouz Taslakian. Colorful strips. *Graphs Comb.*, 27(3):327–339, 2011. doi:10.1007/S00373-011-1014-5.

8 Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, David Orden, and Pedro Ramos. Decomposition of multiple coverings into more parts. *Discret. Comput. Geom.*, 44(3):706–723, 2010. doi:10.1007/S00454-009-9238-3.

9 Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Approximate polytope membership queries. *SIAM J. Comput.*, 47(1):1–51, 2018. Also in *SODA'17*. doi:10.1137/16M1061096.

10 Andrei Asinowski, Jean Cardinal, Nathann Cohen, Sébastien Collette, Thomas Hackl, Michael Hoffmann, Kolja B. Knauer, Stefan Langerman, Michal Lason, Piotr Micek, Günter Rote, and Torsten Ueckerdt. Coloring hypergraphs induced by dynamic point sets and bottomless rectangles. In *In Proceedings of the 13th International Symposium on Algorithms and Data Structures (WADS)*, pages 73–84. Springer, 2013. doi:10.1007/978-3-642-40104-6_7.

11 Imre Bárány, James H. Schmerl, Stuart J. Sidney, and Jorge Urrutia. A combinatorial result about points and balls in Euclidean space. *Discret. Comput. Geom.*, 4:259–262, 1989. doi:10.1007/BF02187727.

12 Ahmad Biniaz, Jean-Lou De Carufel, Anil Maheshwari, Michiel Smid, Shakhar Smorodinsky, and Milos Stojakovic. Polychromatic coloring of tuples in hypergraphs. *CoRR*, 2025. doi:10.48550/arXiv.2503.22449.

13 Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989. doi:10.1145/76359.76371.

14 Béla Bollobás, David Pritchard, Thomas Rothvoß, and Alex D. Scott. Cover-decomposition and polychromatic numbers. *SIAM J. Discret. Math.*, 27(1):240–256, 2013. Also in *ESA'11*. doi:10.1137/110856332.

15 Sarit Buzaglo, Rom Pinchasi, and Günter Rote. *Topological Hypergraphs*, pages 71–81. Springer New York, 2013. doi:10.1007/978-1-4614-0110-0_6.

16 Jean Cardinal, Kolja Knauer, Piotr Micek, Dömötör Pálvölgyi, Torsten Ueckerdt, and Narmada Varadarajan. Colouring bottomless rectangles and arborescences. *Comput. Geom.*, 115:102020, 2023. doi:10.1016/J.COMGEO.2023.102020.

17 Jean Cardinal, Kolja B. Knauer, Piotr Micek, and Torsten Ueckerdt. Making triangles colorful. *J. Comput. Geom.*, 4(1):240–246, 2013. doi:10.20382/JOCG.V4I1A10.

18 Jean Cardinal, Kolja B. Knauer, Piotr Micek, and Torsten Ueckerdt. Making octants colorful and related covering decomposition problems. *SIAM J. Discret. Math.*, 28(4):1948–1959, 2014. Also in *SODA'14*. doi:10.1137/140955975.

19 Timothy M. Chan. Improved deterministic algorithms for linear programming in low dimensions. *ACM Trans. Algorithms*, 14(3):30:1–30:10, 2018. Also in *SODA'16*. doi:10.1145/3155312.

20 Vera Chekan and Torsten Ueckerdt. Polychromatic colorings of unions of geometric hypergraphs. In *Proceedings of the 48th International Workshop Graph-Theoretic Concepts in Computer Science (WG)*, pages 144–157. Springer, 2022. doi:10.1007/978-3-031-15914-5_11.

21 Xiaomin Chen, János Pach, Mario Szegedy, and Gábor Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Struct. Algorithms*, 34(1):11–23, 2009. Also in *SODA'08*. doi:10.1002/RSA.20246.

22 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discret. Comput. Geom.*, 4(1):387–421, 1989. Also in *SoCG'88*. doi:10.1007/BF02187740.

23 Kunal Dutta, Arijit Ghosh, Bruno Jartoux, and Nabil H. Mustafa. Shallow packings, semialgebraic set systems, Macbeath regions, and polynomial partitioning. *Discret. Comput. Geom.*, 61(4):756–777, 2019. Also in *SoCG'17*. doi:10.1007/S00454-019-00075-0.

24 Herbert Edelsbrunner, N. Hasan, Reimund Seidel, and Xiaojun Shen. Circles through two points that always enclose many points. *Geometriae Dedicata*, 32:1–12, 1989. doi:10.1007/BF00181432.

25 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V.T. Sós, editors, *Infinite and finite sets (to P. Erdős on his 60th birthday)*, volume II, pages 609–627. North-Holland, 1975.

26 Matt Gibson and Kasturi R. Varadarajan. Optimally decomposing coverings with translates of a convex polygon. *Discret. Comput. Geom.*, 46(2):313–333, 2011. doi:10.1007/S00454-011-9353-9.

27 David Haussler and Emo Welzl. epsilon-nets and simplex range queries. *Discret. Comput. Geom.*, 2:127–151, 1987. Also in *SoCG'86*. doi:10.1007/BF02187876.

28 Ryan Hayward. A note on the circle containment problem. *Discret. Comput. Geom.*, 4:263–264, 1989. doi:10.1007/BF02187728.

29 Ryan Hayward, David Rappaport, and Rephael Wenger. Some extremal results on circles containing points. *Discret. Comput. Geom.*, 4(3):253–258, 1989. doi:10.1007/BF02187726.

30 Balázs Keszegh and Dömötör Pálvölgyi. The geometric hypergraph zoo (beta). URL: https://coge.elte.hu/cogezoo.html.

31 Balázs Keszegh and Dömötör Pálvölgyi. More on decomposing coverings by octants. *J. Comput. Geom.*, 6(1):300–315, 2015. doi:10.20382/JOCG.V6I1A13.

32 János Komlós, János Pach, and Gerhard J. Woeginger. Almost tight bounds for epsilon-nets. *Discret. Comput. Geom.*, 7:163–173, 1992. doi:10.1007/BF02187833.

33 Jirí Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discret. Comput. Geom.*, 31(2):251–255, 2004. doi:10.1007/S00454-003-2859-Z.

34 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11:1–11:15, 2010. doi:10.1145/1667053.1667060.

35 Dhruv Mubayi and Andrew Suk. *A Survey of Hypergraph Ramsey Problems*, pages 405–428. Springer International Publishing, 2020. doi:10.1007/978-3-030-55857-4_16.

36 Nabil Mustafa and Kasturi Varadarajan. Epsilon-approximations and epsilon-nets. In *Handbook of Discrete and Computational Geometry*. HAL open science, 2017.

37 Nabil H. Mustafa and Saurabh Ray. ϵ-Mnets: Hitting geometric set systems with subsets. *Discret. Comput. Geom.*, 57(3):625–640, 2017. doi:10.1007/S00454-016-9845-8.

38 Victor Neumann-Lara and Jorge Urrutia. A combinatorial result on points and circles on the plane. *Discret. Math.*, 69(2):173–178, 1988. doi:10.1016/0012-365X(88)90015-5.

39 János Pach. Decomposition of multiple packing and covering. *Kolloquium über Diskrete Geometrie, Salzburg, Inst. Math. U. Salzburg*, pages 169–178, 1980.

40 János Pach. Covering the plane with convex polygons. *Discret. Comput. Geom.*, 1:73–81, 1986. doi:10.1007/BF02187684.

41 János Pach, Dömötör Pálvölgyi, and Géza Tóth. Survey on decomposition of multiple coverings. *Geometry – Intuitive, Discrete, and Convex: A Tribute to László Fejes Tóth*, pages 219–257, 2013.

42 János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. In *China-Japan Conference on Discrete Geometry, Combinatorics and Graph Theory (CJCDGCGT)*, pages 135–148. Springer, 2005. doi:10.1007/978-3-540-70666-3_15.

43 János Pach and Géza Tóth. Decomposition of multiple coverings into many parts. *Comput. Geom.*, 42(2):127–133, 2009. Also in *SoCG'07*. doi:10.1016/J.COMGEO.2008.08.002.

44 János Pach and Dömötör Pálvölgyi. Unsplittable coverings in the plane. *Advances in Mathematics*, 302:433–457, 2016. doi:10.1016/j.aim.2016.07.011.

45 Dömötör Pálvölgyi and Géza Tóth. Convex polygons are cover-decomposable. *Discret. Comput. Geom.*, 43(3):483–496, 2010. doi:10.1007/S00454-009-9133-Y.

46 Tim Planken and Torsten Ueckerdt. Polychromatic colorings of geometric hypergraphs via shallow hitting sets. In *In Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, pages 74:1–74:14, 2024. doi:10.4230/LIPICS.SOCG.2024.74.

47 Pedro A. Ramos and Raquel Viaña. Depth of segments and circles through points enclosing many points: a note. *Comput. Geom.*, 42(4):338–341, 2009. doi:10.1016/J.COMGEO.2008.07.001.

48 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.

49 James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. doi:10.1007/BF02579368.

50 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41:247–261, 1972. doi:10.2140/pjm.1972.41.247.

51 Shakhar Smorodinsky, Marek Sulovský, and Uli Wagner. On center regions and balls containing many points. In *Proceedings of the 14th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 363–373. Springer, 2008. doi:10.1007/978-3-540-69733-6_36.

52 Shakhar Smorodinsky and Yelena Yuditsky. Polychromatic coloring for half-planes. *J. Comb. Theory A*, 119(1):146–154, 2012. Also in *SWAT'10*. doi:10.1016/J.JCTA.2011.07.001.

Polynomial-Time Algorithms for Contiguous Art Gallery and Related Problems

Ahmad Biniaz
School of Computer Science, University of Windsor, Canada

Anil Maheshwari
School of Computer Science, Carleton University, Canada

Magnus Christian Ring Merrild
Department of Computer Science, Aarhus University, Denmark

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

Saeed Odak
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

Valentin Polishchuk
Communications and Transport Systems, Linköping Univeristy, Sweden

Eliot W. Robson
Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA

Casper Moldrup Rysgaard
Department of Computer Science, Aarhus University, Denmark

Jens Kristian Refsgaard Schou
Department of Computer Science, Aarhus University, Denmark

Thomas Shermer
School of Computing Science, Simon Fraser University, Burnaby, Canada

Jack Spalding-Jamieson
Independent, Vancouver, Canada

Rolf Svenning
Department of Computer Science, Aarhus University, Denmark

Da Wei Zheng
Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA

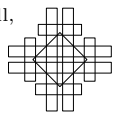
Abstract

We introduce the *contiguous art gallery problem* which is to guard the boundary of a simple polygon with a minimum number of guards such that each guard covers *exactly one* contiguous portion of the boundary. Art gallery problems are often NP-hard. In particular, it is NP-hard to minimize the number of guards to see the boundary of a simple polygon, without the contiguity constraint.

This paper is a merge of three concurrent works [17, 53, 62] each showing that (surprisingly) the contiguous art gallery problem is solvable in polynomial time. The common idea of all three approaches is developing a greedy function that maps a point on the boundary to the furthest point on the boundary so that the contiguous interval along the boundary between them could be guarded by one guard. Repeatedly applying this function immediately leads to an OPT+1 approximation. By studying this greedy algorithm, we present three different approaches that achieve an optimal solution. The first and second approach apply this greedy algorithm from different points on the boundary that could be found in advance or on the fly while traversing along the boundary (respectively). The third approach represents this function as a piecewise linear rational function, which can be reduced to an abstract arc cover problem involving infinite families of arcs. We identify other problems that can be represented by similar functions, and solve them via the third approach.

From the combinatorial point of view, we show that any n-vertex polygon can be guarded by at most $\lfloor \frac{n-2}{2} \rfloor$ guards. This bound is tight because there are polygons that require this many guards.

© Ahmad Biniaz, Anil Maheshwari, Magnus Christian Ring Merrild, Joseph S. B. Mitchell, Saeed Odak, Valentin Polischuk, Eliot W. Robson, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou, Thomas Shermer, Jack Spalding-Jamieson, Rolf Svenning, and Da Wei Zheng;
licensed under Creative Commons License CC-BY 4.0



41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang; Article No. 20; pp. 20:1–20:21

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

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Art Gallery Problem, Computational Geometry, Combinatorics, Discrete Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.20

Related Version *This paper is a merge of 3 submissions: Biniaz, Maheshwari, Mitchell, Odak, Polishchuk, and Shermer*: https://arxiv.org/abs/2412.15053 [17]
Merrild, Rysgaard, Schou, and Svenning: https://arxiv.org/abs/2412.13938 [53]
Robson, Spalding-Jamieson, and Zheng: https://arxiv.org/abs/2412.15567 [62]

Supplementary Material
Software (Source Code): https://github.com/RolfSvenning/ContiguousArtGallery [68]
archived at swh:1:dir:7cfaba2c09d953feb90a49f0e26370ea3f7719a7

Funding *Ahmad Biniaz*: Research supported in part by NSERC.
Anil Maheshwari: Research supported in part by NSERC.
Magnus Christian Ring Merrild: Research supported by Independent Research Fund Denmark (DFF), grant 10.46540/3103-00334B.
Joseph S. B. Mitchell: Partially supported by the National Science Foundation (CCF-2007275).
Saeed Odak: Research supported by NSERC.
Valentin Polishchuk: Partially supported by the Swedish Transport Administration and the Swedish Research Council.
Casper Moldrup Rysgaard: Research supported by Independent Research Fund Denmark (DFF), grant 9131-00113B.
Jens Kristian Refsgaard Schou: Research supported by Independent Research Fund Denmark (DFF), grant 9131-00113B.
Thomas Shermer: Research supported by NSERC.
Rolf Svenning: Research supported by Independent Research Fund Denmark (DFF), grant 9131-00113B.
Da Wei Zheng: Research supported in part by an NSERC PGSD.

Acknowledgements We are grateful to the anonymous reviewers for helpful suggestions, and we thank Joseph O'Rourke for organizing the open problem session at the Canadian Conference on Computational Geometry 2024 (CCCG24), where Thomas C. Shermer proposed the contiguous art gallery problem. The authors further thank the participants of CCCG24 for their lively discussions of the problem at the conference. We thank the members of the Stony Brook CG Group open problem seminar (2022), where Joe Mitchell posed questions regarding a greedy algorithm for the "CBG" (Contiguous Boundary Guarding) problem and its variants.

1 Introduction

The *art gallery problem*, introduced in 1973 by Victor Klee [20, 58], asks for a minimum number of guards that see every point of a given polygon P – the guards can lie anywhere in the polygon. This problem is central to computational geometry and is an active research area. It is long known to be NP-hard [48] and APX-hard [32]; see also [5, 58, 60]. A recent result by Abrahamsen, Adamaszek, and Miltzow [3, 2] shows that the problem is $\exists\mathbb{R}$-complete even if the vertices have integer coordinates. In fact, [1] and [52] present instances with optimal solutions of size 3 and 2 that require guards with irrational coordinates. The problem is notoriously difficult, as the best known approximation algorithms have logarithmic approximation factors [18, 27, 30, 31]; see also [32] for hardness of approximation.

The *boundary guarding* problem is a variant of the art gallery problem in which the goal is to guard only the boundary of P with a minimum number of guards that can lie anywhere in P. One may think of this as guarding only the walls of an art gallery, with no interior sculptures. By extending ideas of [3, 2], Stade has shown that this variant is also $\exists\mathbb{R}$-complete [67]. In this paper we introduce the *contiguous* version of this problem. In the *contiguous art gallery* problem, the goal is to guard the boundary of P such that each guard covers a contiguous portion of the boundary. While the visibility polygon of a guard may have several connected components on the boundary, the guard is assigned to only one component.

The contiguous art gallery problem is a natural modification of the boundary guarding problem: Here, the responsibility of a guard is a part of the boundary that could be contiguously scanned by a rotating security camera (or human) that cannot pause its rotation to completely re-focus (or perform an autofocus). The problem appeared in open problems from CCCG 2024 [8], motivated by the fact that the hardness proofs for typical art gallery variants [3, 2, 67] require guards to cover several connected components on the boundary. Figure 1 illustrates the difference between standard boundary guarding and the contiguous version. In this paper, we give polynomial-time algorithms for contiguous boundary guarding.

We present three approaches to the problem. All three start from the observation that there is a function G called the *greedy function*, mapping points along the boundary of the polygon to maximal point counterclockwise along the boundary so that a guard point exists that can guard the entire boundary between the two points. We discuss this function in Section 1.4, including how it can be used to obtain an additive $+1$ approximation. At a high-level, the three different approaches extend this in different manners: The first applies G repeatedly until it produces an optimal solution, going around the polygon a polynomial number of times. The second studies the specific structure of an optimal solution, showing that a polynomial-sized finite subset of the domain of G needs to be considered. The third shows that G has a representation as a piecewise linear rational function, which can be reduced to an abstract arc cover problem involving infinite families of arcs, resulting in the *analytic arc cover problem*. This third approach can also be applied to solve minimum polygon separation (see Figure 2) and polytope carving (see Figure 3) in polynomial time.

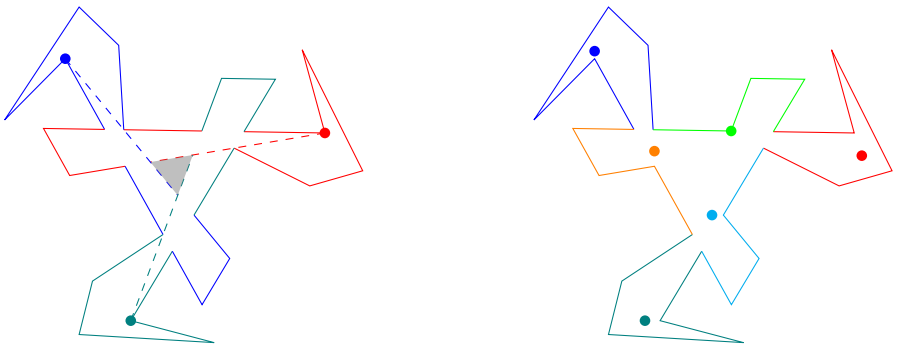


Figure 1 (Left) 3 optimal boundary-covering guards (the interior gray triangle is not seen). (Right) 6 guards that optimally contiguously cover the boundary. Guards are visualized as points whose color matches the boundary portion they guard.

1.1 Related work and results

The art gallery problem and its variants have been extensively studied. Variants arise from constraints on the allowable input polygon, the allowable guard locations, and the portion of the polygon that must be guarded. Essentially all nontrivial variants are NP-hard.

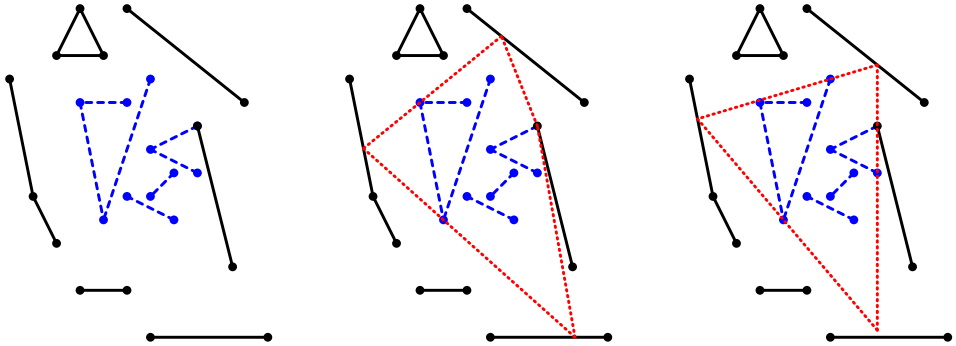


Figure 2 (Left) An instance of the *segment separation problem*, which asks for a convex polygon with the fewest vertices that separates two input sets of line segments. (Middle) A suboptimal solution. (Right) One possible optimal solution.

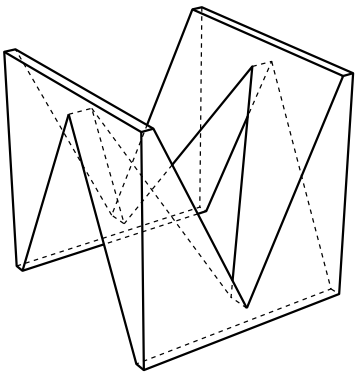


Figure 3 The *polytope half-plane carving problem* asks for the minimum number of half-plane cuts needed to separate a specific target polytope from the ambient material. Pictured is a polytope that can be carved by 10 half-plane cuts.

Hardness and approximations. The *vertex-guard problem* requires guards to be at vertices of the polygon. This version is also NP-hard [48, 60] and APX-hard [32] with known logarithmic factor [37] and sub-logarithmic factor [43] approximation algorithms. A similar sub-logarithmic approximation is known for the case where the guards can lie anywhere on the boundary of the polygon [43]. Better approximation algorithms are known for monotone polygons [46] and weakly-visible polygons [11, 12, 14].

Terrain guarding is a related problem in which we are given a 1.5-dimensional terrain (i.e., an x-monotone polygonal chain), and we want to guard the terrain with a minimum number of guards on the terrain – this is a variant of the boundary guarding problem. This problem is also NP-hard [19, 45, 44], and admits a polynomial time approximation scheme [11, 12, 38].

Both versions of the problem (point-guard and vertex-guard) remain NP-hard in orthogonal polygons [65], even if we want to guard only the vertices of the polygon [42].

Polynomial-time exact algorithms. For some trivial versions, the art gallery problem can be solved in polynomial time, such as for convex, star-shaped, and spiral polygons. Finding non-trivial instances that admit polynomial-time algorithms is of particular interest. To the best of our knowledge, there are only a few such versions. These instances usually enforce constraints on the polygon itself or on the definition of visibility. For example, the authors in [21] present polynomial-time algorithms when the polygon is uni-monotone (i.e., it is

monotone and one of its two monotone polygonal chains is a straight line segment)[1] and for 1.5-dimensional terrains where all the guards must be placed on a horizontal line above the terrain. The art gallery problem can also be solved in polynomial time on orthogonal polygons under *rectilinear* visibility [76] and *staircase* visibility [56, 55]. In the former case, two points are visible if their minimum orthogonal bounding box lies in the polygon, and in the latter case, two points are visible if there is an x- and y-monotone path between them in the polygon. All of these algorithms [21, 56, 55, 76] employ the notion of *perfectness* [9, 10] of P, which means that the minimum number of guards is equal to the maximum number of points that can be placed in P such that their visibility polygons are pairwise disjoint.

Combinatorial bounds. While most variants of the art gallery problem are algorithmically hard, tight combinatorial bounds are known on the numbers of guards that are always sufficient and sometimes necessary to guard a polygon with n vertices. For the original art gallery problem the bound is $\lfloor n/3 \rfloor$ for both point-guards and vertex-guards [20, 35], and it is $\lfloor n/4 \rfloor$ for orthogonal polygons [40, 50, 64, 74]. Tight bounds are also known for variants in which an entire edge or diagonal can serve as a guard [15, 57, 66].

Contiguity. The contiguous guarding is particularly related to guarding polygons with cameras/guards that have a bounded field of view (i.e., the maximum angle a camera can cover is limited) [71, 72] and to the so-called *floodlight* [59, 73] and *city guarding* problems [13, 16, 24]. It is also related to contiguous guarding of points on a 1.5-dimensional terrain by placing at most k watchtowers such that each watchtower sees a contiguous sequence of points [41].

Circle-cover minimization. When partitioning the boundary of a simple polygon, it is natural to view it as a circle and the polygonal chains as arcs or intervals; see Figure 4. By doing so, finding a minimal partition is similar to finding a *minimal circle-cover* among a set of intervals C. This problem was studied by Lee and Lee [47] for finite cardinality C, giving an algorithm that runs in $\mathcal{O}(|C| \log |C|)$ time. Thus, given a finite set C of polygon chains of the boundary of P such that an optimal solution is contained in C, the problem can be solved in $\mathcal{O}(|C| \log |C|)$ time, see Figure 4. However, it is not easy to define a set C that is guaranteed to contain an optimal solution.

Minimum Polygon Separation. Given a convex polygon P contained inside another convex polygon Q, the minimum polygon separation problem asks for a polygon S with the minimum number of vertices such that S contains P and is contained within Q. This is the *polygon separation problem*. Polygon separation was first studied by Aggarwal, Booth, O'Rourke, Suri, and Yap [6, 7], and generalized to multiple connected domains by Mitchell and Suri [54]. One can consider a similar problem: The problem of separating two collections of line segments by a polygon with the minimum number of vertices. This is the *segment separation* problem.

Polytope Carving. Related to polygon separation is the problem of *carving* one shape out of another, or out of an ambient space. Two-dimensional carving was first studied by Overmars and Welzl [61], where they aimed to find the cheapest sequence of line cuts to carve a convex polygon out of a piece of flat material. This has also been studied in the context of rays cuts [22, 23, 69] and line segments cuts [25, 26, 28, 29], as well as in 3D with half-plane cuts [63].

[1] This type of polygons are also referred to as monotone mountains.

1.2 Preliminaries

Let P be a simple polygon with n vertices. We denote the boundary of P by ∂P. A *reflex vertex* of P is a vertex with an interior angle greater than π. A non-reflex vertex is a *convex vertex*. For two points p and q in the plane, we denote by $\overline{p\,q}$ the straight line segment between p and q. The *ray* from p toward q is denoted $\overrightarrow{p\,q}$. A *wedge* is a region of the plane bounded by two rays starting from the same point known as the *apex* of the wedge.

We assume that ∂P is a directed cycle, with vertices ordered counter-clockwise. We index the edges e_i and the vertices v_i of P modulo n, that is, $e_n = e_0$ and $v_n = v_0$. We define a *polygonal chain* as a connected portion of ∂P. For an ordered pair (p_1, p_2) of two points on ∂P, we denote by $[p_1, p_2]$ the interval of ∂P from p_1 to p_2 in counter-clockwise direction – $[p_1, p_2]$ is a directed path, see Figure 4. We refer to p_1 and p_2 as the *first* and the *last* endpoints of $[p_1, p_2]$, respectively. Since $[p_1, p_2]$ is directed, for any two points p and q on $[p_1, p_2]$, we can tell if p appears *before* or *after* q. We define a *guardable* interval $[p_1, p_2]$ to be one for which there exists a point $g \in P$ (guard) such that for all $x \in [p_1, p_2]$ we have $\overline{x\,g} \subset P$. Now, a *contiguous boundary guarding (CBG)* is a collection of guards $\Gamma = \{g_i\}_i$ with guardable intervals $\partial(g_i) = [a_i, b_i]$ such that $\cup_i \partial(g_i) = \partial P$. Multiple guards could be collocated ($g_i = g_j$) while guarding different intervals ($\partial(g_i) \neq \partial(g_j)$). We say that Γ is *minimal* if the removal of any guard from Γ results in an uncovered portion of ∂P. The goal of the *contiguous art gallery problem* is to find a smallest minimal guarding, and we define k^* to be the size of such a guarding.

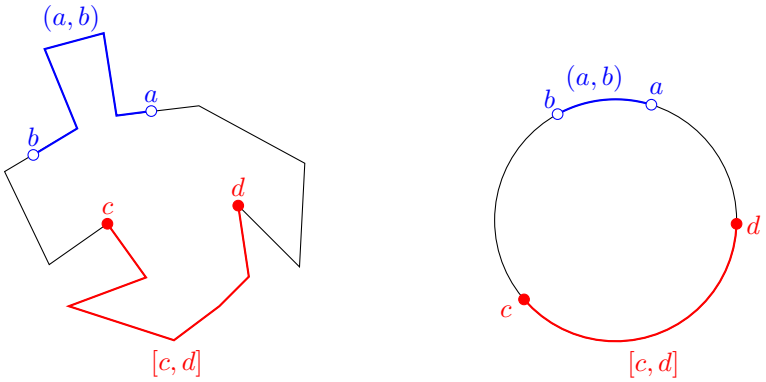


Figure 4 A simple polygon visualized as a circle, with intervals of the polygonal boundary represented as arcs on the circle.

Let $p_1(g)$ and $p_2(g)$ denote the first and last endpoints of $\partial(g)$. We denote by $w(g)$ the wedge from the ray $\overrightarrow{g\,p_1(g)}$ to the ray $\overrightarrow{g\,p_2(g)}$ intersected with P, see Figure 5. Further note that for every guard g in a minimal contiguous guarding, there exists a point g_p of ∂P that is guarded by g alone, because otherwise we could remove g from the guard set. For any (minimal) contiguous guarding Γ, the counter-clockwise order of $\{g_p : g \in \Gamma\}$ along ∂P induces an order of the guards in Γ. Thus, the terms *previous* guard, *next* guard, and *consecutive* guards are well defined. The following simple observation is valid for any guarding with at least two guards.

▶ **Observation 1.** *Let g be a guard in a guarding of ∂P. Then, the first endpoint of $\partial(g)$ is covered by g and by the previous guard. Similarly, the last endpoint of $\partial(g)$ is covered by g and the next guard.*

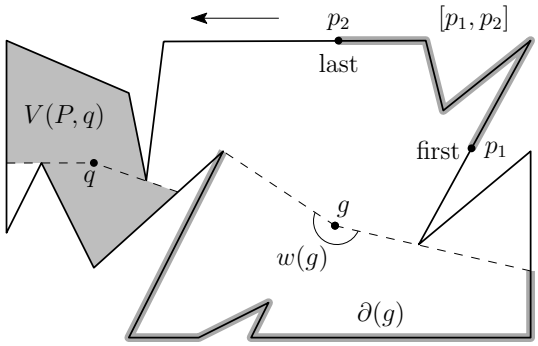


Figure 5 The boundary of the polygon is assumed to be directed counter-clockwise.

1.3 Combinatorial Bounds

In this section, we determine the number of guards that are always sufficient and sometimes necessary to guard an n-vertex polygon. It is easily seen that a chain with n edges can be covered by $\lfloor \frac{n+1}{2} \rfloor$ guards – this can be achieved by placing a guard at every second vertex. Thus it follows that a polygon with n vertices (and hence n edges) can be covered by $\lfloor \frac{n+1}{2} \rfloor$ guards. As we will see later, this bound is very close to the best achievable bound but is not tight. We need stronger ingredients, such as the following lemma, to obtain a tight bound. In this section we say that a vertex "covers" an edge if the vertex sees both endpoints.

▶ **Lemma 2.** *For any polygon P with at least 8 vertices, one of the following statements holds.*

1. *There is a vertex that covers 6 consecutive edges of ∂P.*
2. *There are two vertices that collectively cover 7 consecutive edges of ∂P.*
3. *There are two vertices that collectively cover 8 edges of ∂P such that each vertex covers 4 consecutive edges.*

▶ **Theorem 3.** *In any polygon P with $n \geq 4$ vertices there exists a set of at most $\lfloor \frac{n-2}{2} \rfloor$ points that guard the boundary of P contiguously. This bound is tight.*

Proof. If $n = 4, 5$, then one vertex of the polygon covers the entire boundary (to verify this, observe that in any triangulation of a 5-gon, there is a vertex that is incident to the three resulting triangles). If $n = 6$, then any triangulation contains a diagonal that partitions the boundary of P into 2 and 4 consecutive edges;[2] each partite can be covered by one vertex. If $n = 7$, then there is a diagonal that partitions the boundary of P into 3 and 4 consecutive edges; again, each partite can be covered by one vertex.

Assume that $n \geq 8$. Then, at least one of the cases in Lemma 2 holds. For each case, we show how to cover ∂P by the number of guards that is claimed.

- *Statement 1 holds.* We cover 6 consecutive edges by 1 guard and the remaining chain of $n-6$ edges by at most $\lfloor \frac{n-5}{2} \rfloor$ guards. Thus, the total number of guards is at most $\lfloor \frac{n-3}{2} \rfloor$.
- *Statement 2 holds.* We cover 7 consecutive edges by 2 guards and the remaining chain of $n-7$ edges by at most $\lfloor \frac{n-6}{2} \rfloor$ guards. The total number of guards is at most $\lfloor \frac{n-2}{2} \rfloor$.
- *Statement 3 holds.* We cover the 8 edges by two guards. If 8 edges are consecutive, then we cover the remaining chain as in the previous case. Assume they are not consecutive and thus split into two sets, each having four consecutive edges. The remaining $n-8$ edges of ∂P form two disjoint chains of n_1 and n_2 edges that can be covered by at

[2] For any $k \geq 2$ there is a diagonal that cuts off at least k and at most $2k-2$ edges of the polygon; see [15].

most $\lfloor \frac{n_1+1}{2} \rfloor$ and $\lfloor \frac{n_2+1}{2} \rfloor$ guards, respectively. The total number of guards is at most $\lfloor \frac{n_1+1}{2} \rfloor + \lfloor \frac{n_2+1}{2} \rfloor + 2 \leq \frac{n-2}{2}$. This equals $\lfloor \frac{n-2}{2} \rfloor$ when n is even. When n is odd, one of the chain sizes, say n_1, is even, and hence the chain itself can be covered by at most $\lfloor \frac{n_1}{2} \rfloor$ guards, which would result in at most $\frac{n-3}{2} \leq \lfloor \frac{n-2}{2} \rfloor$ total guards.

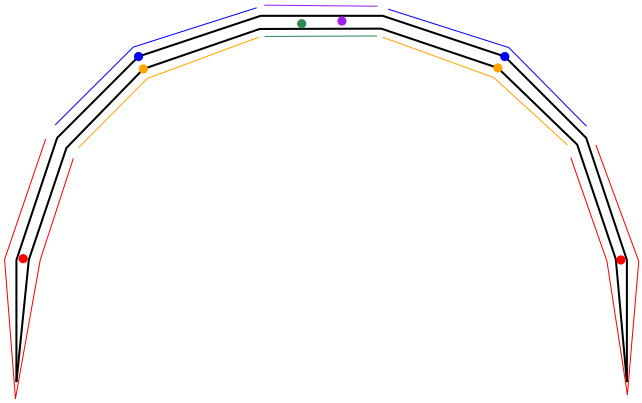


Figure 6 Illustration of the lower bound $\lfloor \frac{n-2}{2} \rfloor$ guards. The polygonal arcs around the boundary are maximal chains that can each be guarded by a single guard.

To verify the lower bound, see the polygon in Figure 6 of size $n = 4k+2$, for some integer k, which consists of two convex chains of odd length $\frac{n}{2}$. We need at least $2\lfloor \frac{n/2+1}{2} \rfloor - 2 = \lfloor \frac{n-2}{2} \rfloor$ guards to cover the boundary of this polygon contiguously. By adding a vertex on any edge of the polygon in Figure 6, a lower bound example for odd n is obtained. ◀

1.4 Greedy Steps

Our overall approach to solving the contiguous art gallery problem studied in this paper is based on the idea to take a point $x \in \partial P$ and find the furthest counter clockwise point $\mathrm{G}(x) \in \partial P$ such that $[x, \mathrm{G}(x)]$ is guardable. We introduce the GreedyInterval algorithm that given x on edge $e_{i-1} = \overline{v_{i-1}\, v_i}$ computes $\mathrm{G}(x)$ progressively by intersecting the visibility polygons (V) of vertices $v_i, \ldots, v_{i+j}$ such that the feasible region $F_j := \cap_{h=0}^{j} V(P, v_{i+h}) \neq \emptyset$ and $F_{j+1} = \emptyset$, then the feasible region can be used to compute the maximal point $\mathrm{G}(x)$ on edge $e_{i+j} = \overline{v_{i+j}\, v_{i+j+1}}$ such that a guard can be placed in P that can see from x to $\mathrm{G}(x)$, here maximality means that $\mathrm{G}(x)$ is closest possible to v_{i+j+1} on ∂P such that $[x, \mathrm{G}(x)]$ is guardable.

▶ **Lemma 4.** *The* GreedyInterval *algorithm can be computed in* $\mathcal{O}(jn \log n)$ *arithmetic operations using poly*(n) *bits on* $x \in \partial P$ *where* j *is the number of vertices in* $[x, G(x)]$.

Proof Sketch. Visibility polygons can be computed in $\mathcal{O}(n)$ by an algorithm of Gindy and Avis [33]. The number of vertices of the feasible region can be shown to be $\mathcal{O}(n)$, independent of j. An algorithm by Martínez, Rueda, and Feito [51] shows that the intersections can be computed in $\mathcal{O}((n_1 + n_2 + h) \log(n_1 + n_2))$ time which together implies that the final feasible region can be computed in $\mathcal{O}(jn \log n)$ time. Finally, to compute how far along edge $\overline{v_{i+j}\, v_{i+j+1}}$ $\mathrm{G}(x)$ can be placed, we construct a new blocking polygon from following the rightmost tangent of F_j through v_{i+j+1} and the pieces of ∂P that stick below this tangent. If this would touch F_j, we instead connect it to a point r above the tangent and finish the construction by connecting this back to v_{i+j+1}. Now, $\mathrm{G}(x)$ can be computed using a common

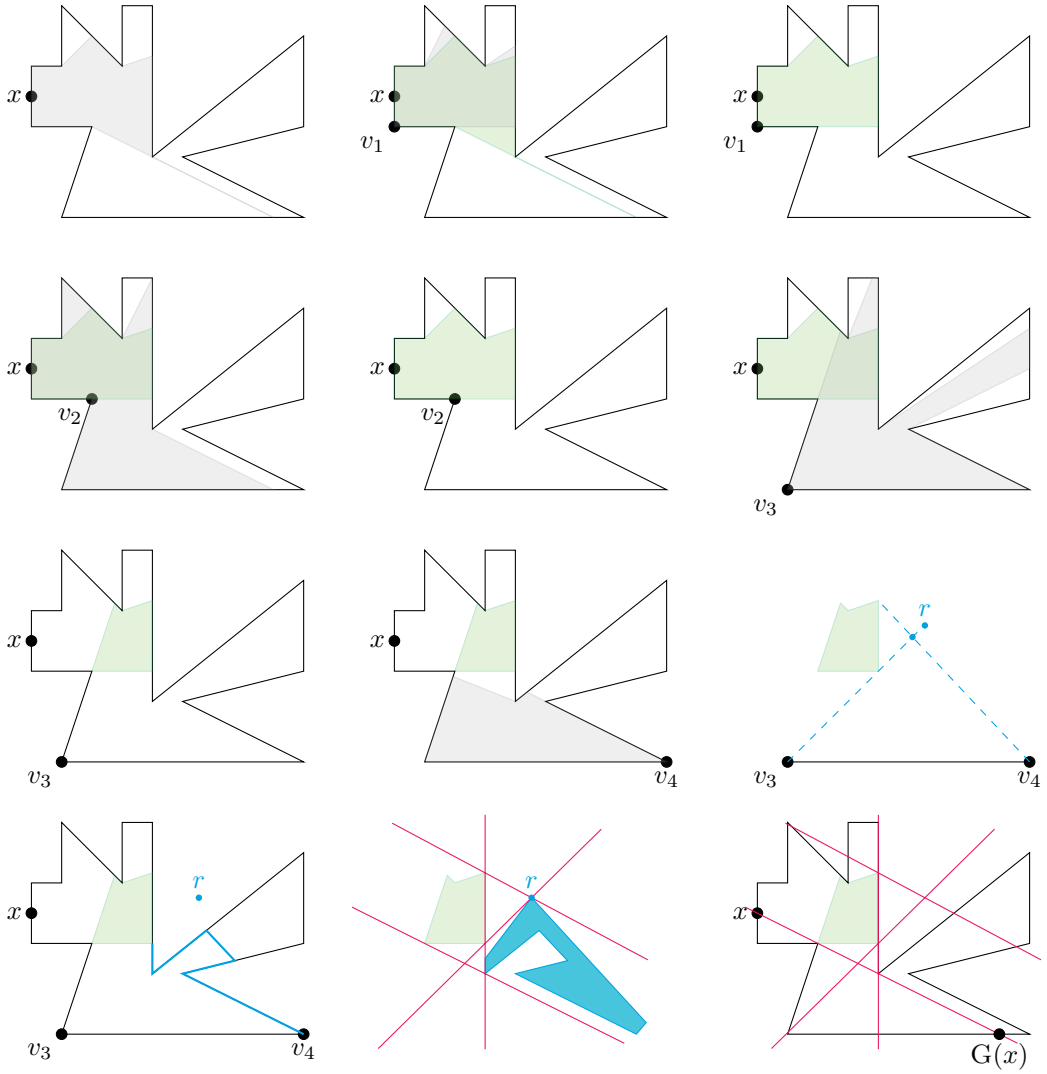


Figure 7 The steps of the GreedyInterval algorithm animated. First successive iterations of computing visibility polygons (in gray) and intersecting with previous feasible regions (in green) and finally computing the blocking polygon (in blue), to find $\mathrm{G}(x)$ through a common tangent.

tangent between F_j and the blocking polygon. This common tangent can be computed with an algorithm of Abrahamsen and Walczak [4]. This process takes $\mathcal{O}(n \log n)$ time, so in total, we use $\mathcal{O}(jn \log n)$ time to compute $\mathrm{G}(x)$. The algorithm is illustrated in Figure 7. ◀

The GreedyRevolution algorithm starts at some point x_0 on the boundary and computes a sequence $x_{i+1} = \mathrm{G}(x_i)$ until x_k is between x_0 and x_1.

▶ **Theorem 5.** *The* GreedyRevolution *algorithm, starting from an arbitrary point $x \in \partial P$, returns a guarding Γ of size at most k^*+1 in $\mathcal{O}\big(n^2 \log n\big)$ arithmetic operations. Moreover, if x is covered by two guards in some optimal solution, then $|\Gamma| = k^*$.*

There is a natural trade-off when guards are moved a little which motivates that it is indeed difficult to check if a candidate solution is optimal or not, see Figure 8.

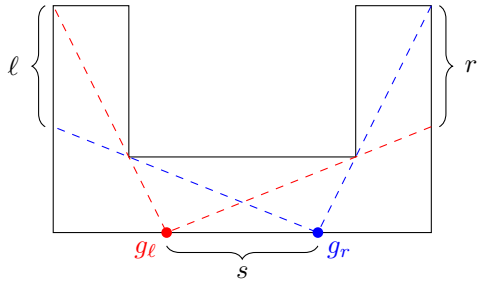


Figure 8 A polygon where the blue guard g_r sees all of r and none of ℓ, and vice-versa for g_ℓ. Any guard placed along s will yield a trade-off between how much of ℓ and r it sees.

In the remainder of this paper, we present three approaches to proving that the contiguous art gallery problem can be solved using GreedyRevolution in polynomial time.

▶ **Theorem 6.** *The contiguous art gallery problem is a member of the complexity class P, with a runtime of* $\mathcal{O}(n^5 k^* \log n)$

1.5 Organization

This paper is the combination of three independent studies [17, 53, 62] of the contiguous art gallery problem. In Section 2 we show that if we continue applying GreedyRevolution $\mathcal{O}(n^3 k^*)$ times then the last candidate solution will be optimal. In Section 3 we show that generating $\mathcal{O}(n^4)$ candidate solutions by starting GreedyRevolution from a carefully selected set of points, at least one of these will be optimal. Finally, in Section 4 we explore how GreedyInterval can be expressed as a piecewise linear rational function and show that repeatedly composing GreedyInterval with itself leads to a piecewise linear rational function representing *all* optimal solutions.

2 Finding an Optimal Solution via Repeated GreedyRevolution

The approach of Merrild, Rysgaard, Schou and Svenning [53] to solve the contiguous art gallery problem is to define the infinite *greedy sequence* $(x_i)_{i=0}^{\infty}$ with $x_0 = x$ and $x_i = \mathrm{G}(x_{i-1})$ and prove that after at most $\mathcal{O}(n^3 k^*)$ revolutions the sequence has converged to an optimal solution, leading to $\mathcal{O}(n^5 k^* \log n)$ runtime.

2.1 Combinatorial optimality conditions

We initially study finite greedy sequences that do not contain optimal solutions to the contiguous art gallery problem, which we define as non-optimal greedy sequences.

If P is star-shaped, i.e., one guard can see all of P and thus all of ∂P we will be guarded after one call to GreedyInterval. Thus, we assume that this is not the case. This together with some elementary properties of G and guardable intervals gives the following lemma, which acts like combinatorial axioms:

▶ **Lemma 7** (Properties of G and guardable intervals). *Let x, y, z, w be points on ∂P.*

1. $G(x) \neq x$.
2. *If $[x, y] \subseteq [z, w]$ and $[z, w]$ is guardable then $[x, y]$ is guardable.*
3. $[x, y] \subseteq [x, G(x)]$ *if and only if* $[x, y]$ *is guardable.*

Using these properties, we derive the following

▶ **Proposition 8** (Behavior of non-optimal greedy sequences). *Let $x \in \partial P$ and assume that the finite greedy sequence $(x_i)_{i=0}^{k \cdot N}$ starting at x does not contain an optimal solution to the* ContiguousArtGallery *problem, where k is the length of a revolution and N is some positive integer. Let $y_0, y_1, \ldots, y_{k^*-1}$ be an optimal solution with $y_0 \in (x_0, x_1]$. Then $x_{ik+m} \in [y_{m-1}, x_{(i-1)k+m})$ for all $m \in \{1, \ldots, k\}$ and $i \in \{1, 2, \ldots, N-1\}$.*

The proof of this proposition follows from considering the greedy steps performed and bounding the result with previous greedy endpoints and optimal endpoints, see Figure 9.

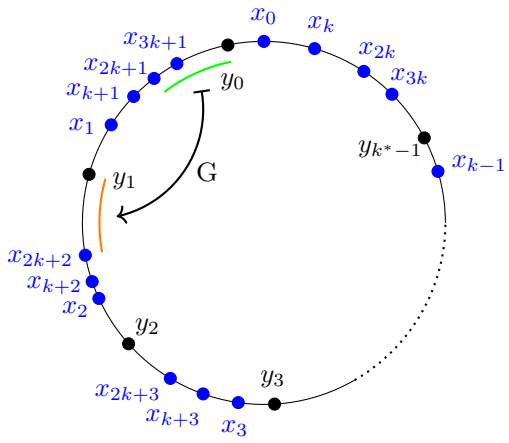


Figure 9 For a non-optimal greedy sequence G maps points from $[y_0, x_{2k+1}]$ (green) to $[y_1, x_{2k+2}]$ (orange), and in the next revolution from $[y_0, x_{3k+1}]$ to $[y_1, x_{3k+2}]$.

From this we see that the subsequence $x_1, x_{k+1}, x_{2k+1}, \ldots$ must be in the interval $[y_0, y_1]$ and the endpoints get closer and closer to y_0. Likewise, all subsequences $x_m, x_{k+m}, x_{2k+m}, \ldots$ lie in $[y_{m-1}, y_m]$ and approach y_{m-1}. Thus, we give these subsequences a name:

▶ **Definition 9** (Local Greedy Sequence). *Let $x \in \partial P$, consider the greedy sequence $(x_i)_{i=0}^{\infty}$ starting at x. Then $(x_i^m)_{i=0}^{\infty}$ for $m = 1, \ldots, k$ are* local greedy sequences, *where $x_i^m = x_{ik+m}$.*

Now, any greedy sequence that does not look like those classified in Proposition 8 must contain an optimal solution. The next step of our analysis is to study geometric polygonal patterns that lead to deviations from this.

2.2 Achieving optimality conditions using Geometry

We now define patterns in which progress toward an optimal solution is achieved:

▶ **Definition 10** (Fingerprint). *Let $(x_i^m)_{i=0}^{\infty}$ be the m'th local greedy sequence. For $i \geq 1$ we say that x_i^m has a* negative fingerprint *if $x_i^m \in [x_{m-1}, x_{i-1}^m)$. Otherwise, we say that x_i^m has a* positive fingerprint.

▶ **Definition 11** (Edge jumps). *Let $(x_i^m)_{i=0}^{\infty}$ be the m'th local greedy sequence. We say that $(x_i^m)_{i=0}^{\infty}$ has an* edge jump *at i if x_i^m is a vertex of P or x_i^m and x_{i+1}^m are on different edges of P and neither is a vertex of P.*

By comparing these with Proposition 8, we get the following optimality conditions:

▶ **Lemma 12** (Progress conditions). *If any of the following three conditions are met, the greedy sequence will contain an optimal solution to the* ContiguousArtGallery *problem:*

1. *If there is a positive fingerprint.*
2. *If there is repetition in the greedy sequence.*
3. *If there are $n+1$ edge jumps.*

Thus, the intermediate goal is to prove the following:

▶ **Theorem 13** (GreedyRevolution will reach a progress condition). *Running* GreedyRevolution $\mathcal{O}(k^* n^2)$ *times guarantees the appearance of a progress condition.*

We show the theorem by zooming in on one greedy step from one local greedy sequence to another. We let $(y_i)_{i=0}^{\infty}$ and $(z_i)_{i=0}^{\infty}$ be two local greedy sequences such that $z_i = \mathrm{G}(y_i)$.

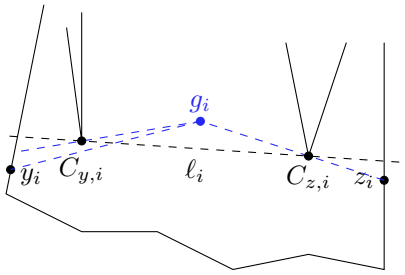


Figure 10 A greedy step from y_i to $z_i := \mathrm{G}(y_i)$ guarded by g_i. The pivot point $C_{z,i}$ blocks g_i from seeing further than z_i and the other pivot $C_{y,i}$ blocks g_i in the other direction. The relative position of the line $\ell_i := L(C_{z,i}, C_{y,i})$ between pivot points to g_i is essential for the casework.

We denote the guard, which can see $[y_i, z_i]$ by g_i and let $C_{z,i}$ and $C_{y,i}$ be the vertices of P that block g_i from seeing more than, respectively z_i and y_i on ∂P, see Figure 10. We define these vertices as *pivot points*. We let ℓ_i be the line through $C_{y,i}$ and $C_{z,i}$. We define this line as the *pivot line* and consider the cases where g_i lies above or below ℓ_i.

If g_i is above ℓ_i, we define the shadow $S(a)$ of a point a as the region bounded by the lines $L(C_{y,i}, a)$ and $L(C_{z,i}, a)$ and above a, see Figure 11. If we let F denote the feasible region of the vertices of ∂P between y_i and z_i, one gets the following:

▶ **Lemma 14** (The feasible region is contained in the shadow). *Assume that F lies above ℓ_i and let a be a vertex of F closest to ℓ_i. Then $F \subseteq S(a)$.*

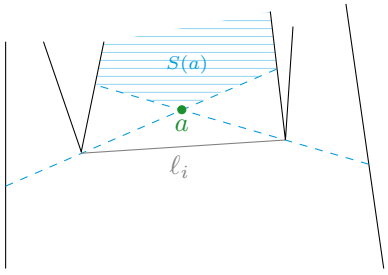


Figure 11 Placing a guard in the shadow $S(a)$ of a leads to a worse guard.

That is, when F lies above ℓ_i, there is only one best guard that guards this interval (we do not have the trade-off as in Figure 8). So we have:

▶ **Proposition 15** (Feasible region over repeated pivot line implies a progress condition)**.** *If $\ell_i = \ell_j$ for some $i < j$ and F lies above ℓ_i, then we get a progress condition.*

If the guard g_i lies below the pivot line ℓ_i, we can consider the guard that guards one of the pivot points and get the following result:

▶ **Proposition 16** (Guard placed below pivot implies other guard placed strictly above pivot)**.** *Let g_i lie below the pivot line, then one of the following will occur:*

1. *In the next revolution, we will see a progress condition.*
2. *There is some guard $\widehat{g}$ found in the next revolution, which is below its pivot line $\widehat{\ell}$.*

We can now use Proposition 15 and Proposition 16 to prove Theorem 13: There are $\mathcal{O}(k^*)$ greedy steps that take place around ∂P at a time, and each of them has $\mathcal{O}(n^2)$ possible pivot lines to consider. Each time we see a guard below a pivot line, it either gives a progress condition or forces another guard to lie above its pivot line. By the pigeonhole principle, after seeing $\mathcal{O}(k^* n^2)$ guards above their pivot lines, we must have seen a guard above the same pivot line twice, and Proposition 15 gives that we must have achieved a progress condition.

2.3 Proof of Theorem 6

Proof of Theorem 6. Running GreedyRevolution for $\mathcal{O}(kn^2) = \mathcal{O}(k^* n^2)$ revolutions guarantees a progress condition, i.e., a positive fingerprint, a repetition, or an edge jump by Theorem 13. Thus, repeating $n+1$ times guarantees at least one positive fingerprint, at least one repetition, or $n+1$ edge jumps. By Lemma 12 the greedy sequence will contain optimal revolutions. Finally, it can be proven that once an optimal solution appears in the greedy sequence, all subsequent revolutions will also be optimal. This requires $\mathcal{O}(k^* n^2 \cdot (n+1) \cdot n^2 \log n) = \mathcal{O}(n^5 k^* \log n)$ arithmetic operations. ◀

An open source implementation using CGAL [34, 36, 39, 70, 75] in C++ is available online [68].

3 Simplifying the Domain of GreedyInterval

In this section we present a polynomial-time exact algorithm for contiguous guarding of the boundary ∂P of a simple polygon P. Recall the notation from Figure 5 in subsection 1.2. We assume that P is not star-shaped, so at least 2 guards are needed.

Without loss of generality, we assume this maximality property: *any guard g covers a maximal contiguous portion of ∂P, i.e., $\partial(g)$ is maximal.* This implies the following lemma.

▶ **Lemma 17.** *Let g be a guard in a guarding of ∂P such that $\partial(g)$ is maximal. Then, each boundary ray of $w(g)$ goes through at least one reflex vertex of P.*

Algorithm (in a nutshell)**.** Our algorithm is fairly simple. It finds a polynomial-size set S of starting points on ∂P such that at least one of the points is covered by two guards in some optimal solution. We execute the greedy algorithm for each point in S and return the smallest guard set over all the points in S. From Theorem 5, it follows that the algorithm returns an optimal solution. The algorithm takes polynomial time because S has a polynomial size, and the greedy algorithm runs in polynomial time.

It remains to compute S. This is the most crucial part of the algorithm. First, we find a polynomial-size point set Q such that at least one guard in some optimal solution lies at a point of Q. For each point $q \in Q$, we compute a set $S(q)$ of all potential first endpoints of $\partial(q)$ for a guard at q. By Lemma 17, these points are the intersections of ∂P with the rays

that start from q and pass through reflex vertices of P. Thus, for every reflex vertex r that is visible from q, we add the first intersection point of the ray $\overrightarrow{qr}$ with ∂P to $S(q)$. We then define the set S as the union of $S(q)$ over all points $q \in Q$. The set S has a polynomial size because $|Q|$ and the number of reflex vertices are bounded by polynomials.

Let g^* be a guard in an optimal solution Γ^* that lies at a point $q \in Q$. Then the first endpoint f^* of $\partial(g^*)$ is in $F(q)$, and hence in S. By Observation 1, f^* is covered by g^* and the previous guard in Γ^*. Therefore, f^* is a point in S with our desired property of being covered by two guards in an optimal solution.

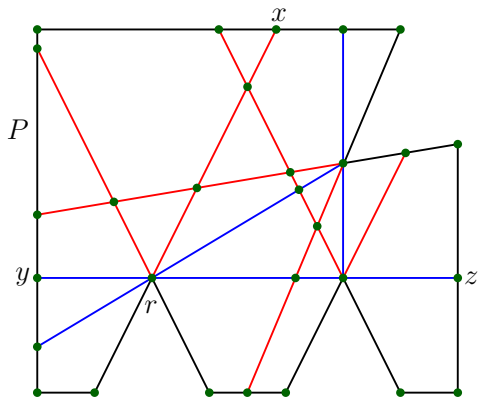


Figure 12 Edge-extensions are in red and vertex-extensions are in blue. The points in Q are represented by small green disks.

Now it remains to compute Q. For every reflex vertex r of P, extend each edge incident to r inside P until hitting ∂P at some point x; see Figure 12. We refer to the segment rx as an *edge-extension*. Add a segment between every two reflex vertices of P that are visible to each other and extend it from both endpoints inside P until hitting ∂P at points y and z. We refer to the segment yz by *vertex-extension*. An *extension* is either an edge-extension or a vertex-extension. We define Q as the set containing the following points

- all vertices of P,
- all intersection points between extensions and ∂P,
- all intersection points between edge-extensions,
- all intersection points between an edge-extension and a vertex-extension.

The following structural lemma shows that Q has our desired property.

▶ **Lemma 18.** *Some optimal solution has a guard at a point of Q.*

If P has n vertices, then Q has $O(n^3)$ points because there are $O(n)$ edge-extensions and $O(n^2)$ vertex-extensions. The set S has size $O(n^4)$. Recall that the greedy algorithm takes $O(n^2 \log n)$ time for a given starting point. Therefore, the total running time of the optimal algorithm, which invokes the greedy algorithm from each starting point in S, is $O(n^6 \log n)$. This is a conservative upper bound on the running time and surely can be improved. For example, one might adopt ideas from the $O(n)$-time incremental algorithm for the kernel of a polygon [49] to achieve a better running time for the greedy algorithm. Our main goal here is a proof that the decision version of the contiguous boundary guarding problem is in the complexity class **P**. The following theorem summarizes our result.

▶ **Theorem 19.** *The problem of contiguous boundary guarding of a polygon with the minimum number of guards can be solved in polynomial time.*

Remark. One might wonder if Lemma 18 holds for every optimal solution, i.e., whether every optimal solution has a guard in Q. Also, one might wonder whether there always exists an optimal solution in which two guards cover the same vertex of the polygon. If this were true, then running the greedy algorithm from all vertices of the polygon would suffice. However, none of these statements are true as shown in Figure 13. (For the second statement, one can verify that no guard can cover a red cross and a blue cross simultaneously. Therefore, any optimal solution must have a guard in the red triangle and a guard in the blue triangle. Then, the contiguous coverages of such guards do not have any vertex in common.)

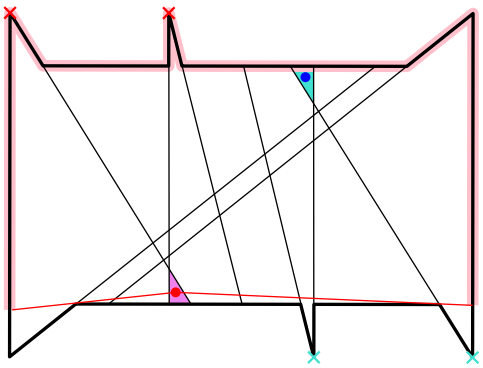


Figure 13 An optimal solution, with two guards, none of which is in Q. The coverages of no two guards, in any optimal solution, share a vertex of the polygon.

4 The Analytic Arc Cover Problem

The approach of Robson, Spalding-Jamieson, and Zheng [62] for the contiguous art gallery problem was to consider the *analytic arc cover problem*. This problem is a version of the interval cover problem on a circle with infinite number of intervals. Each interval is described by the counterclockwise segment between two points a and b on the circle S^1, denoted by $[a, b]$. This infinite set of intervals is given implicitly as a function, as can be seen in the following definition.

▶ **Definition 20.** *Let $G : S^1 \to S^1$ be a function that maps points on the unit circle S^1 to other points on the unit circle S^1. The* analytic arc cover problem *asks to find the minimum set $X \subset S^1$ such that the set of counter-clockwise arcs $\{[x, G(x)] : x \in X\}$ covers S^1.*

Note that we require some additional properties for this problem to be algorithmically solvable and avoid pathalogical examples (e.g. that there exists a finite cover). We leave these details to the full version of this approach [62].

We show that the contiguous boundary art gallery problem reduces to analytic arc cover with piecewise linear-rational functions over a unit-interval representation, i.e., the function G is a piecewise-continuous function that is decomposable into linear rational functions $f_1, f_2, \ldots, f_\ell$. See Figure 14 for an illustration of this. We crucially use the fact that a compositions of two linear rational functions yields another linear rational function. Using this fact, we show that these instances of the analytic arc cover problem are solvable in polynomial time. In addition, we show that minimizing the number of half-plane cuts to carve a 3D polytope reduces to multiple instances of segment separation. We can conclude that finding the minimum cuts to carve a polytope is also solvable in polynomial time.

▶ **Theorem 21.** *Minimizing the number of half-plane cuts to carve a 3D polytope is in* P.

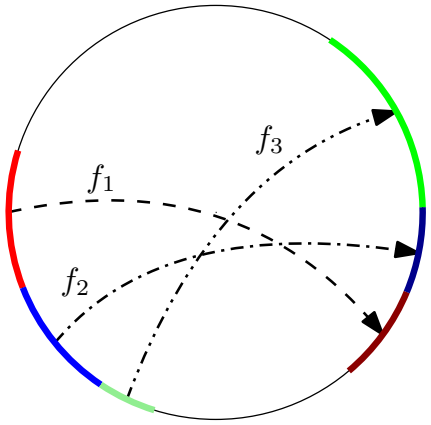


Figure 14 A mapping between parts of the unit circle where the functions f_1, f_2, and f_3 are linear rational functions.

4.1 Outline of approach

At a high level, we show that for instances of the analytic arc cover problem where we can compute a greedy function G that is piecewise linear, then the problem can be solved in polynomial time.

Analytic Arc Cover

We show that the analytic arc cover problem is solvable in polynomial time when the input function G is a piecewise linear rational function with rational coefficients.

▶ **Observation 22.** *The following is true of piecewise linear rational functions.*

- *For two piecewise linear rational functions G_1 and G_2, where G_1 consists of n_1 pieces and G_2 has n_2 pieces, the composition $G = G_1 \circ G_2$ is a piecewise linear rational function of $n_1 + n_2$ parts.*
- *For a piecewise linear rational function $G : S^1 \to S^1$ that consists of n pieces, it is possible to determine if the function "wraps around" in $O(n)$ time.*

The formal definition of "wrapping around" the circle is omitted for brevity (this requires lifting to a covering space), and is given in full detail in [63]. Using these observations, it is straightforward to prove the following theorem.

▶ **Theorem 23.** *Let $G : S^1 \to S^1$ be a piecewise linear rational function. Let I^* be a solution to the analytic arc cover problem that consists of k arcs. Then the analytic arc cover problem on can be solved in $O(nk)$ time.*

Proof. Let $\mathrm{G}^{(j)}$ denote the function $\widetilde{G}$ composed with itself j times, so $\mathrm{G}^{(j)} = \mathrm{G} \circ \mathrm{G}^{(j-1)}$. Thus we can compute $\mathrm{G}^{(j)}$ for each i via function compositions. For each $\mathrm{G}^{(j)}$ we can determine if there is a solution of size j, and halt if so. Note that $\mathrm{G}^{(j)}$ is a piecewise linear rational function with jn pieces. This algorithm terminates after $k-1$ function compositions, taking a total of $O(nk)$ time. ◀

Contiguous Art Gallery

We reduce the contiguous art gallery problem to the analytic arc cover problem for the greedy function G which is piecewise linear rational. (See Figure 8 for an illustration of this).

To prove this (in a constructive manner), we give a series of piecewise linear rational functions mapping an individual "input" segment of the polygon to an "output" segment of the polygon. The final in each series are combined to form one piecewise linear rational function for the entire polygon. Each function in the sequence leverages the last, so that each one incorporates more elements of the problem constraining the final function in various ways. In particular, the constraints that are progressively introduced are:

- The input and output segments themselves constraint the possible guard locations.
- The segments between the input and output segment must be guarded and thus constrain the possible guard locations.
- The portion of the polygon *not* being guarded constrains the possible guard locations.

In the full version [62], the functions presented in "most-constraining" to "least-constraining" order. At each step, functions for different subsets of constraints can be combined with min and max operators, which preserve the piecewise linear rational structure. Some care is needed for this part to ensure that the number of pieces is polynomial, and that the endpoints attained in the algorithm do not have exponential bit complexity.

Polygon Separation and Polytope Carving

Similarly, we can reduce the problem of segment separation, where we wish to separate two sets of line segments to an instance of analytic arc cover with a piecewise rational function. The function is in some sense less complex than the one for the contiguous art gallery problem, in that it requires fewer steps, but we make use of a special case of piecewise linear rational functions over two variables instead of one.

We also show that the problem of determining the minimum number of cuts to cut out a 3D polytope reduces to a polynomial number of instances of segment separation, and hence this problem can also be solved in strongly polynomial time.

5 Future work

In this paper, we have presented an $\mathcal{O}(n^5 k^* \log n)$ solution to the contiguous art gallery problem. We believe that the solution using repeated GreedyRevolution can find an optimal solution in as few as $\mathcal{O}(\text{polylog}\, n)$ revolutions, maybe only constantly many, as our experiments never needed more than four revolutions. Furthermore, we believe that GreedyRevolution can be implemented to run in $\mathcal{O}(n \log n)$ time; we are unsure if, in fact, linear time may be possible. In total, a final runtime of $\mathcal{O}(n\, \text{polylog}\, n)$ should be possible. Variants of the contiguous art gallery problem can be considered. For instance, 3D contiguous art gallery problem or higher-dimensional contiguous art gallery problem, or variants where each guard can see at most m contiguous intervals. We believe that all these variants are NP-hard.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG)*, pages 3:1–3:15, 2017. doi:10.4230/LIPICS.SOCG.2017.3.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. 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 65–73. ACM, 2018. doi:10.1145/3188745.3188868.

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 and Bartosz Walczak. Common tangents of two disjoint polygons in linear time and constant workspace. *ACM Trans. Algorithms*, 15(1), December 2018. doi:10.1145/3284355.

5 Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects*. PhD thesis, The Johns Hopkins University, 1984.

6 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee-Keng Yap. Finding minimal convex nested polygons. In Joseph O'Rourke, editor, *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 296–304. ACM, 1985. doi:10.1145/323233.323271.

7 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee-Keng Yap. Finding minimal convex nested polygons. *Inf. Comput.*, 83(1):98–110, 1989. doi:10.1016/0890-5401(89)90049-7.

8 Reymond Akpanya, Bastien Rivier, and Frederick B. Stock. Open problems CCCG 2024. In *Proceedings of the 36th Canadian Conference on Computational Geometry*, pages 167–170, 2024.

9 Yoav Amit, Joseph S. B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. In *Proc. Ninth Workshop on Algorithm Engineering and Experiments*, Proceedings in Applied Mathematics, pages 120–134. SIAM, 2007. doi:10.1137/1.9781611972870.12.

10 Yoav Amit, Joseph S. B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications*, 20(5):601–630, October 2010. doi:10.1142/S0218195910003451.

11 Stav Ashur, Omrit Filtser, Matthew J. Katz, and Rachel Saban. Terrain-like graphs: Ptass for guarding weakly-visible polygons and terrains. In Evripidis Bampis and Nicole Megow, editors, *Approximation and Online Algorithms - 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*, volume 11926 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2019. doi:10.1007/978-3-030-39479-0_1.

12 Stav Ashur, Omrit Filtser, Matthew J Katz, and Rachel Saban. Terrain-like graphs: PTASs for guarding weakly-visible polygons and terrains. *Computational Geometry*, 101:101832, 2022. doi:10.1016/J.COMGEO.2021.101832.

13 Lichen Bao, Sergey Bereg, Ovidiu Daescu, Simeon C. Ntafos, and Junqiang Zhou. On some city guarding problems. In *Proceedings of the 14th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 600–610. Springer, 2008. doi:10.1007/978-3-540-69733-6_59.

14 Pritam Bhattacharya, Subir Kumar Ghosh, and Bodhayan Roy. Approximability of guarding weak visibility polygons. *Discrete Applied Mathematics*, 228:109–129, 2017. doi:10.1016/J.DAM.2016.12.015.

15 Ahmad Biniaz. Art galleries and mobile guards: Revisiting O'rourke's proof. In *Proceedings of the 32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 27:1–27:4, 2024. doi:10.4230/LIPICS.ESA.2024.27.

16 Ahmad Biniaz and Mohammad Hashemi. City guarding with cameras of bounded field of view. In *Proceedings of the 35th Canadian Conference on Computational Geometry (CCCG)*, pages 71–76, 2023.

17 Ahmad Biniaz, Anil Maheshwari, Joseph S. B. Mitchell, Saeed Odak, Valentin Polishchuk, and Thomas C. Shermer. Contiguous boundary guarding, 2024. doi:10.48550/arXiv.2412.15053.

18 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *33rd International Symposium on Computational Geometry (SoCG)*, volume 77 of *LIPIcs*, pages 20:1–20:15, 2017. doi:10.4230/LIPICS.SOCG.2017.20.

19 Danny Z. Chen, Vladimir Estivill-Castro, and Jorge Urrutia. Optimal guarding of polygons and monotone chains. In *Proceedings of the 7th Canadian Conference on Computational Geometry, (CCCG)*, pages 133–138, 1995. URL: http://www.cccg.ca/proceedings/1995/cccg1995_0022.pdf.

20 Vasek Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

21 Ovidiu Daescu, Stephan Friedrichs, Hemant Malik, Valentin Polishchuk, and Christiane Schmidt. Altitude terrain guarding and guarding uni-monotone polygons. *Computational Geometry*, 84:22–35, 2019. doi:10.1016/J.COMGEO.2019.07.004.

22 Ovidiu Daescu and Jun Luo. Cutting out polygons with lines and rays. In Rudolf Fleischer and Gerhard Trippen, editors, *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*, volume 3341 of *Lecture Notes in Computer Science*, pages 669–680. Springer, 2004. doi:10.1007/978-3-540-30551-4_58.

23 Ovidiu Daescu and Jun Luo. Cutting out polygons with lines and rays. *Int. J. Comput. Geom. Appl.*, 16(2-3):227–248, 2006. doi:10.1142/S0218195906002014.

24 Ovidiu Daescu and Hemant Malik. New bounds on guarding problems for orthogonal polygons in the plane using vertex guards with halfplane vision. *Theor. Comput. Sci.*, 882:63–76, 2021. doi:10.1016/J.TCS.2021.06.012.

25 Erik D. Demaine, Martin L. Demaine, and Craig S. Kaplan. Polygons cuttable by a circular saw. In *Proceedings of the 12th Canadian Conference on Computational Geometry, Fredericton, New Brunswick, Canada, August 16-19, 2000*, 2000. URL: http://www.cccg.ca/proceedings/2000/36.ps.gz.

26 Erik D. Demaine, Martin L. Demaine, and Craig S. Kaplan. Polygons cuttable by a circular saw. *Comput. Geom.*, 20(1-2):69–84, 2001. doi:10.1016/S0925-7721(01)00036-0.

27 Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time $O(\log n)$-approximation algorithm for art gallery problems. In *Proceedings of the 10th International Workshop on Algorithms and Data Structures (WADS)*, volume 4619, pages 163–174, 2007. doi:10.1007/978-3-540-73951-7_15.

28 Adrian Dumitrescu and Masud Hasan. Cutting out polygons with a circular saw. In Takao Asano, Shin-Ichi Nakano, Yoshio Okamoto, and Osamu Watanabe, editors, *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, volume 7074 of *Lecture Notes in Computer Science*, pages 230–239. Springer, 2011. doi:10.1007/978-3-642-25591-5_25.

29 Adrian Dumitrescu and Masud Hasan. Cutting out polygons with a circular saw. *Int. J. Comput. Geom. Appl.*, 23(2):127–140, 2013. doi:10.1142/S0218195913600030.

30 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. In Ricardo A. Baeza-Yates, Ugo Montanari, and Nicola Santoro, editors, *Foundations of Information Technology in the Era of Networking and Mobile Computing, IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science (TCS 2002), August 25-30, 2002, Montréal, Québec, Canada*, volume 223 of *IFIP Conference Proceedings*, pages 181–192. Kluwer, 2002. doi:10.1007/978-0-387-35608-2_16.

31 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/J.IPL.2006.05.014.

32 Stephan J. Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001. doi:10.1007/S00453-001-0040-8.

33 H El Gindy and D Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981. doi:10.1016/0196-6774(81)90019-5.

34 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, 2000. doi:10.1002/1097-024X(200009)30:11\%3C1167::AID-SPE337\%3E3.0.CO;2-B.

35 Steve Fisk. A short proof of chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978. doi:10.1016/0095-8956(78)90059-X.

36 Efi Fogel, Ophir Setter, Ron Wein, Guy Zucker, Baruch Zukerman, and Dan Halperin. 2D regularized boolean set-operations. 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#PkgBooleanSetOperations2.

37 Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. *Discret. Appl. Math.*, 158(6):718–722, 2010. Also in Canad. Information Processing Soc. Congress 1987. doi:10.1016/J.DAM.2009.12.004.

38 Matt Gibson, Gaurav Kanade, Erik A. Krohn, and Kasturi R. Varadarajan. Guarding terrains via local search. *J. Comput. Geom.*, 5(1):168–178, 2014. doi:10.20382/JOCG.V5I1A9.

39 Michael Hemmer, Kan Huang, Francisc Bungiu, and Ning Xu. 2D visibility computation. 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#PkgVisibility2.

40 Jeff Kahn, Maria Klawe, and Daniel Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal on Algebraic Discrete Methods*, 4(2):194–206, 1983.

41 Byeonguk Kang, Junhyeok Choi, Jeesun Han, and Hee-Kap Ahn. Guarding points on a terrain by watchtowers. In *Proceedings of the 36th Canadian Conference on Computational Geometry*, pages 41–47, 2024.

42 Matthew J. Katz and Gabriel S. Roisman. On guarding the vertices of rectilinear domains. *Comput. Geom.*, 39(3):219–228, 2008. doi:10.1016/J.COMGEO.2007.02.002.

43 James King and David G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discret. Comput. Geom.*, 46(2):252–269, 2011. doi:10.1007/S00454-011-9352-X.

44 James King and Erik Krohn. Terrain guarding is np-hard. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1580–1593. SIAM, 2010. doi:10.1137/1.9781611973075.128.

45 James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM J. Comput.*, 40(5):1316–1339, 2011. doi:10.1137/100791506.

46 Erik A. Krohn and Bengt J Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66:564–594, 2013. doi:10.1007/S00453-012-9653-3.

47 C.C. Lee and D.T. Lee. On a circle-cover minimization problem. *Information Processing Letters*, 18(2):109–115, 1984. doi:10.1016/0020-0190(84)90033-4.

48 D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theory*, 32(2):276–282, 1986. doi:10.1109/TIT.1986.1057165.

49 D. T. Lee and Franco P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, 1979. doi:10.1145/322139.322142.

50 Anna Lubiw. Decomposing polygonal regions into convex quadrilaterals. In *Proceedings of the first annual symposium on Computational geometry*, pages 97–106, 1985. doi:10.1145/323233.323247.

51 Francisco Martínez, Antonio Jesús Rueda, and Francisco Ramón Feito. A new algorithm for computing boolean operations on polygons. *Computers & Geosciences*, 35(6):1177–1185, 2009. doi:10.1016/j.cageo.2008.08.009.

52 Lucas Meijer and Tillmann Miltzow. Sometimes two irrational guards are needed. *ArXiv*, abs/2212.01211, 2022. doi:10.48550/arXiv.2212.01211.

53 Magnus Christian Ring Merrild, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou, and Rolf Svenning. The contiguous art gallery problem is solvable in polynomial time, 2024. doi:10.48550/arXiv.2412.13938.

54 Joseph S. B. Mitchell and Subhash Suri. Separation and approximation of polyhedral objects. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '92, pages 296–306, USA, 1992. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=139404.139467.

55 Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. In Herbert Edelsbrunner, editor, *Proceedings of the Fourth Annual Symposium on Computational Geometry, Urbana-Champaign, IL, USA, June 6-8, 1988*, pages 211–223. ACM, 1988. doi:10.1145/73393.73415.

56 Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. Syst. Sci.*, 40(1):19–48, 1990. doi:10.1016/0022-0000(90)90017-F.

57 Joseph O'Rourke. Galleries need fewer mobile guards: A variation on Chvátal's theorem. *Geometriae Dedicata*, 14:273–283, 1983.

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

59 Joseph O'Rourke, Thomas C. Shermer, and Ileana Streinu. Illuminating convex polygons with vertex floodlight. In *Proceedings of the 7th Canadian Conference on Computational Geometry (CCCG)*, pages 151–156, 1995. URL: `http://www.cccg.ca/proceedings/1995/cccg1995_0025.pdf`.

60 Joseph O'Rourke and Kenneth Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–190, 1983. `doi:10.1109/TIT.1983.1056648`.

61 Mark H. Overmars and Emo Welzl. The complexity of cutting paper (extended abstract). In *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 316–321. ACM, 1985. `doi:10.1145/323233.323274`.

62 Eliot W. Robson, Jack Spalding-Jamieson, and Da Wei Zheng. The analytic arc cover problem and its applications to contiguous art gallery, polygon separation, and shape carving, 2024. `doi:10.48550/arXiv.2412.15567`.

63 Eliot W. Robson, Jack Spalding-Jamieson, and Da Wei Zheng. Carving polytopes with saws in 3d. In Rahnuma Islam Nisha, editor, *Proceedings of the 36th Canadian Conference on Computational Geometry, CCCG 2024, Brock University, St. Catharines, Ontario, Canada, July 17 - August 19, 2024*, pages 145–152, 2024.

64 Jörg-Rüdiger Sack and Godfried T. Toussaint. Guard placement in rectilinear polygons. In *Machine Intelligence and Pattern Recognition*, volume 6, pages 153–175. Elsevier, 1988.

65 Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Math. Log. Q.*, 41:261–267, 1995. `doi:10.1002/MALQ.19950410212`.

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

67 Jack Stade. The point-boundary art gallery problem is $\exists\mathbb{R}$-hard. *Symposium on Computational Geometry*, 2025.

68 Rolf Svenning. RolfSvenning/ContiguousArtGallery. Software, swhId: `swh:1:dir:7cfaba2c09d953feb90a49f0e26370ea3f7719a7` (visited on 2025-04-24). URL: `https://github.com/RolfSvenning/ContiguousArtGallery`, `doi:10.4230/artifacts.23018`.

69 Xuehou Tan. Approximation algorithms for cutting out polygons with lines and rays. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, volume 3595 of *Lecture Notes in Computer Science*, pages 534–543. Springer, 2005. `doi:10.1007/11533719_54`.

70 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.0.1 edition, 2024. URL: `https://doc.cgal.org/6.0.1/Manual/packages.html`.

71 Csaba D Tóth. Art gallery problem with guards whose range of vision is 180. *Computational Geometry*, 17(3-4):121–134, 2000. `doi:10.1016/S0925-7721(00)00023-7`.

72 Csaba D Tóth. Art galleries with guards of uniform range of vision. *Computational Geometry*, 21(3):185–192, 2002. `doi:10.1016/S0925-7721(01)00024-4`.

73 Csaba D. Tóth. Illumination of polygons by 45°-floodlights. *Discret. Math.*, 265(1-3):251–260, 2003. `doi:10.1016/S0012-365X(02)00583-6`.

74 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`.

75 Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. 2D arrangements. 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#PkgArrangementOnSurface2`.

76 Chris Worman and J Mark Keil. Polygon decomposition and the orthogonal art gallery problem. *International Journal of Computational Geometry & Applications*, 17(02):105–138, 2007. `doi:10.1142/S0218195907002264`.

Structure and Independence in Hyperbolic Uniform Disk Graphs

Thomas Bläsius
Karlsruhe Institute of Technology, Germany

Jean-Pierre von der Heydt
Karlsruhe Institute of Technology, Germany

Sándor Kisfaludi-Bak
Aalto University, Espoo, Finland

Marcus Wilhelm
Karlsruhe Institute of Technology, Germany

Geert van Wordragen
Aalto University, Espoo, Finland

Abstract

We consider intersection graphs of disks of radius r in the hyperbolic plane. Unlike the Euclidean setting, these graph classes are different for different values of r, where very small r corresponds to an almost-Euclidean setting and $r \in \Omega(\log n)$ corresponds to a firmly hyperbolic setting. We observe that larger values of r create simpler graph classes, at least in terms of separators and the computational complexity of the Independent Set problem.

First, we show that intersection graphs of disks of radius r in the hyperbolic plane can be separated with $\mathcal{O}((1 + 1/r) \log n)$ cliques in a balanced manner. Our second structural insight concerns Delaunay complexes in the hyperbolic plane and may be of independent interest. We show that for any set S of n points with pairwise distance at least $2r$ in the hyperbolic plane, the corresponding Delaunay complex has outerplanarity $1 + \mathcal{O}(\frac{\log n}{r})$, which implies a similar bound on the balanced separators and treewidth of such Delaunay complexes.

Using this outerplanarity (and treewidth) bound we prove that Independent Set can be solved in $n^{\mathcal{O}(1+\frac{\log n}{r})}$ time. The algorithm is based on dynamic programming on some unknown sphere cut decomposition that is based on the solution. The resulting algorithm is a far-reaching generalization of a result of Kisfaludi-Bak (SODA 2020), and it is tight under the Exponential Time Hypothesis. In particular, Independent Set is polynomial-time solvable in the firmly hyperbolic setting of $r \in \Omega(\log n)$. Finally, in the case when the disks have ply (depth) at most ℓ, we give a PTAS for Maximum Independent Set that has only quasi-polynomial dependence on $1/\varepsilon$ and ℓ. Our PTAS is a further generalization of our exact algorithm.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Computational geometry; Theory of computation → Graph algorithms analysis

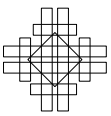
Keywords and phrases hyperbolic geometry, unit disk graphs, independent set, treewidth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.21

Related Version *Full Version*: `https://arxiv.org/abs/2407.09362` [9]

Funding *Jean-Pierre von der Heydt*: Funded by the pilot program Core-Informatics of the Helmholtz Association (HGF).
Sándor Kisfaludi-Bak: Supported by the Research Council of Finland, Grant 363444.
Marcus Wilhelm: funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 524989715.

© Thomas Bläsius, Jean-Pierre von der Heydt, Sándor Kisfaludi-Bak, Marcus Wilhelm, and Geert van Wordragen;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Given a set of disks in the plane, one can assign to them a *geometric intersection graph* whose vertices are the disks, and edges are added between pairs of intersecting disks. The study of intersection graphs is usually motivated by physically realized networks: such networks require spatial proximity between vertices for successful connections, and the simplest model allows for connections within a distance 2, which is equivalent to the intersection graph induced by disks of radius 1 centered at the vertices. Such graphs are called *unit disk graphs*.

Unit disk graphs have received a lot of attention in the theoretical computer science literature. Their intriguing structural properties yield profound mathematical insights and facilitate the development of efficient algorithms. Historically, unit disk graphs have been studied in the Euclidean plane, where they are well motivated due to their relevance for, e.g., sensor networks. The relevance of unit disk graphs in the hyperbolic plane is less obvious. However, originating in the network science community, it has been observed [26] that the intersection graph of randomly sampled disks of equal radius r yields graphs that resemble complex real-world networks in regards to important properties[1]. They are, e.g., heterogeneous with a degree distribution following a power law [19], have high clustering coefficient [19], and exhibit the small-world property [18, 35].

Besides numerous structural results, these hyperbolic random graphs also allow for the design of more efficient algorithms [7, 8, 4, 5]. However, all these results rely on the fact that the disks are chosen randomly. So far, there is only little research on hyperbolic uniform disk graphs from a deterministic, more graph-theoretic perspective. It is important to note that for the intersection graphs of radius-r disks the resulting graph classes are different for different values of r. Moreover, it is natural to allow for the radius r to be a (monotone) function of the number of vertices. Unlike the Euclidean setting where a simple scaling shows that the choice of "unit" does not matter, it is a crucial parameter in the hyperbolic setting.

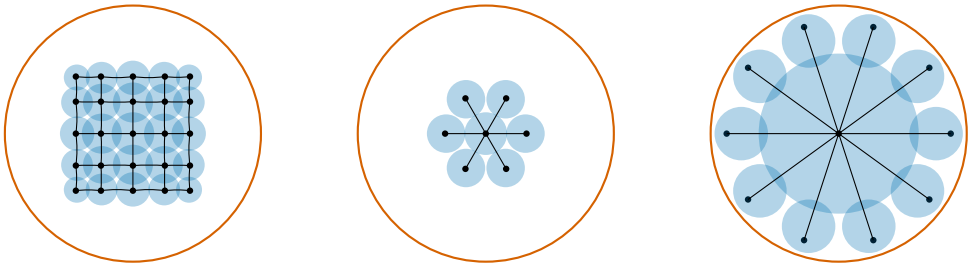
For those unfamiliar with hyperbolic geometry there is a way to conceptually understand these graphs in a Euclidean setting as follows. One can think of a hyperbolic disk graph of radius r disks as a Euclidean disk graph where the radius of a disk of center p is set to $f_r(\text{dist}(o,p))$, where f_r is decreasing very quickly at a rate set by r, and $\text{dist}(o,p)$ is the distance between the origin and the disk center p. In particular, as r goes to 0, the rate of decrease is negligible, and we get almost Euclidean unit disk graphs, while large r corresponds to a very different graph class. See Figure 1 for an illustration.

This observation is formalized by stating that hyperbolic disk graphs of uniform (that is, equal) radius $1/n^3$ and uniform radius $\log n$ are incomparable: The former class contains the $k \times k$ grid graph with $n = k^2$ for all k, but does not contain stars of size $n \geq 8$. The latter contains all star graphs but does not contain any $k \times k$ grid of size $k \geq 5$. See Figure 1 for an illustration and the full version of this paper for a formal proof of these claims. We call the regime $r \leq 1/\sqrt{n}$ *almost Euclidean*, while $r \in \Omega(\log n)$ is called *firmly hyperbolic*.

The primary goal of our paper is to explore the following:

How does the structure of hyperbolic uniform disk graphs depend on the radius $r = r(n)$?

[1] For this to work, the choice of the radius r is crucial. It is chosen as $r = \log n + c$ for a constant c that controls the average degree, and the disk centers are all sampled within a disk of radius $2r$.



(a) Smaller r enables larger grids. **(b)** Only small stars for small r. **(c)** Larger r enables larger stars.

Figure 1 Realizing a grid is only possible for small radii, while large stars are only possible in the firmly hyperbolic setting of $r \in \Omega(\log n)$.

We note that one of the main challenges for large radii is that unlike the Euclidean plane, the kissing number[2] of hyperbolic disks of large radius is unbounded, and as a result, there may be arbitrarily many "nearby disks" to any given disks in the independent set. At the same time the larger the hyperbolic disks are, the harder it is to "surround" them with equal radius disks, which intuitively leads to a simpler structure.

The case of constant radius hyperbolic disks (with constant kissing number) has been studied by Kisfaludi-Bak [23], but these methods fail for super-constant kissing number. The case of radius roughly $\log n$ has only been studied in a more specialized setting (inspired by hyperbolic random graph models) in [6]. They call the relevant graph class *strongly hyperbolic uniform disk graphs*. The above-mentioned hyperbolic random graphs are randomly sampled strongly hyperbolic uniform disk graphs. Apart from these particular choices of the radius, we have very limited understanding about the structure of hyperbolic uniform disk graphs, and the results of [6] do not imply anything for our hyperbolic uniform disk graphs.

The radius' impact on problem complexity was studied in the context of the traveling salesman problem, where one can use the minimum distance between points as a parameter with a similar flavor. In the hyperbolic plane, Kisfaludi-Bak [24] showed that the complexity of TSP decreases as one increases the minimum pairwise distance r among input points. In this paper, we show that the structure of hyperbolic uniform disk graphs behaves similarly: their structure becomes easier for larger values of r, and Independent Set can be solved faster for larger r.

Let $\mathrm{HUDG}(r(n))$ denote the set of intersection graphs where each n-vertex graph can be realized as the intersection graph of disks of uniform (equal) radius $r(n)$ in the hyperbolic plane of Gaussian curvature[3] -1. We denote by HUDG the union of these classes for all radii, that is, $\mathrm{HUDG} = \bigcup_{\rho>0} \mathrm{HUDG}(\rho)$. Let UDG denote the class of unit disk graphs in the Euclidean plane. The authors in [6] have shown that $\mathrm{UDG} \subset \mathrm{HUDG}$. When a graph is given without the geometric realization, it is NP-hard (even $\exists\mathbb{R}$-complete) to decide if the graph is a UDG [33] or HUDG [3]; for this reason, we will assume throughout this article that the input intersection graphs of our algorithms are given by specifying their geometric realization. In the hyperbolic setting, one can specify the disk centers using a so-called *model* of the hyperbolic plane, which is simply an embedding of the hyperbolic plane into some specific part of the Euclidean plane, e.g., inside the Euclidean unit disk. See our preliminaries for a brief introduction to Euclidean models of the hyperbolic plane.

[2] The kissing number of radius r disks is the maximum number of interior-disjoint disks of radius r that can touch a fixed disk of radius r.

[3] Equivalently, one can fix the radius to be 1 and set the Gaussian curvature to be $-r^2$.

From the perspective of graph algorithms, unit disk graphs in the Euclidean plane have been serving an important role as a graph class that is not comparable, but similar to planar graphs. The circle packing theorem [25] states that every planar graph can be realized as an intersection graph of disks (of arbitrary radii); thus, disk graphs serve as a common generalization of planar and unit disk graphs. Using *conformal* hyperbolic models, one can observe that hyperbolic disks appear as Euclidean disks[4] in the model, which means that all HUDGs are realized as Euclidean disk graphs. Denoting Euclidean disk graphs with DG, we thus have UDG $\subseteq$ HUDG $\subseteq$ DG[5].

A key structural tool in the study of both planar and (unit) disk graphs has been *separator theorems*. By a result of Lipton and Tarjan[27], any n-vertex planar graph can be partitioned into three vertex sets, A, B, and the separator S, such that no edges go between A and B, the separator set S has size $\mathcal{O}(\sqrt{n})$, and $\max\{|A|, |B|\} \leq \frac{2}{3}n$. Separator theorems are closely related to the *treewidth*[6] of graphs [10], for example, the above separator implies a treewidth bound of $\mathcal{O}(\sqrt{n})$ on planar graphs.

For (unit) disk graphs, similar separators and treewidth bounds are not possible, because one can represent cliques of arbitrary size. There have been at least three different approaches to deal with large cliques. The first option is to bound the cliques in some way. One can, for example, obtain separators for a set of disks of bounded *ply*, i.e., assuming that each point of the ambient space is included in at most ℓ disks. There are several separators for disks (and even balls in higher dimensions) that involve ply. The strongest and most general among these is by Miller, Teng, Thurston, and Vavasis [34]. The second way to deal with large cliques is to use so-called clique-based separators [15, 16], where the separator is decomposed into cliques and the cliques are sometimes assigned some small weight depending on their size. In the hyperbolic setting, Kisfaludi-Bak [23] showed that for any constant c, the graph class HUDG(c) admits a balanced separator consisting of $\mathcal{O}(\log n)$ cliques. The same paper shows that HUDG(c) also has clique-based separators of weight $\mathcal{O}(\log^2 n)$ and extends these techniques to balls of constant radius in higher-dimensional hyperbolic spaces, along with much of the machinery of [15]. The third option is to use random disk positions to disperse large cliques with high probability. Bläsius, Friedrich, and Krohmer [7] gave high-probability bounds on separators and treewidth for certain radius regimes in hyperbolic random graphs.

The above separators and treewidth can be used to obtain fast divide-and-conquer or dynamic programming algorithms [28, 15]. In the Independent Set problem, the goal is to decide if there are k pairwise non-adjacent vertices in the graph. In general graphs, the best known algorithms run in $2^{\mathcal{O}(n)}$ or $n^{\mathcal{O}(k)}$ time, and these running times are optimal under the Exponential Time Hypothesis (ETH) [14] (see [22] for the definition of ETH). In planar and disk graphs however, the separators allow algorithms with running time $2^{\mathcal{O}(\sqrt{n})}$ or $n^{\mathcal{O}(\sqrt{k})}$ [14, 32]. Moreover, the separator implies that planar graphs have treewidth[6] $\mathcal{O}(\sqrt{n})$ and unit disk graphs have so-called $\mathcal{P}$*-flattened treewidth* $\mathcal{O}(\sqrt{n})$ for some clique-partition $\mathcal{P}$. As a consequence, one can adapt treewidth-based algorithms [14] to these graph classes. In the hyperbolic setting, the ($\mathcal{P}$-flattened) treewidth bounds yield an $n^{\mathcal{O}(\log n)}$ quasi-polynomial algorithm for Independent Set and many other problems on graphs in HUDG(c) for any constant c, which is matched by a lower bound under ETH [23] in case of Independent Set.

[4] In the Poincaré disk and half-plane models, all hyperbolic disks are Euclidean disks, but the Euclidean and hyperbolic radii of these disks are different.
[5] With a little effort, one can show that both containments are strict: UDG $\subsetneq$ HUDG $\subsetneq$ DG.
[6] See the full version of this paper for definitions of treewidth and $\mathcal{P}$-flattened treewidth.

When approximating Independent Set, the main algorithmic tool in planar graphs is Baker's shifting technique [2] that gives a $(1-\varepsilon)$-approximate independent set in $2^{\mathcal{O}(1/\varepsilon)}n$ time. A conceptually easier grid shifting was discovered for unit disks by Hochbaum and Maas [21], which was later (implicitly) improved by Agarwal, van Kreveld, and Suri [1]. This line of research culminated in the algorithm of Chan [13] with a running time of $n^{\mathcal{O}(1/\varepsilon)}$ for disks, which also can be generalized for balls in higher dimensions. Baker's algorithm and Chan's algorithm (even for unit disks) are optimal under standard complexity assumptions [30]. To the best of our knowledge, there are no published approximation algorithms for HUDG$(r(n))$ or HUDG other than what is already implied from the algorithms on disk graphs. However, for hyperbolic random graphs, there is an $\mathcal{O}(m \log n)$ algorithm that yields a $1 - o(1)$-approximation for Independent Set asymptotically almost surely [5].

An interesting setting, where the treewidth of planar graphs has been used for intersection graphs, is that of covering and packing problems in the work of Har-Peled and later Marx and Pilipczuk [20, 32]. For the Independent Set problem on unit disk graphs, they consider the *Voronoi diagram* of the disk centers of a maximum independent set of size k. For a set of points (often called *sites*) S, the Voronoi diagram is a planar subdivision (a planar graph) that partitions the plane according to the closest site of S, that is, the *cell* of a site $s \in S$ consists of those points whose nearest neighbor from S is s. As the Voronoi diagram is a planar graph on $\mathcal{O}(k)$ vertices, it has a balanced separator of size $\mathcal{O}(\sqrt{k})$. Although this Voronoi diagram is based on the solution, and thus unknown from the algorithmic perspective, the idea of Marx and Pilipczuk is that one can enumerate all possible separators of the Voronoi diagram in $n^{\mathcal{O}(\sqrt{k})}$ time, and one of these separators can then be used to separate the (still unknown) solution in a balanced manner. A similar "separator guessing" technique has been used elsewhere in approximation and parameterized algorithms [31, 12].

Our contribution. Our first structural result is a balanced clique-based separator theorem. We note that all of our results in hyperbolic uniform disk graphs assume that the graph is given via some geometric representation, i.e., with the Euclidean coordinates of its disk centers in some Euclidean model of the hyperbolic plane.

▶ **Theorem 1.** *Let G be a hyperbolic uniform disk graph with radius r. Then G has a separator S that can be covered with $\mathcal{O}\big(\log n \cdot \big(1 + \frac{1}{r}\big)\big)$ cliques, such that all connected components of $G - S$ have at most $\frac{2}{3}n$ vertices. The separator can be computed in $\mathcal{O}(n \log n)$ time.*

Our separator is a carefully chosen line; the proof bounds the number of cliques intersecting the separator via decomposing some neighborhood of the chosen line into a small collection of small diameter regions. Notice that the theorem guarantees a separator with $\mathcal{O}(\log n)$ cliques in the hyperbolic setting, and deteriorates linearly as a function of $1/r$. In the almost-Euclidean setting of $r = 1/\sqrt{n}$ the separator has $\mathcal{O}(\sqrt{n} \log n)$ cliques, which almost matches the $\mathcal{O}(\sqrt{n})$ clique separator for unit disk graphs [15]. This is also a direct generalization of the 2-dimensional separator of Kisfaludi-Bak [23] from $r \in \Theta(1)$ to general radii.

Moreover, we observe that the techniques of [15, 23] can be utilized as the above separator implies $\mathcal{P}$-flattened treewidth $\mathcal{O}\left(\log^2 n \left(1 + \frac{1}{r}\right)\right)$ for any natural clique partition $\mathcal{P}$.

▶ **Corollary 2.** *Let G be a hyperbolic uniform disk graph with radius r. Then a maximum independent set of G can be computed in $n^{\mathcal{O}((1+1/r)\log^2 n)}$ time.*

Thus, a quasi-polynomial $n^{\mathcal{O}(\log^2 n)}$ algorithm is possible for all $r \in \Omega(1)$. If $r \leq 1$ – or even $r \in \mathcal{O}(1)$ – then for any $G \in$ HUDG(r) the neighborhood of a disk can be covered by a constant number of cliques of G (see also [23] for the case $r \in \Theta(1)$). Together with our separator in Theorem 1, the machinery of [15, 23] now yields the following result.

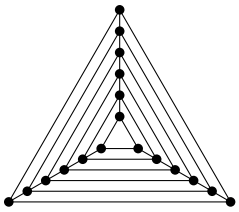


Figure 2 A Euclidean Delaunay complex of n points with outerplanarity $n/3$.

▶ **Corollary 3.** *Let G be a hyperbolic uniform disk graph with radius $r \in \mathcal{O}(1)$. Then* Dominating Set, Vertex Cover, Feedback Vertex Set, Connected Dominating Set, Connected Vertex Cover, Connected Feedback Vertex Set *can be solved in $n^{\mathcal{O}((1/r)\log n)}$ time, and q-*Coloring *for $q \in \mathcal{O}(1)$,* Hamiltonian Path *and* Hamiltonian Cycle *can be solved in $n^{\mathcal{O}(1/r)}$ time.*

While the above separator is already powerful and can be used as a basis for quasi-polynomial divide-and-conquer algorithms, the separator size stops improving when increasing the radius r beyond a constant. Nonetheless, a more powerful result is possible for super-constant radius r, which requires a different type of separator. Rather than separating hyperbolic uniform disk graphs, our second structural result is about separating the Delaunay complex, which is the dual of the Voronoi diagram, of a set of sites with pairwise distance at least $2r$. Note that in some graph $G \in \mathrm{HUDG}(r)$, the disk centers corresponding to an independent set will have pairwise distance more than $2r$, which is what motivated us to study such point sets.

A plane graph (a planar graph with a fixed planar embedding) is called *outerplanar* or 1-*outerplanar* if all of its vertices are on the unbounded face. A plane graph is called k-outerplanar if deleting the vertices of the unbounded face (and all edges incident to them) yields a $(k-1)$-outerplanar graph. Our result concerns the outerplanarity of a Delaunay complex. We prove the following tight result on the outerplanarity of the Delaunay complex $\mathcal{D}(S)$ of such a site set S. We believe that this result may be of independent interest.

▶ **Theorem 4.** *Let S be a set of n sites (points) in $\mathbb{H}^2$ with pairwise distance at least $2r$. Then the Delaunay complex $\mathcal{D}(S)$ is $1 + \mathcal{O}\big(\frac{\log n}{r}\big)$-outerplanar.*

In plane graphs, the treewidth, the outerplanarity, and the dual graph's treewidth and outerplanarity are all within a constant multiplicative factor of each other [10, 11]. Consequently, we get the bound $\mathcal{O}\big(1 + \frac{\log n}{r}\big)$ for the treewidth and outerplanarity of the Voronoi diagram and Delaunay complex of such point sets. Note that this implies constant outerplanarity for the firmly hyperbolic setting of $r \in \Omega(\log n)$.

In particular, the theorem shows that for some constant c and $r > c \log n$ the Delaunay complex is outerplanar. We recover the outerplanarity bound of [23] for $r = \Theta(1)$, and get a sublinear outerplanarity bound ($\mathcal{O}(\sqrt{n}\log n)$) even in the almost-Euclidean setting of $r = 1/\sqrt{n}$. This is surprising in light of the fact that Delaunay-triangulations can have outerplanarity $\Omega(n)$ in the Euclidean setting, as demonstrated by the set of sites in Figure 2. Notice however that the construction requires a point set where the ratio of the maximum and minimum distance among the points is $\Omega(n)$; mimicking this construction in $\mathbb{H}^2$ with a point set of minimum distance $r = 1/\sqrt{n}$ is not possible, as at distance $n \cdot r = \sqrt{n}$ the hyperbolic distortion is very significant compared to the Euclidean setting.

Using the outerplanarity bound, we are able to give the following algorithm for Independent Set.

▶ **Theorem 5.** *Let G be a hyperbolic uniform disk graph with radius r and let $k \geq 0$. Then we can decide if there is an independent set of size k in G in* $\min\left\{n^{\mathcal{O}\left(1+\frac{1}{r}\log k\right)}, 2^{\mathcal{O}(\sqrt{n})}, n^{\mathcal{O}(\sqrt{k})}\right\}$ *time.*

Our algorithm yields the first component of the running time ($n^{\mathcal{O}\left(1+\frac{1}{r}\log k\right)}$), but for very small r we can switch to the general disk graph algorithms with running times $2^{\mathcal{O}(\sqrt{n})}$ by the authors in [15] and $n^{\mathcal{O}(\sqrt{k})}$ by Marx and Pilipczuk [32]. Our algorithm is best possible under ETH for $r = \Theta(1)$ as [23] showed a lower bound of $n^{\Omega(\log n)}$ (for large k). Notice moreover that the almost-Euclidean case of $r = 1/\sqrt{n}$ gives a running time of $n^{\mathcal{O}(\sqrt{n}\log k)}$, which almost recovers the Euclidean running time of $2^{\mathcal{O}(\sqrt{n})}$. Recall that the Euclidean running time is also ETH-tight for Euclidean unit disks [15]. While this Euclidean lower bound cannot be directly applied to HUDG$(1/\sqrt{n})$, it can be adapted to the this setting, so the running time lower bound $2^{\Omega(\sqrt{n})}$ holds also in the hyperbolic plane (assuming ETH). See [24] for a similar adaptation of a Euclidean lower bound to the setting of $r = 1/\sqrt{n}$.

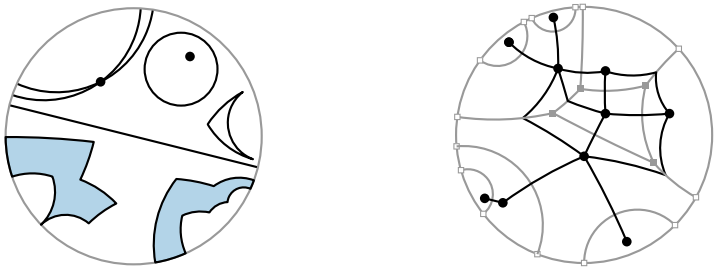
Finally, but most surprisingly, we get a polynomial running time for Independent Set in the firmly hyperbolic setting of $r \in \Omega(\log n)$; in fact, our algorithm is polynomial already for $r \in \Omega(\log k)$. In particular, this provides a polynomial exact algorithm for hyperbolic random graphs. It thereby directly generalizes the algorithm of [4] which gives a polynomial running time in hyperbolic random graphs with high probability: our algorithm has no assumptions on the input distribution and provides a polynomial worst-case running time.

The underlying idea of our exact algorithm can be regarded as a dynamic programming algorithm along the (unknown) tree decomposition of the Voronoi diagram of the solution disks, inspired by Marx and Pilipczuk [32]. More precisely, both [32] and the present paper use so-called *sphere cut decompositions*, a variant of branch decompositions for plane graphs [17] to guide the algorithm. However, due to our setting, a simple adaptation of the divide-and-conquer approach of [32] does not give the desired running time for us (it loses another logarithmic factor in the exponent). To get around this problem, we extend Marx and Pilipczuk's technique of noose-based separators to a noose-based dynamic programming algorithm. Additionally, we introduce *well-spaced* and *valid* sphere cut decompositions, and prove that they are in one-to-one correspondence with independent sets of G.

Finally, we consider approximation algorithms for Maximum Independent Set when the graph in question has low ply, that is, each point of $\mathbb{H}^2$ is covered by at most ℓ disks. We show the following.

▶ **Theorem 6.** *Let $\varepsilon \in (0,1)$ and let G be a hyperbolic uniform disk graph of radius r and ply ℓ. Then a $(1-\varepsilon)$-approximate maximum independent set of G can be found in* $\mathcal{O}\left(n^4 \log n\right) + n \cdot \left(\frac{\ell}{\varepsilon}\right)^{\mathcal{O}(1+\frac{1}{r}\log\frac{\ell}{\varepsilon})}$ *time.*

An important component of the algorithm's correctness proof is bounding the so-called *degeneracy* of hyperbolic uniform disk graphs in terms of their clique size and in terms of their ply. This generalizes the Euclidean results of [29]. When compared to the $n^{\mathcal{O}(1/\varepsilon)}$ algorithm for disk graphs by Chan [13] or the $2^{\mathcal{O}(1/\varepsilon)}n$ algorithm in planar graphs by Baker [2], both of which are conditionally optimal [30], our algorithm has a surprising quasi-polynomial dependence on $1/\varepsilon$ rather than exponential. The dependence on r exhibits the classic Euclidean-to-hyperbolic scaling seen in our earlier structural results. For $\varepsilon = 1/n$, we are guaranteed to get the optimal solution, and setting $\ell = n$ covers the general case. The resulting running time for $1/\varepsilon = \ell = n$ is $n^{\mathcal{O}\left(1+\frac{1}{r}\log n\right)}$, which matches our exact algorithm. Consequently, our approximation scheme is a generalization of our exact algorithm.



(a) (b)

Figure 3 Part (a) shows a visualization of the Poincaré disk including (parallel) lines, a circle with its center, a triangle, and polygons that include ideal points. Part (b) shows a set of sites $S \subseteq \mathbb{H}^2$ in the Poincaré disk model, $\mathcal{D}(S)$ (black), and $\mathcal{V}(S)$ (gray); Voronoi vertices are filled squares, ideal Voronoi vertices are empty squares.

2 Preliminaries

We introduce fundamental definitions and notation used throughout the paper. Additional concepts and notation are introduced in their relevant sections as needed.

Problem definition. An *independent set* in a graph is a set of pairwise non-adjacent vertices. In the Independent Set problem, we are given[7] a graph G and a number k, and the goal is to decide if there is an independent set of size k. In Maximum Independent Set, the input is a graph G, and the goal is to output the maximum value k such that G has an independent set of size k.

Let $G = (V, E)$ be a graph. A vertex set $S \subseteq V$ is a *separator* if V can be partitioned into non-empty subsets S, V_1, and V_2 such that there is no edge between V_1 and V_2, i.e., for every $v_1 \in V_1$ and $v_2 \in V_2$, it holds that $\{v_1, v_2\} \notin E$. The separator is *β-balanced* if $|V_1|, |V_2| \leq \beta n$ for some $\beta < 1$.

Hyperbolic geometry. We write $\mathbb{H}^2$ to refer to the hyperbolic plane and $\mathbb{R}^2$ to refer to the Euclidean plane. The *Poincaré disk* is a model of $\mathbb{H}^2$ that maps the whole hyperbolic plane into the interior of a Euclidean unit disk; see Figure 3a. We refer to the center of the Poincaré disk as *origin*. Straight lines in $\mathbb{H}^2$ are represented as circular arcs perpendicular to the boundary of the Poincaré disk or as chords through the origin. Hyperbolic circles are represented as Euclidean circles. The center of the hyperbolic circle is, however, farther from the origin than the Euclidean center of its representation. The Poincaré disk is *conformal*, i.e., angle preserving. Points on the boundary of the Poincaré disk are called *ideal* points. These are not part of the hyperbolic plane, but form a boundary of infinitely far points.

Let a be the sum of interior angles of a hyperbolic triangle. Then $a < \pi$ and the area of the triangle is $\pi - a$. Note that this implies that the area of a triangle is upper bounded by π. More generally, the area of a k-gon is bounded by $(k-2)\pi$. The area of a disk with hyperbolic radius r is $4\pi \sinh^2(r/2)$, which is in $\Theta(e^r)$ for $r \to \infty$ and in $\Theta(r^2)$ for $r \to 0$.

Delaunay complexes and Voronoi diagrams. We consider Voronoi diagrams with, for the most part, the usual definition: given a set $S \subset \mathbb{H}^2$ of sites, the Voronoi diagram $\mathcal{V}(S)$ is a plane graph where every inner face (*Voronoi cell*) corresponds to a site $s \in S$ and consists

[7] In this article, the graphs are always intersection graphs given through their geometric representation.

of the points in $\mathbb{H}^2$ for which s is the closest site. However, we deviate from the Euclidean definition by setting the outer face of $\mathcal{V}(S)$ to be the boundary of the Poincaré disk, which means that some of the Voronoi vertices are ideal points. This is shown in Figure 3b.

The weak dual of the Voronoi diagram is called the *Delaunay complex* and denoted by $\mathcal{D}(S)$; it is illustrated in black in Figure 3b. Here, sites $s_1 \in S$ and $s_2 \in S$ are connected in $\mathcal{D}(S)$ if and only if their Voronoi cells share a boundary.

3 Overview of our techniques

3.1 Balanced separators in HUDGs

Let G be a hyperbolic uniform disk graph with n vertices and radius r. We show that G has a balanced separator that can be covered by $\mathcal{O}((1+1/r)\log n)$ cliques; see Theorem 1. The overall argument is as follows. We find a double wedge bounded by two lines ℓ_1 and ℓ_2 that contains no vertex in its interior and separates the other vertices of G in a balanced fashion, which is ensured by selecting the center of the double wedge to be the hyperbolic *centerpoint* [24]. In Figure 4a, the double wedge with apex p is shaded gray and contains no vertices and the regions above and below the wedge each contain a constant fraction of the vertices.

Let m be the angular bisector of the wedge. As separator, we use the set of all vertices that have distance at most r from m. The set of points with distance exactly r from m forms two curves that are called *hypercycles* with axis m. Thus, vertices belong to the separator if they lie on or between the two hypercycles. The hypercycles are shown in Figure 4a and the region where separator vertices can lie is shaded blue. In the Poincaré disk, hypercycles are arcs of Euclidean circles that meet the boundary of the disk at the same ideal points as their axis but at a non-right angle. Observe that any vertex from above the top hypercycle has distance greater than $2r$ to any vertex below the bottom hypercycle. Thus, the vertices in the blue region indeed form a balanced separator.

It remains to show that the graph induced by vertices in the blue region can be covered with few cliques. For this, we cover the blue region with boxes as shown in Figure 4b. We show that each of these boxes has diameter at most $2r$, implying that the vertices within one box induce a clique. Moreover, we show that we only need $\mathcal{O}((1+1/r)\log n)$ such boxes. For the latter, we need a lower bound on the opening angle of the empty double wedge. Observe that a larger opening angle, φ, in Figure 4c, shrinks the blue region, which reduces the number of boxes required to cover it.

In the full version of this article, we first give a bound on the number of cliques required to cover the separator, parameterized by the opening angle of the empty double wedge. Afterwards, we show that there is a wedge with not too small opening angle that yields a balanced separator. The proof is constructive, i.e., given the graph with vertex positions, we can efficiently find the wedge and thereby the balanced separator of Theorem 1.

3.2 Outerplanarity of hyperbolic Delaunay complexes

Let S be a set of n sites in $\mathbb{H}^2$ with pairwise distance at least $2r$. Note that S interpreted as a hyperbolic uniform disk graph with radius r forms an independent set. To prove Theorem 4, giving an upper bound on the outerplanarity of the Delaunay complex $\mathcal{D}(S)$, we use two different types of arguments. The first argument is based on the observation that sufficiently large radius implies that any inner vertex of $\mathcal{D}(S)$ requires many other sites around it to shield it from being an outer vertex. This is similar to the observation in the introduction

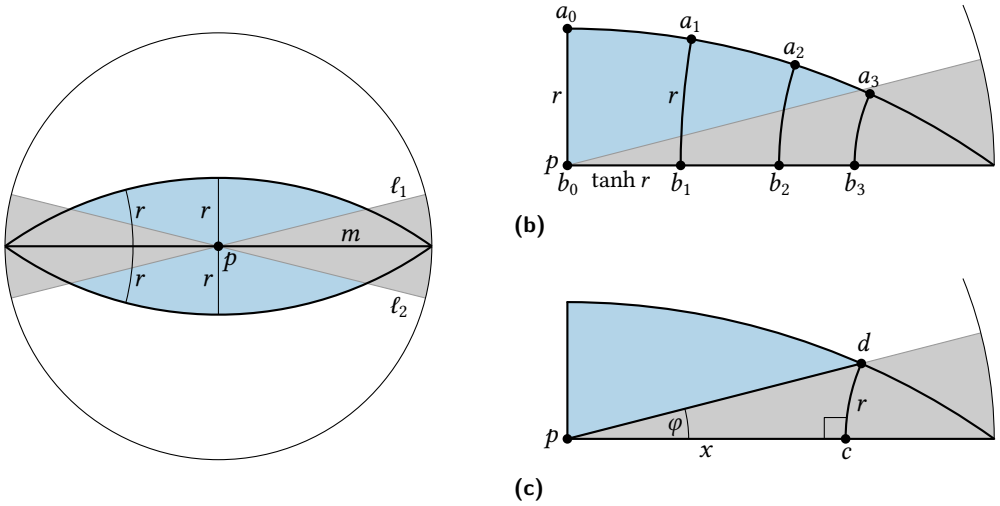


Figure 4 Illustration of our separator using the Poincaré disk model. (a) The axis m is chosen such that it separates the vertices in a balanced fashion. The double wedge between ℓ_1 and ℓ_2 contains no vertices. The separator consists of the blue set of all vertices of distance at most r to the axis m. (b) We cover the separator with boxes that have diameter $2r$ and thus form cliques. (c) If the opening angle φ is sufficiently large, then x is sufficiently small and we only need few boxes to cover the whole separator.

(recall Figure 1) that for high radius r, large stars can be realized. The argument then goes roughly as follows: If all inner vertices have degree more than 6, then we need a linear fraction of vertices on the outer face to account for the fact that the average degree in planar graphs is bounded by 6. This already implies that iteratively deleting the vertices on the outer face takes at most $\log n$ steps as each step deletes a constant fraction of the vertices. A larger radius r leads to higher degrees of inner vertices, which in turn yields stronger bounds for the outerplanarity.

This type of argument can only work if the radius r is sufficiently large. For smaller radii, the second type of argument considers layers of bounded Voronoi cells (which correspond to inner vertices of $\mathcal{D}(S)$) around one fixed center cell. As the union of these layers is bounded by a polygon with a linear number of vertices, its area is linear. In contrast to that, we will see that the area of the layers grows exponentially with the layer (with the base depending on r), showing that there cannot be too many layers.

In the following, we give a sketch of how the second argument based on area works for small radii. Although this is the argument of choice for radii that are small constants or even decreasing with n, we first give a simple area-based argument for why $r \geq \log n$ implies that the Delaunay complex has no inner vertices. This gives a good intuition how the area behaves in the hyperbolic plane and how this can help us in proving outerplanarity, even though our argument for small radii is more involved. Let $s \in S$ be a site. If s is an inner vertex of $\mathcal{D}(S)$, then its Voronoi cell is bounded and its boundary is a polygon of at most $n-1$ vertices. Thus, the area of this Voronoi cell is less than $(n-3)\pi$. Simultaneously, the disk of radius r around s is included in the Voronoi cell of s as all sites have pairwise distance at least $2r$. The area of this disk is $4\pi \sinh^2(\frac{r}{2})$ which is larger than e^r for sufficiently large r. From this it follows that bounded cells can only exist when $r < \log n$.

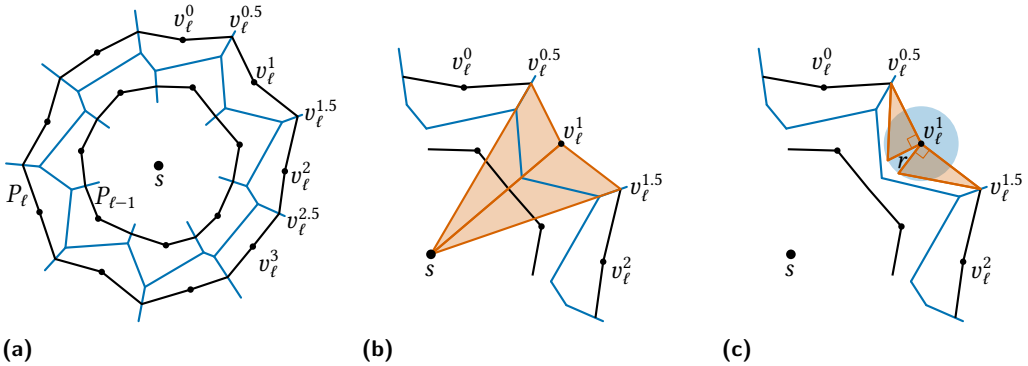


Figure 5 Illustration of the small-radius case. (a) shows two consecutive layer polygons $P_{\ell-1}$ and P_ℓ in the Delaunay complex around s. The Voronoi diagram is shown in blue. We show an upper bound of the area inside P_ℓ by covering it with the red triangles (b). We show a lower bound for the area between $P_{\ell-1}$ to P_ℓ via the red triangles in (c). Relating these two bounds yields an exponential area growth with a basis depending on r.

To make an argument that works for smaller r, let $s \in S$ be an arbitrary but fixed vertex of the Delaunay complex $\mathcal{D}(S)$. We partition the vertices of $\mathcal{D}(S)$ into *layers* by hop distance from s in $\mathcal{D}(S)$, i.e., V_ℓ is the set of vertices with distance ℓ from s. Let L be the largest integer such that for all $\ell \leq L$ the layer V_ℓ contains only inner vertices. Note that our goal is to prove an upper bound on L, as this bounds the distance of s to the outer face.

As the Delaunay complex is an internally triangulated plane graph, each layer V_ℓ induces a cycle. We denote the vertices in V_ℓ with v_ℓ^i and number the vertices modulo $|V_\ell|$, such that vertex v_ℓ^i is adjacent to vertices v_ℓ^{i-1} and v_ℓ^{i+1}; see Figure 5a. Moreover, we assume the Delaunay graph to be drawn as follows. For every edge $\{v_\ell^i, v_\ell^{i+1}\}$ we choose a point on its dual edge, i.e., the boundary between the two corresponding Voronoi cells, denote it with $v_\ell^{i+0.5}$, and connect v_ℓ^i and v_ℓ^{i+1} with two line segments via $v_\ell^{i+0.5}$. We denote the resulting polygon with P_ℓ and call it a *layer polygon* of s. Note that this yields a sequence of nested polygons where $P_{\ell-1}$ lies in the interior of P_ℓ.

Our goal then is to show that the area inside P_ℓ, denoted by $\text{Area}(P_\ell)$, grows exponentially in ℓ with a basis $b(r)$ depending on r. As $\text{Area}(P_L)$ for the outermost layer L is upper bounded by something linear in n, there cannot be too many layers. The interesting part is proving the exponential growth. For this, we show that the area gain $\text{Area}(P_\ell) - \text{Area}(P_{\ell-1})$ in layer ℓ makes up at least some sufficiently large fraction of the area $\text{Area}(P_\ell)$. For this, we give an upper bound for $\text{Area}(P_\ell)$ and relate it to a lower bound for $\text{Area}(P_\ell) - \text{Area}(P_{\ell-1})$.

How we derive these bounds is illustrated in Figure 5b and Figure 5c, respectively. For the upper bound on $\text{Area}(P_\ell)$, we cover P_ℓ with triangles connecting every edge of P_ℓ with the vertex s; see the two red triangles in Figure 5b for the two edges $\{v_\ell^{0.5}, v_\ell^1\}$ and $\{v_\ell^1, v_\ell^{1.5}\}$ of P_ℓ. For the lower bound, we find a set of disjoint triangles that lie between P_ℓ and $P_{\ell-1}$. For this, observe that for every vertex $v_\ell^i \in V_\ell$ in layer ℓ, the Voronoi cell of v_ℓ^i completely contains the disk of radius r around v_ℓ as the sites have pairwise distance at least $2r$. Thus, the two triangles illustrated in red for v_ℓ^1 in Figure 5c satisfy the property of lying between P_ℓ and $P_{\ell-1}$. As the two triangles can in principle intersect, we choose for each vertex in layer ℓ the larger of the two triangles. Note that this gives a collection of $|V_\ell|$ disjoint triangles, as each chosen triangle lies in a different Voronoi cell. Thus, the total area of these triangles gives a lower bound for $\text{Area}(P_\ell) - \text{Area}(P_{\ell-1})$.

It then remains to relate the upper bound for $\text{Area}(P_\ell)$, i.e., the area of the red triangles in Figure 5b, with the lower bound for $\text{Area}(P_\ell) - \text{Area}(P_{\ell-1})$, i.e., the area of larger red triangle in Figure 5c. Intuitively, this does not seem too far fetched for the following reasons. First, the triangles in Figure 5b share a side with the triangles in Figure 5c. Secondly, the other sides of the triangles in Figure 5c have length at least r. Thus, the triangles in Figure 5c cannot be too much smaller than those in Figure 5b.

The details of this area-based argument for small radii can be found in the full version of this paper. Together with the degree-based bound for large radii we obtain Theorem 4.

3.3 Exact algorithm

We give a brief description of the exact algorithm for Theorem 5 and refer to the full version of this paper for the remaining details. Instead of computing the independent set directly, we define some additional structure for a given independent set. The algorithm then finds this additional structure such that the corresponding independent set is maximized. The additional structure is illustrated for two example independent sets in Figure 6. In the following, we explain this from top to bottom.

Let G be a hyperbolic uniform disk graph of radius r together with an independent set S (top row in Figure 6). As there are no edges between vertices in S, they have pairwise distance at least $2r$. Thus, the Delaunay complex $\mathcal{D}(S)$ (second row in Figure 6) has low outerplanarity due to Theorem 4. Low outerplanarity implies small treewidth, which implies small branchwidth [36]. Moreover, as $\mathcal{D}(S)$ is a planar graph, we can wish for a so-called *sphere cut decomposition* of low width, which is a branch decomposition with some additional properties [37, 17, 32]. In a nutshell, we get a hierarchy of *nooses*, where each noose is a closed curve that intersects the graph only at vertices and functions as a vertex separator, splitting interior edges from exterior ones. The hierarchy is a binary tree in the sense that every noose containing more than one edge in its interior has two child nooses partitioning these edges further. Row three in Figure 6 shows three nooses of the Delaunay complex $\mathcal{D}(S)$. The solid blue noose is the parent of the dashed red and green noose.

The width of a sphere cut decomposition is the maximum number of vertices on a noose. As $\mathcal{D}(S)$ has low outerplanarity, we have a low-width sphere cut decomposition, i.e., each noose goes through only few vertices. As a last step, we normalize the nooses by specifying their exact geometry, keeping their combinatorial structure intact; see the last row of Figure 6. At its core, the normalized nooses are polygons that alternate between vertices in S and Voronoi vertices, with some special treatment for the outer face of $\mathcal{D}(S)$.

Observe that the two examples (left and right column) illustrate different independent sets S, but share the green normalized noose. This is not a coincidence, but actually happens a lot. In fact, despite the potentially exponential number of independent sets, we can bound the total number of normalized nooses that we can obtain in this way by $n^{\mathcal{O}(1+\frac{\log n}{r})}$. This follows from the fact that each noose visits only $\mathcal{O}(1 + \frac{\log n}{r})$ vertices due to Theorem 4. Thus, the normalized nooses are essentially polygons with $\mathcal{O}(1 + \frac{\log n}{r})$ corners and the set of potential corners is polynomial, as each corner is a vertex of G or a Voronoi vertex (which is defined by three vertices of G).

Our independent set algorithm works as follows. We start by listing all candidates for normalized nooses. We then use a dynamic program to combine nooses into a hierarchy of polygons. We make sure that the resulting hierarchy is valid in the sense that it corresponds to a sphere cut decomposition. Additionally, we enforce that it is *well-spaced*, which lets us show that all vertices of G visited by the polygons in the hierarchy form an independent set. Then, finding a maximum independent set is equivalent to maximizing the number of vertices visited by polygons in this polygon hierarchy, which can be done with the dynamic program.

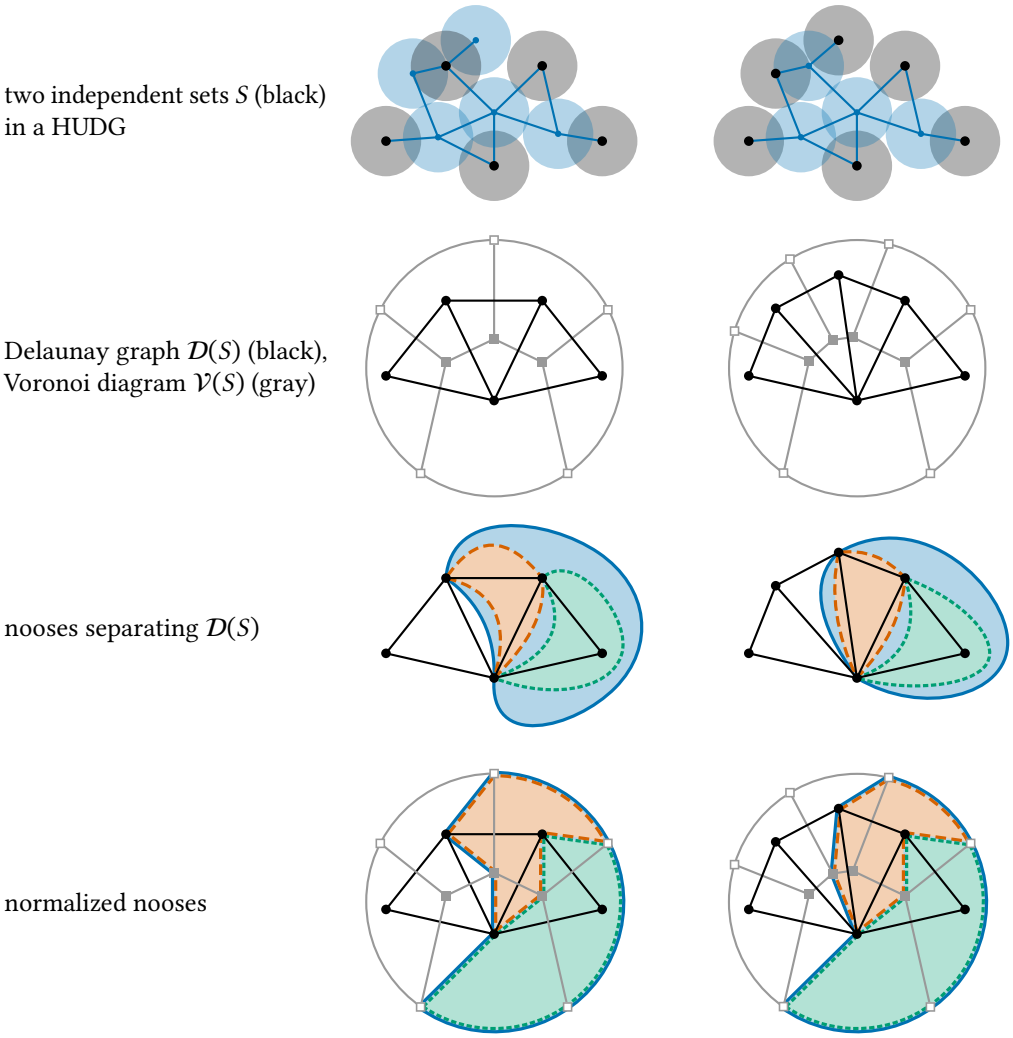


Figure 6 Illustration of our exact algorithm in Theorem 5.

3.4 Approximation algorithm

The approximation algorithm of Theorem 6 (see also the full version of this paper) works similarly to Lipton and Tarjan's version for planar graphs [28]: we repeatedly apply a separator until we get small patches and then solve each patch individually using the exact algorithm of Theorem 5. Vertices in the separators get ignored, so we get an $(1-\varepsilon)$-approximation when these make up only an ε fraction of the maximum independent set. We guarantee this using a lower bound of $\Omega(n/\ell)$ on the size of a maximum independent set, which follows from a proof that hyperbolic uniform disk graphs of ply ℓ always have a vertex of degree at most $4\ell-1$, making them $(4\ell-1)$-degenerate.

4 Future directions

There are several intriguing future directions to explore in this space. First, is there a fully polynomial approximation scheme for Independent Set for $r \geq 1$? If not, is there at least an approximation scheme that is polynomial in n, quasi-polynomial in $1/\varepsilon$, but independent of the ply?

Second, it would be interesting to explore separators in higher-dimensional hyperbolic spaces. The only case studied there is $r = \Theta(1)$ [23]. One may expect that separator size should also decrease with growing r. Is there a constant c_d such that Independent Set can be solved in polynomial time in $\mathbb{H}^d$ when $r \geq c_d \log n$?

Third, we should explore uniform disk graphs in surfaces of constant curvature, i.e., on the sphere, flat torus, and especially hyperbolic surfaces. What is the size of the best clique-based separator for a uniform disk graph of radius r on a hyperbolic surface of genus g?

Fourth, it would be interesting to investigate the complexity of problems other than Independent Set, and study how their complexity scales with the radius of the disks, or equivalently, with the curvature of the underlying space.

Finally, it is unclear whether our bound on balanced separators is tight for all values of r. For constant r, regular hyperbolic tilings are hyperbolic uniform disk graphs with constant clique number and logarithmic treewidth, making our bound asymptotically tight. However, for larger radii, it could be possible to reduce the logarithmic factor.

References

1 Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Comput. Geom.*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.

2 Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

3 Nicholas Bieker, Thomas Bläsius, Emil Dohse, and Paul Jungeblut. Recognizing unit disk graphs in hyperbolic geometry is $\exists\mathbb{R}$-complete. *CoRR*, abs/2301.05550, 2023. doi:10.48550/arXiv.2301.05550.

4 Thomas Bläsius, Philipp Fischbeck, Tobias Friedrich, and Maximilian Katzmann. Solving vertex cover in polynomial time on hyperbolic random graphs. *Theory Comput. Syst.*, 67(1):28–51, 2023. doi:10.1007/S00224-021-10062-9.

5 Thomas Bläsius, Tobias Friedrich, and Maximilian Katzmann. Efficiently approximating vertex cover on scale-free networks with underlying hyperbolic geometry. *Algorithmica*, 85(12):3487–3520, 2023. doi:10.1007/S00453-023-01143-X.

6 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, and Daniel Stephan. Strongly hyperbolic unit disk graphs. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 254 of *LIPIcs*, pages 13:1–13:17, 2023. doi:10.4230/LIPICS.STACS.2023.13.

7 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *24th Annual European Symposium on Algorithms, ESA*, volume 57 of *LIPIcs*, pages 15:1–15:16, 2016. doi:10.4230/LIPICS.ESA.2016.15.

8 Thomas Bläsius, Cedric Freiberger, Tobias Friedrich, Maximilian Katzmann, Felix Montenegro-Retana, and Marianne Thieffry. Efficient shortest paths in scale-free networks with underlying hyperbolic geometry. *ACM Transactions on Algorithms (TALG)*, 18(2):19:1–19:32, 2022. doi:10.1145/3516483.

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. *Computing Research Repository (CoRR)*, abs/2407.09362, 2024. doi:10.48550/arXiv.2407.09362.

10 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

11 Vincent Bouchitté, Frédéric Mazoit, and Ioan Todinca. Chordal embeddings of planar graphs. *Discret. Math.*, 273(1-3):85–102, 2003. doi:10.1016/S0012-365X(03)00230-9.

12 Karl Bringmann, Sándor Kisfaludi-Bak, Michal Pilipczuk, and Erik Jan van Leeuwen. On geometric set cover for orthants. In *27th Annual European Symposium on Algorithms, ESA*, volume 144 of *LIPIcs*, pages 26:1–26:18, 2019. doi:10.4230/LIPICS.ESA.2019.26.

13 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.

14 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

15 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.

16 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.

17 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

18 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.

19 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering. In *39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 573–585, 2012. doi:10.1007/978-3-642-31585-5_51.

20 Sariel Har-Peled. Quasi-polynomial time approximation scheme for sparse subsets of polygons. In *30th Annual Symposium on Computational Geometry, SoCG*, page 120. ACM, 2014. doi:10.1145/2582112.2582157.

21 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.

22 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

23 Sándor Kisfaludi-Bak. Hyperbolic intersection graphs and (quasi)-polynomial time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1621–1638. SIAM, 2020. doi:10.1137/1.9781611975994.100.

24 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.

25 Paul Koebe. *Kontaktprobleme der konformen Abbildung*. Hirzel, 1936.

26 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82:036106, 2010. doi:10.1103/PhysRevE.82.036106.

27 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

28 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. doi:10.1137/0209046.

29 Madhav V. Marathe, H. Breu, Harry B. Hunt III, S. S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995. doi:10.1002/NET.3230250205.

30 Dániel Marx. On the optimality of planar and geometric approximation schemes. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 338–348. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.26.

31 Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 474–484. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00052.

32 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams. *ACM Trans. Algorithms*, 18(2):13:1–13:64, 2022. doi:10.1145/3483425.

33 Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *J. Comb. Theory B*, 103(1):114–143, 2013. doi:10.1016/J.JCTB.2012.09.004.

34 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

35 Tobias Müller and Merlijn Staps. The diameter of KPKVB random graphs. *Advances in Applied Probability*, 51(2):358–377, 2019. doi:10.1017/apr.2019.23.

36 Neil Robertson and P. D Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. doi:10.1016/0095-8956(91)90061-N.

37 Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Comb.*, 14(2):217–241, 1994. doi:10.1007/BF01215352.

Transforming Dogs on the Line: On the Fréchet Distance Under Translation or Scaling in 1D

Lotte Blank
University of Bonn, Germany

Jacobus Conradi
University of Bonn, Germany

Anne Driemel
University of Bonn, Germany

Benedikt Kolbe
University of Bonn, Germany

André Nusser
Université Côte d'Azur, CNRS, Inria, Sophia Antipolis, France

Marena Richter
University of Bonn, Germany

Abstract

The Fréchet distance is a computational mainstay for comparing polygonal curves. The Fréchet distance under translation, which is a translation invariant version, considers the similarity of two curves independent of their location in space. It is defined as the minimum Fréchet distance that arises from allowing arbitrary translations of the input curves. This problem and numerous variants of the Fréchet distance under some transformations have been studied, with more work concentrating on the discrete Fréchet distance, leaving a significant gap between the discrete and continuous versions of the Fréchet distance under transformations. Our contribution is twofold: First, we present an algorithm for the Fréchet distance under translation on 1-dimensional curves of complexity n with a running time of $\mathcal{O}(n^{8/3}\log^3 n)$. To achieve this, we develop a novel framework for the problem for 1-dimensional curves, which also applies to other scenarios and leads to our second contribution. We present an algorithm with the same running time of $\mathcal{O}(n^{8/3}\log^3 n)$ for the Fréchet distance under scaling for 1-dimensional curves. For both algorithms we match the running times of the discrete case and improve the previously best known bounds of $\tilde{\mathcal{O}}(n^4)$. Our algorithms rely on technical insights but are conceptually simple, essentially reducing the continuous problem to the discrete case across different length scales.

2012 ACM Subject Classification Theory of computation → Computational geometry

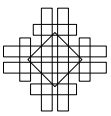
Keywords and phrases Fréchet distance under translation, Fréchet distance under scaling, time series, shape matching

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.22

Related Version *Full Version*: https://arxiv.org/abs/2501.12821

Funding *Lotte Blank*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459420781 (FOR AlgoForGe).
Jacobus Conradi: Funded by the iBehave Network: Sponsored by the Ministry of Culture and Science of the State of North Rhine-Westphalia.
Anne Driemel: Affiliated with Lamarr Institute for Machine Learning and Artificial Intelligence.
Benedikt Kolbe: This work was partially supported by the Lamarr Institute for Machine Learning and Artificial Intelligence.
André Nusser: This work was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of Excellence of Université Côte d'Azur under reference number ANR-15-IDEX-01.

© Lotte Blank, Jacobus Conradi, Anne Driemel, Benedikt Kolbe, André Nusser, and Marena Richter;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

The Fréchet distance [26] is one of the most well-studied distance measures for polygonal curves, with a long history in both applications [7, 17, 13] and algorithmic research [1, 21, 27]. For many areas of interest, including the analysis of stock market trends, electrocardiograms or electroencephalograms, the underlying behavior to be analyzed crucially involves the behavior of 1-dimensional curves, or *time series* [5, 29]. One of the advantages that the Fréchet distance offers over competing distance measures such as the Hausdorff distance is that the Fréchet distance considers a joint traversal of both curves, which naturally takes the orientation and ordering of the vertices of both curves into account. For 1-dimensional curves, the difference is particularly jarring, as representing a *continuous* curve by the points lying on it essentially discards all information concerning the underlying dynamics when measuring the Hausdorff distance. Intuitively, the Fréchet distance between two curves can be thought of the shortest possible length of a rope that can connect two climbers that cooperatively travel along their respective routes from start to end. Another well-known analogy stems from the metaphor of a person walking a dog. In this setting, the connecting rope corresponds to a dog leash and we ask for the shortest leash that can be used for the walk.

The algorithmic complexity of the computation of the Fréchet distance has been thoroughly investigated and is well understood. For curves of complexity n, the Fréchet distance can be computed in time $\tilde{\mathcal{O}}(n^2)$ [1, 12, 16], where $\tilde{\mathcal{O}}$ hides polylogarithmic factors in n. Furthermore, this is hard in the sense that there cannot exist an $\mathcal{O}(n^{2-\epsilon})$ time algorithm in the plane for any $\epsilon > 0$, unless the Strong Exponential Time Hypothesis (SETH) fails even in one dimension [8, 15].

The Fréchet distance has been studied in many different contexts, leading to a plethora of variants, inspired both by applications as well as theoretical considerations, see [14, 19, 20, 24]. Each of the available variants fits into one of two classes, according to whether only the distances of vertices of the curves are considered (the discrete version), or whether the computed distance depends on all points along the straight line segments connecting them (the continuous variant).

While the continuous Fréchet distance naturally considers all points between two vertices and tends to give better practical results (e.g., no discretization artifacts), the discrete Fréchet distance is often easier to handle algorithmically as well as implementation-wise. One overarching goal of Fréchet related research is to match the algorithmic results that are attained in the discrete version with those in the continuous case.

Fréchet distance under translation or scaling. An important aspect of detecting movement patterns in different application areas is to consider the movement independent of its absolute location and scale, i.e., we want to know for which location and scale do the patterns look most similar. Consider, for example, a cardiogram of a patient with heart arrhythmia [5]. Intuitively, some of the shared characteristics of the observed heartbeats do not depend on their absolute location/scaling but instead on the relative movement of the wave. Similarly, identifying a trend in the stock market does not necessarily depend on the exact values of the stocks involved. Instead, a group of stocks may display similar relative behavior although their absolute values are far apart, in which case a natural approach is to allow arbitrary translation and rescaling of the curves to distill their concurrent behavior. These motivations directly lead to the idea of the Fréchet distance under translation or scaling. Concretely, the Fréchet distance under translation (resp. scaling) is the minimum Fréchet distance that we obtain by allowing an arbitrary translation (resp. scaling) of one curve.

Related work. As mentioned previously, due to the simpler setting, there are often more results in the discrete setting, as is also the case for the *discrete* Fréchet distance under translation. This problem was previously mainly considered in the Euclidean plane. The first algorithm due to Jiang, Xu, and Zhu [28] builds an arrangement of size $\mathcal{O}(n^4)$ over the translation space, and then queries an arbitrary representative of each cell of this arrangement with the quadratic-time Fréchet distance algorithm, leading to an $\mathcal{O}(n^6 \log n)$ running time (where the logarithmic factor stems from the usage of parametric search). Avraham, Kaplan, and Sharir [4] crucially observed that instead of re-computing the Fréchet distance for each cell of the arrangement, one can instead formulate the problem as an online dynamic reachability problem on a grid graph and provide a data structure with update time $\mathcal{O}(n)$, leading to an $\mathcal{O}(n^5 \log n)$ algorithm. The current best-known algorithm observes that these updates are offline, i.e., they can be known in advance, and design a tailored data structure for this problem reducing the update time to $\mathcal{O}(n^{2/3} \log^2 n)$, leading to a running time of $\mathcal{O}(n^{4+\frac{2}{3}} \log^3 n)$.[1] This result is complemented by a lower bound of $\mathcal{O}(n^{4-o(1)})$, conditional on the Strong Exponential Time Hypothesis (SETH), for the discrete Fréchet distance under translation in the plane [11].

The currently best algorithm for discrete time series that is explicitly described in the literature has running time $\mathcal{O}(n^3)$ [25], based on the solution to the decision version of the problem in [4]. We note that the approach of Bringmann, Künnemann, and Nusser [11] is also directly applicable to the 1-dimensional problem, which leads to an $\mathcal{O}(n^{8/3} \log^3 n)$ algorithm.[2]

The *continuous* Fréchet distance under translations was first studied in 2001 [2, 3, 23]. Shortly after, Wenk [30] considered the Fréchet distance under scaling and developed the currently best published algorithms for both variants (translation or scaling) in the continuous setting in d-dimensional Euclidean space. For time series, these algorithms achieve a running time of $\mathcal{O}(n^5 \log n)$. We note that for time series there is an unpublished $\tilde{\mathcal{O}}(n^4)$ algorithm for both problems that can be considered folklore. We briefly sketch the idea here. First, we build the arrangement of intervals separated by events at which the decision problem for the Fréchet distance under translation/scaling is subject to change. The crux is that in one dimension, there are only $\mathcal{O}(n^2)$ such events, as the edge-vertex-vertex events that appear in higher dimensions can be omitted as they are captured by the vertex-vertex events (see Corollary 23 and the full version for more details). Thus, computing the Fréchet distance for an arbitrary representative translation/scaling in each cell together with parametric search [18] yields an $\tilde{\mathcal{O}}(n^4)$ algorithm. We want to stress that, while indeed the events in one dimension only depend on vertex-vertex alignments, this does not directly enable the usage of the grid reachability data structures of [4] and [11], as we still have to compute the *continuous* Fréchet distance for each cell.

1.1 Our Contributions

We present the first new result on the continuous Fréchet distance under translation and the continuous Fréchet distance under scaling since the results by Wenk [30] in 2002. Concretely, we obtain the following two theorems:

[1] Meanwhile, this problem was also tackled from the algorithm engineering side, making use of Lipschitz optimization instead of dynamic graph algorithms [10].

[2] This is a result of the arrangement size in $\mathbb{R}$ being n^2, while in $\mathbb{R}^2$ it is n^4. Hence, the running time is reduced by a factor of n^2.

▶ **Theorem 1.** *There exists an algorithm to compute the continuous Fréchet distance under translation between two time series of complexity n in time $\mathcal{O}(n^{8/3}\log^3 n)$.*

▶ **Theorem 2.** *There exists an algorithm to compute the continuous Fréchet distance under scaling between two time series of complexity n in time $\mathcal{O}(n^{8/3}\log^3 n)$.*

These results are made possible by the introduction of a novel framework for studying (continuous) time series [6] and the offline dynamic grid reachability data structure of [11]. Our approach essentially reduces (in an algorithmic sense) the continuous problems to their discrete counterparts and hence surprisingly matches the results for the two distance measures in the discrete setting. We note that in the Euclidean plane, there still is a large gap between the discrete and continuous setting and, while there was significant progress on the discrete side, there was no progress for the continuous Fréchet distance under translation or scaling. We believe that our approach is also of independent interest and can potentially be applied to other continuous 1-dimensional Fréchet distance problems, to match the running times of the discrete setting.

The asymmetric case. We remark that the offline dynamic grid reachability data structure of [11] is only described for the symmetric case in which both curves have the same complexity. While we believe that this restriction can be eliminated, it would drastically complicate the exposition of our results, which is why we also make this assumption in the statements of our results. Otherwise this restriction would not be necessary.

1.2 Technical Overview

On a high level, our approach for the decision problem for a given $\delta > 0$ for both the translation and scaling invariant Fréchet distance can be summarized in the following three steps.

1. The curves are represented by their (slightly adapted) δ-signatures [22], which are simplifications at the given length-scale δ and encode the large scale behavior of the curve in terms of a subset of selected vertices that essentially form a discrete curve. Curves naturally decompose into three parts, the beginning, middle, and end, with the middle part given by the δ-signature. The task is to design and choose data structures for all parts in such a way that they can be efficiently updated and combined when the transformation changes.
2. To process the beginning and end of the curves, we introduce another simplification, and subsequently identify the first point in the beginning of one curve that matches the entire beginning of the other, and likewise for the end parts, where we instead identify the last point that allows a matching. The crux is that we need to be able to do this efficiently for different transformations. We achieve this by introducing the structural notion of deadlocks.
3. We identify and precompute a set of $\mathcal{O}(n^2)$ representative transformations at which the answer to the decision problem is subject to change, and for which the answer can be updated efficiently as we sweep over them. Since each part of the curves essentially behaves like a discrete curve, the decision problem ultimately reduces to a reachability question for an object very similar to the well-known free space matrix used in the decision problem for the discrete Fréchet distance. Similar to the best algorithm for the discrete Fréchet distance under translation, we then make use of the offline dynamic grid reachability data structure [11] to implement updates and queries efficiently.

A note on parametric search. As is common, in this work we mostly describe how we solve the decision version, to then subsequently apply parametric search [18]. We note that the algorithm for the discrete Fréchet distance under translation on time series of [25] cleverly avoids parametric search, and hence shaves a logarithmic factor that would otherwise appear when directly applying [4]. One reason why we cannot take a similar approach – ignoring the fact that this technique is used for the discrete setting – is that the avoidance of parametric search in [25] leads to updates that crucially depend on the results of the decider calls; the updates are therefore *online*. Hence, a combination of [25] with the *offline dynamic* data structure of [11] seems infeasible.

2 Preliminaries

We denote with $[n]$ the set $\{1, \dots, n\}$. For any two points $p_1, p_2 \in \mathbb{R}$, we denote with $\overline{p_1 p_2}$ the directed line segment from p_1 to p_2. A *time series* of complexity n is a 1-dimensional curve formed by n points $P(1), \dots, P(n) \in \mathbb{R}$ and the ordered line segments $\overline{P(i)P(i+1)}$ for $i = 1, \dots, n-1$. Such a time series can be viewed as a function $P : [1, n] \to \mathbb{R}$, where $P(i+\alpha) = (1-\alpha)P(i) + \alpha P(i+1)$ for $i = 1, \dots, n-1$ and $\alpha \in [0, 1]$. Sometimes, we denote P as $\langle P(1), P(2), \dots, P(n)\rangle$. The points $P(i)$ are called *vertices* of P and $\overline{P(i)P(i+1)}$ are called *edges* of P for $i = 1, \dots, n-1$. For $1 \le s \le t \le n$, we denote by $P[s, t]$ the subcurve of P obtained from restricting the domain to the interval $[s, t]$. Further, we denote with $B(P, \delta)$ the set of points having distance at most δ to a point of P, i.e., $B(P, \delta) \coloneqq \{x \mid \exists a \in [1, n] \text{ s.t. } |x - P(a)| \le \delta\}$. We define $\mathrm{im}(P)$ to be the image of the time series P, meaning that $\mathrm{im}(P) \coloneqq \{P(a) \mid a \in [1, n]\}$.

Degenerate point configurations. Our results do not depend on the assumption that the time series are in general position, as all such occurrences can be easily dealt with using arbitrary tie-breaks, or *symbolic perturbation*, where necessary. Every such instance is a result of degenerate vertex-vertex distances, and we discuss more concretely how they are dealt with when they arise.

To define the Fréchet distance, let P and Q be two time series of complexity n and m. Further, let $\mathcal{H}_n$ be the set of all continuous non-decreasing functions $h : [0, 1] \to [1, n]$ with $h(0) = 1$ and $h(1) = n$. The *continuous Fréchet distance* is defined as

$$d_F(P, Q) = \min_{h_P \in \mathcal{H}_n, h_Q \in \mathcal{H}_m} \max_{\alpha \in [0,1]} |P(h_P(\alpha)) - Q(h_Q(\alpha))|.$$

In this paper, we aim to determine the optimal translation or scaling of a time series that minimizes the Fréchet distance. For a time series $P = \langle P(1), \dots, P(n)\rangle$ and a value $t \in \mathbb{R}$, the *translated time series* P_t is $\langle P(1)+t, \dots, P(n)+t\rangle$ and the *Fréchet distance under translation* is defined as

$$d_F^T(P, Q) = \min_{t \in \mathbb{R}} d_F(P, Q_t).$$

For a value $s \in \mathbb{R}$, we define the *scaled time series* sP of P to be $sP = \langle sP(1), \dots, sP(n)\rangle$ and the *Fréchet distance under scaling* is defined as

$$d_F^S(P, Q) = \min_{s \in \mathbb{R}_{\ge 0}} d_F(P, sQ).$$

In contrast to the Fréchet distance under translation, the Fréchet distance under scaling is not symmetric. The above is the directed version, and one obtains the undirected version by minimizing over both directed variants. Whenever we just refer to the Fréchet distance, we mean the continuous Fréchet distance.

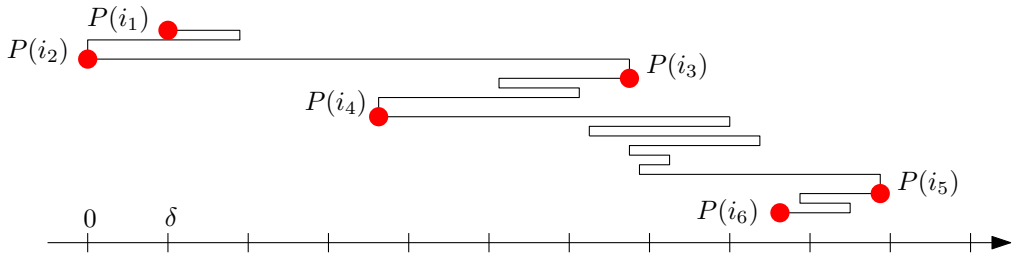


Figure 1 Throughout this paper, vertices of time series are drawn as vertical segments for clarity. The red vertices of the time series P are its δ-signature vertices. After linearly interpolating those red vertices, we get the extended δ-signature of P.

Figure 2 The left (resp. right) time series is δ-monotone increasing (resp. decreasing).

2.1 Signatures and Coupled Visiting Orders

A δ-signature of a time series P is a time series that consists of important maxima and minima of P. The concept of signatures was introduced by Driemel, Krivošija and Sohler [22]. Later, Blank and Driemel [6] extended the δ-signature by adding one vertex in the beginning and one in the end to obtain an extended δ-signature. In this paper, we only use extended δ-signatures. See Figure 1 for an example. Hence, whenever we say signature we mean extended signature. To define extended δ-signatures, we use the notion of δ-monotone time series. Examples of δ-monotone time series can be found in Figure 2.

▶ **Definition 3.** *A time series $P : [1, n] \to \mathbb{R}$ is* δ-monotone increasing (resp. decreasing) *if it holds that $P(s) \leq P(s') + \delta$ (resp. $P(s) \geq P(s') - \delta$) for all $s < s' \in [1, n]$.*

▶ **Definition 4** (extended δ-signature)**.** *Let $P = \langle P(1), \ldots, P(n)\rangle$ be a time series and $\delta \geq 0$. Then, an* extended δ-signature *$P' = \langle P(i_1), \ldots, P(i_t)\rangle$ with $1 = i_1 \leq i_2 < \ldots < i_{t-1} \leq i_t = n$ of P is a time series with the following properties:*

(a) (non-degenerate) *For $k = 2, \ldots, t-1$, it holds that $P(i_k) \notin \overline{P(i_{k-1})P(i_{k+1})}$.*

(b) (2δ-monotone) *For $k = 1, \ldots, t-1$, $P[i_k, i_{k+1}]$ is 2δ-monotone increasing or decreasing.*

(c) (minimum edge length) *If $t > 4$, then for $k = 2, \ldots, t-2$, $|P(i_k) - P(i_{k+1})| > 2\delta$.*

(d) (range)

- *For $k = 2, \ldots, t-2$, it holds that $\mathrm{im}(P[i_k, i_{k+1}]) = \overline{P(i_k)P(i_{k+1})}$, and*
- *$\mathrm{im}(P[1, i_2]) \subset [P(i_2), P(i_2) + 2\delta]$ or $\mathrm{im}(P[1, i_2]) \subset [P(i_2) - 2\delta, P(i_2)]$, and*
- *$\mathrm{im}(P[i_{t-1}, n]) \subset [P(i_{t-1}),\ P(i_{t-1}) + 2\delta]$ or $\mathrm{im}(P[i_{t-1}, n]) \subset [P(i_{t-1}) - 2\delta, P(i_{t-1})]$.*

The vertices of P' are called δ-signature vertices *of P.*

Driemel, Krivošija and Sohler [22] showed that there always exists a δ-signature and that it can be computed in $\mathcal{O}(n)$ time.[3] Bringmann, Driemel, Nusser and Psarros [9] defined δ-visiting orders and Blank and Driemel [6] used this concept to define coupled δ-visiting orders (see Figure 3), which can be used to decide whether the Fréchet distance between two time series is at most δ.

[3] They proved this statement for δ-signatures, but it can easily be shown for extended δ-signatures as well.

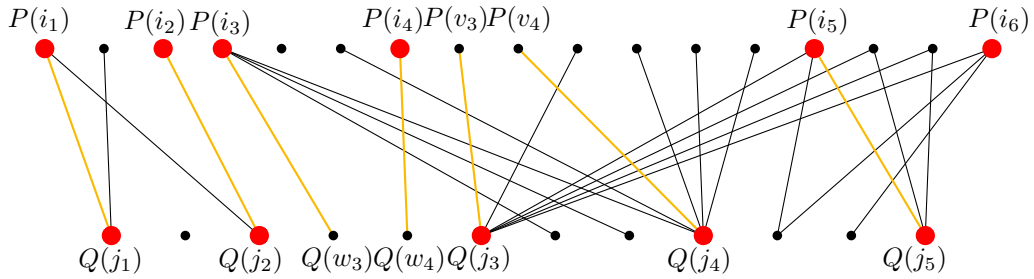


Figure 3 Visualization of a coupled δ-visiting order. Edges are drawn between vertices $P(i)$ and $Q(j)$, when one is a δ-signature vertex and $|P(i) - Q(j)| \leq \delta$. Then, a coupled δ-visiting order consists of a subset of the drawn edges, shown as an orange bipartite graph, where no two edges cross and it contains one incident edge for every δ-signature vertex.

▶ **Definition 5** (coupled δ-visiting order). *Consider two time series $P = \langle P(1), \dots, P(n)\rangle$ and $Q = \langle Q(1), \dots, Q(m)\rangle$. Then, a δ-visiting order of Q on P for the δ-signature vertices $Q(j_1), \dots, Q(j_{s_Q})$ is a sequence of indices $x_1 \leq \cdots \leq x_{s_Q}$ such that $|P(x_k) - Q(j_k)| \leq \delta$ for all k. We similarly define a δ-visiting order $y_1 \leq \cdots \leq y_{s_P}$ of P on Q for the δ-signature vertices $P(i_1), \dots, P(i_{s_P})$ of P. These two δ-visiting orders are said to be crossing-free if there exists no k, l such that $i_k < x_l$ and $j_l < y_k$, or $x_k < i_l$ and $y_l < j_k$. In this case, the ordered sequence containing all tuples (x_k, j_k) and (i_l, y_l), where $k = 1, \dots, s_Q$ and $l = 1, \dots, s_P$, is called* coupled δ-visiting order.

▶ **Lemma 6** (Lemma 9 of [6]). *Let P, Q be two time series and $\langle P(i_1), \dots, P(i_{s_P})\rangle$ and $\langle Q(j_1), \dots, Q(j_{s_Q})\rangle$ their δ-signatures. Then, $d_F(P, Q) \leq \delta$ if and only if there exist a coupled δ-visiting order $((v_1, w_1), (v_2, w_2), \dots, (v_t, w_t))$ of P and Q such that*

- **(i)** *if $v_2 = i_2$ (resp. $w_2 = j_2$), then $\exists w^* \in [w_2, \min\{w_2 + 1, j_2\}]$ (resp. $v^* \in [v_2, \min\{v_2 + 1, i_2\}]$) such that $d_F(P[1, v_2], Q[1, w^*]) \leq \delta$ (resp. $d_F(P[1, v^*], Q[1, w_2]) \leq \delta$), and*
- **(ii)** *if $v_{t-1} = i_{s_P-1}$ (resp. $w_{t-1} = j_{s_Q-1}$), then $\exists w^{**} \in [\max\{j_{s_Q-1}, w_{t-1} - 1\}, w_{t-1}]$ (resp. $v^{**} \in [\max\{i_{s_P-1}, v_{t-1} - 1\}, v_{t-1}]$) such that $d_F(P[v_{t-1}, v_t], Q[w^{**}, w_t]) \leq \delta$ (resp. $d_F(P[v^{**}, v_t], Q[w_{t-1}, w_t]) \leq \delta$).*

There is a small difference to the cited lemma, which we discuss in the full version.

2.2 Grid Reachability

▶ **Definition 7** (Offline Dynamic Grid Reachability). *Let G be the directed $n \times n$-grid graph in which every node is either activated or deactivated. We are given updates $u_1, \dots, u_U$, which are of the form "activate node (i, j)" or "deactivate node (i, j)" in an offline manner. The task of* offline dynamic grid reachibility *is to compute for all $1 \leq \ell \leq U$ if (n, n) can be reached by $(1, 1)$ after updates $u_1, \dots, u_\ell$ are performed.*

Our main results depend on the following theorem.

▶ **Theorem 8** (Theorem 3.1 of [11]). *Offline Dynamic Grid Reachability can be solved in time $\mathcal{O}(n^2 + Un^{2/3} \log^2 n)$.*

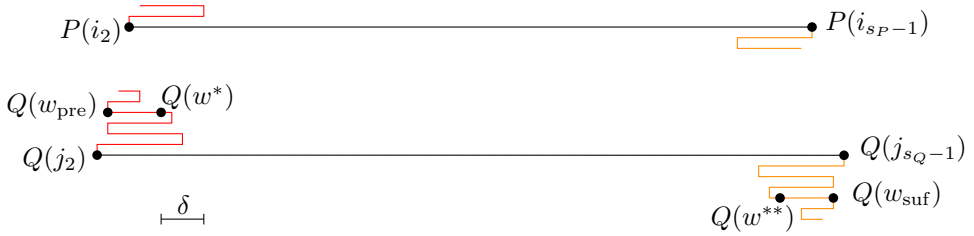


Figure 4 Minimal $\text{pre}(P)$-matcher on $\text{pre}(Q)$, w_{pre}, and minimal $\text{suf}(P)$-matcher on $\text{suf}(Q)$, w_{suf}.

3 The Static Algorithm

First, we discuss how the *modified free-space matrix* can be used to decide whether the Fréchet distance between two time series is at most a given threshold δ. Lemma 6 implies that, except for the beginning and the end, it is enough to look at pairs $(P(i), Q(j))$ of vertices, where one of the two vertices is a δ-signature vertex. Missing proofs for this section can be found in the full version.

▶ **Definition 9** (prefix and suffix). *Let P be a time series of complexity n and let $P(i_2)$ denote the second δ-signature vertex of P, and $P(i_{s_P-1})$ denote the penultimate δ-signature vertex of P. Then $P[1, i_2]$ is the prefix $\text{pre}(P)$ and $P[i_{s_P-1}, n]$ in reverse order defines the suffix $\text{suf}(P)$ of P. That is, the time series $P[i_{s_P-1}, n]$ is $\widehat{\text{suf}(P)}$, where $\widehat{P}$ denotes the time series defined by the vertices of P in reverse order.*

Observe that by the definition of the δ-signature vertices the time series $\text{pre}(P)$ and $\text{suf}(P)$ are each contained in some ball of radius δ.

▶ **Definition 10** (minimal matcher). *Let P and Q be two time series of complexity n and m respectively. The* minimal P-matcher on Q *is the smallest $w \in [m]$ such that*
1. $|P(n) - Q(w)| \leq \delta$, *and*
2. *there is a $w^* \in [w, \min(w+1, m)]$ such that $\text{d}_F(P, Q[1, w^*]) \leq \delta$.*

If no such value exists, we set it to ∞.

We denote by w_{pre} the minimal $\text{pre}(P)$-matcher on $\text{pre}(Q)$ and by v_{pre} the minimal $\text{pre}(Q)$-matcher on $\text{pre}(P)$. For the suffix, we denote by w_{suf} the index of the vertex in Q corresponding to the minimal $\text{suf}(P)$-matcher on $\text{suf}(Q)$ (and similarly v_{suf} when the roles of P and Q are swapped). See Figure 4 for an example.

Due to the next observation, we will only discuss how to process the prefix in Section 3.1, Section 4.2 and Section 5.

▶ **Observation 11.** *Let $\widehat{w_{\text{pre}}}$ be the minimal $\text{pre}(\widehat{P})$-matcher on $\text{pre}(\widehat{Q})$. If m is the complexity of Q, then $w_{\text{suf}} = m - (\widehat{w_{\text{pre}}} - 1)$.*

With these notions, we define the modified free-space matrix. See Figure 5 for an example.

▶ **Definition 12** (Modified Free-Space Matrix). *Let P and Q be two time series of complexity n and m. Further, let $\langle P(i_1), \ldots, P(i_{s_P})\rangle$ and $\langle Q(j_1), \ldots, Q(j_{s_Q})\rangle$ be their δ-signatures. We construct a matrix $M_\delta \in \{0,1\}^{n \times m}$, where the entry $M_\delta(i,j) = 1$ if*
a) *$P(i)$ and $Q(j)$ are both not δ-signature vertices, or*
b) *$|P(i) - Q(j)| \leq \delta$ and $(i,j) \notin [1, i_2] \times [1, j_2] \cup [i_{s_P-1}, n] \times [j_{s_Q-1}, m]$, or*
c) *$|P(i) - Q(j)| \leq \delta$ and $(i,j) \in \{(1,1), (n,m)\}$, or*
d) *$|P(i) - Q(j)| \leq \delta$ and $(i,j) = (i_2, j_2)$ and $d_F(\text{pre}(P), \text{pre}(Q)) \leq \delta$, or*

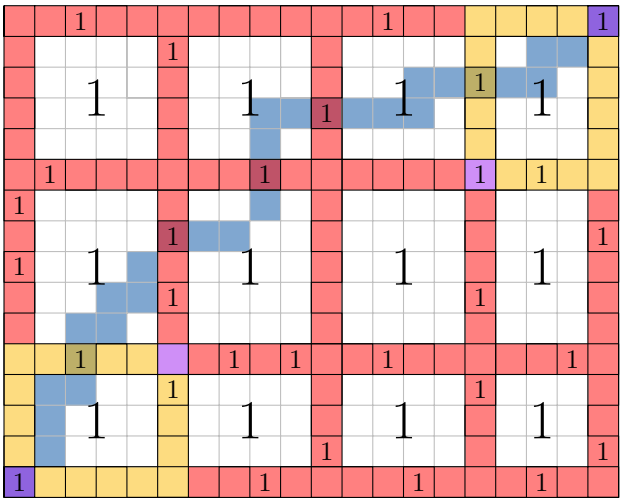


Figure 5 Example of a Modified Free-Space Matrix M_δ. The colored columns (resp. rows) correspond to the δ-signature vertices of P (resp. Q). The white entries are all 1 by Definition 12 a), the red entries are defined by b), the purple entries by c) and d) and e), and the yellow entries by f). The traversal drawn in blue uses only 1-entries of M_δ. Hence, $M_\delta(n, m)$ is reachable.

e) $|P(i) - Q(j)| \leq \delta$ *and* $(i, j) = (i_{s_P-1}, j_{s_Q-1})$ *and* $d_F(\mathrm{suf}(P), \mathrm{suf}(Q)) \leq \delta$, *or*
f) $(i, j) \in \{(i_2, w_{\mathrm{pre}}), (v_{\mathrm{pre}}, j_2), (i_{s_P-1}, w_{\mathrm{suf}}), (v_{\mathrm{suf}}, j_{s_Q-1})\}$.
Otherwise, the entry $M_\delta(i, j) = 0$. *Further, we say that an entry* $M_\delta(i, j)$ *is reachable if there exists a traversal* $(1, 1) = (a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k) = (i, j)$ *such that* $M_\delta(a_l, b_l) = 1$ *and* $(a_l, b_l) \in \{(a_{l-1} + 1, b_{l-1}), (a_{l-1}, b_{l-1} + 1), (a_{l-1} + 1, b_{l-1} + 1)\}$ *for all* $l = 2, \ldots, k$.

The importance of the modified free-space matrix is that it can be used to answer the decision problem of whether or not $d_F(P, Q) \leq \delta$ essentially in the same way as it is done for the discrete Fréchet distance using the free-space matrix. The next lemma follows mainly from a result of [6]. For completeness, a proof is contained in the full version.

▶ **Lemma 13.** *It holds that* $d_F(P, Q) \leq \delta$ *if and only if* $M_\delta(n, m)$ *is reachable. This can be checked in* $\mathcal{O}(nm)$ *time.*

3.1 Prefix and Suffix

This section is dedicated to identifying the minimal $\mathrm{pre}(P)$-matcher on $\mathrm{pre}(Q)$. We often use the fact that $\mathrm{pre}(P)$ and $\mathrm{pre}(Q)$ are each contained in some δ-ball. We will show that the vital behavior of such constricted time series can be captured by their extreme points.

▶ **Definition 14** (extreme point sequence). *Let* P *be a time series of complexity* n, *such that* $P(n)$ *is a global extremum of* P *and* P *is contained in some* δ*-ball. We define the extreme point sequence* $1 = a_1 < a_2 < \cdots < a_p = n$ *to be a sequence of indices, such that*
- *(range-preserving)* $\mathrm{im}(P[1, a_{k+1}]) = \overline{P(a_k)P(a_{k+1})}$ *for all* k, *and*
- *(extreme point)* $P(a_k) = \min(P[1, a_k])$ *or* $P(a_k) = \max(P[1, a_k])$.

See Figure 6 and 7 for an example of both an extreme point sequence and their associated preliminary assignments, defined next.

▶ **Definition 15** (preliminary assignment). *Let* P *and* Q *be two time series such that the image of each time series is contained in some ball of radius* δ. *Let* $a_1 < \ldots < a_p$ *and* $b_1 < \ldots < b_q$ *be extreme point sequences of time series* P *and* Q *respectively. Define the*

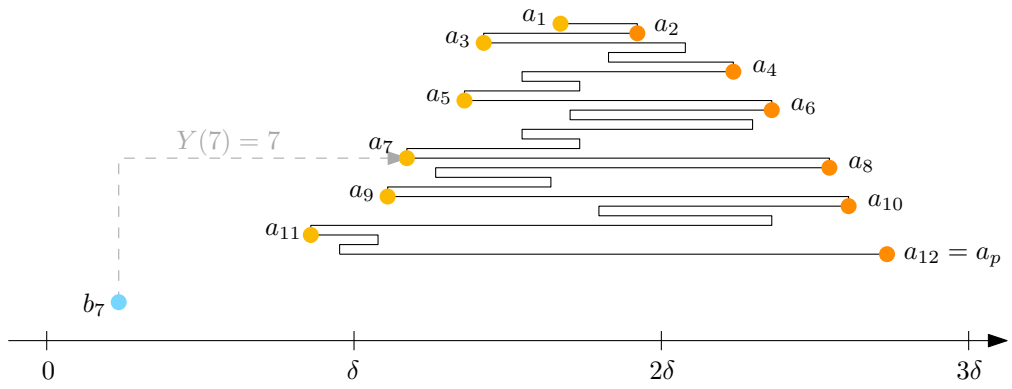


Figure 6 An extreme point sequence of the depicted time series P is $1 = a_1 < a_2 < \cdots < a_p$ and the preliminary assignment $Y(7)$ of Q on P is 7, since $Q(a_7)$ is the first vertex on P in $B(Q(b_7), \delta)$.

preliminary assignment $X(k)$ *of* P *on* Q *for every* $k \leq p$ *to be the smallest index such that* $|P(a_k) - Q(b_{X(k)})| \leq \delta$. *If no such index exists, we set* $X(k) = \infty$. *Let similarly* $Y(l)$ *be the preliminary assignment of* Q *on* P. *We say that* $X(k)$ *and* $Y(l)$ *form a* deadlock, *if* $l < X(k)$ *and* $k < Y(l)$.

Before we show how the preliminary assignment can be used to find the minimal pre(P)-matcher on pre(Q), we state some of its structural properties.

▶ **Lemma 16.** *Let* P *and* Q *be two time series such that the image of each time series is contained in some ball of radius* δ. *Let* $a_1 < \ldots < a_p$ *and* $b_1 < \ldots < b_q$ *be extreme point sequences of time series* P *and* Q. *If* $X(1) = 1$, *then the following holds.*

- **i)** *For any* $3 \leq k \leq p$ *it holds that* $X(k) \geq X(k-2)$.
- **ii)** *It holds that* $X(2k) = 1$ *for all* k *or* $X(2k-1) = 1$ *for all* k.
- **iii)** *If* $k < p$ *and* $X(k) \neq 1$, *then for any index* $l \geq X(k)$ *there is a* $t \in [b_{l-1}, b_l]$ *such that* $\mathrm{im}(P[1, a_{k+1}]) \subset B(Q(t), \delta)$.
- **iv)** *If* $X(k) > 1$ *and* $Y(l)$ *form a deadlock, then* $X(k)$ *and* $Y(X(k)-1)$ *form a deadlock.*

The importance of deadlocks is summarized in the following pivotal lemma.

▶ **Lemma 17.** *Let* P *and* Q *be time series such that the image of each time series is contained in some ball of radius* δ. *Let* $a_1 < \ldots < a_p$ *and* $b_1 < \ldots < b_q$ *be extreme point sequences of* P *and* Q. *For any* $w^* \leq b_q$, *it holds that* $d_F(P[1, a_p], Q[1, w^*]) \leq \delta$ *if and only if*

- **(i)** $|P(1) - Q(1)| \leq \delta$ *and* $|P(a_p) - Q(w^*)| \leq \delta$,
- **(ii)** $\mathrm{im}(P[1, a_p]) \subset B(Q[1, w^*], \delta)$,
- **(iii)** $\mathrm{im}(Q[1, w^*]) \subset B(P[1, a_p], \delta)$, *and*
- **(iv)** $X(k)$ *and* $Y(l)$ *do not form a deadlock for all* $k = 1, \ldots, p$ *and* $l = 1, \ldots, q$.

Observe that if $X(1)$ and $Y(1)$ do not form a deadlock then already $|P(1) - Q(1)| \leq \delta$.

Proof Sketch. First assume that $d_F(P[1, a_p], Q[1, w^*]) \leq \delta$. Then Condition (i)-(iii) hold by definition. Observe that if $X(k)$ and $Y(l)$ form a deadlock, then $|P(a_k) - Q(t)| > \delta$ for all $t \leq b_l$. However, in this case any traversal of P and Q would have to match $Q(b_l)$ with some $P(s)$ with $s < a_k$, since the part of Q before $Q(b_l)$ has to be traversed before reaching $P(a_k)$ as $d_F(P[1, a_p], Q[1, w^*]) \leq \delta$. As $X(k)$ and $Y(l)$ form a deadlock, also $|Q(b_l) - P(s)| > \delta$ for all $s \leq a_k$. Therefore, any traversal induces a cost greater than δ. Thus $X(k)$ and $Y(l)$ cannot form a deadlock, implying Condition (iv).

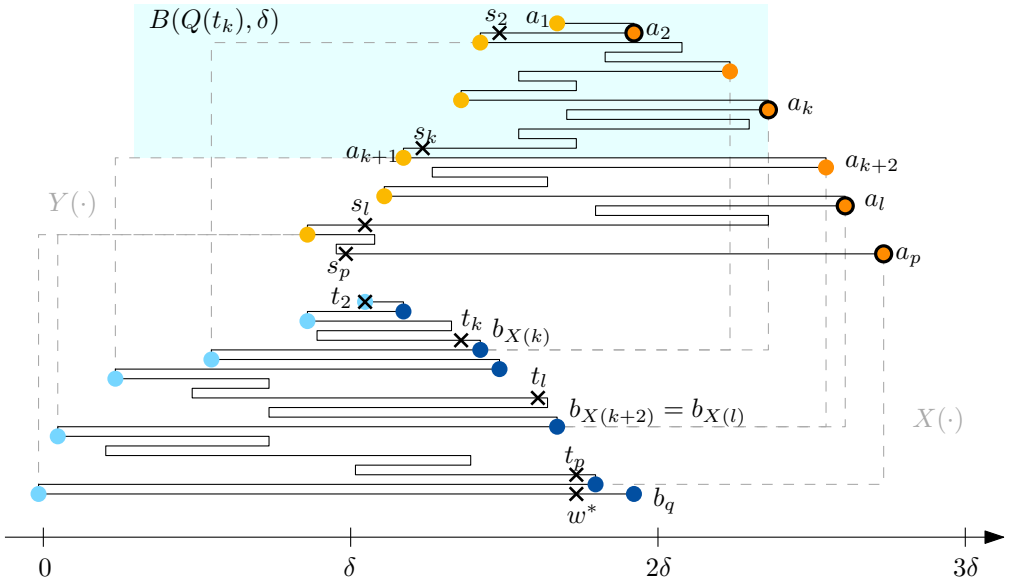


Figure 7 The gray dashed lines mark the preliminary assignments $X(\cdot)$ and $Y(\cdot)$. The points defined in the proof of Lemma 17 are marked. The circled points are the transition indices.

Now assume that Condition (i)-(iv) hold. Then, we explicitly construct a traversal of the two time series P and Q with cost $\leq \delta$. If $X \equiv 1$, then by Condition (iii) and (iv) there exists a point $s \in [a_{p-1}, a_p]$, such that $\mathrm{im}(Q[1, w^*]) \subset B(P(s), \delta)$ and $\mathrm{im}(P[s, a_p]) \subset B(Q(w^*), \delta)$. Then, matching $P[1, s]$ to $Q(1)$, and $Q[1, w^*]$ to $P(s)$, and $P[s, a_p]$ to $Q(w^*)$ results in a matching with cost $\leq \delta$. Otherwise, $X \not\equiv 1$. Let $\mathcal{K} = \{k \leq p-2 \mid X(k) \neq X(k+2)\} \cup \{p-1 \leq k \mid X(k) \neq 1\}$ be the set of *transition indices*, at which $X(k)$ is about to change when ignoring all 1-entries. To construct a matching between $P[1, a_p]$ and $Q[1, w^*]$, we define values $s_k \in [1, a_p]$ and $t_k \in [1, w^*]$ for all transition indices $k \in \mathcal{K}$ with the following properties. See Figure 7 and 8. The existence of those values is shown in the full version.

a) There is a $t_k \in [b_{X(k)-1}, \min\{w^*, b_{X(k)}\}]$ such that $\mathrm{im}(P[1, a_{k+1}]) \subset B(Q(t_k), \delta)$, with $a_{p+1} = a_p$.

b) If $k < p-1$, then there is an $s_k \in [a_k, a_{k+1}]$ such that $\mathrm{im}(Q[t_k, t_l]) \subset B(P(s_k), \delta)$, where l is the transition index after k.

c) If $k \geq p-1$, then there is an $s_k \in [a_{p-1}, a_p]$ such that $\mathrm{im}(Q[t_k, w^*]) \subset B(P(s_k), \delta)$ and $\mathrm{im}(P[s_k, a_p]) \subset B(Q(w^*), \delta)$.

Utilizing a)-d), we construct a traversal of P and Q as follows: Let k be the smallest transition index. Match $P[1, s_k]$ to $Q(t_k)$ with $t_k = 1$ via a). Now let k be the last handled transition index and let l be the next transition index after k. First match $Q[t_k, t_l]$ to $P(s_k)$ via b) (lightblue in Figure 8) and then match $P[s_k, s_l]$ to $Q(t_l)$ via a) (yellow in Figure 8). Lastly, let k be the last transition index. So far, the constructed traversal matches $Q[1, t_k]$ to $P[1, s_k]$. We now match $Q(t_k, w^*)$ to $P(s_k)$ and lastly $P[s_k, a_p]$ to $Q(w^*)$ via d) (darkblue and orange in Figure 8). This yields a traversal with cost $\leq \delta$, which concludes the proof. ◀

We use the next lemma to decide whether a minimal $\mathrm{pre}(P)$-matcher on $\mathrm{pre}(Q)$ exists.

▶ **Lemma 18.** *Let P and Q be two time series and $P(i_2)$ be the second δ-signature vertex of P and $Q(j_2)$ of Q. There exists a minimal $\mathrm{pre}(P)$-matcher on $\mathrm{pre}(Q)$ if and only if $X(k) < \infty$, and $X(k)$ and $Y(l)$ do not form a deadlock for all $k \in [p]$ and $l \in [q]$. If it exists, it is*

$$\min\{w = 1, \ldots, j_2 \mid |P(i_2) - Q(w)| \leq \delta,\ \mathrm{im}(\mathrm{pre}(P)) \subset B(Q([1, w+1]), \delta)\}.$$

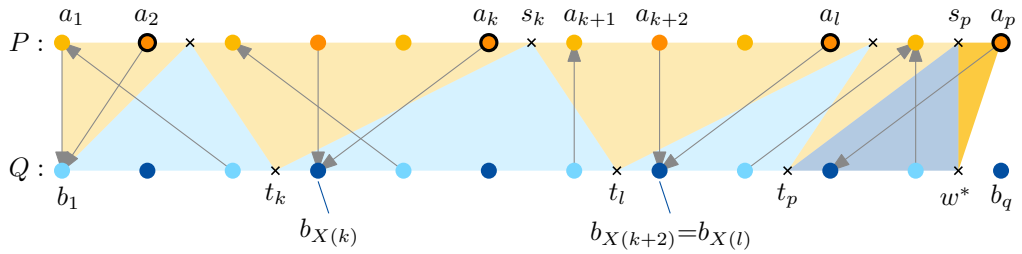


Figure 8 The transition indices are circled in black. The arrows link a_k and $X(k)$ as well as b_k and $Y(k)$ and for all a_k (resp. b_k) without an outgoing arrow it holds $X(k) = 1$ (resp. $Y(k) = 1$). The traversal described in the proof of Lemma 17 is visualized by the triangles.

Finally, we are ready to show how the minimal pre(P)-matcher on pre(Q) can be computed.

▶ **Lemma 19.** *Let P and Q be time series and let* im($Q[1, j]$) *be given for every $j = 1, \dots, q$. Further, assume the preliminary assignments X and Y of* pre(P) *and* pre(Q) *do not form a deadlock. Then we can compute the minimal* pre(P)*-matcher on* pre(Q) *in $\mathcal{O}(\log n)$ time.*

▶ **Lemma 20.** *Let P and Q be two time series and let extreme point sequences of* pre(P) *and* pre(Q) *be given. If the preliminary assignments of* pre(P) *and* pre(Q) *do not form a deadlock, then we can decide whether $d_F(\text{pre}(P), \text{pre}(Q)) \leq \delta$ in $\mathcal{O}(1)$ time. Otherwise, it holds that $d_F(\text{pre}(P), \text{pre}(Q)) > \delta$.*

The results of this section show how, using deadlocks, we can fill in the modified free-space matrix. In the following, we show that deadlocks allow efficient processing of updates of the transformations, which we crucially exploit for our algorithms.

4 1D Continuous Fréchet Distance Under Translation

Missing proofs for the statements in this section can be found in the full version.We first describe the reduction of the 1D continuous Fréchet distance under translation to an offline dynamic grid graph reachability problem. Subsequently we describe how we solve this problem for two time series P and Q both of complexity n in time $\mathcal{O}(n^{8/3} \log^3 n)$.

4.1 The $\mathcal{O}(n^2)$ Events

In this section, we show how to compute a set of $\mathcal{O}(n^2)$ translation values such that the Fréchet distance under translation between P and Q is at most δ if and only if there exists a translation value t in this set such that the Fréchet distance between P and Q_t is at most δ. We start by observing that it is sufficient to consider vertex-vertex events.

▶ **Lemma 21.** *Let $M(P, Q) = (m_{i,j})_{(i,j)\in[n]\times[n]}$ be the matrix where $m_{i,j} = \mathbb{1}_{|P(i)-Q(j)|\leq\delta}$. Let $t, t' \in \mathbb{R}$ be given. If $M(P, Q_t) = M(P, Q_{t'})$, then $d_F(P, Q_t) \leq \delta \iff d_F(P, Q_{t'}) \leq \delta$.*

▶ **Observation 22.** *Let P and Q be two time series of complexity n. There are at most $\mathcal{O}(n^2)$ many values t such that there are values $i, j \in [n]$ such that $|P(i)-Q_t(j)| = \delta$. These values are precisely the interval boundaries of intervals of the form $[P(i) - Q(j) - \delta, P(i) - Q(j) + \delta]$.*

We briefly discuss how to deal with degenerate cases, that is, cases where it is not true that for every $t \in \mathbb{R}$ exactly two values $i \neq j \in [n]$ exist such that $|P(i) - Q_t(j)| = \delta$, in the full version.

▶ **Corollary 23** (translation representatives). *There exist a sorted set $\mathfrak{T} \subset \mathbb{R}$ containing $\mathcal{O}(n^2)$ points and computable in $\mathcal{O}(n^2 \log n)$ time with the following properties.*

- *It holds that $d_F^T(P,Q) \leq \delta$ if and only if $\exists t \in \mathfrak{T}$ such that $d_F(P,Q_t) \leq \delta$.*
- *For two consecutive t, t' in $\mathfrak{T}$, there exist only one pair of indices $k, l \in [n]$ such that $|P(k) - Q_t(l)| \leq \delta$ and $|P(k) - Q_{t'}(l)| > \delta$, or the other way round. Further, for the set of all pairs of consecutive $t, t' \in \mathfrak{T}$, those indices can be computed in $\mathcal{O}(n^2 \log n)$ time.*

We call the set $\mathfrak{T}$ of Corollary 23 $\mathcal{O}(n^2)$ points the *translation representatives*.

4.2 Prefix and Suffix under Translation

In Section 3.1, we showed that the preliminary assignment can be used to determine the minimal $\mathrm{pre}(P)$-matcher on $\mathrm{pre}(Q)$. In particular, the minimal $\mathrm{pre}(P)$-matcher on $\mathrm{pre}(Q)$ can be computed in $\mathcal{O}(\log n)$ time, if we know whether there exists a deadlock or not. Therefore, we define the translated preliminary assignment and show how to keep track of whether a deadlock exists while translating the time series Q with the values in $\mathfrak{T}$.

▶ **Definition 24** (translated preliminary assignment). *Let P and Q be time series and let $t \in \mathbb{R}$. Define the* translated preliminary assignment *X_t to be the preliminary assignment of $\mathrm{pre}(P)$ on $\mathrm{pre}(Q_t)$. Let similarly Y_t be the preliminary assignment of $\mathrm{pre}(Q_t)$ on $\mathrm{pre}(P)$. We say that $X_t(k)$ and $Y_t(l)$ form a* deadlock*, if $l < X_t(k)$ and $k < Y_t(l)$.*

Given two time series P and Q together with extreme point sequences $a_1, \ldots, a_p$ and $b_1, \ldots, b_q$ of $\mathrm{pre}(P)$ and $\mathrm{pre}(Q)$, we show that we can compute the set $\mathcal{T} \subset \mathfrak{T}$ of translations for which X_t and Y_t do not form a deadlock in total time $\mathcal{O}(n^2 \log n)$.

▶ **Observation 25.** *For every $t \in \mathbb{R}$, the extreme point sequences of Q_t and Q coincide.*

▶ **Observation 26.** $\mathcal{T} \subset \{t \in \mathbb{R} | X_t(1) = 1\} = \{t \in \mathbb{R} | Y_t(1) = 1\}$.

The next observation and next lemma show that the preliminary assignments of two consecutive translations of the translation representatives differ only slightly.

▶ **Observation 27.** *Let two consecutive translation representatives t, t' be given. Then, X_t and $X_{t'}$ (resp. Y_t and $Y_{t'}$) agree everywhere except at the index that participates in the point-point distance inducing the arrangement boundary separating t and t'.*

▶ **Lemma 28.** *Let two consecutive translation representatives t, t' be given. Let k be the index of P and l the index of Q participating in the point-point distance inducing the arrangement boundary separating t and t'. Then, it holds that*

1. $|Y_t(l) - Y_{t'}(l)| \leq 2$,
2. $Y_t(l) = \infty$ *and* $Y_{t'}(l) \in \{p-1, p\}$, *or*
3. $Y_{t'}(l) = \infty$ *and* $Y_t(l) \in \{p-1, p\}$.

Similarly, it holds that

1. $|X_t(k) - X_{t'}(k)| \leq 2$,
2. $X_t(k) = \infty$ *and* $X_{t'}(k) \in \{q-1, q\}$, *or*
3. $X_{t'}(k) = \infty$ *and* $X_t(k) \in \{q-1, q\}$.

▶ **Lemma 29.** *There is an algorithm that correctly computes the set $\mathcal{T}$ of translations $t \in \mathfrak{T}$ for which X_t and Y_t do not form a deadlock in $\mathcal{O}(n^2 \log n)$. This is the set of translations in $\mathfrak{T}$ for which the preliminary assignments of $\mathrm{pre}(P)$ and $\mathrm{pre}(Q_t)$ do not form a deadlock. Similarly, we can compute the set of translations in $\mathfrak{T}$ for which $\mathrm{suf}(P)$ and $\mathrm{suf}(Q_t)$ do not form a deadlock in $\mathcal{O}(n^2 \log n)$ time.*

4.3 Computing the Fréchet Distance under Translation

We are now ready to prove our result for the decision problem of the Fréchet distance under translation of time series.

▶ **Lemma 30.** *We can decide the Fréchet distance under translation between two time series in time* $\mathcal{O}(n^{8/3}\log^2 n)$.

Proof. The goal is to apply Theorem 8 to maintain reachability in the modified free-space matrix defined in Definition 12 for all translations in $\mathfrak{T}$ from Corollary 23.

In a pre-processing step, we first compute the signatures of both input time series and extreme point sequences of $\text{pre}(P), \text{pre}(Q), \text{suf}(P), \text{suf}(Q)$, all in time $\mathcal{O}(n)$. We use Corollary 23 to compute the set $\mathfrak{T}$ of candidate translations that we have to check reachability for, and fix the order in which we check them (e.g., from left to right). Additionally, we invoke Lemma 29 to compute the set $\mathcal{T} \subset \mathfrak{T}$ for which there are no deadlocks of the prefixes and suffixes. The above steps take time $\mathcal{O}(n^2 \log n)$ each. It now suffices to show that if we consider the translations $\mathfrak{T}$ in order, there are only $\mathcal{O}(1)$ many updates per translation to the modified free-space matrix of Definition 12, for each type of entry (a) to (f). Furthermore, we need to determine all of these updates in time $\mathcal{O}(n^2 \log n)$.

Combining Corollary 23 and the information from the pre-computed signatures, we can pre-compute the $\mathcal{O}(1)$ many updates per translation in $\mathfrak{T}$ of type (a),(b), and (c) in time $\mathcal{O}(n^2 \log n)$. For entries of type (d) and (e), we first check whether the translation at hand incurs a deadlock or not (i.e., we check for containment in the pre-computed set $\mathcal{T}$). If it does not incur a deadlock, then we use Lemma 20 to determine in constant time whether $d_F(\text{pre}(P), \text{pre}(Q)) \leq \delta$ and $d_F(\text{suf}(P), \text{suf}(Q)) \leq \delta$. For type (f), we use the pre-computed signatures and Lemma 19 to compute $w_{\text{pre}}, v_{\text{pre}}, w_{\text{suf}}, v_{\text{suf}}$ in $\mathcal{O}(\log n)$ time.

Note that by the above, each entry type incurs at most $\mathcal{O}(1)$ updates per translation, and we therefore have $\mathcal{O}(n^2)$ updates in total. All of these updates can be pre-determined, i.e., they are offline. Consequently, we can apply Theorem 8 and obtain a running time of $\mathcal{O}(n^2 \cdot n^{2/3} \log^2 n) = \mathcal{O}(n^{8/3} \log^2 n)$. ◀

Using parametric search, the standard technique for solving the optimization problem of computing the Fréchet distance from the solution to the decision problem, we obtain the following theorem, proved in more detail in the full version.

▶ **Theorem 1.** *There exists an algorithm to compute the continuous Fréchet distance under translation between two time series of complexity* n *in time* $\mathcal{O}(n^{8/3}\log^3 n)$.

5 1D Continuous Fréchet Distance Under Scaling

Our algorithm for the Fréchet distance under scaling works in a similar way as our algorithm for translations. An additional obstacle to overcome is that scaling a time series can change the δ-signature. The next lemma shows that the inclusion of δ-signature vertices is monotonously increasing in the scaling. We can prove it using the observation that, if Q' is the δ-signature of Q, then sQ' is the δ-signature of sQ and previous results about signatures in [22].

▶ **Lemma 31.** *We can compute in* $\mathcal{O}(n \log n)$ *time all values* s_j *with* $j = 1, \ldots, n$ *such that the following holds. The vertex* $sQ(j)$ *is a* δ*-signature vertex of* sQ *if and only if* $s > s_j$.

As a result, by adding the events where the δ-signature changes to the arrangement for the scaling parameters, the decision problem can be solved similarly to the translation case, yielding the following theorem. See the full version for details.

▶ **Theorem 2.** *There exists an algorithm to compute the continuous Fréchet distance under scaling between two time series of complexity* n *in time* $\mathcal{O}(n^{8/3}\log^3 n)$.

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1–2):75–91, 1995. doi:10.1142/S0218195995000064.

2 Helmut Alt, Christian Knauer, and Carola Wenk. Matching polygonal curves with respect to the Fréchet distance. In Afonso Ferreira and Horst Reichel, editors, *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science, Dresden, Germany, February 15-17, 2001, Proceedings*, volume 2010 of *Lecture Notes in Computer Science*, pages 63–74. Springer, 2001. doi:10.1007/3-540-44693-1_6.

3 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004. doi:10.1007/S00453-003-1042-5.

4 Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. A faster algorithm for the discrete Fréchet distance under translation. *CoRR*, abs/1501.03724, 2015. arXiv:1501.03724.

5 Abdullah Biran and Aleksandar Jeremic. Ecg bio-identification using Fréchet classifiers: A proposed methodology based on modeling the dynamic change of the ecg features. *Biomedical Signal Processing and Control*, 82:104575, 2023. doi:10.1016/j.bspc.2023.104575.

6 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.

7 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proc. 31st International Conf. Very Large Data Bases (VLDB'05)*, pages 853–864, 2005. URL: http://www.vldb.org/archives/website/2005/program/paper/fri/p853-brakatsoulas.pdf.

8 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Ann. IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

9 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.

10 Karl Bringmann, Marvin Künnemann, and André Nusser. When Lipschitz walks your dog: Algorithm engineering of the discrete Fréchet distance under translation. In *28th Annual European Symposium on Algorithms (ESA)*, volume 173 of *LIPIcs*, pages 25:1–25:17, 2020. doi:10.4230/LIPIcs.ESA.2020.25.

11 Karl Bringmann, Marvin Künnemann, and André Nusser. Discrete Fréchet distance under translation: Conditional hardness and an improved algorithm. *ACM Trans. Algorithms*, 17(3):25:1–25:42, 2021. doi:10.1145/3460656.

12 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog - with an application to Alt's conjecture. In *Proc. 25th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA'14)*, pages 1399–1413, 2014. doi:10.1137/1.9781611973402.103.

13 Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 496–499. ACM, 2019. doi:10.1145/3347146.3359111.

14 Kevin Buchin, André Nusser, and Sampson Wong. Computing continuous dynamic time warping of time series in polynomial time. 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 22:1–22:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.SOCG.2022.22.

15 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.

16 Siu-Wing Cheng and Haoqiang Huang. Fréchet distance in subquadratic time. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5100–5113, 2025. doi:10.1137/1.9781611978322.173.

17 Ian R. Cleasby, Ewan D. Wakefield, Barbara J. Morrissey, Thomas W. Bodey, Steven C. Votier, Stuart Bearhop, and Keith C. Hamer. Using time-series similarity measures to compare animal movement trajectories in ecology. *Behavioral Ecology and Sociobiology*, 73(11):151, November 2019. doi:10.1007/s00265-019-2761-1.

18 Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34(1):200–208, 1987. doi:10.1145/7531.7537.

19 Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. ($1 + \epsilon$) -ANN Data Structure for Curves via Subspaces of Bounded Doubling Dimension. *Computing in Geometry and Topology*, 3(2):1–22, 2024. doi:10.57717/cgt.v3i2.45.

20 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.

21 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, SoCG '10, pages 365–374, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1810959.1811019.

22 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. 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 766–785. SIAM, 2016. doi:10.1137/1.9781611974331.CH55.

23 Alon Efrat, Piotr Indyk, and Suresh Venkatasubramanian. Pattern matching for sets of segments. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 295–304, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365463.

24 Omrit Filtser and Matthew J. Katz. Algorithms for the discrete Fréchet distance under translation. *J. Comput. Geom.*, 11(1):156–175, 2020. doi:10.20382/JOCG.V11I1A7.

25 Omrit Filtser and Matthew J. Katz. Variants of the discrete Fréchet distance under translation. *Journal of Computational Geometry*, 11(1):156–175, 2020.

26 M Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, 1906.

27 Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1):3:1–3:22, 2014. doi:10.1145/2532646.

28 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure–structure alignment with discrete Fréchet distance. *J. Bioinformatics and Computational Biology*, 6(01):51–64, 2008. doi:10.1142/S0219720008003278.

29 Hongli Niu and Jun Wang. Volatility clustering and long memory of financial time series and financial price model. *Digital Signal Processing*, 23(2):489–498, 2013. doi:10.1016/j.dsp.2012.11.004.

30 Carola Wenk. *Shape matching in higher dimensions*. PhD thesis, Freie Universität Berlin, 2002. PhD Thesis.

On the Twin-Width of Smooth Manifolds

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Kristóf Huszár
Institute of Geometry, Graz University of Technology, Austria

Abstract

Building on Whitney's classical method of triangulating smooth manifolds, we show that every compact d-dimensional smooth manifold admits a triangulation with dual graph of twin-width at most $d^{O(d)}$. In particular, it follows that every compact 3-manifold has a triangulation with dual graph of bounded twin-width. This is in sharp contrast to the case of treewidth, where for any natural number n there exists a closed 3-manifold such that every triangulation thereof has dual graph with treewidth at least n. To establish this result, we bound the twin-width of the dual graph of the d-skeleton of the second barycentric subdivision of the $2d$-dimensional hypercubic honeycomb. We also show that every compact, piecewise-linear (hence smooth) d-dimensional manifold has triangulations where the dual graph has an arbitrarily large twin-width.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Mathematics of computing → Geometric topology

Keywords and phrases Smooth manifolds, triangulations, twin-width, Whitney embedding theorem, structural graph parameters, computational topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.23

Related Version *Full Version*: `https://arxiv.org/abs/2407.10174` [10]

Funding The authors have been supported by the French National Research Agency through the project TWIN-WIDTH with reference number ANR-21-CE48-0014.
Kristóf Huszár: Partially supported by the Austrian Science Fund (FWF) grant P 33765-N.

Acknowledgements We thank the anonymous reviewers for their helpful comments and for raising interesting questions for future research.

1 Introduction

Structural graph parameters have become increasingly important in computational topology in the past two decades. This is mainly due to the emergence of fixed-parameter tractable (FPT) algorithms for problems on knots, links [13, 36, 37, 39] and 3-manifolds [15, 17, 18, 19, 20],[1] most of which are known to be NP-hard in general. Although these FPT algorithms may have exponential worst-case running time, on inputs with bounded *treewidth* they are guaranteed to terminate in polynomial (or even in linear) time.[2] In addition, some of these algorithms have been implemented in software packages such as `Regina`, providing practical tools for researchers in topology [12, 14].[3]

[1] Also see [2] for an FPT algorithm checking tightness of (weak) pseudomanifolds in arbitrary dimensions.
[2] Here the term *input* refers either to a link diagram $\mathcal{D}$, or to a 3-manifold triangulation $\mathcal{T}$. In the first case the treewidth means the treewidth of $\mathcal{D}$ considered as a 4-regular graph, in the second case it means the treewidth of the dual graph $\Gamma(\mathcal{T})$ of $\mathcal{T}$. The running times are measured in terms of the *size* of the input. The size of a link diagram is defined as the number of its crossings, and the size of a 3-manifold triangulation is the number of its tetrahedra. More definitions are given in Section 2.
[3] See [3] for an implemented algorithm to effectively compute certain Khovanov homology groups of knots. This algorithm is conjectured to be FPT in the *cutwidth* of the input knot diagram, cf. [3, Section 6].

© Édouard Bonnet and Kristóf Huszár;
licensed under Creative Commons License CC-BY 4.0

The success of the above algorithms naturally leads to the following question. Given a 3-manifold $\mathcal{M}$ (resp. knot $\mathcal{K}$), what is the smallest treewidth that the dual graph of a triangulation of $\mathcal{M}$ (resp. a diagram of $\mathcal{K}$) may have?[4] Motivated by this challenge, in recent years several results have been obtained that reveal quantitative connections between *topological invariants* of knots and 3-manifolds, and *width parameters* associated with their diagrams [22, 35] and triangulations [26, 27, 28, 29, 30, 40], respectively. It turns out that topological properties of 3-manifolds may prohibit the existence of "thin" triangulations.[5,6] At the same time, geometric or topological descriptions of 3-manifolds can also give strong hints on how to triangulate them so that their dual graphs have constant pathwidth or treewidth, or at least bounded in terms of a topological invariant of these 3-manifolds.[7]

In this work we establish similar results for another graph parameter called *twin-width*. Introduced in [11], this notion has been subject of growing interest and found many algorithmic applications in recent years.[8] Namely, on classes of effectively[9] bounded twin-width, first-order properties can be decided efficiently[10] [11] (also see [8] for improved running times on specific problems definable in first-order logic), first-order queries can be enumerated fast [24], and enhanced approximation algorithms can be designed for several graph optimization problems [4]. Besides, classes of bounded twin-width are fairly general and broad. They for instance include classes of bounded tree-width, and even its dense analogue, clique-width, classes excluding a minor, proper permutation classes, d-dimensional grid graphs [11], some classes of cubic expanders [9], segment intersection graphs without biclique subgraphs of a fixed size [7], and modifications definable in first-order logic (called first-order *transductions*) of all these classes [11]. We observe that the definition of twin-width can readily be lifted to binary structures (i.e., edge-colored multigraphs).

Our first result shows that a compact d-dimensional smooth manifold always has a triangulation with dual graph of twin-width bounded in terms of d.

▶ **Theorem 1.** *Any compact d-dimensional smooth manifold $\mathcal{M}$ admits a triangulation (more specifically, a* Whitney triangulation[11]*) with dual graph of twin-width at most $d^{O(d)}$.*

Let us emphasize that, in an appropriate computational model, Whitney triangulations of smooth manifolds can be constructed algorithmically [5]. As every 3-manifold is smooth [41] (see also [38]), the next corollary follows immediately from Theorem 1. We recall that $\mathrm{tww}(G)$ denotes the twin-width of a graph G, and $\Gamma(\mathcal{T})$ denotes the dual graph of a triangulation $\mathcal{T}$.

▶ **Corollary 2.** *There exists a universal constant $C > 0$ such that every compact 3-dimensional manifold $\mathcal{M}$ admits a triangulation $\mathcal{T}$ with $\mathrm{tww}(\Gamma(\mathcal{T})) \leqslant C$.*

---

[4] For links and knots, this question was respectively asked in [37, Section 4] and [16, p. 2694].

[5] For non-Haken 3-manifolds of large Heegaard genus [30] or Haken 3-manifolds with a "complicated" JSJ decomposition [29], the dual graph of *any* triangulation must also have large treewidth.

[6] For results where the treewidth of a knot diagram is bounded below by topological properties of the underlying knot, see [22, 35].

[7] This is case with Seifert fibered spaces [28] or hyperbolic 3-manifolds [27, 40].

[8] For an introduction to twin-width and an overview of its applications, see [6] and the references therein.

[9] A class has *effectively bounded twin-width* if it has bounded twin-width, and contraction sequences of width $O(1)$ (objects witnessing the twin-width upper bound) can be found in polynomial time; see Section 2.1 for the definitions of *contraction sequences* and *twin-width*.

[10] More precisely, there is a *fixed-parameter tractable* algorithm that, given a first-order sentence φ and an n-vertex graph G with a contraction sequence of width d, decides if G satisfies φ in time $f(\varphi, d) \cdot n$, for some computable function f.

[11] We call a triangulation $\mathcal{T}$ of a compact smooth manifold $\mathcal{M}$ a *Whitney triangulation* of $\mathcal{M}$, if $\mathcal{T}$ is obtained via Whitney's method discussed in Section 3.

This is in sharp contrast to the case of treewidth, for which it is known that for every $n \in \mathbb{N}$ there are infinitely many 3-manifolds where the smallest treewidth of the dual graph of *every* triangulation is at least n [29, 30]. Complementing Theorem 1, we also show that for any fixed $d \geqslant 3$, the d-dimensional triangulations of large twin-width are abundant (Theorem 19). Moreover we show that any piecewise-linear (hence smooth) manifold of dimension at least three admits triangulations with dual graph of arbitrarily large twin-width.

▶ **Theorem 3.** *Let $d \geqslant 3$ be an integer. For every compact d-dimensional piecewise-linear manifold $\mathcal{M}$ and natural number $n \in \mathbb{N}$, there is a triangulation $\mathcal{T}$ of $\mathcal{M}$ with* $\mathrm{tww}(\Gamma(\mathcal{T})) \geqslant n$.

▶ Remark 4. The assumption of $d \geqslant 3$ in Theorem 3 is essential. Indeed, the dual graph of any triangulation of the genus-g surface S_g is, in particular, a graph that embeds into S_g and such graphs are known to have twin-width bounded above by $c(\sqrt{g}+1)$ for some universal constant $c > 0$ [32].[12]

Outline of the paper. In Section 2 we review the relevant notions from graph theory and topology. In Section 3 we recall Whitney's seminal work on triangulating smooth manifolds. This is followed by a detailed proof of Theorem 1 in Section 4. Finally, in Section 5 we prove the complementary results about triangulations with dual graph of large twin-width.

2 Preliminaries

Basic notation. For a finite set S we let $|S|$ denote its cardinality, while for a real number x we let $|x|$ denote its absolute value. For a positive integer $k \leqslant |S|$, we let $\binom{S}{k}$ denote the set of k-element subsets of S. For a positive integer n, we let $[n]$ denote the set of all positive integers up to n, and for two real numbers $a \leqslant b$, we use $[a, b]$ to denote the closed interval $\{x \in \mathbb{R} : a \leqslant x \leqslant b\}$. For a vector $y = (y_1, \ldots, y_d) \in \mathbb{R}^d$ we use $\|y\|$ to denote its Euclidean norm, i.e, $\|y\|^2 = \sum_{i=1}^{d} y_i^2$. However, if $\mathcal{X}$ is a (cubical or simplicial) complex, then $\|\mathcal{X}\|$ refers to its geometric realization.

2.1 Trigraphs, contraction sequences and twin-width

Following [11, Sections 3 and 4], in this section we review the graph-theoretic notions central to our work and collect some basic, yet important facts about twin-width.

Trigraphs. A *trigraph* G is a triple $G = (V, E, R)$, where V is a finite set of *vertices*, and $E, R \subseteq \binom{V}{2}$ are two disjoint subsets of pairs of vertices called *black edges* and *red edges*, respectively. We also refer to the sets of vertices, black edges and red edges of a given trigraph G as $V(G)$, $E(G)$ and $R(G)$, respectively. Any simple graph $G = (V, E)$ may be regarded as a trigraph (V, E, R) with $R = \emptyset$. For a vertex $v \in V(G)$ the *degree* $\deg(v)$ *of* v is the number of edges incident to it, i.e., $\deg(v) = |\{e \in E \cup R : v \in e\}|$. Additionally, the *red degree* $\deg_R(v)$ *of* v is the number of red edges incident to it, i.e., $\deg_R(v) = |\{e \in R : v \in e\}|$. A trigraph G for which $\deg_R(G) = \max_{v \in V(G)} \deg_R(v) \leqslant b$ is called a *b-trigraph*. Given two trigraphs $G = (V, E, R)$ and $G' = (V', E', R')$, we say that G' is a *subtrigraph* of G, if $V' \subseteq V$, $E' \subseteq E \cap \binom{V'}{2}$ and $R' \subseteq R \cap \binom{V'}{2}$.[13] In addition, if $E' = E \cap \binom{V'}{2}$ and $R' = R \cap \binom{V'}{2}$, then we say that G' is an *induced subtrigraph* of G. For a trigraph G and a subset $S \subseteq V(G)$ of its vertices, $G - S$ denotes the induced subtrigraph of G with vertex set $V(G) \setminus S$.

[12] This bound is sharp up to a constant multiplicative factor [32]. For $g = 0$, we know that planar graphs have twin-width at most eight [25], and there are planar graphs with twin-width equal to seven [31].
[13] As usual, subtrigraphs of graphs (those without any red edges) will also be called subgraphs.

Contraction sequences and twin-width. Let $G = (V, E, R)$ be a trigraph and $u, v \in V$ be two arbitrary distinct vertices of G. We say that the trigraph $G/u, v = (V', E', R')$ is obtained from G by *contracting* u and v into a new vertex w if **1.** $V' = (V \setminus \{u, v\}) \cup \{w\}$, **2.** $G - \{u, v\} = (G/u, v) - \{w\}$ and **3.** for any $x \in V' \setminus \{w\} = V \setminus \{u, v\}$ we have

- $\{w, x\} \in E'$ if and only if $\{u, x\} \in E$ and $\{v, x\} \in E$,
- $\{w, x\} \notin E' \cup R'$ if and only if $\{u, x\} \notin E \cup R$ and $\{v, x\} \notin E \cup R$, and
- $\{w, x\} \in R'$ otherwise.

We call the trigraph $G/u, v$ a *contraction* of G. A sequence $\mathfrak{S} = (G_1, \ldots, G_m)$ of trigraphs is a *contraction sequence* if G_{i+1} is a contraction of G_i for every $1 \leqslant i \leqslant m - 1$. Note that $|V(G_{i+1})| = |V(G_i)| - 1$. We use the notation "$\mathfrak{S} \colon G_1 \rightsquigarrow G_m$" to indicate that the trigraphs G_1 and G_m are initial and terminal entries of the contraction sequence $\mathfrak{S}$. The *width* $\mathrm{w}(\mathfrak{S})$ of a contraction sequence $\mathfrak{S} = (G_1, \ldots, G_m)$ is defined as $\mathrm{w}(\mathfrak{S}) = \max_{1 \leqslant i \leqslant m} \deg_R(G_i)$, i.e., the largest red degree of any vertex of any trigraph in $\mathfrak{S}$. Now, the *twin-width* $\mathrm{tww}(G)$ of a trigraph G is defined as the smallest width of any contraction sequence $(G_1, \ldots, G_{|V(G)|})$ with $G_1 = G$ and $G_{|V(G)|} = \bullet$, where $\bullet$ denotes the trigraph consisting of a single vertex.

Some properties of twin-width; grid graphs

We conclude this section by collecting some properties of twin-width that we will rely on later. The first one states that twin-width is monotonic under taking induced subtrigraphs and is a simple consequence of the definitions (cf. [11, Section 4.1]).

▶ **Proposition 5.** *If H is an induced subtrigraph of a trigraph G, then* $\mathrm{tww}(H) \leqslant \mathrm{tww}(G)$.

Smallness. An infinite class $\mathcal{G}$ of graphs is *small* if there exists a constant $c > 1$ such that for every $n \in \mathbb{N}$ the class $\mathcal{G}$ contains at most $n! c^n$ labeled graphs on n vertices. The next theorem says that every graph class of bounded twin-width – i.e., for which there exists a constant $C > 0$, such that $\mathrm{tww}(G) \leqslant C$ for every graph G in the class – is small.

▶ **Theorem 6** ([9, Theorem 2.5])**.** *Every graph class with bounded twin-width is small.*

Now let s be a non-negative integer. The *s-subdivision* of G is the graph $\mathrm{subd}_s(G)$ obtained from G by subdividing each edge in $E(G)$ exactly s times. A simple counting argument together with Theorem 6 yields the following:

▶ **Proposition 7.** *For any fixed integers $k \geqslant 4$ and $s \geqslant 0$, the class* $\mathrm{subd}_s(\mathcal{G}_k)$ *of s-subdivisions of k-regular*[14] *simple graphs is not small, hence has unbounded twin-width.*

This proposition follows from the adaptation of an argument given in the first paragraph of [23, Section 3]. For completeness, we spell out this proof below.

Proof of Proposition 7. Let $N_k(m)$ be the number of labeled k-regular simple graphs on m vertices. Note that if $N_k(m) > 0$, then km is even; which we now assume. It is known (cf. [42, Section 6.4.1]) that, asymptotically

$$N_k(m) \sim \exp\left(\frac{1 - k^2}{4}\right) \frac{(km)!}{(km/2)! \cdot 2^{km/2} \cdot (k!)^m} = \Omega\left(\frac{(km/2)!}{(k!)^m}\right), \tag{1}$$

which readily implies that k-regular graphs (for $k \geqslant 3$) do not form a small class.

[14] A simple graph $G = (V, E)$ is *k-regular* if every vertex $v \in V$ has degree $\deg(v) = k$.

Now, let $N_k^{(s)}(n)$ be the number of n-vertex graphs in the class $\mathrm{subd}_s(\mathcal{G}_k)$ of s-subdivisions of k-regular simple graphs. If a graph $G \in \mathrm{subd}_s(\mathcal{G}_k)$ has n vertices, then $n = m + \frac{km}{2}s$, where m is the number of vertices of G of degree k. Note that such a graph G can be obtained by first choosing a k-regular labeled graph H on m vertices, then ordering the remaining $n-m = kms/2$ vertices arbitrarily and evenly distributing them to the edges of H according to some fixed ordering of $E(H)$. It follows that

$$N_k^{(s)}(n) \geqslant \binom{n}{m} \cdot N_k(m) \cdot (n-m)! = n! \frac{N_k(m)}{m!} \overset{(1)}{=} n! \cdot \Omega\left(\frac{(km/2)!}{m! \cdot (k!)^m}\right). \tag{2}$$

Since $k \geqslant 4$, we have $km/2 \geqslant 2m$. This, together with $(2m)! \geqslant 2^m (m!)^2$ and (2) implies

$$N_k^{(s)}(n) \geqslant n! \cdot \Omega\left(\frac{m!}{(2 \cdot k!)^m}\right). \tag{3}$$

Recall that $m = 2n/(2+ks)$. In particular, m is proportional to n. Hence $m!/(2 \cdot k!)^m$ grows faster than c^n for any fixed constant $c > 1$. Thus the graph class $\mathrm{subd}_s(\mathcal{G}_k)$ is not small. ◀

Grid graphs. The *d-dimensional n-grid* P_n^d is the graph with vertex set $V(P_n^d) = [n]^d$, and $\{u, v\} \in E(P_n^d)$ for two vertices $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$ if and only if $\sum_{i=1}^d |u_i - v_i| = 1$. The next result states that $\mathrm{tww}(P_n^d) \leqslant 3d$ irrespective of the value of n.

▶ **Theorem 8** (Theorem 4 in [11]). *For every positive d and n, the d-dimensional n-grid P_n^d has twin-width at most $3d$.*

The *d-dimensional n-grid with diagonals* $D_{n,d}$ is the graph with $V(D_{n,d}) = [n]^d$, and $\{u, v\} \in E(D_{n,d})$ for two vertices $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$ if and only if $\max_{i=1}^d |u_i - v_i| \leqslant 1$. Now, for a given trigraph $G = (V, E, R)$ we set $\mathrm{red}(G) = (V, \emptyset, E \cup R)$. In words, $\mathrm{red}(G)$ is the trigraph obtained from G by replacing every black edge of G by a red edge between the same vertices. With this notation $\mathrm{red}(D_{n,d})$ is the *d-dimensional* ***red*** *n-grid with diagonals*, i.e., $\mathrm{red}(D_{n,d}) = ([n]^d, \emptyset, E(D_{n,d}))$. Clearly, $\mathrm{tww}(D_{n,d}) \leqslant \mathrm{tww}(\mathrm{red}(D_{n,d}))$.

▶ **Theorem 9** (Lemma 4.4 in [11]). *For every positive d and n, every subtrigraph of the d-dimensional red n-grid with diagonals $\mathrm{red}(D_{n,d})$ has twin-width at most $2(3^d - 1)$.*

2.2 Background in topology

For general background in (combinatorial and differential) topology we refer to [43].

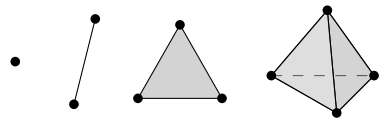
2.2.1 Simplicial and cubical complexes

Abstract simplicial complexes. Given a finite ground set $\mathcal{S}$, an *abstract simplicial complex* (or *simplicial complex*, for short) $\mathcal{X}$ over $\mathcal{S}$ is a downward closed subset of the power set $2^{\mathcal{S}}$, i.e., $\mathcal{F} \in \mathcal{X}$ and $\mathcal{F}' \subset \mathcal{F}$ imply $\mathcal{F}' \in \mathcal{X}$. Any element of $\mathcal{X}$ is called a *face* or *simplex* of $\mathcal{X}$, and for $\sigma \in \mathcal{X}$ the *dimension of* σ is defined as $\dim \sigma = |\sigma| - 1$. The *dimension of* $\mathcal{X}$, denoted by $\dim \mathcal{X}$, is then the maximum dimension of a face of $\mathcal{X}$. If $\dim \mathcal{X} = d$, we also say that $\mathcal{X}$ is a *simplicial d-complex*. For $0 \leqslant i \leqslant \dim \mathcal{X}$, we let $\mathcal{X}(i) = \{\sigma \in \mathcal{X} : \dim \sigma = i\}$ denote the set of i-dimensional faces (or i-faces, or i-simplices) of $\mathcal{X}$. The *i-skeleton* $\mathcal{X}_i = \bigcup_{j=0}^i \mathcal{X}(j)$ of $\mathcal{X}$ is the union of all faces of $\mathcal{X}$ up to dimension i. Note that any simple graph $G = (V, E)$ can be regarded as a 1-dimensional simplicial complex $\mathcal{X}_G$ with $\mathcal{X}_G(0) = \{\{v\} : v \in V\}$ and $\mathcal{X}_G(1) = E$. For $0 \leqslant i \leqslant 3$ the i-simplices of a simplicial complex $\mathcal{X}$ are respectively called the *vertices*, *edges*, *triangles*, and *tetrahedra* of $\mathcal{X}$.

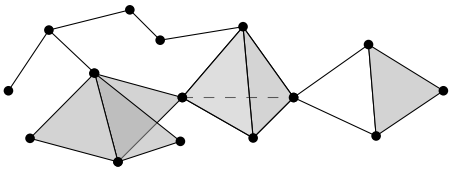
Geometric realization of simplicial complexes. Every abstract simplicial complex $\mathcal{X}$ may be realized geometrically as follows. To each abstract i-simplex $\sigma = \{v_0, v_1, \ldots, v_i\} \in \mathcal{X}$ we associate a *geometric i-simplex* $\|\sigma\| = [v_0, v_1, \ldots, v_i] \subset \mathbb{R}^i$ defined as the convex hull of $i+1$ affinely independent points in $\mathbb{R}^i$. We equip $\|\sigma\|$ with the subspace topology inherited from $\mathbb{R}^i$. Next, we consider the disjoint union $\bigsqcup_{\sigma\in\mathcal{X}} \|\sigma\|$ of these geometric simplices, and perform identifications along their faces that reflect their relationship in $\mathcal{X}$. The resulting space $\|\mathcal{X}\|$, equipped with the quotient topology, is called the *geometric realization of* $\mathcal{X}$, see Figure 1. The geometric realization of a simplicial complex is unique up to homeomorphism.

Cubical complexes. Analogous to simplicial complexes, a *cubical complex* $\mathcal{X}$ over a ground set $\mathcal{S}$ is a set system $\mathcal{X} \subset 2^{\mathcal{S}}$ that consists of "cubes" instead of simplices. The terminology is the same as in the simplicial case. The only difference is that a *geometric i-cube* is a topological space homeomorphic to $[0,1]^i$, where $[0,1]$ denotes the closed unit interval.

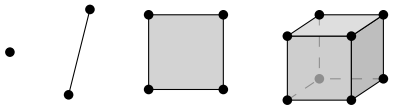
Cubical or simplicial complexes in this paper will typically be defined geometrically, and as such they will naturally come with a geometric realization.



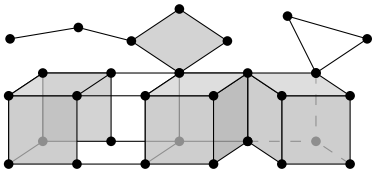
(a) Geometric i-simplices ($i = 0, 1, 2, 3$).



(b) Geometric realization of a simplicial 3-complex.



(c) Geometric i-cubes ($i = 0, 1, 2, 3$).



(d) Geometric realization of a cubical 3-complex.

Figure 1 The geometric perspective on simplicial and cubical complexes.

The hypercubic honeycomb. Let n and d be positive integers and consider the d-dimensional cube $[1,n]^d \subset \mathbb{R}^d$. The *$d$-dimensional hypercubic honeycomb* $\mathbf{H}^{d,n}$ is a cubical d-complex that decomposes $[1,n]^d$ into $(n-1)^d$ geometric cubes. The properties of this familiar object play an important role in this work, so we describe it for completeness. We define $\mathbf{H}^{d,n}$ geometrically, in a bottom-up fashion. First, the vertex set $\mathbf{H}^{d,n}(0)$ consists of precisely those points in $[1,n]^d$, which have only integral coordinates. Next, a 1-cube (i.e., an edge) is attached along its endpoints to vertices $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$ in $\mathbf{H}^{d,n}(0)$ if and only if $\sum_{i=1}^{d} |u_i - v_i| = 1$. Thus the 1-skeleton $\mathbf{H}_1^{d,n}$ is just the d-dimensional grid graph P_n^d encountered in the end of Section 2.1. Finally, the higher dimensional skeleta of $\mathbf{H}^{d,n}$ are *induced* by its 1-skeleton: for each subcomplex $\mathcal{Y} \subset \mathbf{H}_i^{d,n}$ isomorphic to the boundary of a $(i+1)$-cube, we attach an $(i+1)$-cube to $\mathbf{H}_i^{d,n}$ along $\mathcal{Y}$. In words, starting from the 1-skeleton $\mathbf{H}_1^{d,n}$, whenever we have the possibility to attach a cube of dimension at least two (because its boundary is already present), we attach it. See, e.g., Figure 2.

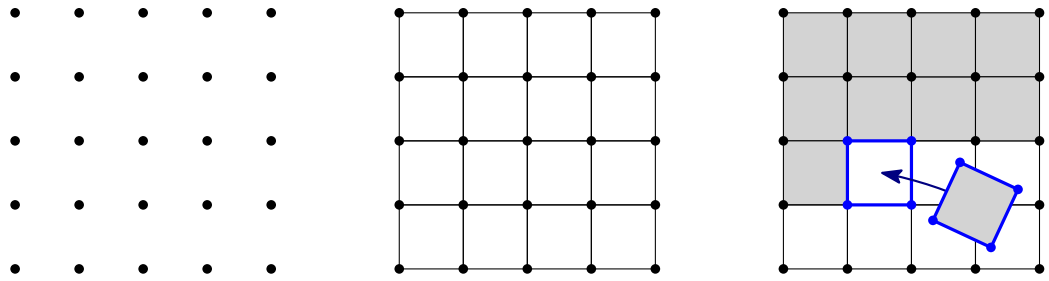
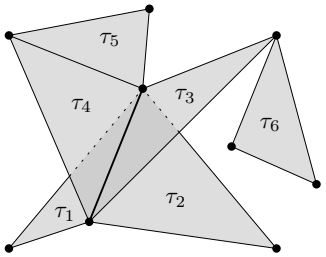
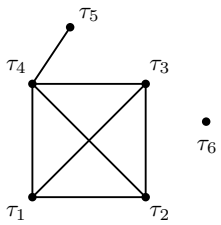


Figure 2 Constructing the complex $\mathbf{H}^{2,5}$. Its 1-skeleton $\mathbf{H}^{2,5}_1$ is isomorphic to the grid P^2_5.

Pure complexes and their dual graphs. A (cubical or simplicial) complex $\mathcal{X}$ is *pure* if every face of $\mathcal{X}$ is contained in a face of dimension $\dim \mathcal{X}$. It follows that, for every i with $0 \leqslant i \leqslant \dim \mathcal{X}$, the i-skeleton $\mathcal{X}_i$ of a pure complex $\mathcal{X}$ is also pure. Examples of pure complexes include nonempty graphs without isolated vertices, or triangulations of manifolds (see Section 2.2.3). Given a pure complex $\mathcal{X}$, the *dual graph* $\Gamma(\mathcal{X}_i) = (V, E)$ of its i-skeleton is defined as the graph, where the vertex set V corresponds to the set $\mathcal{X}(i)$ of i-faces, and $\{\sigma, \tau\} \in E$ if and only if σ and τ share an $(i-1)$-dimensional face in $\mathcal{X}$, see Figure 3.



(a) A pure simplicial 2-complex $\mathcal{X}$ formed by six triangles τ_1, τ_2, τ_3, τ_4, τ_5, and τ_6, four of which meet along a single edge.



(b) The dual graph $\Gamma(\mathcal{X}_2)$ of $\mathcal{X}_2$. As τ_6 shares no edge with any other triangle in $\mathcal{X}$, its corresponding vertex is isolated.

Figure 3 Example of a pure simplicial 2-complex and its dual graph.

2.2.2 Barycentric subdivisions

Given a (cubical or simplicial) complex $\mathcal{X}$, its *barycentric subdivision* is a **simplicial** complex $\mathcal{X}'$ defined abstractly as follows. For the ground set $\mathcal{S}'$ of $\mathcal{X}'$ we have $\mathcal{S}' = \mathcal{X}$. A $(k+1)$-tuple $\{\sigma_0, \ldots, \sigma_k\} \subset \mathcal{S}'$ forms a k-simplex of $\mathcal{X}'$ if and only if $\sigma_i \subset \sigma_j$ for every $0 \leqslant i < j \leqslant k$. We denote the 2^{nd} (resp. ℓ^{th}) iterated barycentric subdivision of a complex $\mathcal{X}$ by $\mathcal{X}''$ (resp. $\mathcal{X}^{(\ell)}$). See [10, Appendix A] for examples. The following are simple consequences of the definitions.

▶ **Observation 10.** *For any $0 \leqslant l \leqslant k$, a k-simplex has $\binom{k}{l}$ l-faces.*

▶ **Observation 11.** *The barycentric subdivision of a k-simplex contains $(k+1)!$ k-simplices.*

▶ **Observation 12.** *The barycentric subdivision of a k-cube contains $2^k k!$ k-simplices.*

See [10, Appendix A] for a proof of Observation 12.

2.2.3 Manifolds and their triangulations

Manifolds can be regarded as higher dimensional analogs of surfaces. A *d-dimensional topological manifold with boundary* (or *d-manifold*, for short) is a topological space[15] $\mathcal{M}$, where every point has an open neighborhood homeomorphic to $\mathbb{R}^d$, or to the closed upper half-space $\{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_1 \geqslant 0\}$. The points of $\mathcal{M}$ that do **not** have a neighborhood homeomorphic to $\mathbb{R}^d$ constitute the *boundary* $\partial\mathcal{M}$ of $\mathcal{M}$. If a manifold $\mathcal{M}$ satisfies $\partial\mathcal{M} = \emptyset$, then $\mathcal{M}$ is called a *closed* manifold. In this paper $\mathcal{M}$ always denotes a compact manifold.

Smooth manifolds. The main result of this paper (Theorem 1) applies for manifolds that have an additional property, namely *smoothness*. As we will not need to work with the actual definition of smoothness, but merely rely on it, we only give a brief definition here. For more background on smooth manifolds, we refer to [43, Chapter 5] and [33].

Given a connected and open subset $U \subset \mathcal{M}$ and a homeomorphism $\varphi\colon U \to \varphi(U)$ onto an open subset of $\mathbb{R}^d$, the pair (U, φ) is called a *chart*. Given two charts $(U_\alpha, \varphi_\alpha)$ and (U_β, φ_β) with $U_\alpha \cap U_\beta \neq \emptyset$, the map $\tau_{\alpha,\beta}\colon \varphi_\alpha(U_\alpha \cap U_\beta) \to \varphi_\beta(U_\alpha \cap U_\beta)$ defined via $\tau_{\alpha,\beta} = \varphi_\beta \circ \varphi_\alpha^{-1}$ is called a *transition map*. A *smooth structure* on a topological manifold $\mathcal{M}$ with $\partial\mathcal{M} = \emptyset$ is a collection $\mathcal{A} = \{(U_\alpha, \varphi_\alpha) : \alpha \in A\}$ of charts that satisfies the following three properties.

1. The sets U_α cover $\mathcal{M}$, that is $\bigcup_{\alpha\in A} U_\alpha = \mathcal{M}$.
2. For any $\alpha, \beta \in A$ with $U_\alpha \cap U_\beta \neq \emptyset$, the transition map $\tau_{\alpha,\beta}$ is smooth.[16]
3. The collection $\mathcal{A}$ is maximal in the sense that if (U, φ) is a chart and for every $\alpha \in A$ with $U \cap U_\alpha \neq \emptyset$ the transition maps $\varphi \circ \varphi_\alpha^{-1}$ and $\varphi_\alpha \circ \varphi^{-1}$ are smooth, then $(U, \varphi) \in \mathcal{A}$.

A topological manifold together with a smooth structure is called a *smooth manifold*. By an appropriate modification of property **2** above, the definition extends to manifolds with non-empty boundary as well. We refer to [43, Section 5.1.1] for details.

Triangulations. Let $\mathcal{M}$ be a compact topological d-manifold. A simplicial complex $\mathcal{T}$ whose geometric realization $\|\mathcal{T}\|$ is homeomorphic to $\mathcal{M}$ is called a *triangulation* of $\mathcal{M}$. It follows that $\mathcal{T}$ is a pure simplicial complex of dimension d (see Section 2.2.1). For $d \leqslant 3$, every topological d-manifold admits a triangulation [41, 44], however, for $d > 3$ this is generally not true (see [38] for an overview). Smooth manifolds can nevertheless always be triangulated, irrespective of their dimension, e.g., by work of Whitney [48, Chapter IV.B] (cf. Section 3).

Given a d-dimensional triangulation $\mathcal{T}$, recall that its *dual graph* $\Gamma(\mathcal{T})$ is the graph with vertices corresponding to the d-simplices of $\mathcal{T}$, and edges to the face gluings, i.e, those $(d-1)$-simplices of $\mathcal{T}$ that are contained in precisely two d-simplices. Note that $\deg(v) \leqslant d+1$ for any vertex v of $\Gamma(\mathcal{T})$. The following proposition is a consequence of the definitions.

▶ **Proposition 13.** *Let $\mathcal{T}$ be a triangulation of a d-manifold $\mathcal{M}$ and $\mathcal{U}$ be a collection of d-simplices of $\mathcal{T}$ that define a submanifold of $\mathcal{M}$. Then $\Gamma(\mathcal{U})$ is an induced subgraph of $\Gamma(\mathcal{T})$.*

3 Whitney's method

A seminal result of Whitney states that a smooth d-dimensional manifold $\mathcal{M}$ always admits a smooth embedding into a $2d$-dimensional Euclidean space.

[15] As a topological space a manifold is required to be *second countable* [43, p. 2] and *Hausdorff* [43, p. 87].
[16] See [43, p. 185] for a discussion of (smooth) maps of manifolds.

▶ **Theorem 14** (strong Whitney embedding theorem [47, Theorem 5], cf. [33, Theorem 6.19]). *For $d > 0$, every smooth d-manifold admits a smooth embedding into $\mathbb{R}^{2d}$.*

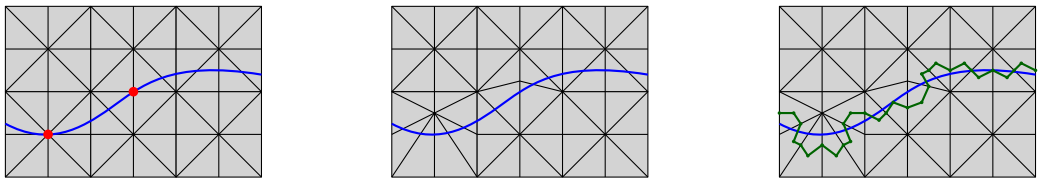
An important consequence of Theorem 14 is that smooth manifolds can be triangulated.

▶ **Theorem 15** (triangulation theorem [48, Chapter IV.B], cf. [5, Theorem 1.1]). *Every compact, smooth d-manifold $\mathcal{M}$ embedded in some Euclidean space $\mathbb{R}^m$ admits a triangulation.*

Next, we give a very brief and high-level overview of Whitney's method of triangulating smooth manifolds based on [48, Chapter IV.B] sufficient for our purposes. To the interested reader we also recommend [5], where Whitney's method is recast in a computational setting.

Triangulating smooth manifolds. Let $\mathcal{M}$ be a compact smooth d-manifold. By Theorem 14 there exists a smooth embedding $\iota\colon \mathcal{M} \to \mathbb{R}^{2d}$. Given such an embedding, we choose a sufficiently fine (with respect to ι) hypercubic honeycomb decomposition of $\mathbb{R}^{2d}$, denoted by L_0. Next, we pass to the first barycentric subdivision L of L_0. (Geometrically, L is obtained from L_0 by subdividing each k-dimensional hypercube of L_0 into $(2k)!!$ simplices.) By slightly perturbing the vertices of L we obtain a triangulation L^* of $\mathbb{R}^{2d}$, which is combinatorially isomorphic to L, but is in *general position* with respect to $\iota(\mathcal{M}) \subset \mathbb{R}^{2d}$. Now, by the work of Whitney, L^* induces a triangulation $\mathcal{T}$ of $\mathcal{M}$, where, importantly, $\mathcal{T}$ is a subcomplex of the d-skeleton of the barycentric subdivision $(L^*)'$ of L^*.

See Figure 4 for an illustration of this procedure for $d = 1$.



(a) The triangulation L and the image $\iota(\mathcal{M})$ (blue). The red points indicate the vertices of L contained by $\iota(\mathcal{M})$.

(b) The perturbed triangulation L^*, which in general position with respect to $\iota(\mathcal{M})$.

(c) The resulting triangulation $\mathcal{T}$ of $\mathcal{M}$ (dark green). $\mathcal{T}$ is a subcomplex of $(L^*)'$.

Figure 4 Illustration of Whitney's method of triangulating ambient submanifolds ($d = 1$).

Similar to Proposition 13, the next proposition is a direct consequence of the definitions.

▶ **Proposition 16.** *For any Whitney triangulation $\mathcal{T}$ of a compact smooth d-manifold $\mathcal{M}$, we have that the dual graph $\Gamma(\mathcal{T})$ is an induced subgraph of $\Gamma(((\mathbf{H}^{2d,n})'')_d)$.*

4 The proof of Theorem 1

In this section we prove our main result, i.e., every compact smooth d-manifold $\mathcal{M}$ has twin-width $\operatorname{tww}(\mathcal{M}) \leqslant d^{O(d)}$. To streamline the notation, we let $G_{d,n} = \Gamma(((\mathbf{H}^{2d,n})'')_d)$, i.e., $G_{d,n}$ denotes the dual graph of the d-skeleton[17] of the second barycentric subdivision of the hybercubic honeycomb $\mathbf{H}^{2d,n}$. The result is based on the following property of $G_{d,n}$.

[17] Recall that the dual graph $\Gamma(\mathcal{X})$ of a pure k-dimensional complex $\mathcal{X}$ has vertices corresponding to the k-faces of $\mathcal{X}$ and two vertices are connected if and only if their corresponding k-faces share a $(k-1)$-face.

▶ **Theorem 17.** *For the twin-width of the dual graph $G_{d,n} = \Gamma(((\mathbf{H}^{2d,n})'')_d)$ of the d-skeleton of the second barycentric subdivision of the hypercubic honeycomb $\mathbf{H}^{2d,n}$ we have*

$$\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}.$$

Before proving Theorem 17, we show how it implies Theorem 1.

Proof of Theorem 1. Let $\mathcal{M}$ be a compact, smooth d-dimensional manifold. Consider a Whitney triangulation $\mathcal{T}$ of $\mathcal{M}$. By Proposition 16, $\Gamma(\mathcal{T})$ is an induced subgraph of $G_{d,n}$ and by Theorem 17, $\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. Hence, since twin-width is monotone under taking induced subtrigraphs (Proposition 5), we obtain $\operatorname{tww}(\Gamma(\mathcal{T}))) \leqslant \operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. ◀

To complete the proof of Theorem 1, it remains to show Theorem 17.

Proof of Theorem 17. We establish the theorem by exhibiting a $d^{O(d)}$-contraction sequence $\mathfrak{S}\colon G_{d,n} \rightsquigarrow \bullet$. We will obtain $\mathfrak{S}$ by concatenating two contraction sequences $\mathfrak{S}_1\colon G_{d,n} \rightsquigarrow G^*_{d,n}$ and $\mathfrak{S}_2\colon G^*_{d,n} \rightsquigarrow \bullet$, referred to as the *first* and the *second epoch*, where $G^*_{d,n}$ is an appropriate subtrigraph of $\operatorname{red}(D_{n,2d})$, the $2d$-dimensional red n-grid with diagonals. In the following we regard $\mathbf{H}^{2d,n}$ and its subdivisions mainly as abstract complexes, but we will also take advantage of their geometric nature.

Preparations. Consider the $(2d+1)$-coloring $\mathcal{C}\colon \mathbf{H}^{2d,n} \to \{0,\dots,2d\}$, where we color the cubes of $\mathbf{H}^{2d,n}$ by their dimension, that is, for $c \in \mathbf{H}^{2d,n}$ we set $\mathcal{C}(c) = \dim(c)$ (Figure 5a).

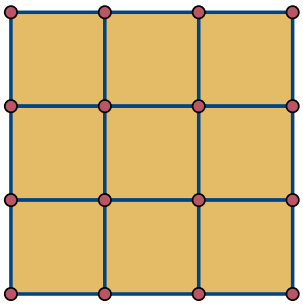
The coloring $\mathcal{C}$ induces a $(2d+1)$-coloring $\mathcal{C}''\colon (\mathbf{H}^{2d,n})'' \to \{0,\dots,2d\}$ of the simplices of the second barycentric subdivision $(\mathbf{H}^{2d,n})''$ as follows. The vertices of the first barycentric subdivision $(\mathbf{H}^{2d,n})'$ are in one-to-one correspondence with the cubes in $\mathbf{H}^{2d,n}$, hence $\mathcal{C}$ induces a coloring $\mathcal{C}'_0\colon (\mathbf{H}^{2d,n})'(0) \to \{0,\dots,2d\}$ via $\mathcal{C}'_0(v_c) = \mathcal{C}(c)$, where v_c denotes the vertex of $(\mathbf{H}^{2d,n})'$ corresponding to the cube $c \in \mathbf{H}^{2d,n}$ (Figure 5b). Geometrically, v_c is in the barycenter of the cube c. When we pass to the second barycentric subdivision, the vertices of $(\mathbf{H}^{2d,n})'$ become vertices of $(\mathbf{H}^{2d,n})''$, thus there is a natural inclusion $\iota\colon (\mathbf{H}^{2d,n})'(0) \to (\mathbf{H}^{2d,n})''(0)$. Let $\mathcal{V} = \operatorname{im}(\iota) \subset (\mathbf{H}^{2d,n})''(0)$ be the image of $(\mathbf{H}^{2d,n})'(0)$ under this inclusion ι. We color the elements of $\mathcal{V}$ identically to $\mathcal{C}'_0$, that is, we consider the coloring $\mathcal{C}''_{\mathcal{V}}\colon \mathcal{V} \to \{0,\dots,2d\}$ defined as $\mathcal{C}''_{\mathcal{V}}(v) = \mathcal{C}'_0(\iota^{-1}(v))$, see Figure 5c. Now, pick a simplex $\sigma \in (\mathbf{H}^{2d,n})''$ and note that $\bigcup_{v\in\mathcal{V}} \overline{\operatorname{st}}(v) = (\mathbf{H}^{2d,n})''$, i.e., the closed stars of the vertices $v \in \mathcal{V}$ cover $(\mathbf{H}^{2d,n})''$. Let $\mathcal{V}_\sigma = \{v \in \mathcal{V} : \sigma \in \overline{\operatorname{st}}(v)\}$. We now define $\mathcal{C}''(\sigma)$ as

$$\mathcal{C}''(\sigma) = \min\{\mathcal{C}''_{\mathcal{V}}(v) : v \in \mathcal{V}_\sigma\}. \tag{4}$$

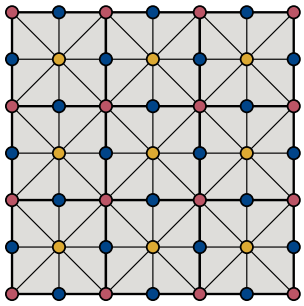
In words, $\mathcal{C}''(\sigma)$ is defined as the smallest dimension of any cube $c \in \mathbf{H}^{2d,n}$ such that σ belongs to the closed star of the vertex $\iota(v_c)$ in $(\mathbf{H}^{2d,n})''$, where v_c is the vertex of $(\mathbf{H}^{2d,n})'$ corresponding to c. We refer to Figure 5d for an example.

The first epoch. We start the description of the *first epoch* $\mathfrak{S}_1\colon G_{d,n} \rightsquigarrow G^*_{d,n}$ by considering the restriction $\mathcal{C}''_d\colon (\mathbf{H}^{2d,n})''(d) \to \{0,\dots,2d\}$ of the coloring $\mathcal{C}''$ to the d-simplices of $(\mathbf{H}^{2d,n})''$, see Figure 5e. Note that for each $i \in \{0,\dots,2d\}$, the i-colored d-simplices of $(\mathbf{H}^{2d,n})''$ form a family $\mathcal{F}_i = \{F_{i,c} : c \in \mathbf{H}^{2d,n}(i)\}$ of pairwise-disjoint connected subcomplexes of the d-skeleton of $(\mathbf{H}^{2d,n})''$, where $F_{i,c}$ denotes the subcomplex induced by the i-colored d-simplices of $(\mathbf{H}^{2d,n})''$ belonging to the closed star of the vertex $\iota(v_c)$, where v_c is the vertex of $(\mathbf{H}^{2d,n})'$ corresponding to the i-cube c in $\mathbf{H}^{2d,n}$ (Figure 5e). We also let $\mathcal{F} = \bigcup_{i=0}^{2d} \mathcal{F}_i$.

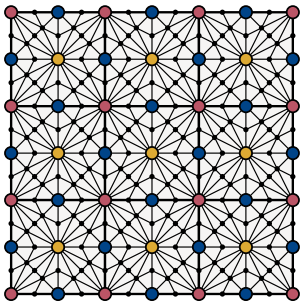
Since $G_{d,n}$ is defined as the dual graph of the d-skeleton of $(\mathbf{H}^{2d,n})''$, the nodes of $G_{d,n}$ are in one-to-one correspondence with the d-simplices of $(\mathbf{H}^{2d,n})''$. Let $\gamma\colon (\mathbf{H}^{2d,n})''(d) \to V(G_{d,n})$ denote the bijection realizing this correspondence. Henceforth, by a slight abuse of notation,



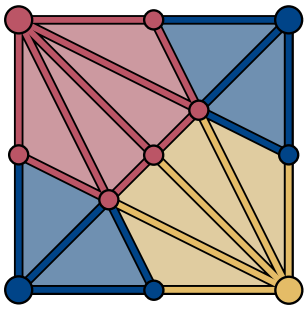
(a) The coloring $\mathcal{C}$ of the cubes of $\mathbf{H}^{2d,n}$ by their dimension.



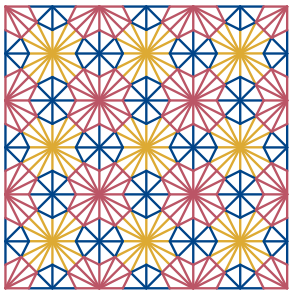
(b) The complex $(\mathbf{H}^{2d,n})'$ with its vertices colored by $\mathcal{C}'_0$.



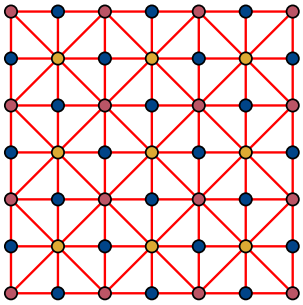
(c) The complex $(\mathbf{H}^{2d,n})''$ with its vertices in $\mathcal{V}$ colored by $\mathcal{C}''_{\mathcal{V}}$.



(d) The coloring $\mathcal{C}''$ near the upper left corner of $(\mathbf{H}^{2d,n})''$. The four larger disks represent vertices that belong to $\mathcal{V}$.



(e) The coloring $\mathcal{C}''$ restricted to the d-simplices of $(\mathbf{H}^{2d,n})''$. The color classes correspond to the families $\mathcal{F}_0$, $\mathcal{F}_1$ and $\mathcal{F}_2$.



(f) The trigraph $G^*_{d,n}$ obtained from $G_{d,n}$ by contracting each subtrigraph $C_{i,c}$ to a single node. All edges in $G^*_{d,n}$ are red.

Figure 5 (a)–(c) Illustrations of the cubical complex $\mathbf{H}^{2d,n}$ for $d = 1$ and $n = 4$, and of its first and second barycentric subdivisions (which are simplicial complexes) displaying the colorings described in the proof of Theorem 17. **(d)–(f)** Three essential steps in the proof of Theorem 17.

we also consider $\mathcal{C}''_d$ to be a $(2d+1)$-coloring of $V(G_{d,n})$ via $\mathcal{C}''_d(v) = \mathcal{C}''_d(\gamma^{-1}(v))$. Let $C_{i,c}$ denote the subtrigraph of $G_{d,n}$ induced by the nodes $\{\gamma(\sigma) : \sigma \in F_{i,c}\}$. Note that $\mathcal{C}''_d$ assigns the color i to all nodes of $C_{i,c}$. The first epoch $\mathfrak{S}_1 \colon G_{d,n} \rightsquigarrow G^*_{d,n}$ is merely the contraction sequence, where we contract each $C_{i,c}$ (where $0 \leqslant i \leqslant 2d$ and c is running over $\mathbf{H}^{2d,n}$) to a single node, in any order, obtaining a trigraph $G^*_{d,n}$ (Figure 5f).

The second epoch. $\mathfrak{S}_2 \colon G^*_{d,n} \rightsquigarrow \bullet$ is defined as an optimal contraction sequence of the trigraph $G^*_{d,n}$ to a single vertex. Since $G^*_{d,n}$ is a subtrigraph of $\mathrm{red}(D_{n,2d})$, the *$2d$-dimensional red n-grid with diagonals*, by Theorem 9 the width of $\mathfrak{S}_2$ is at most $2(3^{2d} - 1)$.

Bounding the width of the first epoch. We now give a rough estimate to show that the first epoch $\mathfrak{S}_1 \colon G_{d,n} \rightsquigarrow G^*_{d,n}$ is a $d^{O(d)}$-contraction sequence. The estimate is based on Claim 18 below, which follows from elementary properties of the hypercubic honeycomb and barycentric subdivisions.

▷ Claim 18. For any color $i \in \{0, \dots, 2d\}$ and cube $c \in \mathbf{H}^{2d,n}$, the subcomplex $F_{i,c}$ of $\mathbf{H}^{2d,n}$ (resp. the subtrigraph $C_{i,c}$ of $G_{d,n}$) defined above

1. has less than $h_1(d) = 4^d((2d)!)^2\binom{2d}{d}$ d-simplices (resp. nodes), and
2. less than $h_2(d) = 9^d$ incident subcomplexes $F_{i',c'} \in \mathcal{F}$ (resp. adjacent subtrigraphs $C_{i',c'}$).

Indeed, these two facts imply that by sequentially contracting each $C_{i,c}$ into a single node, the maximum red degree remains bounded by $O(h_1(d)^3 h_2(d))$ throughout the first epoch.

Proof of Claim 18. To bound the number of d-simplices in $F_{i,c}$, note that
- $F_{i,c}$ is covered by an appropriate translate of a $2d$-dimensional cube of $\mathbf{H}^{2d}$,
- the barycentric subdivision of a $2d$-cube contains $2^{2d}(2d)!$ $2d$-simplices (Observation 12),
- the barycentric subdivision of a $2d$-simplex contains $(2d)!$ $2d$-simplices (Observation 11),
- a $2d$-simplex has $\binom{2d}{d}$ faces of dimension d (Observation 10).

Multiplying these numbers, we obtain the first part of the claim.

To bound the number of subcomplexes $F_{i',c'} \in \mathcal{F}$ incident to $F_{i,c}$, note that the incidences between these subcomplexes reflect those between the handles in the canonical handle decomposition of $\mathbf{H}^{2d,n}$ obtained by "thickening up" its cubical cells. Thus, the number of subcomplexes in $\mathcal{F}$ incident to $F_{i,c}$ equals $3^{2d} - 1$ if $i \in \{0, 2d\}$ and $3^i + 3^{2d-i} - 2$ otherwise. Both of these numbers are less than 9^d, so the second part of the claim also holds. ◁

The width of $\mathfrak{S}$ is the maximum of the widths of $\mathfrak{S}_1$ and $\mathfrak{S}_2$, hence $\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. ◀

5 Triangulations with dual graph of arbitrary large twin-width

In Section 4 we showed that every compact, smooth d-manifold admits a triangulation with dual graph of twin-width at most $d^{O(d)}$. In this section we prove complementary results, showing that triangulations with dual graphs of large twin-width are abundant. We shed light on this fact in two ways. First, we show that for any fixed dimension $d \geqslant 3$, the class of $(d+1)$-regular graphs that can be dual graphs of triangulated d-manifolds is *not* small. Second, we show that the d-dimensional ball admits triangulations with a dual graph of arbitrarily large twin-width, which extends to every piecewise-linear (hence smooth) d-manifold. Both of these results rely on counting arguments, and thus are not constructive.

5.1 The class of dual graphs of triangulations is not small

Let us fix an integer $d \geqslant 3$. Following [21], we let $M_d(n)$ denote the number of colored[18] triangulations of closed orientable d-dimensional manifolds consisting of n d-simplices labeled from 1 to n. We assume n to be even. By [21, Theorem 1.1] we have[19]

$$n! \cdot n^{n/(2d)} \preceq M_d(n). \tag{5}$$

Ignoring the colors gives an uncolored triangulation, but any such d-dimensional triangulation can be obtained from $(d+1)!$ colored triangulations, since the $d+1$ colors can always be permuted. Now, any $(d+1)$-regular graph G with n vertices can be the dual graph of at most $d!^{n(d+1)/2}$ different d-dimensional triangulations. Indeed, if G is the dual graph of some d-dimensional triangulation, then each of the $n(d+1)/2$ edges of G corresponds to a face gluing, i.e., an identification of two $(d-1)$-dimensional simplices via a simplicial isomorphism, of which there are $d!$ many.[20] Hence by (5), for the number $\Gamma_d(n)$ of $(d+1)$-regular graphs on n labeled vertices that are dual graphs of some d-dimensional triangulations, we have

$$\frac{n! \cdot n^{n/(2d)}}{(d+1)! \cdot d!^{n(d+1)/2}} \preceq \Gamma_d(n). \tag{6}$$

As the left-hand side of (6) grows super-exponentially in n, the next theorem directly follows.

[18] We refer to [21, Section 2.1] for the precise definitions.

[19] Here "$\preceq$" denotes a comparison where exponential factors are ignored: more precisely, $f(n) \preceq g(n)$ means that there exists some constant $K > 0$, such that for n large enough, we have $f(n) \leqslant K^n g(n)$.

[20] This is because a simplicial isomorphism between two $(d-1)$-simplices σ and τ is determined by a perfect matching between the d vertices of σ and the d vertices of τ.

▶ **Theorem 19.** *For every $d \geqslant 3$, the class of $(d+1)$-regular graphs that are dual graphs of triangulations of d-manifolds is not small. In particular, there are graphs with arbitrarily large twin-width in this class.*

5.2 Complicated triangulations of the d-dimensional ball

In this section we show Theorem 3, according to which every piecewise-linear (PL) d-manifold ($d \geqslant 3$) admits triangulations with a dual graph of arbitrary large twin-width. To show the existence of such triangulations for every PL-manifold, we rely on the monotonicity of twin-width with respect to taking induced subtrigraphs (Proposition 5), the fact that the class of d-subdivisions of $(d+1)$-regular graphs is *not* small (Proposition 7) together with the following classical result from the theory of PL-manifolds. See [45] for an introduction.

▶ **Theorem 20** ([1, Corollary 1])**.** *Any triangulation of the boundary of a compact piecewise-linear (PL) manifold can be extended to a triangulation of the whole manifold.*

We first show that already the d-dimensional ball $\mathcal{B}^d = \{x \in \mathbb{R}^d : \|x\| \leqslant 1\}$ admits triangulations with dual graph of arbitrary large twin-width. More precisely, we prove:

▶ **Theorem 21.** *For every $m \in \mathbb{N}$ there is a triangulation $\mathcal{T}_m$ of the d-dimensional ball $\mathcal{B}^d$, such that $\operatorname{tww}(\Gamma(\mathcal{T}_m)) \geqslant m$ and $\partial\mathcal{T}_m = \partial\Delta^d$, the boundary of the standard d-simplex Δ^d.*

Proof. Let G_m be a d-subdivision of a $(d+1)$-regular graph G such that $\operatorname{tww}(G_m) \geqslant m$. The existence of such a graph is guaranteed by Proposition 7. Let $\mathcal{N}$ be a d-manifold homeomorphic to a closed regular neighborhood of a straight-line embedding of G in $\mathbb{R}^d$. Informally, $\mathcal{N}$ can be seen as a d-dimensional thickening of the graph G.

Construct an abstract triangulation of $\mathcal{N}$ as follows. Take a d-simplex σ_v for each node v of G, and fix a one-to-one correspondence between the $d+1$ facets of σ_v and the $d+1$ arcs incident to v in G. For every arc $\{u,v\} \in E(G)$, take a simplicial d-prism $P_{\{u,v\}} = \sigma_{\{u,v\}} \times [0,1]$, where $\sigma_{\{u,v\}}$ is a $(d-1)$-simplex, and attach $P_{\{u,v\}}$ to the simplices σ_u and σ_v by identifying $\sigma_{\{u,v\}} \times \{0\}$ (resp. $\sigma_{\{u,v\}} \times \{1\}$) with the facet of σ_u (resp. σ_v) that corresponds to the arc $\{u,v\}$. Now triangulate each prism $P_{\{u,v\}}$ with a minimal triangulation consisting of d d-simplices stacked onto each other, see [10, Appendix B].

Let $\mathcal{T}_{\mathcal{N}}$ denote the resulting triangulation of $\mathcal{N}$.

▷ Claim 22. For the dual graph of the triangulation $\mathcal{T}_{\mathcal{N}}$ we have $\Gamma(\mathcal{T}_{\mathcal{N}}) = G_m$.

Proof. The claim follows immediately from the facts that the dual graph of the constructed triangulation of the simplicial d-prism is merely a path of length d, and G_m is the d-subdivision of the $(d+1)$-regular graph G on which the triangulation $\mathcal{T}_{\mathcal{N}}$ is modeled. ◁

Pick a sufficiently large $\ell \in \mathbb{N}$ such that the ℓ^{th} iterated barycentric subdivision $\mathcal{T}_{\mathcal{N}}^{(\ell)}$ of $\mathcal{T}_{\mathcal{N}}$ embeds linearly in $\mathbb{R}^d$ and fix a simplex-wise linear embedding $\mathcal{E}\colon \|\mathcal{T}_{\mathcal{N}}^{(\ell)}\| \to \mathbb{R}^d$. Consider a large geometric d-simplex $\Sigma \subset \mathbb{R}^d$ that contains the image $\operatorname{im}(\mathcal{E})$ of $\mathcal{E}$ in its interior. Now $\Sigma_\circ = \Sigma \setminus \operatorname{int}(\operatorname{im}(\mathcal{E}))$ is a PL manifold[21] with triangulated boundary, so by Theorem 20 this boundary triangulation can be extended to a triangulation $\mathcal{T}_{\Sigma_\circ}$ of the entire manifold $\Sigma_\circ$. The boundary of $\mathcal{T}_{\Sigma_\circ}$ has two connected components: $\partial_1\mathcal{T}_{\Sigma_\circ} \cong \partial(\mathcal{T}_{\mathcal{N}}^{(\ell)})$ and $\partial_2\mathcal{T}_{\Sigma_\circ} \cong \partial\Delta^d$.

Let $\mathcal{Q}$ be a d-manifold homeomorphic to $\partial\mathcal{T}_{\mathcal{N}} \times [0,1]$. Take a triangulation $\mathcal{T}_{\mathcal{Q}}$ of $\mathcal{Q}$, such that for the two boundary components we have $\partial_1\mathcal{T}_{\mathcal{Q}} \cong \partial\mathcal{T}_{\mathcal{N}}$ and $\partial_2\mathcal{T}_{\mathcal{Q}} \cong \partial(\mathcal{T}_{\mathcal{N}}^{(\ell)})$. One way to construct such a triangulation is as follows. First, consider the decomposition $\mathcal{P}_{\mathcal{Q}}$ of $\mathcal{Q}$

[21] It is folklore that every codimension zero submanifold of a Euclidean space is a PL manifold, see, e.g., [34, p. 118] or [46, Remark 1.1.10].

into simplicial d-prisms induced by the product structure $\partial\mathcal{T}_\mathcal{N} \times [0,1]$. That is, $\mathcal{P}_\mathcal{Q}$ consists of d-prisms $P_\sigma \cong \sigma \times [0,1]$, one for each $(d-1)$-simplex σ of $\partial\mathcal{T}_\mathcal{N}$, glued together along their vertical boundary prisms the same way as the simplices of $\partial\mathcal{T}_\mathcal{N}$. The boundary $\partial\mathcal{P}_\mathcal{Q}$ of $\mathcal{P}_\mathcal{Q}$ has two connected components: $\partial_1\mathcal{P}_\mathcal{Q}$ and $\partial_2\mathcal{P}_\mathcal{Q}$, each combinatorially isomorphic to $\partial\mathcal{T}_\mathcal{N}$. Next, pass to the ℓ^{th} iterated barycentric subdivision of $\partial_2\mathcal{P}_\mathcal{Q}$. This operation turns each prism P_σ of $\mathcal{P}_\mathcal{Q}$ into a polyhedral cell R_σ. These cells form a polyhedral decomposition $\mathcal{R}_\mathcal{Q}$ of $\mathcal{Q}$, where $\partial_1\mathcal{R}_\mathcal{Q} \cong \partial\mathcal{T}_\mathcal{N}$ and $\partial_2\mathcal{R}_\mathcal{Q} \cong (\partial\mathcal{T}_\mathcal{N})^{(\ell)} = \partial(\mathcal{T}_\mathcal{N}^{(\ell)})$. Triangulate R_σ as follows. Consider an order $c_1 \prec c_2 \prec \cdots$ of the vertical cells[22] of R_σ, where $c_i \prec c_j$ implies $\dim(c_i) \leqslant \dim(c_j)$. Place a new vertex v_i in the barycenter of c_i and, iterating over the vertical cells in the above order, triangulate c_i by coning from v_i over its (already triangulated) boundary ∂c_i. It is clear that the resulting triangulation of R_σ is symmetric with respect to the symmetries of its base simplex σ. Applying this procedure for each polyhedral cell R_σ of $\mathcal{R}_\mathcal{Q}$ yields a triangulation $\mathcal{T}_\mathcal{Q}$ of $\mathcal{Q}$ with the desired properties.

Now the triangulation $\mathcal{T}_m$ of $\mathcal{B}^d$ is obtained by gluing together $\mathcal{T}_\mathcal{N}$, $\mathcal{T}_\mathcal{Q}$ and $\mathcal{T}_{\Sigma_\circ}$ via the identity maps along the isomorphic boundary-pairs $\partial\mathcal{T}_\mathcal{N} \cong \partial_1\mathcal{T}_\mathcal{Q}$ and $\partial_2\mathcal{T}_\mathcal{Q} \cong \partial_1\mathcal{T}_{\Sigma_\circ}$.

To conclude, note that the d-simplicies of $\mathcal{T}_\mathcal{N}$ triangulate a submanifold of $\|\mathcal{T}_m\|$, hence by Proposition 13 the graph $\Gamma(\mathcal{T}_\mathcal{N})$ is an induced subgraph of $\Gamma(\mathcal{T}_m)$. By Claim 22, $\Gamma(\mathcal{T}_\mathcal{N}) = G_m$ and by the initial assumption $\operatorname{tww}(G_m) \geqslant m$, hence by Proposition 5, $\Gamma(\mathcal{T}_m) \geqslant m$ as well. ◀

Proof of Theorem 3. Let $\mathcal{M}$ be an arbitrary PL-manifold possibly with non-empty boundary and $\mathcal{T}_\circ$ be a simplicial triangulation of $\mathcal{M}$. Consider a triangulation $\mathcal{T}_m$ of the d-ball with $\operatorname{tww}(\Gamma(\mathcal{T}_m)) \geqslant m$ as in Theorem 21. Let Δ be a d-simplex of $\mathcal{T}_\circ$ that is disjoint from $\partial\mathcal{M}$ (if $\mathcal{T}_\circ$ does not contain such a simplex, just replace $\mathcal{T}_\circ$ with its second barycentric subdivision). Since $\mathcal{T}_\circ$ is simplicial, Δ is embedded in $\mathcal{T}_\circ$ and is topologically a d-ball. Now replace Δ with $\mathcal{T}_m$ by first removing Δ from $\mathcal{T}_\circ$, thereby creating a boundary component isomorphic to $\partial\Delta$, then gluing $\partial\mathcal{T}_m$ to this new boundary component via a simplicial isomorphism. (Note that this is possible since $\partial\mathcal{T}_m \cong \partial\Delta$.) Let $\mathcal{T}$ denote the resulting triangulation of $\mathcal{M}$. By Propositions 5 and 13, it follows that $\operatorname{tww}(\Gamma(\mathcal{T})) \geqslant \operatorname{tww}(\Gamma(\mathcal{T}_m)) \geqslant m$. ◀

References

1 M. A. Armstrong. Extending triangulations. *Proc. Amer. Math. Soc.*, 18:701–704, 1967. doi:10.2307/2035442.

2 B. Bagchi, B. A. Burton, B. Datta, N. Singh, and J. Spreer. Efficient algorithms to decide tightness. In *32nd Int. Symp. Comput. Geom. (SoCG 2016)*, volume 51 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 12:1–12:15. Schloss Dagstuhl–Leibniz-Zent. Inf., 2016. doi:10.4230/LIPIcs.SoCG.2016.12.

3 D. Bar-Natan. Fast Khovanov homology computations. *J. Knot Theory Ramifications*, 16(3):243–255, 2007. doi:10.1142/S0218216507005294.

4 P. Bergé, É. Bonnet, H. Déprés, and R. Watrigant. Approximating highly inapproximable problems on graphs of bounded twin-width. In *40th Int. Symp. Theor. Aspects Comput. Sci. (STACS 2023)*, volume 254 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 10:1–10:15. Schloss Dagstuhl. Leibniz-Zent. Inform., 2023. doi:10.4230/lipics.stacs.2023.10.

5 J.-D. Boissonnat, S. Kachanovich, and M. Wintraecken. Triangulating submanifolds: an elementary and quantified version of Whitney's method. *Discrete Comput. Geom.*, 66(1):386–434, 2021. doi:10.1007/s00454-020-00250-8.

6 É. Bonnet. *Twin-width and contraction sequences.* Habilitation thesis, ENS de Lyon, April 2024. URL: https://perso.ens-lyon.fr/edouard.bonnet/text/hdr.pdf.

7 É. Bonnet, D. Chakraborty, E. J. Kim, N. Köhler, R. Lopes, and S. Thomassé. Twin-width VIII: delineation and win-wins. In *17th Int. Symp. Parametr. Exact Comput. (IPEC 2022)*, volume 249 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 9:1–9:18. Schloss Dagstuhl. Leibniz-Zent. Inform., 2022. doi:10.4230/LIPIcs.IPEC.2022.9.

[22] These are precisely those cells that are not contained in the boundary of $\mathcal{R}_\mathcal{Q}$.

8 É. Bonnet, C. Geniet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In *48th Int. Colloq. Autom. Lang. Prog. (ICALP 2021)*, volume 198 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 35:1–35:20. Schloss Dagstuhl. Leibniz-Zent. Inform., 2021. doi:10.4230/LIPIcs.ICALP.2021.35.

9 É. Bonnet, C. Geniet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width II: small classes. *Comb. Theory*, 2(2):Paper No. 10, 42, 2022. doi:10.5070/C62257876.

10 É. Bonnet and K. Huszár. On the twin-width of smooth manifolds, 2024. arXiv:2407.10174.

11 É. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width I: Tractable FO model checking. *J. ACM*, 69(1):Art. 3, 46, 2022. doi:10.1145/3486655.

12 B. A. Burton. Computational topology with Regina: algorithms, heuristics and implementations. In *Geometry and Topology Down Under*, volume 597 of *Contemp. Math.*, pages 195–224. Am. Math. Soc., Providence, RI, 2013. doi:10.1090/conm/597/11877.

13 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.

14 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2023. Version 7.3. URL: https://regina-normal.github.io.

15 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.

16 B. A. Burton, H. Edelsbrunner, J. Erickson, and S. Tillmann, editors. *Computational Geometric and Algebraic Topology*, volume 12 of *Oberwolfach Rep.* EMS Publ. House, 2015. doi:10.4171/OWR/2015/45.

17 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.

18 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.

19 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.

20 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.

21 G. Chapuy and G. Perarnau. On the number of coloured triangulations of d-manifolds. *Discrete Comput. Geom.*, 65(3):601–617, 2021. doi:10.1007/s00454-020-00189-w.

22 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.

23 Z. Dvořák and S. 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.

24 J. Gajarský, M. Pilipczuk, W. Przybyszewski, and Sz. Toruńczyk. Twin-width and types. In *49th Int. Conf. Autom. Lang. Prog. (ICALP 2022)*, volume 229 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 123:1–123:21. Schloss Dagstuhl. Leibniz-Zent. Inform., 2022. doi:10.4230/lipics.icalp.2022.123.

25 P. Hliněný and J. Jedelský. Twin-width of planar graphs is at most 8, and at most 6 when bipartite planar. In *50th Int. Colloq. Autom. Lang. Prog. (ICALP 2023)*, volume 261 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 75, 18. Schloss Dagstuhl. Leibniz-Zent. Inform., 2023. doi:10.4230/lipics.icalp.2023.75.

26 K. Huszár. *Combinatorial width parameters for 3-dimensonal manifolds*. PhD thesis, IST Austria, June 2020. doi:10.15479/AT:ISTA:8032.

27 K. Huszár. On the pathwidth of hyperbolic 3-manifolds. *Comput. Geom. Topol.*, 1(1):1–19, 2022. doi:10.57717/cgt.v1i1.4.

28 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.

29 K. Huszár and J. Spreer. On the width of complicated JSJ decompositions. In *39th Int. Symp. Comput. Geom. (SoCG 2023)*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 42:1–42:18. Schloss Dagstuhl–Leibniz-Zent. Inf., 2023. doi:10.4230/LIPIcs.SoCG.2023.42.

30 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.

31 D. Kráľ and A. Lamaison. Planar graph with twin-width seven. *Eur. J. Comb.*, page 103749, 2023. doi:10.1016/j.ejc.2023.103749.

32 D. Kráľ, K. Pekárková, and K. Štorgel. Twin-Width of Graphs on Surfaces. In *49th Int. Symp. Math. Found. Comput. Sci. (MFCS 2024)*, volume 306 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 66:1–66:15. Schloss Dagstuhl–Leibniz-Zent. Inf., 2024. doi:10.4230/LIPIcs.MFCS.2024.66.

33 J. M. Lee. *Introduction to smooth manifolds*, volume 218 of *Grad. Texts Math.* Springer, New York, second edition, 2013. doi:10.1007/978-1-4419-9982-5.

34 N. Levitt and A. Ranicki. Intrinsic transversality structures. *Pacific J. Math.*, 129(1):85–144, 1987. doi:10.2140/pjm.1987.129.85.

35 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.

36 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.

37 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.

38 C. Manolescu. Lectures on the triangulation conjecture. In *Proc. 22nd Gökova Geom.-Topol. Conf. (GGT 2015)*, pages 1–38. Int. Press Boston, 2016. URL: https://gokovagt.org/proceedings/2015/manolescu.html.

39 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.

40 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.

41 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.

42 M. Noy. Graphs. In *Handbook of Enumerative Combinatorics*, Discrete Math. Appl., pages 397–436. Chapman & Hall / CRC, 2015. doi:10.1201/b18255-12.

43 V. V. Prasolov. *Elements of combinatorial and differential topology*, volume 74 of *Grad. Stud. Math.* Am. Math. Soc., Providence, RI, 2006. Translated from the 2004 Russian original by Olga Sipacheva. doi:10.1090/gsm/074.

44 T. Radó. Über den Begriff der Riemannschen Fläche. *Acta Sci. Math.*, 2(2):101–121, 1925.

45 C. P. Rourke and B. J. Sanderson. *Introduction to piecewise-linear topology.* Springer Study Edition. Springer-Verlag, Berlin-New York, 1982. Reprint. doi:10.1007/978-3-642-81735-9.

46 F. Waldhausen, B. Jahren, and J. Rognes. *Spaces of PL manifolds and categories of simple maps*, volume 186 of *Annals Math. Stud.* Princeton Univ. Press, Princeton, NJ, 2013. doi:10.1515/9781400846528.

47 H. Whitney. The self-intersections of a smooth n-manifold in $2n$-space. *Ann. of Math. (2)*, 45:220–246, 1944. doi:10.2307/1969265.

48 H. Whitney. *Geometric Integration Theory.* Princeton Univ. Press, Princeton, NJ, 1957. doi:10.1515/9781400877577.

An 11/6-Approximation Algorithm for Vertex Cover on String Graphs

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Paweł Rzążewski
Warsaw University of Technology, Poland
University of Warsaw, Poland

Abstract

We present a 1.8334-approximation algorithm for Vertex Cover on string graphs given with a representation, which takes polynomial time in the size of the representation; the exact approximation factor is $\frac{11}{6}$. Recently, the barrier of 2 was broken by Lokshtanov, Panolan, Saurabh, Xue, and Zehavi [SoGC '24] with a 1.9999-approximation algorithm. Thus we increase by three orders of magnitude the distance of the approximation ratio to the trivial bound of 2. Our algorithm is very simple. The intricacies reside in its analysis, where we mainly establish that string graphs without odd cycles of length at most 11 are 8-colorable. Previously, Chudnovsky, Scott, and Seymour [JCTB '21] showed that string graphs without odd cycles of length at most 7 are 80-colorable, and string graphs without odd cycles of length at most 5 have bounded chromatic number.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms, string graphs, Vertex Cover, Coloring, odd girth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.24

Related Version *Full Version*: `https://arxiv.org/abs/2409.18820` [4]

Funding For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.
Édouard Bonnet: supported by the French National Research Agency through the project TWIN-WIDTH with reference number ANR-21-CE48-0014.
Paweł Rzążewski: funded in part by National Science Centre SONATA BIS grant number 2024/54/E/ST6/00094.

1 Introduction

The Vertex Cover problem[1] is one of Karp's 21 NP-complete problems [12]. While it admits several easy polynomial-time 2-approximation algorithms, it is an open question if an approximation factor of $2-\varepsilon$ can be achieved for some $\varepsilon > 0$. If the unique games conjecture (UGC) holds, then the answer to the latter question is negative [15]. Under the sole $\mathsf{P} \neq \mathsf{NP}$ assumption, it is currently only known that Vertex Cover cannot be approximated within ratio less than $\sqrt{2}$ [13, 9, 14].

On several graphs classes, better approximation algorithms of Vertex Cover exist. For instance, this problem can be exactly solved in polynomial time on bipartite graphs, as it reduces to a maximum flow problem [16]. It also admits a polynomial-time approximation scheme (PTAS) on planar graphs by Baker's technique [2]. On the contrary, a $(2-\varepsilon)$-approximation algorithm for Vertex Cover with $\varepsilon \in (0, 1/2]$ on triangle-free graphs would

[1] As an optimization problem, given a graph G, find a smallest possible vertex subset S such that $G - S$ (i.e., the subgraph of G induced by all the vertices *not* in S) is edgeless.

© Édouard Bonnet and Paweł Rzążewski;
licensed under Creative Commons License CC-BY 4.0

imply the same holding for general graphs. Indeed, one can observe that any vertex cover intersects each triangle on at least two vertices. One can thus repeatedly include all three vertices of a triangle in the approximate solution. Once the input graph has no triangle left, one calls the $(2-\varepsilon)$-approximation algorithm. This strategy can be seen to achieve approximation factor $\max(3/2, 2-\varepsilon)$ on general graphs. In particular, the former algorithm on triangle-free graphs would refute the UGC. Thus we say that $(2-\varepsilon)$-approximating Vertex Cover on triangle-free graphs is UGC-hard. It would be interesting to establish a dichotomy splitting the hereditary (i.e., closed under vertex removals) graph classes on which we know a $(2-\varepsilon)$-approximation algorithm from those on which this task is UGC-hard.

The class of *string graphs* consists of every intersection graph of connected sets in a planar graph. Alternatively string graphs are the intersection graphs of curves in the planes, called *strings*. The set of strings is called a *geometric representation* or *string representation*. The geometric representation may be convenient for drawing and in the proofs, but it is not so easy to handle as input. For that, we will also adopt a more combinatorial approach, based on the first definition of string graphs. A *representation* of a string graph G is a planar graph P and as many connected vertex subsets of P as G has vertices; two vertices are joined by an edge if and only if their corresponding subsets intersect. A caveat is that finding a (geometric) representation is NP-hard [19]. Furthermore, some string graphs G require representations whose size (i.e., the number of vertices of P or the number of crossings in a geometric representation) is exponential in $|V(G)|$ [20].

Recently Lokshtanov, Panolan, Saurabh, Xue, and Zehavi [23] presented a $(2-\varepsilon)$-approximation algorithm for Vertex Cover on string graphs, for some small positive $\varepsilon \approx 10^{-4}$. In this paper, we give a simpler algorithm with improved approximation factor $11/6 = 1.8333\ldots$

▶ **Theorem 1.** *Vertex Cover admits an $\frac{11}{6}$-approximation algorithm in string graphs given with a representation, whose running time is polynomial in the size of a representation.*

Our algorithm works as follows. We first get rid of all the odd cycles of length at most 11. This is done as in the abovementioned reduction to triangle-free graphs. While there is an odd cycle C with $|V(C)| \leqslant 11$, any vertex cover contains at least 6 vertices of C. We thus include all vertices of C in our approximate solution, and remove them from the graph. This ensures an 11/6 approximation factor, if such a factor can be achieved in the resulting graph. We can thus assume that our input graph has *odd girth*[2] more than 11. As the standard linear-programming (LP) formulation of Vertex Cover is half-integral [26], we can further assume that the vertex cover contains at least half of the vertices (see Theorem 14). Note that these opening steps are also present in [23].

We now deviate from the previous algorithm, and simply bound the chromatic number of string graphs of odd girth larger than 11.

▶ **Theorem 2.** *Every string graph of odd girth larger than 11 is 8-colorable, and given a representation, an 8-coloring can be found in time polynomial in the representation.*

A largest color class in a proper 8-coloring has size at least $n/8$ in an n-vertex graph. Thus, its complement is a vertex cover of size at most $7n/8$. As the optimum solution has size at least $n/2$, this yields a $7/4 = 1.75$ factor, and we conclude. Therefore, the main technical content of the paper is the proof of Theorem 2.

[2] The (odd) girth of a graph G is the length of a shortest (odd) cycle of G.

Outline of the proof of Theorem 2. Our strategy goes as follows. We make a breadth-first search (BFS) from some arbitrary vertex u_0 in each connected component of the graph G. We reserve colors $1, 2, 3, 4$ for the even-indexed BFS layer, and $5, 6, 7, 8$ for the odd-indexed BFS layer. We thus need to 4-color each connected component X of each subgraph induced by a single layer. Let R be a string representation of the entire graph, and $R[X]$ its restriction to X. As any face in the arrangement $R[X]$ can be made the infinite face, we can assume that the string of u_0 lies on the infinite face of $R[X]$. Now $R[X]$ has a nice property: The minimal topological disk D that contains it is such that each string of $R[X]$ intersects at least one string (in $R \setminus R[X]$) that itself crosses ∂D, the boundary of D. Indeed, such a string is given by a neighbor in the previous layer.

We now consider the string representation R_H made by X and at least one neighbor of the previous layer for each vertex of X, and intersect it by D. Each string of X remains in one piece, whereas the strings of the previous layer are possibly split into several strings in D. Let H be the intersection graph of R_H. We further layer X with the distance in H to a fixed vertex $w \in V(H) \setminus V(X)$. Let X_k be the vertices of H at distance exactly k from w in H.

We are left with proving that $H[X_k]$ is bipartite. For the sake of contradiction, we assume that there is an odd cycle C in $H[X_k]$. Our goal is to show that this odd cycle is contained in a ball of small radius around some vertex, yielding a contradiction in the form of a short odd cycle. For that, we establish that there is another odd cycle C' (built from C) such that the string of w is contained in a face F made by few strings of $N_H[C']$, with most of the rest of C' not intersecting F. A string defining F is then at a small distance of *every* string of C' since a (shortest) path from w to a vertex of the rest of C' has to cross the boundary of F. After which, the path has only a constant number of steps to reach its target since vertices of $N_H[C']$ are at distance $k-1$, k, or $k+1$ from w. The short odd cycle in H implies the existence of a short odd cycle in G, a contradiction.

Chromatic number of intersection graphs without short (odd) cycles. The chromatic number of intersection graphs of a given girth has a long history. Erdős and Gyárfás asked if girth-4 (i.e., triangle-free) segment intersection graphs have bounded chromatic number [10]. Kostochka and Nešetřil [17] raised the same question for 1-string graphs.[3] Both questions were answered in the negative in [27]. It was further shown by Walczak [28] that there are triangle-free segment intersection graphs without independent sets of size linear in the number of vertices. Nevertheless, Kostochka and Nešetřil [18] proved that 1-string graphs of girth at least 8 are 3-colorable, and asked [18, Problem 3] whether 8 could be replaced by 5. This has been confirmed for *outer* 1-strings by Das, Mukherjee, and Sahoo [7].

More relevant to our Vertex Cover application, the chromatic number of (1-)string graphs of a given odd girth has also been considered. (Note indeed that, in the design of a $(2-\varepsilon)$-approximation algorithm for Vertex Cover, whereas one can remove odd cycles up to any fixed size, the same trick does not apply to 4-vertex cycles.) McGuinness established that 1-string graphs of odd girth at least 7 have bounded chromatic number [24, 25]. Chudnovsky, Scott, and Seymour [6, Corollary from 12.4 and 12.5] further showed that string graphs of odd girth at least 9 are 80-colorable, and string graphs of odd girth at least 7 have bounded chromatic number. With proper adjustments, the former result can be turned into an algorithm that inputs a representation of any string graph G of odd girth at least 9, and outputs an 80-coloring of G in time polynomial in the representation. However, this would yield a significantly worse approximation factor for Vertex Cover.

[3] The 1-strings are strings every pair of which intersects at most once; note that it is not a property of particular objects but rather their arrangement.

The reason intersection graphs of 1-strings with girth at least 8 are 3-chromatic [18] is that they are 2-degenerate, i.e., (all their induced subgraphs) have a vertex of degree at most 2, which gives a recursive 3-coloring strategy. While string graphs of large odd girth are not d-degenerate for any d (they contain arbitrarily large bipartite complete graphs), we conjecture that they are nevertheless 3-colorable.

▶ **Conjecture 3.** *There is an integer k such that the class of string graphs of odd girth at least k is 3-chromatic.*

A stronger form of Conjecture 3 is that it holds for $k = 7$.

Limits of the method. The following observation summarizes the trade-off between required odd girth and bound on the chromatic number, in how they impact the approximation ratio.

▶ **Observation 4.** *Let $\mathcal{C}$ be a class such that there is a polynomial-time algorithm to c-color the graphs of $\mathcal{C}$ of odd girth at least an odd positive integer g. Then* Vertex Cover *admits a polynomial-time $2\max(\frac{g-2}{g-1}, \frac{c-1}{c})$-approximation algorithm in $\mathcal{C}$.*

We apply Observation 4 on represented string graphs with $g = 13$ and $c = 8$. Showing a counterpart of Theorem 2 with $g = 11$ and $c \leqslant 10$ would improve the ratio to $9/5 = 1.8$, and with $g = 9$ and $c \leqslant 8$, to $7/4 = 1.75$. Past this point, our 8-coloring starts being the bottleneck. The last possible step of this method would be to get parameters $g = 7$ and $c \leqslant 6$, for a ratio of $5/3 = 1.666\ldots$ Indeed we recall that string graphs of odd girth at least 5 have unbounded chromatic number [27], and do not necessarily have linear-size independent sets [28]. On the complexity side, we do not expect that Vertex Cover is APX-hard (i.e., NP-hard to approximate within some constant ratio $r > 1$) on represented string graphs as a quasipolynomial-time approximation scheme (QPTAS) exists [1, 11]. However, if a PTAS also exists, it will have to be found with a different approach than ours.

Let us emphasize that Theorem 2 is the only place where our algorithm actually requires a representation of the input to be given. Thus, an algorithm coloring string graphs with no short odd cycles, that works directly on the input graph G (not its representation), would yield an approximation algorithm for Vertex Cover in string graphs whose complexity is polynomial in the number of vertices of G. However, if we are only interested in the decision variant of the problem, the existential statement in Theorem 2 is sufficient to get the following result.

▶ **Theorem 5.** *Given a graph G and an integer k, in time polynomial in $|V(G)|$ we can distinguish the following cases:*
1. *G is a string graph and Vertex Cover$(G) \leqslant k$, and*
2. *G is a string graph and Vertex Cover$(G) > 11k/6$.*

If none of the cases applies, the algorithm terminates but its output is unspecified.

Let us remark that the authors of [23] claim that their algorithm can actually return a 1.9999-approximate solution in time polynomial in the number of vertices of the input graph but this is, unfortunately, not true. Indeed, one of the crucial steps in their argument is the application of a result of Lee [22, Theorem 4.2], whose proof has recently been realized to be flawed[4] [21, 3]. Seemingly the result [22, Theorem 4.2] can be reproven in a different way but only for *region intersection graphs* (a generalization of string graphs) [8], and the

[4] The inequality at the sixth line of the proof of Lemma 4.14 need not hold.

new approach requires that the representation is given (and still the final approximation ratio is much closer to 2 than the one given by Theorem 1). Similarly, it is claimed that the approach in [23] applies for all proper *induced-minor-closed* classes, but the problem lies in the same place: we are not aware of any way to fix the proof of Lee [22, Theorem 4.2] in such a general setting.

We leave as open questions if Vertex Cover admits, for some constant $\varepsilon > 0$, a $(2-\varepsilon)$-approximation algorithm on string graphs given without representation (which is very likely to be the case), and on classes excluding a fixed graph H as an induced minor.[5]

2 Definitions and preparatory lemmas

If $i \leqslant j$ are two non-negative integers, we denote by $[i,j]$ the set $\{i, i+1, \ldots, j-1, j\}$, and $[i]$ is a short-hand for $[1,i]$.

Graphs and odd cycles. We denote by $V(G)$ and $E(G)$ the set of vertices and edges of a graph G, respectively. A graph H is an *induced subgraph* (resp. *subgraph*) of a graph G if H can be obtained from G by vertex deletions (resp. by vertex and edge deletions). For $S \subseteq V(G)$, the *subgraph of G induced by S*, denoted $G[S]$, is obtained by removing from G all the vertices that are not in S. Then $G - S$ is a short-hand for $G[V(G) \setminus S]$. A set $X \subseteq V(G)$ is connected (in G) if $G[X]$ has a single connected component.

If C is a cycle, once a direction along the cycle is fixed, we may denote by $C[x \to y]$ the subpath of C from $x \in V(C)$ to $y \in V(C)$, turning in this direction.

▶ **Lemma 6.** *Let G be a graph that has an induced odd cycle C, and let $v \in V(G) \setminus V(C)$ have at least two neighbors in $V(C)$. Then there is a subpath of C that forms with v an induced odd cycle of G.*

Proof. Let $v_1, \ldots, v_h$ list the neighbors of v on C, starting at an arbitrary vertex and turning in a fixed (arbitrary) direction. The cycles $C_1, \ldots, C_h$, each with vertex set $\{v\} \cup C[v_i \to v_{i \pmod h + 1}]$ for $i \in [h]$, respectively, have a combined number of edges of $|V(C)| + 2h$; an odd number since $|V(C)|$ is odd. Hence at least one of $C_1, \ldots, C_h$ has odd length. ◀

Let G be a graph and $v \in V(G)$. We denote by $N_G(v)$ and $N_G[v]$, the open, respectively closed, neighborhood of v in G. For a non-negative integer r, we denote by $N_G^{\leqslant r}(v)$ (resp. $N_G^{=r}(v)$) the set of vertices of G at distance at most r (resp. exactly r) from v. We may call $N_G^{\leqslant r}(v)$ the ball of radius r around v (in G). We will rely on the observation that if a possibly long odd cycle is contained in a ball of small radius, then there is also a short odd cycle (in this ball).

▶ **Lemma 7.** *Let G be a graph, $v \in V(G)$, and r be a positive integer. Suppose $G[N_G^{\leqslant r}(v)]$ contains an odd cycle C. Then G (even $G[N_G^{\leqslant r}(v)]$) has an odd cycle of length at most $2r+1$. Furthermore, if $N_G^{=r}(v) \cap V(C)$ is an independent set of G, then G (even $G[N_G^{\leqslant r-1}(v)]$) has an odd cycle of length at most $2r-1$.*

Proof. Perform a breadth-first search (BFS) starting at v in $G[N_G^{\leqslant r}(v)]$. By definition, this breadth-first search has exactly $r+1$ layers $L_0, L_1, \ldots, L_r$, with $L_i := N_G^{=i}(v)$ for each $i \in [0,r]$. At least one L_i is *not* an independent set, otherwise $G[N_G^{\leqslant r}(v)]$ would be bipartite, and would not contain any odd cycle. Say $x, y \in L_i$ are adjacent, and let $z \in L_{i'}$ be their lowest common ancestor in the BFS tree (possibly $z = v$ and $i' = 0$). Then, there is a cycle

[5] i.e, H cannot be obtained from graphs of the class by removing vertices and contracting edges

of length $2(i-i')+1 \leqslant 2r+1$ in $G[N_G^{\leqslant r}(v)]$, consisting of the edge xy and the paths from z to x and from z to y in the BFS tree. Furthermore, if $N_G^{=r}(v) \cap V(C)$ is an independent set of G, then the odd cycle C has at least one of its edges with both endpoints in some $L_i \neq L_r$, which makes an odd cycle of length at most $2r-1$. ◀

String representations, and some useful notions and properties. A *string* is a subset of $\mathbb{R}^2$ homeomorphic to a closed segment. We may call a path-connected subset of a string s a *substring* of s. If R is a collection of strings in the plane, we denote their intersection graph by G_R, with one vertex per string, and an edge between every pair of intersecting strings. For any vertex $v \in V(G_R)$, we denote by $s_R(v)$ the string *representing* v in R. Conversely, we may denote by $v_R(s)$ the vertex represented by the string s. We typically drop the subscript when R is clear from the context. If X is an induced subgraph of G_R, we may denote by $R[X]$ the subset of strings of R corresponding to vertices of X.

A *face* of a geometric arrangement made by a collection $\mathcal{S}$ of strings, or more specifically of curves, in $\mathbb{R}^2$ is a connected component of $\mathbb{R}^2 \setminus \bigcup_{s\in\mathcal{S}} s$. A *closed face* is the topological closure of a face. If D is a topological disk, and more generally a face, we denote by ∂D its boundary. For instance, the curve ∂D defines two faces, one of which is finite ($D \setminus \partial D$), and admits D as a closed face. We may denote by $\overline{F}$ the closure of an (open) face F.

If P is an induced path represented by strings $s_1, \ldots, s_h$ (i.e., s_i and s_j intersect whenever $|i-j| = 1$), $a \in s_1$ and $b \in s_h$, we denote by $s[a, P, b]$ a minimal path-connected subset of $\bigcup_{i\in[h]} s_i$ containing a and b. In particular $s[a, P, b]$ is a string with endpoints a and b, coinciding with each s_i on a substring of positive length. Although $s[a, P, b]$ is not necessarily uniquely defined, every further claim will hold regardless of the actual choice for $s[a, P, b]$. It can thus be thought as a short-hand for: *fix any minimal path-connected subset of* $\bigcup_{i\in[h]} s_i$ *containing* a *and* b*, and denote it* $s[a, P, b]$.

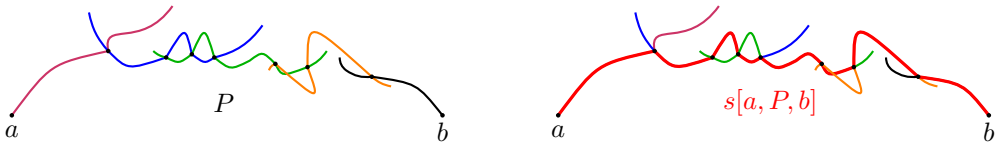


Figure 1 Left: 5-vertex path as the intersection graph of strings, where the first string of the path contains point a, and the last string of the path contains point b. Right: illustration of $s[a, P, b]$.

It will be convenient to extend the latter notation to induced cycles, that is, to allow s_1 and s_h to be adjacent. If P may be a cycle, then $s[a, P, b]$ is defined more specifically in the case when s_1 and s_h indeed intersect. If there is a point $c \in s_1 \cap s_h$ and two substrings $s'_1 \subseteq s_1, s'_h \subseteq s_h$ with endpoints a, c and b, c, respectively, such that $(s'_1 \cup s'_h) \cap \bigcup_{2\leqslant i\leqslant h-1} s_i = \emptyset$, then $s[a, P, b] := s'_1 \cup s'_h$. If not, then there is a (*non-self-intersecting*) string that starts at a, follows non-empty substrings of $s_1, \ldots, s_h$ in this order, and ends at b. We fix, as before, $s[a, P, b]$ to be any such string.

We will need the following lemma.

▶ **Lemma 8.** *Let D be a topological disk in the plane. Let $V \uplus W$ be a set of strings all contained in D, whose intersection graph is a cycle C, such that*

- *$|W| \geqslant 2$,*
- *every string of W has one or both endpoints in ∂D,*
- *no other point of a string of $V \cup W$ intersects ∂D, and*
- *no two strings of W (hence of $V \cup W$) intersect at a point of ∂D.*

Let s_p be a string contained in D with at least one endpoint p in ∂D, such that s_p does not intersect any string of $V \cup W$.

Then there are two strings $s \neq s' \in W$, and h strings $s_1, \ldots, s_h \in V$ with $v(s), v(s_1), \ldots, v(s_h), v(s')$ a subpath of C (or C itself), and s_p is contained in a closed face $\overline{F}$ of the arrangement $\{\partial D, s, s_1, \ldots, s_h, s'\}$ such that $\overline{F}$ does not intersect any string of $V \cup W$ outside of $s, s_1, \ldots, s_h, s'$, the two strings of $V \cup W$ intersecting s (including s_1), and the two string of $V \cup W$ intersecting s' (including s_h).

Proof. Let t, t' be two distinct strings of W, with $q \in t \cap \partial D$ and $q' \in t' \cap \partial D$. Let P_1, P_2 be the two paths from $v(t)$ to $v(t')$ in C. We define the strings $t_1 := s[q, P_1, q']$ and $t_2 := s[q, P_2, q']$. We denote by $\langle q, q' \rangle$ and $\langle q', q \rangle$ the two minimal path-connected subsets of ∂D containing q and q'. Let F_1, F_1' be the two finite (open) faces with boundary $\langle q, q' \rangle \cup t_1$ and $\langle q', q \rangle \cup t_1$, respectively. Similarly let F_2, F_2' be the two finite faces with boundary $\langle q, q' \rangle \cup t_2$ and $\langle q', q \rangle \cup t_2$, respectively.

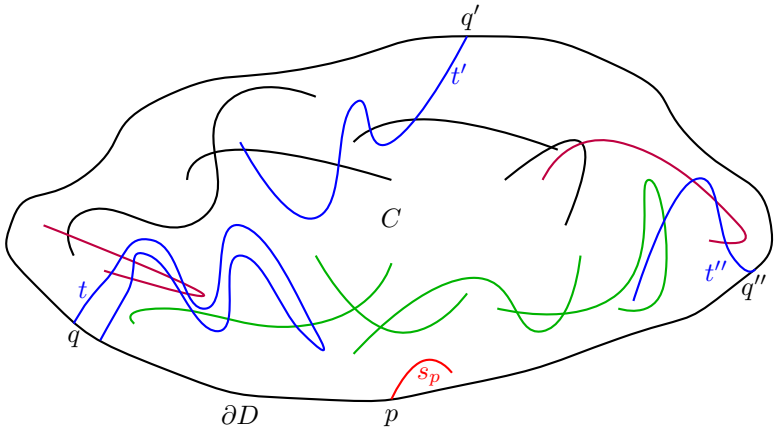


Figure 2 Illustration of the proof of Lemma 8. The strings of W are in blue. Say, we initially pick $t \neq t' \in W$ as depicted. The $v(t)$–$v(t')$ subpath of C enclosing s_p contains a string in W, namely t''. As qpq'' turns as qpq' along ∂D, t' will be set to t'', and the new P (green strings) is entirely in V, so the process stops. The strings $s := t, s' := t''$, the green strings, and ∂D form a closed face containing s_p that, among the strings of the rest of C, only the two purple strings (may) intersect.

As C is an induced cycle, at most one of $\overline{F_1}, \overline{F_1'}$ intersects the string of some vertex in $V(C) \setminus N_C[P_1]$. Without loss of generality, suppose that $\overline{F_1}$ does not intersect any string of $V(C) \setminus N_C[P_1]$. As s_p does not intersect t_1, it is contained in the closed face $\overline{F_1}$ or in the closed face $\overline{F_1'}$. If s_p is contained in $\overline{F_1}$, we set $P := P_1 - \{v(t), v(t')\}$ and $F := F_1$, and proceed to the next paragraph. Otherwise, we observe that s_p is contained in $\overline{F_2'}$, and $\overline{F_2'}$ does not intersect any string of $V(C) \setminus N_C[P_2]$. In which case we set $P := P_2 - \{v(t), v(t')\}$ and $F := F_2'$.

At this point, we have all the requirements of the lemma except that some strings of P may be in W. Let P be $v_1, \ldots, v_\ell$ with v_1 a neighbor of $v(t)$, and v_ℓ a neighbor of $v(t')$. While there is some string $t'' \in W$ with $v(t'') = v_i \in V(P)$, we let $q'' \in t'' \cap \partial D$ and distinguish two cases. If q, p, q'' turn along ∂D in the same direction as q, p, q', we set $t' := t''$ and $P := v_1, \ldots, v_{i-1}$. Otherwise we set $t := t''$ and $P := v_{i+1}, \ldots, v_\ell$. (Note that $|V(P)|$ decreases by at least one in each case.) When we exit the while loop, $s := t, s' := t' \in W$, and the subpath P of C (now only made of strings of V) satisfy the lemma statement. A visual recap is provided by Figure 2. ◀

3 String graphs of odd girth larger than 11 are 8-colorable

This section is devoted to the proof of Theorem 2. Let G be a string graph of odd girth strictly greater than 11, and R be a string representation of G. We may color each connected component of G independently, thus assume that G is connected. Let $u_0 \in V(G)$ be an arbitrary vertex. For each $i \geqslant 0$, let $L_i := N_G^{=i}(u_0) \subseteq V(G)$ be the set of vertices at distance exactly i from u_0. Note that $\{u_0\} = L_0, L_1, L_2, \ldots$ partition $V(G)$, and that there may be an edge between $x \in L_i$ and $y \in L_j$ only if $|i - j| \leqslant 1$.

Our goal is to 4-color $G[L_i]$ for each non-negative integer i. We are then done, since we can use two disjoint 4-color palettes for $G[\bigcup_{i \text{ is odd}} L_i]$ and $G[\bigcup_{i \text{ is even}} L_i]$. We fix i, and assume that $i \geqslant 2$ since L_0 is a singleton, and L_1 is an independent set, due to the condition on the odd girth. Note that it is sufficient to 4-color each connected component of $G[L_i]$. Let X be the vertex set of any connected component of $G[L_i]$.

Definition of the topological disk D. Let $R[X]$ be the string representation of $G[X]$ obtained by keeping the strings of R corresponding to vertices of X. Let D_0 be the topological disk whose boundary is the boundary of the infinite face in the arrangement $R[X]$. If $s(u_0)$ is contained in D_0, we draw R on a sphere, place any arbitrary free point (i.e., not occupied by a string of R) of the face containing $s(u_0)$ in $R[X]$ at its North pole, and consider the stereographic projection of this string representation. The latter is such that $s(u_0)$ is on the infinite face of the string arrangement $R[X]$. Thus we can in fact assume that $s(u_0)$ is not contained in D_0.

Note that every string of $R[X]$ is contained in D_0, and apart from those of $L_{i-1} \cup L_{i+1} \cup X$ no string of R intersects ∂D_0. Let $D \supset D_0$ be a very slightly augmented topological disk such that $\partial D_0 \subset D \setminus \partial D$, the property that no string outside $L_{i-1} \cup L_{i+1} \cup X$ intersects ∂D is preserved, no intersection of two strings of R lies on ∂D, and, for any point $q \in \partial D$, no string of R intersects ∂D at q without crossing it at q. We observe that every string of L_{i-1} with a neighbor in $V(X)$ intersects ∂D, and that every vertex of X has a neighbor in L_{i-1}.

Definition of the auxiliary graph H. Let R_H be the set of strings formed by $D \cap R[(L_{i-1} \cap N_G(V(X))) \cup V(X)]$, and H its intersection graph. Every vertex of X corresponds to a single vertex in H, whereas each vertex of $L_{i-1} \cap N_G(V(X))$ may split in several strings in H as the corresponding string in R may enter and exit D several times. However, the strings of R_H that correspond to the same vertex in $L_{i-1} \cap N_G(V(X))$ are pairwise non-intersecting and thus they form an independent set in H. Fix an arbitrary vertex $w \in L_{i-1} \cap N_G(V(X))$. Let X_j be the subset of vertices of X at distance exactly j from w in H. Again, $X_1, X_2, \ldots$ partition X. By the previous remarks, it is sufficient to show that that for any positive integer k, the graph $G[X_k] = H[X_k]$ is bipartite. We fix k, and show this fact in the next section.

3.1 $G[X_k]$ is bipartite

For the sake of contradiction assume that there is an induced odd cycle C in $G[X_k]$. We denote by $v_1, \ldots, v_h$ the vertices of C. For every $\ell \in [h]$, we fix some $w_\ell \in N_H(v_\ell) \setminus X$. Such a vertex exists since every vertex of L_i has at least one neighbor in L_{i-1} (possibly split into several strings in R_H, at least one of which intersects $s(v_\ell)$). Observe that $w_1, \ldots, w_h$ are not necessarily pairwise distinct. We set $W := \{w_1, \ldots, w_h\}$, with $1 \leqslant |W| \leqslant h$. We say that string $s(v_i)$ (and, by extension, vertex v_i) is *simply attached* in a cycle C' containing v_i if w_i has only one neighbor in $V(C')$, namely v_i. We denote by $r(w_i)$ one (arbitrary, if both endpoints are in ∂D) endpoint of w_i that is in ∂D.

Our plan is to show that there is an odd cycle C' contained in the ball of radius 6 around some vertex z in H such that $N_H^{=6}(z) \cap V(C')$ is an independent set. This implies, by Lemma 7, the existence of an odd cycle of length at most 11 in H, which in turn implies, as we will see, that the same happens in G. To do this, we exhibit a path $\widehat{P}$ in $H[V(C) \cup W]$ on at most four vertices, whose strings define with ∂D a (finite) closed face F containing $s(w)$ (recall that w is the arbitrary BFS root in H), and an odd cycle C' in $H[V(C) \cup W]$ "mostly" contained in $D \setminus F$. We then show that z can be chosen among the vertices of $\widehat{P}$.

Let us first establish the following lemma.

▶ **Lemma 9.** *There is an induced odd cycle C' in H such that $V(C') \subseteq V(C) \cup W$ and one of the following items holds*

- *$C' = C$ and every string of $V(C')$ is simply attached in C', or*
- *$V(C') \cap W = \{w_i\}$, for some $w_i \in W$, and $V(C') \setminus \{w_i\}$ induces a subpath of C whose internal vertices are all simply attached in C', or*
- *$|V(C') \cap W| \geqslant 2$, and there is a subpath $x, v_i, \ldots, v_j, y$ of C' (or C' itself) such that $x \neq y$, $\{x, y\} \subset W$, $\{v_i, \ldots, v_j\} \subset V(C)$ may be empty, $s(v_{i+1}), \ldots, s(v_{j-1})$ are all simply attached in C', and $s(w)$ is contained in a closed face of $\{\partial D, s(x), s(v_i), s(v_{i+1}), \ldots, s(v_{j-1}), s(v_j), s(y)\}$ that does not intersect any string of C' but $s(x), s(v_i), s(v_{i+1}), \ldots, s(v_{j-1}), s(v_j), s(y)$ and the other neighbor in C' of x and of y.*

Proof. If all the strings of C are simply attached, we are done as the first item holds. Otherwise there is some $w_i \in W$ with at least two neighbors in $V(C)$. By Lemma 6, there is an induced odd cycle C_1 such that $V(C_1)$ comprises w_i and a subpath of C.

Let $p \geqslant 1$, and suppose we have defined C_p. If C_p satisfies the second or the third item of the lemma, we are done; so we assume otherwise. We will show that we can find an induced cycle C_{p+1} in $H[V(C) \cup W]$ with fewer vertices in $V(C)$ than C_p has.

First, suppose that $V(C_p) \cap W$ is a singleton $\{x\}$. As C_p does not satisfy the second outcome, there is a vertex $v_i \in V(C_p) \setminus \{x\} \subseteq V(C)$ that is *not* simply attached nor adjacent to x in C_p. We then obtain C_{p+1} by applying Lemma 6 to the pair C_p, w_i.

Suppose instead that C_p has at least two vertices in W. By Lemma 8, $s(w)$ is contained in a closed face F of the arrangement $\{\partial D, s(x), s(v_i), s(v_{i+1}), \ldots, s(v_{j-1}), s(v_j), s(y)\}$ with $x \neq y$ and $\{x, y\} \subset W$ such that F does not intersect any string of C' outside $s(x'), s(x), s(v_i), s(v_{i+1}), \ldots, s(v_{j-1}), s(v_j), s(y), s(y')$ where x', y' are the unique vertices in $N_{C'}(x) \setminus \{v_i\}, N_{C'}(y) \setminus \{v_j\}$, respectively. As C_p does not satisfy the third item, $|j - i| \geqslant 2$ and there is some $v_{i'}$ with $i + 1 \leqslant i' \leqslant j - 1$ such that $s(v_{i'})$ is *not* simply attached in C_p. We then obtain C_{p+1} by applying Lemma 6 to the pair $C_p, w_{i'}$; see Figure 3.

In both cases, the number of vertices of $V(C)$ present in the current odd cycle drops by at least one when going from C_p to C_{p+1}. Thus, after at most $h = |V(C)|$ iterations, the procedure will return an odd cycle C_q in $H[V(C) \cup W]$ that satisfies the second or the third statement of the lemma. ◀

We can then obtain the announced milestone.

▶ **Lemma 10.** *There is an induced odd cycle C' in $H[V(C) \cup W]$, two distinct vertices x, y in W, possibly part of C', and a subpath P of C' on 0, 1, or 2 vertices, all in $V(C)$, such that*

- *if $V(P) \neq \emptyset$, then x is adjacent to one endpoint of P, and y, to the other (possibly equal) endpoint of P, and if $V(P) = \emptyset$, then x and y are adjacent,*
- *one finite closed face F_1 of the arrangement $\{\partial D, s[r(x), xPy, r(y)]\}$ contains $s(w)$, and*
- *the other finite closed face F_2 contains every string of $V(C') \setminus N_{C'}[V(P) \cup \{x, y\}]$.*

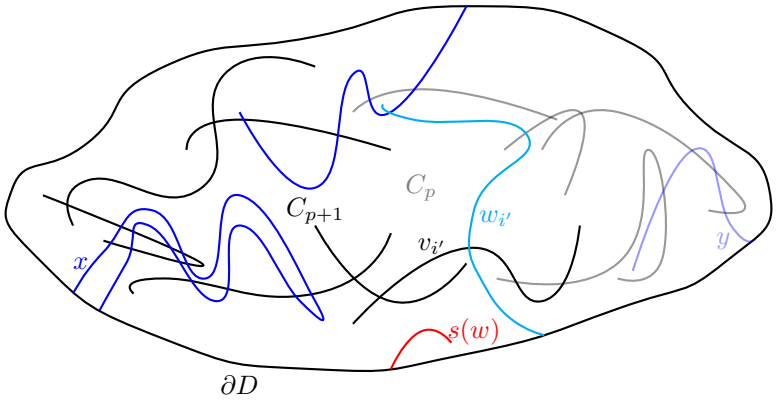


Figure 3 The string $w_{i'} \in W$ with several neighbors in $V(C_p)$, and the new odd induced cycle C_{p+1} obtained by Lemma 6. Lemma 8 will then locate $s(w)$ as enclosed by $x, w_{i'}$ and some (here, three) strings of V. For legibility, a string may be labeled by its corresponding vertex.

Proof. By Lemma 9, there is an induced odd cycle C' in $H[V(C) \cup W]$, two vertices $x' \neq y' \in W$, possibly part of C', and a possibly-empty subpath P of C' such that

- $V(P) \subset V(C)$ and each internal vertex of P is simply attached in C',
- $x'Py'$ is a path or a cycle (we allow x' and y' to be adjacent, even when $V(P) \neq \emptyset$),
- $\{\partial D, s[r(x'), x'Py', r(y')]\}$ has two finite closed faces $F_1 \supset s(w)$, and
- F_2 containing every string of $V(C') \setminus N_{C'}[V(P) \cup \{x', y'\}]$.

Indeed, we directly get this outcome if the third item of Lemma 9 holds. If instead the first item of Lemma 9 holds, we set $x' := w_i$ and $y' := w_j$ for any $v_i \neq v_j \in V(C) = V(C')$ such that $w_iw_j \notin E(H)$. This is possible to ensure since H is triangle-free and $|V(C)| \geqslant 3$. The path P is then chosen as the subpath of C from v_i to v_j such that $\partial D \cup s(x') \cup \bigcup_{u \in V(P)} s(u) \cup s(y')$ separates $s(w)$ from $V(C) \setminus N_C[P]$. If finally the second item of Lemma 9 holds, we set $x' := w_i$ for any $v_i \in V(C')$ such that $w_i \notin V(C')$, and y' is defined as the only vertex of $W \cap V(C')$. We then set P as previously.

Now, while the path P has at least three vertices and $s[r(x'), x'Py', r(y')]$ passes through the strings of P, let v_i be an internal vertex of P. By construction, $s(v_i)$ is simply attached in C'. Let P' be the subpath of P going from the endpoint of P neighboring x' to v_i, and P'' be the subpath of P going from v_i to its other endpoint (neighboring y'). Then, either P' and the pair $x' \neq w_i \in W$ or P'' and the pair $w_i \neq y'$ satisfy the four items of the previous paragraph. In the former case, we set $P := P'$ and $y' := w_i$, whereas in the latter, we set $P := P''$ and $x' := w_i$; see Figure 4.

We are done when $|V(P)| \leqslant 2$ or when $s[r(x'), x'Py', r(y')]$ is contained in $s(x') \cup s(y')$. In both cases, we set $x := x'$ and $y := y'$, and in the latter case, we set P to be empty. ◀

Following the notations of Lemma 10, we define z as the neighbor of x in P if $V(P) \neq \emptyset$, and as x otherwise.

▶ **Lemma 11.** $V(C') \subseteq N_H^{\leqslant 6}(z)$.

Proof. We keep the notations of Lemma 10. First we observe that any vertex in $N_{C'}[V(P) \cup \{x, y\}]$ is at distance at most 3 of z in H. It remains to see that every vertex z' of C' such that $s(z')$ is contained in F_2 is at distance at most 6 from z in H.

Note that every vertex of W is at distance $k-1$, k, or $k+1$ from w in H, since every vertex of C is at distance exactly k from w in H, and every vertex of W has a neighbor in $V(C)$. As $s(w)$ is in F_1, a (shortest) path from w to z' has to contain a vertex y' such

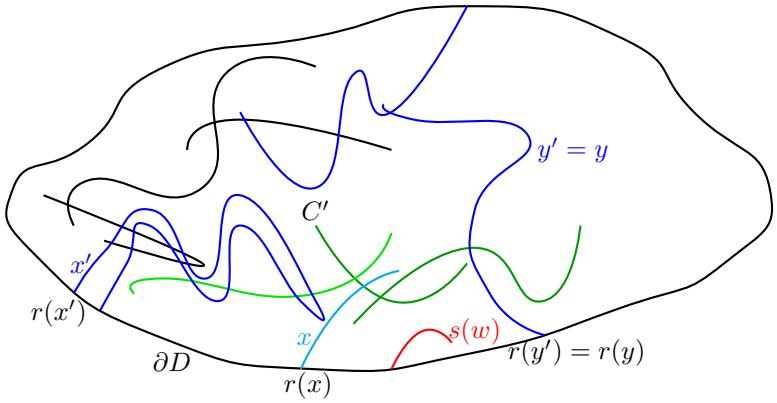


Figure 4 Example of strings $x' \neq y' \in W$ and the subpath P (three green strings) of C'. The internal node of P is simply attached to C'. This is the case when x' should be updated (to the vertex of the cyan string). The new path P (darker green) has two vertices, thus the process stops.

that $s(y')$ intersects $s[r(x), xPy, r(y)]$. Since vertices of $V(P) \cup \{x, y\}$ are at distance at least $k-1$ from w, vertex y' is at distance at least $k-2$ from w. Now, as vertices of C' are at distance at most $k+1$ from w, vertex y' is at distance at most 3 from z'. In turn, z is at distance at most 3 from y', and we conclude. ◀

We further show that no edge of C' can lie within $N_H^{=6}(z)$.

▶ **Lemma 12.** *$N_H^{=6}(z) \cap V(C')$ is an independent set of H or H admits an odd cycle of length at most 9.*

Proof. The proof of Lemma 11 shows that for some vertex $z' \in V(C')$ to be at distance 6 from z in H, there should be a 4-edge path from y to z'. Thus two vertices z', z'' adjacent in C' and at distance 6 from z in H entail an odd cycle of length *at most* 9 in H, built from y, the corresponding two (non-necessarily edge-disjoint) 4-edge paths, z', and z''. ◀

Lemmas 7, 11, and 12 imply the existence of an odd cycle C_o of length at most 11 in H. To reach the final contradiction, we build an odd cycle of at most the same length in G.

▶ **Lemma 13.** *Let C_o be an odd cycle in H. Then G has an odd cycle of length at most $|V(C_o)|$.*

Proof. We initialize a represented odd cycle $\widehat{C}$ to C_o realized by the corresponding strings of R_H. We make an induction on the number of strings of $\widehat{C}$ not part of the representation R of G. When this number is 0, we conclude since G contains the cycle $\widehat{C}$ as a subgraph. Let $y_1, \ldots, y_q \in V(\widehat{C})$ be all the vertices of $\widehat{C}$ whose strings are substrings of $s(y)$ for some $y \in L_{i-1} \cap N_G(V(X))$.

We further assume that starting at $y_1 \in V(\widehat{C})$, and turning in some fixed (arbitrary) direction along $\widehat{C}$, one encounters $y_1, y_2, \ldots, y_q$ in this order. Let, for every $a \in [q]$, ℓ_a be the number of edges of $\widehat{C}[y_a \to y_{a \pmod q+1}]$. As, $|V(\widehat{C})| = \sum_{a \in [q]} \ell_a$, at least one ℓ_a is odd. Recall that $\{y_1, \ldots, y_q\}$ is an independent set in H, hence every ℓ_a is strictly greater than 1. Then the string $s(y)$ and those of the internal vertices in the path between y_a and $y_{a \pmod q+1}$ form an odd cycle of length at least 3 and at most $|V(\widehat{C})|$. This defines the new represented odd cycle $\widehat{C}$, and concludes the proof. ◀

3.2 8-coloring algorithm

To claim Theorem 2, we finally need to check that, given a representation $P, \mathcal{S}$ of a string graph G of odd girth larger than 11, one can compute an 8-coloring of G in time polynomial in $|V(P)|$. Recall that P is a planar graph and $\mathcal{S}$ is a set of non-empty connected sets in P in one-to-one correspondence with $V(G)$ such that two distinct connected sets of $\mathcal{S}$ intersect if and only if the corresponding vertices of G are adjacent. Note that, as G is in particular triangle-free, $|V(G)| \leqslant 2|V(P)|$.

We remind the reader that we compute one BFS in G, and fewer than $|V(G)|$ BFSes in auxiliary graphs H of size at most $(|V(P)|+1)|V(G)|$. After this we simply 2-color bipartite graphs whose combined number of vertices is at most $|V(G)|$. Thus, we shall just detail how to compute each auxiliary graph H.

Let X be the component of $G[L_i]$ giving rise to H. We start by adding every vertex of X to $V(H)$. We compute the set $Y \subseteq V(P)$ of all the vertices contained in an element of $\mathcal{S}$ corresponding to a vertex of X. Let $u' \in V(P)$ be a vertex in the connected set of u_0. Let $Y' \subseteq Y$ be the vertices $v \in Y$ for which there is a path in P from u' to v whose internal vertices are all in $V(P) \setminus Y$. Note that Y' can be computed in polynomial time in $|V(P)|$ by checking if u' and y are in the same connected component of $P - (Y \setminus \{v\})$. The set Y' is the combinatorial counterpart of ∂D. In particular, we do not need to change the representation if u' is in a finite face "made by Y." This was merely helpful in the correctness proof.

For each vertex $v \in L_{i-1} \cap N_G(V(X))$ (alternatively we can only keep at least one neighbor per vertex of X), we add to $V(H)$ one vertex for each connected component of $S \cap Y$ in P, where S is the connected set of v. Note that $S' \cap Y' \neq \emptyset$ for every vertex set S' of such a connected component. This step adds to H fewer than $|V(G)| \cdot |V(P)|$ vertices. Graph H is finally defined as the intersection graph of all the connected sets of vertices in $V(H)$, which can be computed in time polynomial in $|V(P)|$.

4 Approximation algorithm for Vertex Cover

For a graph G, let $\mathsf{vc}(G)$ denote the size of a minimum vertex cover in G. We will use the following result of Chlebík and Chlebíková [5], which is a slight strengthening of the well-known Nemhauser–Trotter theorem [26].

▶ **Theorem 14** (Chlebík and Chlebíková [5], Nemhauser and Trotter [26]). *Given a graph G, one can compute in polynomial time a partition of $V(G)$ into $V_0, V_{1/2}, V_1$, such that*

1. *there are no edges between V_0 and $V_{1/2}$ or within V_0,*
2. $\mathsf{vc}(G[V_{1/2}]) \geqslant \frac{1}{2}|V_{1/2}|$, *and*
3. *every minimum vertex cover S of G satisfies $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$.*

Finally, we are ready to prove Theorem 1, which we restate for convenience.

▶ **Theorem 1.** *Vertex Cover admits an $\frac{11}{6}$-approximation algorithm in string graphs given with a representation, whose running time is polynomial in the size of a representation.*

Proof. Let G be the input graph, given along with a representation. The algorithm has three phases.

Phase one. We initialize $X = \emptyset$. If G contains an odd cycle of length at most 11, we include all its vertices into X and remove them from the graph. We repeat this step exhaustively; clearly this can be done in polynomial time. Let G' be the graph obtained after the last iteration of the process.

Phase two. We call the algorithm from Theorem 14 on the graph G' in order to obtain three sets $V_0, V_{1/2}$, and V_1. We denote $Y = V_1$ and $G'' = G'[V_{1/2}]$. Note that G'' is a string graph with odd girth larger than 11 and the representation of G'' can be easily obtained from the representation of G by removing objects corresponding to vertices in $V(G) \setminus V(G'')$.

Phase three. We apply the algorithm from Theorem 2 to find a proper coloring of G'' with at most 8 colors. Let c be the color that appears most frequently, and let Z be set of vertices of G'' *not* colored c. Clearly $|Z| \leqslant \frac{7}{8}|V(G'')|$, so $|V(G'')| \geqslant \frac{8}{7}|Z|$. The algorithm returns $Q = X \cup Y \cup Z$.

Analysis. First, let us argue that Q is indeed a vertex cover. Since $X \subseteq Q$, it is enough to show that $Q \cap V(G') = Y \cup Z$ is a vertex cover of G'. Note that by the first property in Theorem 14 and since $V_1 = Y$, all edges of G' not contained in G'' are covered by Q. So we are left with showing that $Q \cap V(G'') = Z$ is a vertex cover of G''. This is clearly true, as the complement of Z in G'' is an independent set (of color c).

Now let us analyze the approximation factor. Let S be an optimum solution, i.e., a vertex cover of G of size $\mathsf{vc}(G)$. Note that for each odd cycle C removed in the first phase, $S \cap C$ must contain at least $\frac{|C|}{2}$ vertices in order to cover all the edges of C. As each removed cycle has at most 11 vertices, we conclude that $|S \cap X| \geqslant \frac{6}{11}|X|$.

Note that $S' \setminus X$ is a vertex cover of G'. By the third property in Theorem 14 we observe that $Y = V_1 \subseteq S$, and so $S \setminus (X \cup Y)$ is a vertex cover of G''. By the second property in Theorem 14 we obtain that $|S \setminus (X \cup Y)| \geqslant \frac{1}{2}|V(G'')|$.

Summing up, we obtain

$$\begin{aligned}|S| &= |S \cap X| + |S \cap Y| + |S \setminus (X \cup Y)| \geqslant \frac{6}{11}|X| + |Y| + \frac{1}{2}|V(G'')| \\ &\geqslant \frac{6}{11}|X| + |Y| + \frac{4}{7}|Z| \geqslant \frac{6}{11}|X \cup Y \cup Z| = \frac{6}{11}|Q|,\end{aligned}$$

which means that $|Q| \leqslant \frac{11}{6}|S| = \frac{11}{6}\mathsf{vc}(G)$. This completes the proof. ◀

The proof above is easily adapted to show Theorem 5. The first two phases remain unchanged. Let $\mathcal{C}$ be the family of odd cycles found in phase one and let Y be the set found in phase two. Once we reach phase three, we do not call the coloring algorithm on G'', as its running time might not be polynomial in $|V(G)|$. Instead, we check if $\sum_{C \in \mathcal{C}} \lceil |V(C)|/2 \rceil + |Y| + |V(G'')|/2 > k$ and, if so, we reject the instance. Note that the sum in the expression above is a lower bound on $\mathsf{vc}(G)$, so this step is correct. Otherwise, we accept the instance. The approximation guarantee is estimated as in the proof of Theorem 1.

References

1. Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *J. ACM*, 66(4):29:1–29:40, 2019. doi:10.1145/3326122.
2. Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.
3. Édouard Bonnet, Jędrzej Hodor, Tuukka Korhonen, and Tomáš Masařík. Treewidth is polynomial in maximum degree on weakly sparse graphs excluding a planar induced minor. *CoRR*, abs/2312.07962, 2023. doi:10.48550/arXiv.2312.07962.
4. Édouard Bonnet and Paweł Rzążewski. An 11/6-approximation algorithm for vertex cover on string graphs. *CoRR*, abs/2409.18820, 2024. doi:10.48550/arXiv.2409.18820.

5 Miroslav Chlebík and Janka Chlebíková. Improvement of Nemhauser-Trotter theorem and its applications in parametrized complexity. In Torben Hagerup and Jyrki Katajainen, editors, *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, volume 3111 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2004. doi:10.1007/978-3-540-27810-8_16.

6 Maria Chudnovsky, Alex Scott, and Paul D. Seymour. Induced subgraphs of graphs with large chromatic number. V. Chandeliers and strings. *J. Comb. Theory B*, 150:195–243, 2021. doi:10.1016/J.JCTB.2021.05.001.

7 Sandip Das, Joydeep Mukherjee, and Uma kant Sahoo. Outer 1-string graphs of girth at least five are 3-colorable. In *Extended Abstracts EuroComb 2021: European Conference on Combinatorics, Graph Theory and Applications*, pages 593–598. Springer, 2021.

8 James Davies. private communication.

9 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? 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 376–389. ACM, 2018. doi:10.1145/3188745.3188804.

10 András Gyárfás. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae*, 19(3-4):413–441, 1987.

11 Sariel Har-Peled. Approximately: Independence implies vertex cover. *CoRR*, abs/2308.00840, 2023. doi:10.48550/arXiv.2308.00840.

12 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

13 Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 576–589. ACM, 2017. doi:10.1145/3055399.3055432.

14 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 592–601. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00062.

15 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. doi:10.1016/J.JCSS.2007.06.019.

16 Dénes Kőnig. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

17 Alexandr Kostochka and Jaroslav Nešetřil. Chromatic number of geometric intersection graphs. In *1995 Prague Midsummer Combinatorial Workshop*, volume 95.309, pages 43–45. KAM Series, 1995.

18 Alexandr V. Kostochka and Jaroslav Nešetřil. Coloring relatives of intervals on the plane, I: chromatic number versus girth. *Eur. J. Comb.*, 19(1):103–110, 1998. doi:10.1006/EUJC.1997.0151.

19 Jan Kratochvíl. String graphs. II. recognizing string graphs is np-hard. *J. Comb. Theory B*, 52(1):67–78, 1991. doi:10.1016/0095-8956(91)90091-W.

20 Jan Kratochvíl and Jiří Matoušek. String graphs requiring exponential representations. *J. Comb. Theory B*, 53(1):1–4, 1991. doi:10.1016/0095-8956(91)90050-T.

21 James R. Lee. private communication.

22 James R. Lee. Separators in region intersection graphs. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 1:1–1:8. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ITCS.2017.1.

23 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A 1.9999-approximation algorithm for vertex cover on string graphs. 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 72:1–72:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.72.

24 Sean McGuinness. Colouring arcwise connected sets in the plane I. *Graphs Comb.*, 16(4):429–439, 2000. doi:10.1007/PL00007228.

25 Sean McGuinness. Colouring arcwise connected sets in the plane II. *Graphs Comb.*, 17(1):135–148, 2001. doi:10.1007/PL00007235.

26 George L. Nemhauser and Leslie E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974. doi:10.1007/BF01580222.

27 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *J. Comb. Theory B*, 105:6–10, 2014. doi:10.1016/J.JCTB.2013.11.001.

28 Bartosz Walczak. Triangle-free geometric intersection graphs with no large independent sets. *Discret. Comput. Geom.*, 53(1):221–225, 2015. doi:10.1007/S00454-014-9645-Y.

Approximating Klee's Measure Problem and a Lower Bound for Union Volume Estimation

Karl Bringmann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Kasper Green Larsen
Aarhus University, Denmark

André Nusser
Université Côte d'Azur, CNRS, Inria, I3S, Sophia Antipolis, France

Eva Rotenberg
Technical University of Denmark, Kongens Lyngby, Denmark

Yanheng Wang
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max-Planck-Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Union volume estimation is a classical algorithmic problem. Given a family of objects $O_1, \dots, O_n \subset \mathbb{R}^d$, we want to approximate the volume of their union. In the special case where all objects are boxes (also called hyperrectangles) this is known as Klee's measure problem. The state-of-the-art $(1+\varepsilon)$-approximation algorithm [Karp, Luby, Madras '89] for union volume estimation as well as Klee's measure problem in constant dimension d uses a total of $O(n/\varepsilon^2)$ queries of three types: (i) determine the volume of O_i; (ii) sample a point uniformly at random from O_i; and (iii) ask whether a given point is contained in O_i.

First, we show that if an algorithm learns about the objects only through these types of queries, then $\Omega(n/\varepsilon^2)$ queries are necessary. In this sense, the complexity of [Karp, Luby, Madras '89] is optimal. Our lower bound holds even if the objects are equiponderous axis-aligned polygons in $\mathbb{R}^2$, if the containment query allows arbitrary (not necessarily sampled) points, and if the algorithm can spend arbitrary time and space examining the query responses.

Second, we provide a more efficient approximation algorithm for Klee's measure problem, which improves the running time from $O(n/\varepsilon^2)$ to $O((n+\frac{1}{\varepsilon^2}) \cdot \log^{O(d)}(n))$. We circumvent our lower bound by exploiting the geometry of boxes in various ways: (1) We sort the boxes into classes of similar shapes after inspecting their corner coordinates. (2) With orthogonal range searching, we show how to sample points from the union of boxes in each class, and how to merge samples from different classes. (3) We bound the amount of wasted work by arguing that most pairs of classes have a small intersection.

2012 ACM Subject Classification Theory of computation → Computational geometry

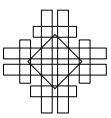
Keywords and phrases approximation, volume of union, union of objects, query complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.25

Related Version *Full Version*: `https://arxiv.org/abs/2410.00996` [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).
Kasper Green Larsen: Supported by a DFF Sapere Aude Research Leader Grant No. 9064-00068B.
André Nusser: This work was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of

© Karl Bringmann, Kasper Green Larsen, André Nusser, Eva Rotenberg, and Yanheng Wang;
licensed under Creative Commons License CC-BY 4.0



Excellence of Université Côte d'Azur under reference number ANR-15-IDEX-01. Part of this work was conducted while the author was at BARC, University of Copenhagen, supported by the VILLUM Foundation grant 16582.
Eva Rotenberg: Supported by DFF Grant 2020-2023 (9131-00044B) "Dynamic Network Analysis", the VILLUM Foundation grant VIL37507 "Efficient Recomputations for Changeful Problems" and the Carlsberg Foundation Young Researcher Fellowship CF21-0302 "Graph Algorithms with Geometric Applications".

1 Introduction

We revisit the classical problem of *union volume estimation*: given objects $O_1, \ldots, O_n \subset \mathbb{R}^d$, we want to estimate the volume of $O_1 \cup \ldots \cup O_n$.[1] This problem has several important applications such as DNF counting and network reliability; see the discussion in Section 1.2.

The state-of-the-art solution [19] works in a model where one has access to each input object O_i by three types of queries: (i) determine the volume of the object, (ii) sample a point uniformly at random from the object, and (iii) ask whether a point is contained in the object. Apart from these types of queries, the model allows arbitrary computations. The complexity of algorithms is thus measured by the number of queries.

After Karp and Luby [18] introduced this model, Karp, Luby and Madras [19] gave an algorithm that $(1+\varepsilon)$-approximates the volume of n objects in this model with constant success probability.[2] It uses $O(n/\varepsilon^2)$ queries plus $O(n/\varepsilon^2)$ additional time, which improves earlier algorithms [18, 22], and only asks containment queries of previously sampled points. The last 35 years have seen no algorithmic improvement. Is this classical upper bound best possible? We resolve the question in this work by providing a matching lower bound.

The union volume estimation problem was also studied recently in the streaming setting [25, 23], where a stream of objects $O_1, \ldots, O_n \subset \Omega$ arrive in order. When we are at position i in the stream, we can only query object O_i. This line of work yields algorithms with constant success probability that use $O(\text{polylog}(|\Omega|)/\varepsilon^2)$ queries and additional time per object (and the same bound also holds for space complexity). Summed over all n objects, the bounds match the classical algorithm up to the $\text{polylog}(|\Omega|)$ factor. So, interestingly, even in the streaming setting the same upper bound can be achieved.[3]

The perhaps most famous application of the algorithm by Karp, Luby, and Madras [19] is *Klee's measure problem* [21]. This is a fundamental problem in computational geometry in which we are given n axis-aligned boxes in $\mathbb{R}^d$ and want to compute the volume of their union. An axis-aligned box is a set in the form $[a_1, b_1] \times \ldots \times [a_d, b_d] \subset \mathbb{R}^d$, and the input consists of the corner coordinates $a_1, b_1, \ldots, a_d, b_d$ of each box. A long line of research on this problem and various special cases (e.g., for fixed dimensions or for cubes) [31, 26, 11, 2, 1, 12, 32, 3] lead to an exact algorithm running in time $O(n^{d/2} + n \log n)$ for constant d [13]. A conditional lower bound suggests that any faster algorithm would require fast matrix multiplication techniques [12], but it is unclear how to apply fast matrix multiplication to this problem. On the approximation side, note that the three queries can be implemented in time $O(d)$ for any d-dimensional axis-aligned box. Thus the union volume estimation algorithm can be applied, and it computes a $(1+\varepsilon)$-approximation to Klee's measure problem in time $O(nd/\varepsilon^2)$, as

[1] Technically, the objects need to be measurable. In the most general form, $O_1, \ldots, O_n$ can be any measurable sets in a measure space, and we want to estimate the measure of their union. However, this work only deals with boxes in $\mathbb{R}^d$ (in our algorithm) and polygons in the plane (in our lower bound).
[2] The success probability can be boosted to $1-\delta$, adding a $\log(1/\delta)$ factor in time and query complexity.
[3] See also [30] for earlier work studying Klee's measure problem in the streaming setting.

has been observed in [4]. This direct application of union volume estimation was the state of the art for approximate solutions for Klee's measure problem until our work. See Section 1.2 for interesting applications of Klee's measure problem.

1.1 Our contribution

Lower bound for union volume estimation

Given the state of the art, a natural question is to ask *whether the query complexity of the general union volume estimation algorithm of [19] can be further improved.* Any such improvement would speed up several important applications, cf. Section 1.2. On the other hand, any lower bound showing that the algorithm of [19] is optimal also implies tightness of the known streaming algorithms (up to logarithmic factors), as the streaming algorithms match the static running time bound.

Previously, a folklore lower bound of $\Omega(n + 1/\varepsilon^2)$ was known. But between this and the upper bound $O(n/\varepsilon^2)$ there is still a large gap; consider for example the regime $\varepsilon = 1/\sqrt{n}$ where the bounds are $\Omega(n)$ and $O(n^2)$, respectively. We close this gap by strengthening the lower bound to $\Omega(n/\varepsilon^2)$, thereby also showing optimality of [19]. Note that the lower bound is about query complexity in the model; it does not make any computational assumption. In particular, it holds even if the algorithm spends arbitrary time and space in computation.

▶ **Theorem 1.** *Any algorithm that computes a $(1+\varepsilon)$-approximation to the volume of the union of n objects via volume, sampling and containment queries with success probability at least $2/3$ must make $\Omega(n/\varepsilon^2)$ queries.*

We highlight that our lower bound holds even for equiponderous, axis-aligned polygons in the plane.

Upper bound for Klee's measure problem

Our lower bound for union volume estimation implies that any improved algorithm for Klee's measure problem must exploit the geometric structure of boxes. To this end, we exploit our knowledge of the corner coordinates and classify the boxes by shapes. In addition, we apply the geometric tool of orthogonal range searching. They allow us to approximate Klee's measure problem faster than the previous $O(n/\varepsilon^2)$ algorithm. Here and throughout, we assume that the dimension d is a constant; in particular, we suppress factors of $2^{O(d)}$ in the time complexity.

▶ **Theorem 2.** *There is an algorithm that runs in time $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ and with probability at least 0.9 computes a $(1+\varepsilon)$-approximation for Klee's measure problem.*

This is a strict improvement when $\varepsilon \leq 1/\log^{d+1}(n)$ is small. Consider for example the regime $\varepsilon = 1/\sqrt{n}$: Our algorithm runs in near-linear time, whereas the previous algorithm runs in quadratic time. We remark that the success probability can be boosted to any $1 - \delta$ by standard techniques which incurs an extra $\log(1/\delta)$ factor in the running time. Finally, we highlight that the core of our algorithm is an efficient method to sample uniformly and independently with a given density from the union of the input boxes. This can be of independent interest outside the context of volume estimation.

1.2 Related work

A major application of union volume estimation is *DNF counting*, in which we are given a Boolean formula in disjunctive normal form and want to count the number of satisfying assignments. Computing the exact number of satisfying assignments is #P-complete, so it likely requires superpolynomial time. Approximating the number of satisfying assignments can be achieved by a direct application of union volume estimation, as described in [19]. Their algorithm remains the state of the art for approximate DNF counting, see e.g. [24]. This has been extended to more general model counting [27, 9, 24], probabilistic databases [20, 15, 28], and probabilistic queries on databases [7].

We also mention *network reliability* as another application for union volume estimation, which was already discussed in [19]. Additionally, Karger's famous paper on the problem [17] uses the algorithm of [19] as a subroutine. However, the current state-of-the-art algorithms no longer use union volume estimation as a tool [8].

Finally, we want to draw a connection to the following well-known query sampling bound. Canetti, Even, and Goldreich [6] showed that approximating the mean of a random variable whose codomain is the unit interval requires $\Omega(1/\varepsilon^2)$ queries, thus obtaining tight bounds for the sampling complexity of the mean estimation problem. Their bound generalize to $\Omega(1/(\mu\varepsilon^2))$ on the number of queries needed to estimate the mean μ of a random variable in general. Before our work it was thus natural to expect that the $1/\varepsilon^2$ dependence in the number of queries for union volume estimation is optimal. However, whether the factor n is necessary, or the number of queries could be improved to, say, $O(n + 1/\varepsilon^2)$, was open to the best of our knowledge.

Klee's measure problem is an important problem in computational geometry. One reason for its significance is that techniques developed for it can often be adapted to solve various related problems, such as the depth problem (given a set of boxes, what is the largest number of boxes that can be stabbed by a single point?) [13] or Hausdorff distance under translation in L_∞ [14]. Moreover, various other problems can be reduced to Klee's measure problem or to its related problems. For example, deciding whether a set of boxes covers its boundary box can be reduced to Klee's measure problem [13]. The continuous k-center problem on graphs (i.e., finding centers that can lie on the edges of a graph that cover the vertices of a graph) is reducible to Klee's measure problem as well [29]. Finding the smallest hypercube containing at least k points among n given points can be reduced to the depth problem [16, 10, 13]. In light of this, it would be interesting to see if our approximation techniques generalize to any of these related problems.

2 Technical overview

We now sketch our upper and lower bounds at an intuitive level. The formal arguments will be presented in Section 3 and Section 4, respectively. In this paper we denote $[n] := \{1, 2 \dots, n\}$.

2.1 Upper bound for Klee's measure problem

Due to our lower bound, we have to exploit the structure of the input boxes to obtain a running time of the form $O\left((n + \frac{1}{\varepsilon^2}) \cdot \text{polylog}(n)\right)$. We follow the common algorithmic approach of sampling. Specifically, we aim to sample a set S from the union of boxes such that each point is selected with probability density p. For appropriately defined p, the value $|S|/p$ is a good estimate of the volume of the union. In the overview we assume that a proper p is given and focus on the main difficulty: creating a sample set S for the given p.

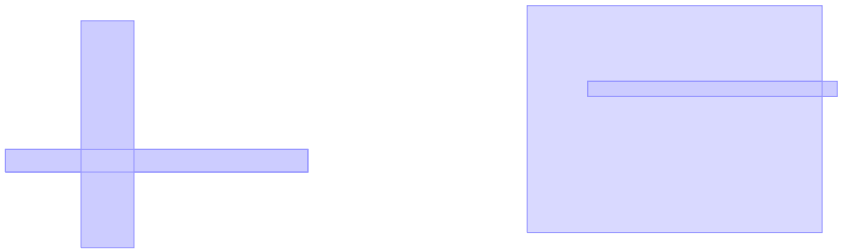


Figure 1 When the side lengths of two boxes differ a lot in at least one of their dimensions (in our examples, the y-axis), their intersection is small compared to their union.

We start by sorting the input boxes into *classes* by shape. Two boxes are in the same class if, in each dimension $k \in [d]$, their side lengths are both in $[2^{L_k}, 2^{L_k+1})$ for some $L_k \in \mathbb{Z}$. We call two classes *similar* if the corresponding side lengths are polynomially related (e.g., within a factor of n^4) in each dimension. We make two crucial observations:

1. Each class is similar to only polylogarithmically many classes (Observation 11).
2. Dissimilar classes have a small intersection compared to their union (Observation 12, Figure 1).

Our algorithm continues as follows. We scan through the classes in an arbitrary order. For each class, we sample with density p from the union of the boxes in this class, but we only keep a point if it is not contained in any class that comes later in the order. To efficiently test containment in a later class, we use an orthogonal range searching data structure (with an additional dimension for the index of the class). When the scan finishes, we obtain a desired sample set S.

Let us elaborate why the algorithm is efficient.

Sampling from a single class. Our approach is simple yet powerful: (i) grid the space into cells of side lengths comparable to the boxes in this class; (ii) sample points from the relevant cells uniformly at random; (iii) discard points outside the union by querying an orthogonal range searching data structure. See Figure 2. All these steps exploit the geometry of boxes. Since the grid cells and boxes have roughly the same shape, a significant fraction of the points sampled in (ii) are contained in the union, i.e., not discarded in (iii). The orthogonal range searching data structure allows us to quickly decide which points to discard. These explain the efficiency of sampling from a class.

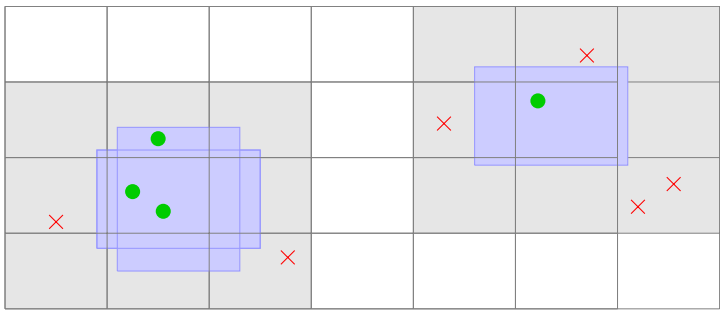


Figure 2 We sample points in the grid cells $\mathcal{G}$ that are intersected by a box $\mathcal{O}_i$ from a fixed class. We then use orthogonal range searching to determine whether a sampled point is in a box from the class and should be kept (•), or is not and should be discarded (×).

Bounding the amount of wasted work. Sampled points that appear in later classes also need to be discarded, and this is a potential (and only) source of inefficiency. So we need to bound the number of them. Such a point shows up later either (i) in a similar class, or (ii) in a dissimilar class. Our first observation stated that a class is similar to at most polylogarithmically many classes. So (i) can happen up to polylogarithmically many times from the perspective of a fixed point in the union. On the other hand, our second observation stated that the intersection of dissimilar classes is small. So on average (ii) rarely happens and does not affect the expected running time significantly.

2.2 Lower bound for union volume estimation

Our lower bound is shown by a reduction from the Query-Gap-Hamming problem: Given two vectors $x, y \in \{-1, +1\}^\ell$, distinguish whether their inner product is greater than $\sqrt{\ell}$ or less than $-\sqrt{\ell}$. It is known that any algorithm distinguishing these two cases with success probability at least $2/3$ must access $\Omega(\ell)$ bits of x and y.

We first sketch why $\Omega(1/\varepsilon^2)$ queries are necessary to $(1+\varepsilon)$-approximate the volume of the union of two objects in the query model. Given a Query-Gap-Hamming instance x, y, we construct two objects $X = \{(j, x_j) : j \in [\ell]\}$ and $Y = \{(j, y_j) : j \in [\ell]\}$. These are discrete point sets, but they can be inflated to polygons such that cardinality corresponds to volume. See Figure 3 for an example. Note that for all $k \in \{0, \dots, \ell\}$, we have

$$|X \cup Y| = \ell + k \iff \langle x, y\rangle = \ell - 2k.$$

If an algorithm $\mathcal{A}$ can $(1+\varepsilon)$-approximate $|X \cup Y|$, then it can approximate k within additive error $2\varepsilon\ell$, thus it can also approximate $\langle x, y\rangle$ within additive error $4\varepsilon\ell$. Setting $\ell = 1/(16\varepsilon^2)$, the additive error is at most $4\varepsilon\ell = 4 \cdot \frac{1}{4\sqrt{\ell}} \cdot \ell = \sqrt{\ell}$, which suffices to distinguish $\langle x, y\rangle = \sqrt{\ell}$ from $\langle x, y\rangle = -\sqrt{\ell}$ and thereby solves the Query-Gap-Hamming instance.

Note that $|X| = |Y| = \ell$, so a volume query does not disclose any information about x and y. Each sample or containment query accesses at most one bit of x or y. Therefore, algorithm $\mathcal{A}$ has to make $\Omega(\ell) = \Omega(1/\varepsilon^2)$ queries to X, Y.

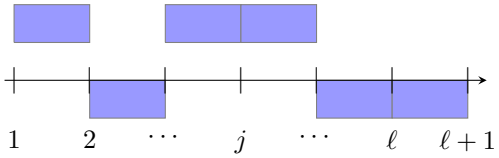


Figure 3 The vector $x = (+1, -1, +1, +1, -1, -1)$ represented as the set $\{(j, x_j) : j \in [6]\}$, where each point is drawn as a rectangle.

In order to generalize this lower bound for the union of two objects to an $\Omega(n/\varepsilon^2)$ lower bound for the union of n objects, we need to ensure that each query gives away only $O(1/n)$ bits of information about x and y. We apply two obfuscations that jointly slow down the exposure of bits; see Figure 4. Firstly, we introduce objects $X_1, \dots, X_n$ whose union is X and objects $Y_1, \dots, Y_n$ whose union is Y. Imagine cutting each X-induced rectangle in Figure 3 into n side-by-side pieces and distributing them randomly among $X_1, \dots, X_n$. Do the same for Y. The idea is that one needs to make $\Omega(n)$ containment queries on the rectangular region in order to hit the correct piece; only then is the corresponding bit revealed. Secondly, we introduce a large band shared by all X_i and Y_i for $i \in [n]$. In Figure 4, this is the long dark-blue rectangle that spans from left to right. Intuitively it enforces $\Omega(n)$ sample queries to obtain a single point that contains any information about x and y.

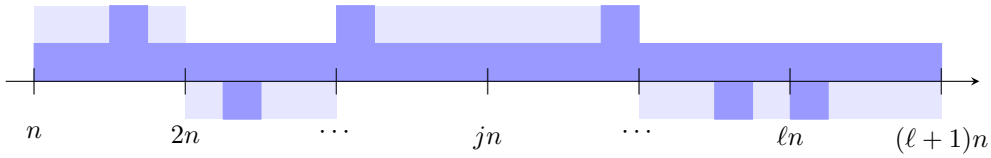


Figure 4 The vector y or $x = (+1, -1, +1, +1, -1, -1)$ gives rise to n polygons; one of these polygons is illustrated in dark blue. The light blue area indicates the union of all these n polygons.

3 Approximation algorithm for Klee's measure problem

In this section, we present our approximation algorithm for Klee's measure problem in constant dimension d:

▶ **Theorem 2.** *There is an algorithm that runs in time* $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ *and with probability at least* 0.9 *computes a* $(1+\varepsilon)$*-approximation for Klee's measure problem.*

Lemmas marked with $\star$ are folklore or straightforward. Their proofs can be found in the full version of this paper [5].

3.1 Preliminaries

In Klee's measure problem we are given *boxes* $O_1, \ldots, O_n \subset \mathbb{R}^d$. Here, a box is an object of the form $O_i = [a_1, b_1] \times \ldots \times [a_d, b_d]$, and as input we are given the coordinates $a_1, b_1, \ldots, a_d, b_d$ of each box. Based on these, it is easy to compute the side lengths and volume of each box.

Throughout we write $V := \text{Volume}(\bigcup_{i=1}^n O_i)$ for the volume of the union of boxes. The goal is to approximate V up to a factor of $1+\varepsilon$. Our approach is based on sampling, so now let us introduce the relevant notions.

Recall the Poisson distribution $\text{Pois}(\lambda)$ with mean and variance λ: It captures the number of active points in a universe, under the assumption that active points occur uniformly and independently at random across the universe, and that λ points are active on average. The following definition is usually referred to as a homogeneous *Poisson point process* at rate p. Intuitively, we activate each point in some universe $U \subset \mathbb{R}^d$ independently with "probability density" p, thus the number of activated points follows the Poisson distribution with mean $p \cdot \text{Volume}(U)$.

▶ **Definition 3** (p-sample)**.** *Let* $U \subset \mathbb{R}^d$ *be a measurable set, and let* $p \in [0,1]$*. We say that a random subset* $S \subseteq U$ *is a* p-sample of U *if for any measurable* $U' \subseteq U$ *we have that* $|S \cap U'| \sim \text{Pois}(p \cdot \text{Volume}(U'))$.

In particular, if S is a p-sample of U, then $|S| \sim \text{Pois}(p \cdot \text{Volume}(U))$. Two more useful properties follow from the definition:

(i) For any measurable subset $U' \subseteq U$, the restriction $S \cap U'$ is a p-sample of U'.

(ii) The union of p-samples of two disjoint sets U, U' is a p-sample of $U \cup U'$.

Besides sampling, we will apply orthogonal range searching to handle the so-called APPEARS(x, i) query: Given $x \in \mathbb{R}^d$ and $i \in \mathbb{N}$, is $x \in O_i \cup \cdots \cup O_n$?

▶ **Lemma 4** ($\star$)**.** *We can build a data structure in* $O(n \log^{d+1}(n))$ *time that answers* APPEARS(x, i) *queries in* $O(\log^{d+1}(n))$ *time.*

For our algorithm to work, we also need a constant-factor approximation of the volume V. It is known that this can be computed in $O(n)$ time [19]. In order to stay simple and self-contained, we state a weaker result by implementing the Karp-Luby algorithm [18] with the help of APPEARS queries.

▶ **Lemma 5** (⋆, adapted from [18]). *Given the data structure from Lemma 4, there exists an algorithm that computes in $O(n \log^{d+1}(n))$ time a 2-approximation to V with probability at least 0.9.*

3.2 Classifying boxes by shapes

As our first step in the algorithm, we classify boxes by their shapes.

▶ **Definition 6.** *Let $L_1, \ldots, L_d \in \mathbb{Z}$. We say that a box $O \subset \mathbb{R}^d$ is of type $(L_1, \ldots, L_d)$ if its side length in dimension k is contained in $[2^{L_k}, 2^{L_k+1})$, for all $k \in [d]$.*

Using this definition, we partition the input boxes $O_1, \ldots, O_n$ into *classes* $C_1, \ldots, C_m$ such that each class corresponds to one type of boxes. The notation is fixed from now on. For each $t \in [m]$, we denote $U_t := \bigcup_{O \in C_t} O \subset \mathbb{R}^d$, namely the union of boxes in class C_t.

Similar to APPEARS, we may use orthogonal range searching to handle the so-called INCLASS(x, t) query: Is a given point $x \in \mathbb{R}^d$ contained in U_t?

▶ **Lemma 7** (⋆). *We can build a data structure in $O(n \log^{d+1}(n))$ time that answers INCLASS(x, t) queries in $O(\log^{d+1}(n))$ time.*

Sampling from a class

The next lemma shows that we can p-sample from any U_t efficiently by rejection sampling.

▶ **Lemma 8.** *Given $t \in [m]$, $p \in [0, 1]$ and the data structure from Lemma 7, one can generate a p-sample of U_t in expected time $O(|C_t| \log |C_t| + p \cdot \text{Volume}(U_t) \cdot \log^{d+1}(n))$.*

Proof. Let $(L_1, \ldots, L_d)$ be the type of class C_t. We subdivide $\mathbb{R}^d$ into the grid

$$\mathcal{G}_\infty := \{[i_1\, 2^{L_1}, (i_1 + 1)\, 2^{L_1}) \times \cdots \times [i_d\, 2^{L_d}, (i_d + 1)\, 2^{L_d}) \mid i_1, \ldots, i_d \in \mathbb{Z}\}.$$

We call each element of $\mathcal{G}_\infty$ a *cell*. Let $\mathcal{G} := \{G \in \mathcal{G}_\infty \mid G \cap U_t \neq \emptyset\}$ be the set of cells that intersect with U_t. Let $U := \bigcup_{G \in \mathcal{G}} G$.

First we create a p-sample S of U as follows. Generate $K \sim \text{Pois}(p \cdot \text{Volume}(U))$, the number of points we are going to include. Then sample K points uniformly at random from U by repeating the following step K times: Select a cell $G \in \mathcal{G}$ uniformly at random and then sample a point from G uniformly at random. The sampled points constitute our set S.

Next we compute $S \cap U_t$: For each $x \in S$, we query INCLASS(x, t); if the answer is true then we keep x, otherwise we discard it. The resulting set $S \cap U_t$ is a p-sample of U_t, since restricting to a fixed subset preserves the p-sample property.

Before we analyze the running time, we show that U_t makes up a decent proportion of U. Recall that every box in class C_t is of type $(L_1, \ldots, L_d)$. Consider the projection to any dimension $k \in [d]$. Each projected box from C_t can intersect at most three projected cells from $\mathcal{G}$. So each box from C_t intersects at most 3^d cells from $\mathcal{G}$, implying that $|\mathcal{G}| \leq 3^d\, |C_t|$. Moreover, since the volume of any cell is at most the volume of a box in C_t, we have $\text{Volume}(U) \leq 3^d\ \text{Volume}(U_t)$.

Recall that we assume d to be constant and hence $3^d = O(1)$. The computation of $\mathcal{G}$ takes $O(|\mathcal{G}| \log |\mathcal{G}|) \subseteq O(|C_t| \log |C_t|)$ time. The remaining time is dominated by the inClass queries. The expected size of S is $p \cdot \text{Volume}(U) \leq 3^d\, p\, \text{Volume}(U_t)$. As we query inClass once for each point of S, the total expected time is $O(p \cdot \text{Volume}(U_t) \cdot \log^{d+1}(n))$ by Lemma 7. ◀

Classes do not overlap much

We show the following interesting property of classes, that the sum of their volumes is within a polylogarithmic factor of the total volume V.

▶ **Lemma 9.** *We have* $\sum_{t=1}^{m} \text{Volume}(U_t) \leq 2^{3d+1} \log^d(n) \cdot V$.

We later exploit it to draw p-samples from $\bigcup_{i=1}^{n} O_i = \bigcup_{t=1}^{m} U_t$ efficiently. Towards proving the lemma, let us collect some simple observations.

▶ **Definition 10.** *We say that a class of type* $(L_1, \ldots, L_d)$ *is* similar *to a class of type* $(L'_1, \ldots, L'_d)$ *if* $2^{|L_k - L'_k|} < n^4$ *for all* $k \in [d]$. *Otherwise they are said to be* dissimilar.

▶ **Observation 11.** *Every class is similar to at most* $8^d \log^d(n)$ *classes.*

Proof. Fix a type $(L_1, \ldots, L_d)$. For each $k \in [d]$, there are at most $8 \log n$ many integers L'_k such that $2^{|L_k - L'_k|} < n^4$. ◀

▶ **Observation 12.** *If two boxes* O, O' *are in dissimilar classes, then* $\text{Volume}(O \cap O') \leq 2V/n^4$.

Proof. Let $(L_1, \ldots, L_d)$ be the type of O, and $(L'_1, \ldots, L'_d)$ be the type of O'. Since the boxes belong to dissimilar classes, there is a dimension $k \in [d]$ such that $2^{|L_k - L'_k|} \geq n^4$. Without loss of generality, assume $2^{L_k - L'_k} \geq n^4$; the other case is symmetric. Let $[a_k, b_k]$ and $[a'_k, b'_k]$ be the projections of O and O' onto dimension k, respectively. Note that $b_k - a_k \in [2^{L_k}, 2^{L_k+1})$ and $b'_k - a'_k \in [2^{L'_k}, 2^{L'_k+1})$. So we have $\frac{b_k - a_k}{b'_k - a'_k} \geq 2^{L_k - (L'_k+1)} \geq n^4/2$. In other words, at most a $2/n^4$ fraction of the interval $[a_k, b_k]$ intersects the interval $[a'_k, b'_k]$. Hence, $\text{Volume}(O \cap O') \leq \text{Volume}(O) \cdot 2/n^4 \leq 2V/n^4$. ◀

Proof of Lemma 9. Without loss of generality assume $\text{Volume}(U_1) \geq \cdots \geq \text{Volume}(U_m)$. We construct a set of indices $T \subseteq [m]$ by the following procedure:

- Initialize $T = \emptyset$.
- For $t = 1, \ldots, m$, if C_t and C_s are dissimilar for all $s \in T$, then add t to T.

For $t \in [m]$, we have $t \notin T$ only if there exists an $s \in T$ such that C_s, C_t are similar and $\text{Volume}(U_s) \geq \text{Volume}(U_t)$; we call s a *witness* of t. If multiple witnesses exist, then we pick an arbitrary one. Conversely, every $s \in T$ can be a witness at most $8^d \log^d(n)$ times by Observation 11. Hence,

$$\sum_{t=1}^{m} \text{Volume}(U_t) \leq 8^d \log^d(n) \cdot \sum_{t \in T} \text{Volume}(U_t). \tag{1}$$

It remains to bound $\sum_{t \in T} \text{Volume}(U_t)$. Consider any distinct $s, t \in T$. By construction, C_s and C_t are dissimilar; and each class contains at most n boxes. So $\text{Volume}(U_s \cap U_t) \leq n^2 \cdot (2V/n^4) = 2V/n^2$ by Observation 12. Using this and inclusion-exclusion, we bound

$$\begin{aligned} \sum_{t \in T} \text{Volume}(U_t) \;&\leq\; \text{Volume}\left(\bigcup_{t \in T} U_t\right) + \sum_{\{s,t\} \subseteq T} \text{Volume}(U_s \cap U_t) \\ &\leq\; V + \binom{m}{2} \frac{2V}{n^2} \;\leq\; 2V. \end{aligned}$$

Plugging this into the right-hand side of Expression (1), we obtain the lemma statement. ◀

3.3 Joining the classes

Recall that $C_1, \ldots, C_m$ are the classes of the input boxes and $U_1, \ldots, U_m$ their respective unions. Assume without loss of generality that the boxes are ordered in accordance with the class ordering, that is, $C_1 = \{O_1, \cdots, O_{i_1}\}$ form the first class, $C_2 = \{O_{i_1+1}, \cdots, O_{i_2}\}$ form the second class, and so on. In general, we assume that $C_t = \{O_{i_{t-1}+1}, \ldots, O_{i_t}\}$ for all $t \in [m]$, where $0 = i_0 < i_1 < \ldots < i_m = n$.

Let $D_t := U_t \setminus (\bigcup_{s=t+1}^{m} U_s)$ be the points in U_t that are not contained in later classes. Note that $D_1, \ldots, D_m$ is a partition of $\bigcup_{t=1}^{m} U_t = \bigcup_{i=1}^{n} O_i$. Hence, to generate a p-sample of $\bigcup_{i=1}^{n} O_i$, it suffices to draw p-samples from each D_t and then take their union.[4] To this end, we draw a p-sample S_t from U_t via Lemma 8. Then we remove all $x \in S_t$ for which $\textsc{appears}(x, i_t + 1) = \text{true}$; these are exactly the points that appear in a later class. What remains is a p-sample of D_t. The union of these sets thus is a p-sample of $\bigcup_{i=1}^{n} O_i$, and we can use the size of this p-sample to estimate the volume V of $\bigcup_{i=1}^{n} O_i$. The complete algorithm is summarized in Algorithm 1.

Algorithm 1 Approximation of Klee's measure.

1. Partition the boxes into classes $C_1, \ldots, C_m$. Relabel the boxes so that their indices are in accordance with the class ordering, i.e., $C_t = \{O_{i_{t-1}+1}, \ldots, O_{i_t}\}$ for all $t \in [m]$.
2. Build the data structures from Lemmas 4 and 7.
3. Obtain a crude estimate $\widetilde{V}$ by Lemma 5. Set $p := 8/(\varepsilon^2 \widetilde{V})$.
4. For $t = 1, \ldots, m$ do:
   - Draw a p-sample S_t from the union $U_t = \bigcup_{O \in C_t} O$ via Lemma 8.
   - Compute $|S'_t|$ where $S'_t := \{x \in S_t : \textsc{appears}(x, i_t + 1) = \text{false}\}$.
5. Output $\sum_{t=1}^{m} |S'_t|/p$.

▶ **Lemma 13.** *Conditioned on $\widetilde{V} \le 2V$, Algorithm 1 outputs a $(1+\varepsilon)$-approximation to V with probability at least $3/4$.*

Proof. Note that for all $t \in [m]$, the set S'_t is a p-sample of D_t. Since $D_1, \ldots, D_m$ partition $\bigcup_{t=1}^{m} U_t = \bigcup_{i=1}^{n} O_i$, their union $\bigcup_{t=1}^{m} S'_t$ is a p-sample of $\bigcup_{i=1}^{n} O_i$. It follows that $N := \sum_{t=1}^{m} |S'_t| \sim \text{Pois}(pV)$. The expectation and variance of N are $pV = 8V/(\varepsilon^2 \widetilde{V}) \ge 4/\varepsilon^2$. So by Chebyshev,

$$\Pr(|N - pV| > \varepsilon pV) \le \frac{\text{Var}[N]}{(\varepsilon pV)^2} \le \frac{1}{4}.$$

Hence, with probability at least $3/4$, the output N/p is a $(1+\varepsilon)$-approximation to V. ◀

▶ **Lemma 14.** *Conditioned on $\widetilde{V} \ge \frac{V}{2}$, Algorithm 1 runs in time $O\left((n + \frac{1}{\varepsilon^2}) \cdot \log^{2d+1}(n)\right)$ in expectation.*

Proof. Step 1 takes $O(n \log n)$ time: we can compute the side lengths of each box, determine its class, then sort the boxes by class index. Step 2 takes $O(n \log^{d+1}(n))$ time by Lemmas 4 and 7. Step 3 takes $O(n \log^{d+1}(n))$ time by Lemma 5.

[4] This idea has previously been used on objects, by considering the difference $D'_i := O_i \setminus (\bigcup_{j=i+1}^{n} O_j)$ [18, 25], while we use this idea on classes.

In step 4, iteration t, sampling S_t costs expected time $O((i_t - i_{t-1})\log(i_t - i_{t-1}) + p\,\text{Volume}(U_t)\cdot\log^{d+1}(n))$ by Lemma 8, and computing S'_t costs expected time $O((1 + p\,\text{Volume}(U_t))\cdot\log^{d+1}(n))$ by Lemma 4. Over all iterations, the expected running time is

$$O\left(\log^{d+1}(n)\cdot\left(n + p\sum_{t=1}^{m}\text{Volume}(U_t)\right)\right).$$

Substituting $p = 8/(\varepsilon^2\tilde{V}) \le 16/(\varepsilon^2 V)$ and applying Lemma 9, we can bound

$$p\sum_{t=1}^{m}\text{Volume}(U_t) \le \frac{16}{\varepsilon^2 V}\sum_{t=1}^{m}\text{Volume}(U_t) \le \frac{2^{3d+5}\log^d(n)}{\varepsilon^2}.$$

Hence the expected running time of step 4 is $O\left(\log^{2d+1}(n)\cdot(n + \frac{1}{\varepsilon^2})\right)$.

Finally, step 5 takes $O(n)$ time. ◀

Proof of Theorem 2. Run Algorithm 1 with a time budget tenfold the bound in Lemma 14; abort the algorithm as soon as it spends more time than the budget allows. So the stated time bound is clearly satisfied. Now consider three bad events:

- $\widetilde{V} \notin [\frac{V}{2}, 2V]$.
- $\widetilde{V} \in [\frac{V}{2}, 2V]$, but the algorithm is aborted.
- $\widetilde{V} \in [\frac{V}{2}, 2V]$ and the algorithm is not aborted, but it does not output a $(1+\varepsilon)$-approximation to V.

By Lemma 5, the first event happens with probability at most 0.1. By Lemma 14 and Markov's inequality, the second event happens with probability at most 0.1. Lastly, by Lemma 13, the third event happens with probability at most 1/4. So the total error probability is at most $0.1 + 0.1 + \frac{1}{4} = \frac{9}{20}$. If none of the bad events happen, then the algorithm correctly outputs a $(1+\varepsilon)$-approximation to V. The success probability of $\frac{11}{20}$ can be boosted to, say, 0.9 by returning the median of a sufficiently large constant number of repetitions. ◀

4 Lower bound for union volume estimation

In this section, any randomized algorithm or protocol is assumed to have success probability at least 2/3. We consider the discrete version of union volume estimation in two dimensions, where each object O_i is a finite subset of the integer lattice $\mathbb{Z}^2$. The goal is to $(1+\varepsilon)$-approximate the cardinality $|O_1 \cup \ldots \cup O_n|$ of their union. The algorithm does not have direct access to the objects, but it can ask three forms of queries:

- Volume(O_i): Return the cardinality $|O_i|$.
- Sample(O_i): Draw a uniform random point from O_i.
- Contains($(a,b), O_i$): Given a point $(a,b) \in \mathbb{Z}^2$, return whether $(a,b) \in O_i$ or not.

We aim to prove that $\Omega(n/\varepsilon^2)$ queries are necessary. Later in Section 5 we translate this to a lower bound in the continuous space $\mathbb{R}^2$, thereby proving Theorem 1.

Our starting point is the Query-Gap-Hamming problem: The inputs are two (hidden) vectors $x, y \in \{-1,1\}^\ell$ and we can access one bit of x or y of our choice at a time. The goal is to distinguish the cases $\langle x, y\rangle \ge \sqrt{\ell}$ and $\langle x, y\rangle \le -\sqrt{\ell}$ using as few accesses as possible. The following lemma is folklore; its proof can be found in the full version of this paper [5].

▶ **Lemma 15 (⋆).** *There exists a constant $\delta \in (0,1)$ with the following property. For every $\ell \in \mathbb{N}$, any randomized algorithm for Query-Gap-Hamming on vectors of length ℓ needs at least $\delta\ell$ accesses in expectation.*

From now on, let us fix the constant $\delta \in (0,1)$ guaranteed by Lemma 15. Let $n \in \mathbb{N}$, $\varepsilon \in (0,1)$ be arbitrary and define $\ell := \frac{1}{36\varepsilon^2}$. We reduce Query-Gap-Hamming on vectors of length ℓ to estimating the cardinality of the union of $2n$ objects in $\mathbb{Z}^2$.

The reduction

From the hidden vectors $x, y \in \{-1,1\}^\ell$ we construct $2n$ objects $X_1, \ldots, X_n, Y_1, \ldots, Y_n \subset \mathbb{Z}^2$. Let $R := \{(n+1, 0), \ldots, (n\ell + n, 0)\}$. Take independent random permutations $\pi_1, \ldots, \pi_\ell$ of $[n]$ and define

$$X_i := R \cup \{(jn + \pi_j(i), x_j) : j \in [\ell]\}$$

for $i \in [n]$. Then take another set of independent random permutations $\tau_1, \ldots, \tau_\ell$ and define

$$Y_i := R \cup \{(jn + \tau_j(i), y_j) : j \in [\ell]\}$$

for $i \in [n]$. Note that R is a subset of all X_i and Y_i.

How is $\langle x, y\rangle$ related to the cardinality of the union $|X_1 \cup \cdots \cup X_n \cup Y_1 \cup \cdots \cup Y_n|$? Consider an arbitrary index $j \in [\ell]$. If $x_j = y_j$ then the point sets $\{(jn + \pi_j(i), x_j) : i \in [n]\}$ and $\{(jn + \tau_j(i), y_j) : i \in [n]\}$ are equal (regardless of the concrete permutations), so they together contribute n to the cardinality of the union. On the other hand, if $x_j \neq y_j$ then they are disjoint and thus contribute $2n$. Furthermore, the point set R is contained in all objects and contributes $n\ell$. Hence, the cardinality of the union is equal to

$$n\ell + \sum_{j : x_j = y_j} n + \sum_{j : x_j \neq y_j} 2n = \frac{5}{2} n\ell - \frac{1}{2} n \cdot \left(\sum_{j : x_j = y_j} 1 + \sum_{j : x_j \neq y_j} (-1) \right) = \frac{5}{2} n\ell - \frac{1}{2} n \langle x, y \rangle.$$

Suppose we get a $(1+\varepsilon)$-approximation ρ to the cardinality of the union, then

$$\rho \in \left[(1-\varepsilon)(\tfrac{5}{2} n\ell - \tfrac{1}{2} n\langle x, y\rangle),\ (1+\varepsilon)(\tfrac{5}{2} n\ell - \tfrac{1}{2} n\langle x, y\rangle) \right].$$

Since $|\langle x, y\rangle| \leq \ell$, by computing $(\frac{5}{2} n\ell - \rho) \cdot \frac{2}{n}$ we obtain a value in $\langle x, y\rangle \pm 6\varepsilon\ell$, that is, an additive $6\varepsilon\ell$ approximation to $\langle x, y\rangle$. Our choice of $\varepsilon = \frac{1}{6\sqrt{\ell}}$ allows to distinguish $\langle x, y\rangle \geq \sqrt{\ell}$ from $\langle x, y\rangle \leq -\sqrt{\ell}$.

The reduction is not yet complete. The catch is that an algorithm for union volume estimation can only learn about the objects via queries. It is left to *us* to answer those queries. In Algorithm 2 we describe how to answer queries on object X_i. Queries on object Y_i can be handled similarly by replacing π_j with τ_j and x_j with y_j. The correctness is easy to verify, and this completes the reduction. So far, the argument works for arbitrary permutations; randomness becomes relevant when we bound the number of bit accesses.

Algorithm 2 Answering queries.

- For query Volume(X_i): Return $(n+1)\ell$.
- For query Sample(X_i): Draw a random block $j \in [\ell]$ and a random shift $s \in [n]$. With probability $1 - \frac{1}{n+1}$ return $(jn + s, 0)$. Otherwise return $(jn + \pi_j(i), x_j)$.
- For query Contains($(a,b), X_i$): If $(a,b) \in R$ then return true. Else, compute $j := \lfloor a/n \rfloor$ and $s := a - jn$. If $j \notin [\ell]$ or $\pi_j(i) \neq s$ then return false. Otherwise, if $x_i = b$ then return true; else return false.

Bounding the number of accesses

Now for the sake of contradiction, suppose there exists a randomized algorithm $\mathcal{A}$ that $(1+\varepsilon)$-approximates the cardinality of the union of any $2n$ objects in $\mathbb{Z}^2$, using less than $\frac{\delta}{360\cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2}$ queries. We run $\mathcal{A}$ on the objects defined earlier (using Algorithm 2 to handle its queries). From the output, we can distinguish $\langle x, y\rangle \geq \sqrt{\ell}$ and $\langle x, y\rangle \leq -\sqrt{\ell}$. We claim that less than $\delta\ell$ bits of x, y are accessed from the side of Algorithm 2.

To this end, note that accesses to x, y only happen in the "otherwise" clauses of Sample() and Contains() in Algorithm 2. For analysis we isolate the following mini game:

▶ **Lemma 16.** *Consider a game of $m := \lfloor 2^{-9/\delta} n \rfloor$ rounds. At the start, Merlin generates a secret random permutation π of $[n]$. In each round, Arthur adaptively sends one of the following two requests to Merlin:*

- Random()*: Answer "yes" with probability $\frac{1}{n+1}$ and "no" with the remaining probability.*
- Probe(i, s)*: Answer "yes" if $\pi(i) = s$ and "no" otherwise.*

No matter what strategy Arthur employs, the probability that Merlin ever answers "yes" in the game is at most $2\delta/5$.

Proof. Let E be the event that Merlin ever answers "yes" *and* the first one was due to a Random() request. Let F be the event that Merlin ever answers "yes" *and* the first one was due to a Probe() request. We will bound $\Pr(E)$ and $\Pr(F)$ separately.

To bound $\Pr(E)$, define events $E_1, \ldots, E_m$ where E_t indicates that Arthur's t-th request is Random() and the answer is "yes". Clearly $\Pr(E_t) \leq \frac{1}{n+1}$, thus $\Pr(E) \leq \sum_{t=1}^{m} \Pr(E_t) \leq \frac{m}{n+1} < 2^{-9/\delta} < \delta/15$.

To bound $\Pr(F)$ we use an entropy argument. In particular, we will describe and analyze an encoding scheme for a random permutation π of $[n]$.

The encoder receives π and acts as Merlin in the game, using π as the secret permutation. It interacts with Arthur till the end of the game. If F does not happen then the encoding is simply a bit `0` followed by π. On the other hand, if F happens, then consider the first round $T \in [m]$ when "yes" appeared; denote that request by Probe(i, s). The encoding starts with a bit `1`, then the number T, and finishes with the ordering π' of $[n] \setminus \{i\}$ induced by π.

The decoder receives the encoding and tries to reconstruct π. First it reads the leading bit of the encoding. If the bit is `0` then π immediately follows. If the bit is `1` then the decoder recovers the number T. It pretends to be Merlin and starts a game with Arthur, using the same random tape as the encoder but without knowledge of π. For the first $T-1$ requests it just answers "no" to Arthur. Now comes the T-th request from Arthur, which must be a Probe(i, s) request with answer "yes". This allows the decoder to deduce $\pi(i) = s$. The remaining entries in π can be fully restored from π'.

Writing $p := \Pr(F)$, the encoding length is at most

$$\begin{aligned}
&(1-p)\Big(1 + \lceil \log(n!) \rceil\Big) + p\Big(1 + \lceil \log(m) \rceil + \lceil \log((n-1)!) \rceil\Big) \\
<&(1-p)\big(\log(n!) + 3\big) + p\big(\log(n) - 9/\delta + \log((n-1)!) + 3\big) \\
=&\log(n!) + 3 - 9p/\delta.
\end{aligned}$$

where we used $m := \lfloor 2^{-9/\delta} n \rfloor$ in the second line. Since the encoding length must be at least the Shannon entropy $\log(n!)$ of the encoded permutation π, we conclude that $\Pr(F) = p \leq \delta/3$.

Putting the two bounds together, we have $\Pr(E \cup F) \leq \Pr(E) + \Pr(F) \leq 2\delta/5$. ◀

Where does the game come into play? The reader might find it useful to recall Figure 4. The space $\mathbb{Z}^2$ is split into ℓ blocks horizontally, where each block $j \in [\ell]$ holds information about the j-th bit of the input vector. Let us focus on a particular block j. As $\mathcal{A}$ runs, some of

its queries "hit" this block while others don't. Precisely, we say that a Sample(X_i) query *hits* j if the random block chosen in Algorithm 2 is j. Likewise, we say that a Contains($(a,b), X_i$) query *hits* j if $\lfloor a/n \rfloor = j$. Now we restrict ourselves to the queries hitting j and ignore all the others. Then, over time, $\mathcal{A}$ (as Arthur) is exactly playing the game with us (as Merlin) on the secret permutation π_j: Sample() queries correspond to Random() requests; Contains() queries correspond to Probe() requests; Algorithm 2 branches to the "otherwise" clauses if and only if the answer is "yes". Whatever information $\mathcal{A}$ collects in the queries hitting other blocks j', it has no effect on the decision in this game, as all other permutations are independent of π_j, and as x, y do not play a role in this game.

With this in mind, let random variable N_j count the number of queries hitting j. Define

$$J := \{j \in [\ell] : x_j \text{ is accessed and } N_j \le m\} \quad \text{and} \quad J' := \{j \in [\ell] : N_j > m\}.$$

where $m := \lfloor 2^{-9/\delta} n \rfloor$. By Lemma 16, $\Pr(j \in J) \le 2\delta/5$ for every $j \in [\ell]$, so $\mathbb{E}[|J|] \le 2\delta\ell/5$. On the other hand, by definition of J' and our assumption that the number of queries is at most $\frac{\delta}{360 \cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2}$, we can bound

$$|J'| < \frac{1}{m+1} \cdot \frac{\delta}{360 \cdot 2^{9/\delta}} \cdot \frac{n}{\varepsilon^2} \le \frac{\delta\ell}{10}.$$

So Algorithm 2 accesses less than $\frac{2\delta\ell}{5} + \frac{\delta\ell}{10} = \frac{\delta\ell}{2}$ bits of x in expectation. Symmetrically, the same bound holds for y. Altogether it accesses less than $\delta\ell$ bits of x, y. But as we argued, the output can be used to distinguish $\langle x, y\rangle \ge \sqrt{\ell}$ and $\langle x, y\rangle \le -\sqrt{\ell}$. This contradicts Lemma 15.

5 Reduction from discrete to continuous space

Union volume estimation and Klee's measure problem were formulated in the continuous space $\mathbb{R}^d$. Their analogs in discrete space $\mathbb{Z}^d$ are only simpler due to the following reduction. Consider the natural embedding φ that blows up each point $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$ into a continuous cube $\varphi(x) = [x_1, x_1 + 1] \times \cdots \times [x_d, x_d + 1] \subset \mathbb{R}^d$. For any finite subset $X \subset \mathbb{Z}^d$, denote its image by $\varphi(X) \subset \mathbb{R}^d$. We have:

- Volume($\varphi(X)$) $= |X|$.
- To sample from $\varphi(X)$ uniformly, we can first sample $x \in X$ uniformly and then sample from $\varphi(x)$ uniformly.
- If X is a discrete box then $\varphi(X)$ is a continuous box, and vice versa.

Given this correspondence, our algorithm for Klee's measure problem can be made to handle discrete boxes in $\mathbb{Z}^d$. On the other hand, the discrete objects in our lower bound construction can be realized as axis-aligned polygons in $\mathbb{R}^2$.

References

1 Pankaj K. Agarwal. An improved algorithm for computing the volume of the union of cubes. In David G. Kirkpatrick and Joseph S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry, Snowbird, Utah, USA, June 13-16, 2010*, pages 230–239. ACM, 2010. doi:10.1145/1810959.1811000.

2 Pankaj K. Agarwal, Haim Kaplan, and Micha Sharir. Computing the volume of the union of cubes. In Jeff Erickson, editor, *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 294–301. ACM, 2007. doi:10.1145/1247069.1247121.

3 Karl Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom.*, 45(5-6):225–233, 2012. doi:10.1016/j.comgeo.2011.12.001.

4 Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.*, 43(6-7):601–610, 2010. doi:10.1016/j.comgeo.2010.03.004.

5 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. *CoRR*, abs/2410.00996, 2024. doi:10.48550/arXiv.2410.00996.

6 Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995. doi:10.1016/0020-0190(94)00171-T.

7 Nofar Carmeli, Shai Zeevi, Christoph Berkholz, Alessio Conte, Benny Kimelfeld, and Nicole Schweikardt. Answering (unions of) conjunctive queries using random access and random-order enumeration. *ACM Trans. Database Syst.*, 47(3):9:1–9:49, 2022. doi:10.1145/3531055.

8 Ruoxu Cen, William He, Jason Li, and Debmalya Panigrahi. Beyond the quadratic time barrier for network unreliability. 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 1542–1567. SIAM, 2024. doi:10.1137/1.9781611977912.62.

9 Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 200–216. Springer, 2013. doi:10.1007/978-3-642-40627-0_18.

10 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

11 Timothy M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM J. Comput.*, 32(3):700–716, 2003. doi:10.1137/S0097539702404389.

12 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. doi:10.1016/j.comgeo.2009.01.007.

13 Timothy M. Chan. Klee's measure problem made easy. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 410–419. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.51.

14 Timothy M. Chan. Minimum L_∞ Hausdorff distance of point sets under translation: Generalizing klee's measure problem. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.24.

15 Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007. doi:10.1007/s00778-006-0004-3.

16 David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discret. Comput. Geom.*, 11:321–350, 1994. doi:10.1007/BF02574012.

17 David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM J. Comput.*, 29(2):492–514, 1999. doi:10.1137/S0097539796298340.

18 Richard M. Karp and Michael Luby. Monte-Carlo algorithms for the planar multiterminal network reliability problem. *J. Complex.*, 1(1):45–64, 1985. doi:10.1016/0885-064X(85)90021-4.

19 Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989. doi:10.1016/0196-6774(89)90038-2.

20 Benny Kimelfeld, Yuri Kosharovsky, and Yehoshua Sagiv. Query efficiency in probabilistic XML models. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 701–714. ACM, 2008. doi:10.1145/1376616.1376687.

21 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.

22 Michael G. Luby. Monte-Carlo methods for estimating system reliability. Technical report, Report UCB/CSD 84/168, Computer Science Division, University of California, Berkeley, 1983.

23 Kuldeep S. Meel, Sourav Chakraborty, and N. V. Vinodchandran. Estimation of the size of union of delphic sets: Achieving independence from stream size. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 41–52. ACM, 2022. doi:10.1145/3517804.3526222.

24 Kuldeep S. Meel, Aditya A. Shrotri, and Moshe Y. Vardi. Not all FPRASs are equal: demystifying FPRASs for DNF-counting. *Constraints An Int. J.*, 24(3-4):211–233, 2019. doi:10.1007/s10601-018-9301-x.

25 Kuldeep S. Meel, N. V. Vinodchandran, and Sourav Chakraborty. Estimating the size of union of sets in streaming models. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 126–137. ACM, 2021. doi:10.1145/3452021.3458333.

26 Mark H. Overmars and Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

27 Aduri Pavan, N. V. Vinodchandran, Arnab Bhattacharyya, and Kuldeep S. Meel. Model counting meets F_0 estimation. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 299–311. ACM, 2021. doi:10.1145/3452021.3458311.

28 Christopher Ré, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 886–895. IEEE Computer Society, 2007. doi:10.1109/ICDE.2007.367934.

29 Qiaosheng Shi and Binay K. Bhattacharya. Application of computational geometry to network p-center location problems. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008.

30 Srikanta Tirthapura and David P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 283–294. ACM, 2012. doi:10.1145/2213556.2213595.

31 Jan van Leeuwen and Derick Wood. The measure problem for rectangular ranges in d-space. *J. Algorithms*, 2(3):282–300, 1981. doi:10.1016/0196-6774(81)90027-4.

32 Hakan Yildiz and Subhash Suri. On Klee's measure problem for grounded boxes. In Tamal K. Dey and Sue Whitesides, editors, *Proceedings of the 28th ACM Symposium on Computational Geometry, Chapel Hill, NC, USA, June 17-20, 2012*, pages 111–120. ACM, 2012. doi:10.1145/2261250.2261267.

Geometric Spanners of Bounded Tree-Width

Kevin Buchin
TU Dortmund, Germany

Carolin Rehs
TU Dortmund, Germany

Torben Scheele
TU Dortmund, Germany

Abstract

Given a point set P in the Euclidean space, a geometric t-spanner G is a graph on P such that for every pair of points, the shortest path in G between those points is at most a factor t longer than the Euclidean distance between those points. The value $t \geq 1$ is called the dilation of G. Commonly, the aim is to construct a t-spanner with additional desirable properties. In graph theory, a powerful tool to admit efficient algorithms is bounded tree-width. We therefore investigate the problem of computing geometric spanners with bounded tree-width and small dilation t.

Let d be a fixed integer and $P \subset \mathbb{R}^d$ be a point set with n points. We give a first algorithm to compute an $\mathcal{O}(n/k^{d/(d-1)})$-spanner on P with tree-width at most k. The dilation obtained by the algorithm is asymptotically worst-case optimal for graphs with tree-width k: We show that there is a set of n points such that every spanner of tree-width k has dilation $\mathcal{O}(n/k^{d/(d-1)})$. We further prove a tight dependency between tree-width and the number of edges in sparse connected planar graphs, which admits, for point sets in $\mathbb{R}^2$, a plane spanner with tree-width at most k and small maximum vertex degree.

Finally, we show an almost tight bound on the minimum dilation of a spanning tree of n equally spaced points on a circle.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Computational Geometry, Geometric Spanner, Tree-width

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.26

Related Version *Full Version*: https://arxiv.org/abs/2412.06316

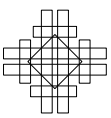
1 Introduction

Geometric spanners are an extensively studied area in computational geometry, see [7, 23] for surveys. A geometric spanner for a point set $P \in \mathbb{R}^d$ is a weighted graph on P, where the weight of an edge is the Euclidean distance between its endpoints, aiming for sparsity while avoiding long paths between two points in P. This is formalised by the dilation $t = \max\left\{\frac{d(p,p')}{|pp'|} \mid p, p' \in P\right\}$, where $d(p, p')$ is the distance between two points in G and $|pp'|$ is the Euclidean distance between p and p'. A geometric spanner with dilation at most t is also called a *t-spanner*.

When geometric spanners were introduced in 1986 [12], sparsity was obtained by requiring planarity. Since then, plane geometric spanners have been widely studied [7]. But also other measures are of interest: Especially spanners with a linearly bounded number of edges or a bounded vertex degree have been widely researched [20, 23, 30, 18, 6, 14, 19, 5].

In graph theory, an important tool to design efficient algorithms for in general NP-hard problems is to restrict these problems to graphs which are bounded in certain parameters. Especially graphs with bounded *tree-width* [25, 26] allow polynomial-time algorithms for many in general NP-hard problems using a dynamic programming algorithm on the tree decomposition. By Courcelle's Theorem, every problem that is describable in monadic second

© Kevin Buchin, Carolin Rehs, and Torben Scheele;
licensed under Creative Commons License CC-BY 4.0



order logic with vertex and edge quantification is solvable in polynomial time for graphs with bounded tree-width [13]. Thus, geometric spanners with bounded tree-width would allow polynomial time solutions for many in general NP-hard problems and thus be a strong tool for many geometric applications. For instance, Cabello and Knauer [9] recently gave some examples for efficient algorithms for geometric problems on graphs with bounded tree-width. They show that via orthogonal range searching, for graphs of n vertices and tree-width k with $k \geq 3$, the sum of the distances between all pairs of vertices can be computed in $\mathcal{O}(n \log^{k-1} n)$ time. Further, they show that the dilation of a geometric graph of bounded tree-width can be computed in $\mathcal{O}(n \log^{k+1} n)$ time. That makes it possible to compute the dilation of every tree-width bounded graph, but does not lead to tree-width bounded spanners with low dilation. In [8], Cabello shows how to compute the inverse geodesic length in graphs of bounded tree-width. Further, Cabello and Rote [10] show how to compute obnoxious centers in graphs with bounded tree-width in $\mathcal{O}(n \log n)$ time.

Therefore, we aim to construct spanners with bounded tree-width. There are spanners with sublinear tree-width. More specifically, since planar graphs have tree-width $\mathcal{O}(\sqrt{n})$ [22] (where n is the number of vertices), for point sets in the Euclidean plane, constant dilation spanners with tree-width $\mathcal{O}(\sqrt{n})$ can be obtained by plane spanner approaches such as the Delaunay triangulation or the greedy triangulation. For point sets in $\mathbb{R}^d$, we point out that the greedy spanner has tree-width $\mathcal{O}(n^{1-1/d})$, using separator results from [21].

While this gives some first results, constructing spanners with tree-width $k \in o(n^{1-1/d})$ will require other constructions. The only known construction for small tree-width and a guarantee on the dilation is the Euclidean minimum spanning tree: it has tree-width $k = 1$ and worst-case dilation $n - 1$.

We present a first algorithm that provides a trade-off between tree-width and the dilation:

▶ **Theorem 1.** *Given a set $P \subset \mathbb{R}^d$ of n points for some fixed d and a positive integer $k \leq n^{1-1/d}$. There is a geometric spanner of tree-width k on P with a dilation of $\mathcal{O}\left(n/k^{d/(d-1)}\right)$ and bounded degree that can be computed in time $\mathcal{O}(n^2 \log n)$.*

We complement our result with a matching lower bound.

▶ **Theorem 2.** *Let $d \geq 2$ be a fixed integer. For positive integers n and $k \leq n^{(d-1)/d} \cdot (5d)^{1/d-2}$ there is a set of n points in $\mathbb{R}^d$, so that every geometric spanner of tree-width k on this set has dilation $\Omega\left(n/k^{d/(d-1)}\right)$.*

This proves that our algorithm for Theorem 1 has a worst-case optimal trade off between tree-width and dilation. We therefore now turn our attention to bounded tree-width spanners with an additional property. The most commonly studied property of geometric spanners is planarity.

For point sets in the Euclidean plane, we can adapt the construction of the spanner for Theorem 1 to obtain a planar spanner with bounded tree-width (and bounded degree), that is not necessarily plane. As a general tool to obtain bounded tree-width plane spanners, we provide the following strong dependency between tree-width and the number of edges in sparse connected planar graphs:

▶ **Theorem 3.** *Let G be a planar connected graph with n vertices and $n + \lfloor (k-1)^2/72 \rfloor$ edges, where $k \geq 2$ is an integer. The tree-width of G is at most k.*

This result is quite surprising, since for non-planar graphs, arbitrarily placed edges increase the tree-width linearly in the number of added edges, as there are graphs with m edges and tree-width at least $m \cdot \varepsilon$ for a constant ε [17].

It further yields an alternative proof for Theorem 1 for $d = 2$, creating a bounded tree-width spanner with additional properties:

▶ **Corollary 4.** *Given a set $P \subset \mathbb{R}^2$ of n points and some positive integer $k \leq 12\sqrt{n-3}$, a plane spanner with tree-width k, maximum vertex degree 4 and dilation $\mathcal{O}(n/k^2)$ can be constructed in $\mathcal{O}(n \log n)$ time.*

This follows immediately using an adapted version of the algorithm provided in [3]: Instead of the Delaunay triangulation, use a plane constant dilation spanner with maximum vertex degree 4, as provided in [19] and the MST of this spanner instead of the EMST.

In the last section, we answer an open question stated in [3] by giving some smaller results for spanners of bounded tree-width on points in convex position. In particular, we show that, for point sets P which are equally spaced on a circle, there is a tree spanner with dilation $\leq \frac{2n}{\pi} + \frac{\pi}{2n}$. This complements a lower bound given in [3]. Further, since plane spanners on points in convex position are outerplanar, there is a 1.88-spanner of tree-width 2 for P. Further, we give a (straight-forward) lower bound: There are sets of points in convex position on which every spanner of tree-width 2 has dilation at least 1.43.

2 Preliminaries

We start giving some preliminary definitions to clarify the notations we use later on.

In this paper, all graphs G are simple and undirected. We refer to the vertex set of G by $V(G)$ and to the edge set of G by $E(G)$. Euclidean graphs are complete and weighted and the edge weight is given by the Euclidean distance of the incident vertices. We denote the *degree* of a vertex $v \in V(G)$ by $\Delta_G(v)$ and the *maximum vertex degree* of a graph by $\Delta(G)$. By K_n we denote the complete graph on n vertices.

A *Euclidean 2-dimensional $n \times n$-grid* is defined as a graph on the point set $\{0, \dots, n-1\} \times \{0, \dots, n-1\}$ in $\mathbb{R}^2$. We call two vertices (x, y) and (x', y') *neighbouring*, if $|x-x'|+|y-y'| = 1$. The edge set of the $n \times n$ grid is then $\{\{(x, y), (x', y')\} \mid (x, y)$ and (x', y') are neighbouring$\}$.

A *Euclidean d-dimensional n^d-grid* is defined as a graph on the point set $\{0, \dots, n-1\}^d$. Here, two vertices $(x_1, \dots, x_d)$ and $(x'_1, \dots, x'_d)$ are called *neighbouring* if $|x_1 - x'_1| + \dots + |x_d - x'_d| = 1$. The edge set of the n^d grid is then $\{\{(x_1, \dots x_d), (x'_1, \dots, x'_d)\} \mid (x_1, \dots x_d)$ and $(x'_1, \dots x'_d)$ are neighbouring$\}$.

We use a standard definition of tree-width [26] in the following notation: A *tree decomposition* (T, X) is a tree T over a set of nodes I and a mapping $X : I \to 2^V$, where $X(i)$ is called the *bag* of i, so that the following conditions hold:

1. $\bigcup_{i \in I} X(i) = V$
2. $\bigcup_{i \in I} E(i) = E$ where $E(i) = \{\{u, v\} \in E \mid u, v \in X(i)\}$
3. If $v \in X(i)$ and $v \in X(k)$ for some $i, k \in I$ then $v \in X(j)$ holds for every j on the path in T between i and k.

The *width* of a tree decomposition is defined as $\max_{i \in I} |X(i)| - 1$. The *tree-width* $\mathrm{tw}(G)$ of G is the minimum width over all tree decompositions of G.

A minor of a graph can be obtained by subgraph operations and edge contractions. As, however, edge contractions in general do not maintain any properties of the embedding, which is essential for geometric graphs, we give the definition of contracting one vertex to another: For a graph G, *contracting vertex u* to v or equivalently *contracting vertex u* along e, we create a minor $G' = (V', E')$ of G with $V' = V \setminus \{u\}$ and $E' = \{\{u', v'\} \mid u', v' \in V', \{u', v'\} \in E\} \cup \{\{u', v\} \mid u' \neq v, \{u', u\} \in E\}$. Note that this is just another way of defining an edge contraction. Given a graph $G = (V, E)$, some vertex $v \in V$ and a path

$\gamma = v, v_2, ..., v_\ell$ inside G, we define the *path contraction* of γ to v to be the series of vertex contractions starting with the contraction of v_2 to v up to the contraction of v_ℓ to v. A graph H is a *minor* of G if it can be obtained by vertex contractions and subgraph operations. A class of graphs $\mathcal{G}$ is considered *minor closed* if every minor H of a graph $G \in \mathcal{G}$ is also in $\mathcal{G}$.

A *t-balanced separator* of size s of a graph G is an edge set $S \subseteq V(G)$, $|S| = s$ such that $V - S$ can be partitioned into two sets A and B with no edges between the vertices of A and B and $|A| \leq t \cdot |V|$ as well as $|B| \leq t \cdot |V|$. If $t = \frac{2}{3}$, we call S a *separator* of G.

3 Bounded tree-width spanners in higher dimensions

In this section, we present an algorithm for constructing bounded tree-width spanners for points in $\mathbb{R}^d$ for constant dimension d. As we will see later on, the trade-off between tree-width and dilation of this spanner is worst-case optimal.

The algorithm makes use of two ingredients: a Euclidean minimum spanning tree ($\mathcal{EMST}$) and a class of spanners with sublinear tree-width. For $d = 2$ a class of planar spanners can be used (and will be used in Section 5). For general d we can use the greedy spanner [2] instead.

▶ **Lemma*** **5.** *Let $P \subset \mathbb{R}^d$ be a set of n points and G the greedy 3/2-spanner of P. G has tree-width at most $15\eta_d c_d 8^d \cdot n^{1-1/d}$, where η_d is the packing constant in d-dimensional Euclidean space and $c_d \leq 2^{\mathcal{O}(d)}$ is a constant.*

The following algorithm computes a bounded tree-width spanner. The constant C is defined as $C = 30\eta_d c_d 8^d$.

Algorithm 1 d-`DimensionalBoundedTreeWidthSpanner`(P, k).

Require: a set of n points $P \subset \mathbb{R}^d$ and a natural number $k \leq n^{1-1/d}$
Ensure: an $\mathcal{O}(n \cdot k^{d/(1-d)})$-spanner $G = (P, E)$ of tree-width k

1: $\mathcal{EMST} \leftarrow$ Euclidean minimum spanning tree of P
2: **if** $k = 1$ **then**
3: **return** $\mathcal{EMST}$
4: $m \leftarrow \left\lceil (k/C)^{d/(d-1)} + 1 \right\rceil$
5: Compute a set $\mathcal{T}$ of m disjoint subtrees of $\mathcal{EMST}$, each containing $\mathcal{O}(n/m)$ points, as explained in the proof of Lemma 9.
6: For $T \in \mathcal{T}$ let $R(T)$ be the vertices in T incident to edges removed by the previous step.
7: For each $T \in \mathcal{T}$ iterate through its edges e from long to short. If e lies on a path between two vertices in $R(T)$, remove the edge. Let $E'(T)$ be the set of remaining edges.
8: $E' \leftarrow \bigcup_{T \in \mathcal{T}} E'(T)$
9: $(R, E'') \leftarrow$ greedy 3/2-spanner for $R = \bigcup_{T \in \mathcal{T}} R(T)$
10: **return** $G = (P, E' \cup E'')$

We now start with a detailed description on how the subtrees (line 5) are constructed. The following is a folklore result and is for instance mentioned in [22].

▶ **Fact*** **6.** *For any tree T on n vertices there is a separator vertex that splits T into subtrees each at most of size $n/2$ and can be found in time $\mathcal{O}(n)$.*

* The proofs of the results marked with $*$ can only be found in the full version due to space restrictions.

Considering the separator vertex v and its edges, we observe that one of the by removal of v induced subtrees must have size at least $n \cdot 1/\Delta(v)$. Let e be the edge that connects v to this subtree. Then, e separates the tree into one subtree of size at least $n \cdot 1/\Delta(v)$, this being the aforementioned subtree, and one of size at most $n \cdot (\Delta(v) - 1)/\Delta(v)$.

▶ **Lemma 7.** *For any tree T of degree Δ with n vertices there is a separator edge, that separates T into two subtrees, one of size at least $n \cdot 1/(\Delta + 1)$ and one of size at most $n \cdot \Delta/(\Delta + 1)$. This edge can be found in time $\mathcal{O}(n)$.*

This result can be extended as follows:

▶ **Lemma 8.** *Given a tree T of degree Δ, that has n vertices and some integer $m \geq 1$, there is a set of $m - 1$ edges of T, whose removal lead to m disjoint subtrees of size $\mathcal{O}(n/m)$. This set of edges can be found in time $\mathcal{O}(n(\Delta + 1)(\log m + \log(\Delta + 1)))$.*

Proof sketch. By Lemma 7 for any tree of size n there is always an edge, whose removal splits the tree into one subtree of size at most $n \cdot \Delta/(\Delta+1)$ and one of size at least $n \cdot 1/(\Delta+1)$. The idea now is to split subtrees until their sizes are sufficiently small. So given a tree on n vertices and some $m \in \mathbb{N}$, we keep splitting the largest subtrees until there is no subtree of size more than $(\Delta + 1) \cdot n/m$ left. Note that this procedure can only lead to subtrees as small as n/m and therefore only to at most m subtrees. To obtain exactly m subtrees we can simply continue splitting the largest remaining subtrees until the number of subtrees is exactly m. The subtrees are now at least of size $n/((\Delta + 1) \cdot m)$ and at most of size $(\Delta + 1) \cdot n/m$ and therefore of size $\mathcal{O}(n/m)$. For the running time, we refer to the full version of the paper. ◀

Since the $\mathcal{EMST}$ is of constant degree, we obtain the following corollary:

▶ **Lemma 9.** *Given the $\mathcal{EMST}$ of n points in $\mathbb{R}^d$ and some integer $m \geq 1$, there is a set of $m - 1$ edges of T, whose removal lead to m disjoint subtrees of size $\mathcal{O}(n/m)$. This set of edges can be found in time $\mathcal{O}(n \log m)$.*

We call a vertex a *representative* of a subtree $T \in \mathcal{T}$ if it was incident to an edge of the $\mathcal{EMST}$ that was removed in the construction of Lemma 8. While bounding the dilation of the spanner requires more work, we can already derive the running time and tree-width.

▶ **Lemma 10.** *Algorithm 1 constructs in $\mathcal{O}(n^2 \log n)$ time a graph with tree-width $\leq k$ and bounded degree.*

Proof. The greedy spanners in line 9 of the algorithm can be computed in $\mathcal{O}(n^2 \log n)$ time [6]. The $\mathcal{EMST}$ can be computed in subquadratic time [1], and all remaining steps take $\mathcal{O}(n \log n)$ time. The greedy t-spanner and the $\mathcal{EMST}$ both have bounded degree for fixed dimension (and t) [27, 23].

The set of representatives of all subtrees has size at most $2m - 2$, since each representative is incident to at least one of the $m - 1$ edges removed. Thus, $\mathcal{GS}$ is the greedy spanner of at most $2m - 2$ points. By Lemma 5 and the choice of the constant C the tree-width of $\mathcal{GS}$ is at most k. After line 7 of the algorithm each of the subtrees of the subtrees $T \in \mathcal{T}$ contains only one representative. Thus, the graph G consists of $\mathcal{GS}$ with a tree attached to each of its vertices. These trees do not increase the tree-width, since every tree contains only one vertex from $\mathcal{GS}$. Thus the tree-width of G is k. ◀

Next we bound the dilation of the computed spanner.

▶ **Lemma*** **11.** *Given a set $P \subset \mathbb{R}^d$ of n points for some fixed d and a positive integer $k \leq n^{1-1/d}$, Algorithm 1 computes an $\mathcal{O}(n/k^{d/(d-1)})$-spanner for P.*

Proof idea. First ignore that we removed edges in line 7. For two points p, q in the same subtree the dilation is small, since the edges of the path between p and q in the $\mathcal{EMST}$ are short, and since the path contains only few edges. For points in different subtrees the path in the $\mathcal{EMST}$ contains a representative of each of the two subtrees. We can take the paths to these representatives and between the representatives the greedy spanner.

Removing edges in line 7 complicates the argument, but we can use the lengths of the edges removed to bound the distances to the representatives. ◀

From the previous lemmas we now directly obtain the main result of this section.

▶ **Theorem 1.** *Given a set $P \subset \mathbb{R}^d$ of n points for some fixed d and a positive integer $k \leq n^{1-1/d}$. There is a geometric spanner of tree-width k on P with a dilation of $\mathcal{O}\left(n/k^{d/(d-1)}\right)$ and bounded degree that can be computed in time $\mathcal{O}(n^2 \log n)$.*

4 Lower bound

In the previous section it was shown that Algorithm 1 gives an upper bound for the dilation of tree-width bounded spanners. In this section, we investigate corresponding lower bounds. We show that the bound given in Theorem 1 for the dilation of a tree-width bounded spanner on Euclidean point sets is asymptotically tight:

▶ **Theorem 2.** *Let $d \geq 2$ be a fixed integer. For positive integers n and $k \leq n^{(d-1)/d} \cdot (5d)^{1/d-2}$ there is a set of n points in $\mathbb{R}^d$, so that every geometric spanner of tree-width k on this set has dilation $\Omega\left(n/k^{d/(d-1)}\right)$.*

To obtain this result, we construct a set of points resembling a grid. While it is commonly known that a two-dimensional grid has high tree-width, the $\left(k^{1/(d-1)}\right)^d$-grid does not necessarily have tree-width k. We thus construct a set of points resembling the $(h+1)^d$-grid, where $h = \left\lceil (9d/2 \cdot (k+2))^{1/(d-1)} - 1 \right\rceil$. To lower bound the tree-width of this grid, we generalise a basic idea for the 3-dimensional grid given by Korhonen in an online forum[1]:

▶ **Lemma*** **12.** *The n^d-grid has tree-width $\geq \frac{2}{9d} \cdot n^{d-1} - 1$.*

Thus, the tree-width of our grid is at least

$$\begin{aligned}
\frac{2}{9d} \cdot (h+1)^{d-1} - 1 &= \frac{2}{9d} \cdot \left(\left\lceil (9d/2 \cdot (k+2))^{1/(d-1)} - 1 \right\rceil + 1 \right)^{d-1} - 1 \\
&\geq \frac{2}{9d} \cdot \left((9d/2 \cdot (k+2))^{1/(d-1)} \right)^{d-1} - 1 \\
&= k+1.
\end{aligned}$$

Using an inductive construction for the $(h+1)^d$-grid, by taking d copies of the $(h+1)^{d-1}$-grid and connecting them using $(d-1) \cdot (h+1)^{d-1}$ edges, we can prove that the $(h+1)^d$-grid has a total of $d \cdot (h+1)^d + d \cdot (h+1)^{d-1}$ edges.

[1] https://cstheory.stackexchange.com/questions/53029/what-is-the-treewidth-of-the-3d-grid-mesh-or-lattice-with-sidelength-n

The constructing of the grid-like set $P_{d,n,k}$ is as follows: Let $m = \lfloor n/(d \cdot (h+1)^d + d \cdot (h+1)^{d-1}) \rfloor$, be the number of points representing an edge of the grid in $P_{d,n,k}$. For every dimension $i \in \{1, \dots, d\}$, we define the set

$$P_i = \bigcup_{j_1=0}^{h} \cdots \bigcup_{j_d=0}^{h} \{(j_1 \cdot m, \dots, j_{i-1} \cdot m, x_i, j_{i+1} \cdot m, \dots, j_d \cdot m) \mid x_i \in \{0, \dots, h \cdot m\}\},$$

which consists of $(h+1)^{d-1}$ sets of collinear points representing the grid edges parallel to the axis of the i-th dimension. Each of these collinear sets of points consists of $h \cdot m + 1$ points and represents h edges of the grid. Hence $|P_i| = (h \cdot m + 1) \cdot (h+1)^{d-1}$ for every i. We can now define the set $P_{d,n,k} = \bigcup_{i=1}^{d} P_i$ which represents the whole $(h+1)^d$-grid. (In Figure 1 an example is shown for a 2-dimensional point set.) Notice that the $(h+1)^d$ points $P_{\boxplus} = \{(j_1 \cdot m, \dots, j_d \cdot m) \mid j_1, \dots, j_d \in \{0, \dots, h\}\}$ are contained in every P_i. We call these points the *grid points* of the set $P_{d,n,k}$ and two grid points are called *neighbouring* if the Euclidean distance between them is m, which corresponds to them being connected through an edge in the $(h+1)^d$-grid.

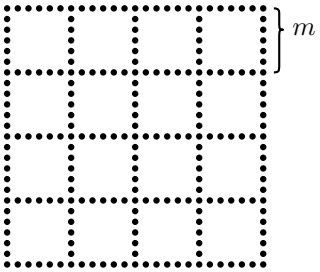


Figure 1 The set of points resembling the $(4+1)^2$-grid where $n = 240$ and $m = 6$.

▶ **Lemma*** **13.** *Let $d \geq 2$ be a fixed integer. The grid-like set $P_{d,n,k}$ is well-defined for every integer $n \geq d \cdot 64^{d \log d}$ and positive integer $k \leq n^{(d-1)/d} \cdot (5d)^{1/d-2}$ and is of size at most n.*

▶ **Lemma 14.** *If G is a geometric $o\left(n/k^{d/(d-1)}\right)$-spanner on $P_{d,n,k}$, then it must be of tree-width at least $k+1$.*

Proof. Let G be a $o(n/k^{d/(d-1)})$-spanner on $P_{d,n,k}$. Let $\mathcal{R}_x := x + [-m/4, m/4]^d$ denote the hypercube of side-length $m/2$ centered at x. For any pair $p, q \in P_{\boxplus}$ of neighbouring grid points, we consider the three hypercubes $\mathcal{R}_p, \mathcal{R}_q$, and $\mathcal{R}_s := s + [-m/4, m/4]^d$, where $s = \frac{p+q}{2}$ is the midpoint between p and q. We argue that there is a path in G from p to q such that: *(i)* the path only visits vertices in $\mathcal{R}_{p,q} := \mathcal{R}_p \cup \mathcal{R}_q \cup \mathcal{R}_s$ and *(ii)* for any $p' \in \mathcal{R}_p$ and $q' \in \mathcal{R}_q$, if the path visits p' and q' then it visits p' before it visits q'.

Let $p_0 = p, p_1, p_2, \dots p_m = q$ the ordered sequence of points representing the edge of the grid between p and q, and further let $s_i := \frac{p_i + p_{i+1}}{2}$. Since G is a $o(n/k^{d/(d-1)})$-spanner, we can assume that there is for every pair p_i, p_{i+1} a shortest path γ_i in G of length at most $m/4 = \Theta(n/k^{d/(d-1)})$. Such a path necessarily will remain with in $\mathcal{R}_{p_i} \cap \mathcal{R}_{p_{i+1}} \subset \mathcal{R}_{s_i}$. For $0 \leq i < m/2$ we have $\mathcal{R}_{s_i} \subset \mathcal{R}_p \cup \mathcal{R}_s$, and for $m/2 \leq i < m$ we have $\mathcal{R}_{s_i} \subset \mathcal{R}_q \cup \mathcal{R}_s$.

Consider the walk from p to q obtained by concatenating the paths $\gamma_0, \gamma_1, \dots \gamma_{m-1}$. It first only visits vertices in $\mathcal{R}_p \cup \mathcal{R}_s$ and then only vertices in $\mathcal{R}_q \cup \mathcal{R}_s$. By removing cycles we obtain a path $\gamma_{p,q}$ from p to q with this property. Thus, we have a path with the properties *(i)*+*(ii)* stated above.

We have such a path $\gamma_{p,q}$ for any pair $p, q \in P_{\boxplus}$. We claim that this implies that G must have a minor isomorphic to the $(h+1)^d$-grid and is therefore of tree-width $> k$. To show this, we perform a series of path contractions on G, obtaining a graph G'. More specifically, we first contract the edges of $\gamma_{p,q}$ starting from p, until we encounter as endpoint the last vertex in $\mathcal{R}_p$, identifying the merged vertices with p. After skipping an edge, we contract the remaining edges of $\gamma_{p,q}$, identifying the merged vertices with q. This is illustrated in Figure 2. After these contractions, the remaining edge of $\gamma_{p,q}$ is an edge from p to q.

Further we observe that *(i)* any vertex p' of $\gamma_{p,q}$ in $\mathcal{R}_p$ has been identified with p, *(ii)* any vertex q' of $\gamma_{p,q}$ in $\mathcal{R}_q$ has been identified with q, and *(iii)* any other vertex p'' of $\gamma_{p,q}$ does not also lie on another path $\gamma_{r,s}$ with $\{p,q\} \neq \{r,s\}$. The latter holds, since p'' cannot lie in $\mathcal{R}_{r,s}$. This means, that the paths $\gamma_{p,q}$ and $\gamma_{r,s}$ are contracted consistently. Thus, regardless of the order in which we contract the paths, we obtain a graph G' which has a minor isomorphic to the $(h+1)^d$-grid. In G', there is an edge between every neighbouring pair of grid points. Therefore the subgraph $G'[P_{\boxplus}]$ containing only the grid points is, if we remove edges between non-neighbouring grid points, isomorphic to the $(h+1)^d$-grid. We conclude that the spanner G has a minor isomorphic to the $(h+1)^d$-grid and by Lemma 12 therefore must be of tree-width at least $\frac{2}{9d}(h+1)^{d-1} - 1 \geq k+1$. ◀

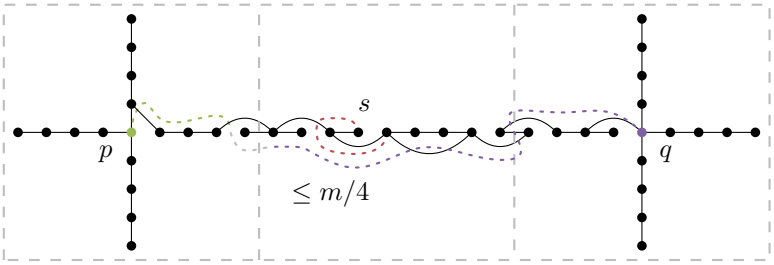


Figure 2 Grid points p and q as well as the three hypercubes centered at p, q and s. The subpath of $\gamma_{p,q}$ that is contracted to p is shown in green and the subpath that is contracted to q in purple. The red path is a detour of length $\leq m/4$ for points p_i and p_{i+1}.

5 Minor-3-cores and the tree-width of planar spanners

We now have given an asymptotically tight algorithm to obtain small bounded tree-width, with a dilation dependent on the chosen tree-width. We now focus on bounded tree-width spanners with the additional property of being plane.

In general, there are graphs with m edges and tree-width at least $m \cdot \varepsilon$ for a constant ε [17]. We show that this is not true for connected planar graphs, yielding a tight dependency between tree-width and the number of edges:

▶ **Theorem 3.** *Let G be a planar connected graph with n vertices and $n + \lfloor (k-1)^2/72 \rfloor$ edges, where $k \geq 2$ is an integer. The tree-width of G is at most k.*

5.1 Minor-3-cores

To obtain this Theorem, we first introduce the concept of *minor-3-cores*, in reference to the k-core which was established by Seidman [29] for the study of social networks. We then use this concept to obtain the tree-width of planar graphs with a certain fixed number of edges.

▶ **Definition 15** (Minor-3-Core). *Let G be a graph and H a minor of G with minimum degree 3. If there is no minor of minimum degree 3 that has more edges than H i.e., H is an edge maximal minor with this property, we call it a minor-3-core of G.*

This definition gives a very clear idea of what a minor-3-core should be, but unfortunately it does not lead to any structural insight concerning the concept. It especially leaves open whether minor-3-cores are unique or if there are multiple non-isomorphic minor-3-cores. We therefore need to further characterise minor-3-cores with regard to algorithmic properties and uniqueness. In order to answer these questions, we first define the disjoint-paths property.

▶ **Definition 16** (Disjoint-Paths Property). *Let G be a graph and let C be a minor-3-core of G as well as v be a vertex of the minor-3-core (and therefore a vertex of G that has not been contracted). We say that v fulfils the disjoint-paths property for C, if for every neighbour $u_1, ..., u_{\Delta_C(v)}$ of v in C, there is a path γ_i in G which leads from v to u_i and $\left(\bigcup_{u \text{ in } \gamma_i} u\right) \cap \left(\bigcup_{w \text{ in } \gamma_j} w\right) = \{v\}$ for every path γ_j for $j \neq i$.*

We now show that we can assume having the disjoint paths property in a minor-3-core:

▶ **Lemma*** **17.** *For every graph G and minor-3-core C of G, there is a minor-3-core isomorphic to C in which every vertex has the disjoint paths property.*

By Lemma 17 we now know that there is a minor-3-core, in which every vertex has the disjoint-paths property. We call a minor with this property a *canonical minor-3-core*:

▶ **Definition 18** (Canonical Minor-3-Core). *Let C be a minor-3-core of a graph G. We call C the canonical minor-3-core, if every vertex of C has the disjoint-paths property. The vertices of C are called the canonical vertices.*

We now show that the canonical minor-3-core is, as the name implies, unique:

▶ **Lemma*** **19.** *Let G be a graph, the canonical minor-3-core of G is unique.*

Combining Lemma 17 and Lemma 19, we can state that every minor-3-core C of G is isomorphic to the canonical minor-3-core of G. Thus we obtain uniqueness and can refer to *the minor-3-core*, instead of a minor-3-core, meaning the canonical minor-3-core.

We will now give an algorithmic characterisation for the minor-3-core. Similarly to k-cores [29] we can compute the minor-3-core by an iterative pruning process:

▶ **Theorem 20.** *Let G be a graph. The minor-3-core of G can be obtained by performing the following operations to exhaustion:*

- *Let e be an edge incident to a vertex v with degree ≤ 2. Contract v along e.*
- *If v is an isolated vertex, delete v.*

Proof. The graph H resulting from the iterative pruning process described above is clearly a minor of G, in which every vertex has degree at least 3. Let C be the canonical minor-3-core of G. Suppose that there is some edge in C that is not in H. For an edge to not be an edge of H one of the vertices of this edge needs to be contracted to one of its neighbours in the construction of H. Let u be the first vertex of C that is contracted in the construction of H. By definition of the pruning process u needs to be of degree ≤ 2 to be contracted. Since u by the definition of the disjoint-paths property starts off as a vertex of degree ≥ 3, there needs to be a contraction that reduces the degree of u to ≤ 2. For this to be possible, there needs to be a contraction of two neighbouring vertices of u, where both of these vertices are canonical vertices. This must be the case, since in G there are at least three disjoint

paths starting in u that lead to at least three different canonical vertices and there cannot be an edge between the vertices of the disjoint paths, since otherwise these vertices would be canonical vertices themselves. The vertices of this edge would need to be canonical as they are both connected to u, to one other canonical vertex each, as well as to each other by disjoint paths. But since we assumed u to be the first canonical vertex that gets contracted in the construction of H, we reach a contradiction. Hence every edge of C is also an edge of H and since C is edge maximal, H must be equal to the canonical minor-3-core C. ◀

In the following we use the minor-3-core in order to bound the tree-width of planar spanners. To begin the exploration of the relationship of minor-3-cores and tree-width we first make a couple of observations.

1. The minor-3-core of a tree is the empty graph.
2. The minor-3-core of a graph with at most one circle is the empty graph.
3. The K_4 is its own minor-3-core.

Even more, every graph without a K_4 minor has an empty minor-3-core.*

Considering a graph and then connecting a tree to some vertex of our graph, like a branch growing out of our graph, does not increase the tree-width of the graph. The same holds for edges that we replace with paths. So we can see that it is possible to increase the size of our graph without affecting the tree-width of the graph. This is a problem if we want to bound the tree-width of our graph through common separator arguments, for example using [24], since the bound on the tree-width there depends on the size of the graph. Hence it would be nice to have a tool that prunes our graph by removing "tree-like" structures from our graph. For that, we use the definition of minor-3-cores:

▶ **Theorem 21.** *Given a graph of tree-width $k \geq 3$, the tree-width of the minor-3-core of G is k as well.*

Proof. Let G be a graph of tree-width $k \geq 3$ and H its minor-3-core. Since H is a minor of G, the tree-width of H can be at most k. To prove that H must be of tree-width at least k, we prove that contracting a vertex of degree at most 2 does not decrease the tree-width. So let G be a graph of tree-width $k \geq 3$. Suppose that there is a vertex v of degree 2 and let u_1 and u_2 be the two vertices adjacent to v. Let G' be the graph resulting from the contraction of v to u_1. We now argue that G' must have tree-width k as well. Assume that G' has tree-width $k' < k$ and let (T', X') be a tree-decomposition of width k' for G'. We now construct a tree-decomposition (T, X) for G of width k' from (T', X'). In (T', X') there must be some bag $X(i)$ which contains both u_1 and u_2, since they are connected by an edge. We now create a new node, whose bag contains v, u_1 and u_2 and connect it to i by an edge. The tree-decomposition (T, X) is now a valid tree-decomposition for G with tree-width $k' < k$, which is a contradiction to G being of tree-width k. If v is of degree 1, the same argument holds, but we only consider u_1, which has to appear in at least one bag of (T', X'). ◀

As mentioned before we want to bound the tree-width of graphs (especially spanners) through the size of their minor-3-core. Therefore it would be useful to have a bound on the size of the minor-3-core of a graph. If a graph is very "tree-like", i.e. the graph can be decomposed into a set of trees by removing only a few edges, we expect it to have a small minor-3-core. We formalise this statement as follows:

▶ **Lemma*** **22.** *Let G be a connected graph with n vertices and $n-1+m$ edges, where $m \geq 1$ is some integer. The minor-3-core of G has at most $2 \cdot (m-1)$ vertices.*

This lemma is the last part we need to prove an upper bound for the tree-width of connected planar graphs. Intuitively the statement of the lemma could just as well be stated as a bound on the size of minor-3-cores of trees with an additional m edges. If we place these additional edges in a planar fashion, we can combine our bound on the size of minor-3-cores and the bound on the tree-width of planar graphs derived from Theorem 6.2 in [24] to obtain a bound on the tree-width of trees with additional edges that were placed without creating edge crossings. Using this, we can finally proof our second main result:

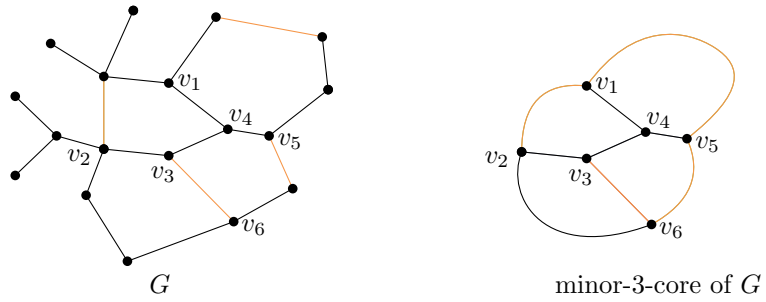


Figure 3 Minor-3-core of a graph G that consists of a tree with 4 additional edges (orange).

Proof of Theorem 3. Since G is a connected graph with $n + \lfloor (k-1)^2/72 \rfloor$ edges, its minor-3-core must by Lemma 22 have at most $\lfloor (k-1)^2/36 \rfloor$ vertices. Also, since the minor-3-core is a minor of G and G is planar, the minor-3-core must be planar as well. The tree-width of a planar graph with n vertices by [24] is at most $6\sqrt{n}+1$. The tree-width of the minor-3-core of G therefore is

$$6\sqrt{\lfloor (k-1)^2/36 \rfloor} + 1 \leq 6\sqrt{(k-1)^2/36} + 1 \leq k.$$

By Lemma 21 we can now conclude that the tree-width of G is either ≤ 2 or exactly k, so in both cases at most k. ◀

5.2 Tree-width of planar spanners

Theorem 3 allows us a stronger result than Theorem 1 in the Euclidean plane: Adjusting the algorithm given in [3], we obtain a plane bounded-degree spanner with bounded tree-width:

▶ **Corollary 4.** *Given a set $P \subset \mathbb{R}^2$ of n points and some positive integer $k \leq 12\sqrt{n-3}$, a plane spanner with tree-width k, maximum vertex degree 4 and dilation $\mathcal{O}(n/k^2)$ can be constructed in $\mathcal{O}(n \log n)$ time.*

Our main adjustment to the algorithm is to chose a plane constant-dilation spanner $\mathcal{S}$ of P with maximum degree 4, as provided in [19], instead of the Delaunay triangulation and to utilise a minimum spanning tree of $\mathcal{S}$ instead of the Euclidean MST. The full version of the algorithm can be found in the full version of the paper. Since the constructed spanner then is a subgraph of $\mathcal{S}$, it is plane and has maximum vertex degree 4. By Theorem 3, its tree-width can then be bounded by the given k. Further, the dilation is not affected, which follows directly by Lemma 4 in [3].

6 Bounded tree-width spanners on convex point sets

We have seen that for unrestricted sets of points in the plane the Delaunay triangulation is an 1.998-spanner of tree-width $6\sqrt{n}+1$. To obtain spanners with smaller tree-widths, we now look at sets of points in the plane that are convex or even lie on a circle.

6.1 Tree spanners

The Euclidean minimum spanning tree is an $(n-1)$-spanner for any set of n points in $\mathbb{R}^d$ of tree-width 1. Aronov et. al. [3] proved that there are sets of points, on which the dilation bound of the minimum spanning tree is almost optimal. To be more precise they showed that there is a set of points, on which any tree-spanner has dilation at least $\frac{2n}{\pi} - 1$. The set they used in their proof was a set of $n \geq 3$ points equally spaced on the unit circle. So, even for tree-spanners on convex sets and sets of points on a circle a dilation of $\Omega(n)$ may be unavoidable. The most obvious tree-spanner on a set of points equally spaced on the unit circle would probably be the spanner that consists of all the edges between points next to each other except for one edge. It is not hard to see that this spanner has dilation $n-1$. But we can show that for points equally spaced on the unit circle, there are tree-spanners with dilation almost $\frac{2n}{\pi} - 1$.

▶ **Lemma*** **23.** *Let P be a set of n points equally spaced on a circle. There is a tree-spanner on P with dilation $\leq \frac{2n}{\pi} + \frac{\pi}{2n}$.*

Proof Idea. Given some $n \geq 2$, let $p_i = \left(\sin\left(\frac{2\pi i}{n}\right), \cos\left(\frac{2\pi i}{n}\right)\right)$. We define $P = \{p_i \mid i \in \{0, \ldots, n-1\}\}$ which is the set of n points equally spaced on the unit circle, where one point is $(0, 1)$. Notice that for any i we have $p_i = p_{i+n}$.

We now construct a tree-spanner G for P with dilation $\leq \frac{2n}{\pi} + \frac{\pi}{2n}$. The spanner G is a path visiting each point on the unit circle in a sawtooth pattern. We define G as the path $p_0, p_{n-1}, p_1, \ldots, p_{\lfloor n/2 \rfloor}$ or more formally as the graph with edges

$$E = \{\{p_i, p_{n-1-i}\} \mid i \in \{0, \ldots, \lfloor n/2 \rfloor - 1\}\} \cup \{\{p_{i+1}, p_{n-i}\} \mid i \in \{0, \ldots, \lfloor (n-1)/2 \rfloor - 1\}\}.$$

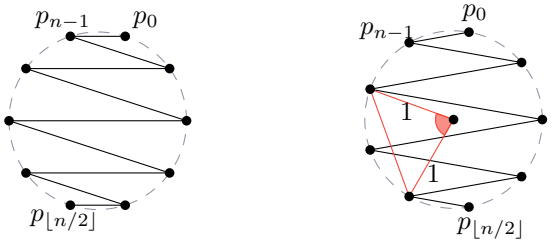


Figure 4 The spanner G on n points equally spaced on the unit circle for $n = 10$ (left) and for $n = 9$ (right).

The Euclidean distance between points p_i and p_j can be derived from the law of cosines:

$$|p_i p_j| = \sqrt{2 - 2\cos\left(\frac{2\pi}{n}(j-i)\right)} = \left|2\sin\left((j-i)\cdot\frac{\pi}{n}\right)\right|.$$

For the distance in G between two points we can add up the Euclidean distances between the points on the path connecting them. Since the spanner G is a path from p_0 to $p_{\lfloor n/2 \rfloor}$, we now let $q_1, ..., q_n$ be the vertices along the path G, where $q_1 = p_0$ and $q_n = p_{\lfloor n/2 \rfloor}$. The distance in G from some q_i to q_j for $i \leq j$ then is

$$d(q_i, q_j) = \sum_{k=i}^{j-1} 2\sin\left(\frac{\pi}{n}\cdot k\right).$$

To complete the proof, we still need to calculate $|q_iq_j|$ and then bound $\frac{d(q_i,q_j)}{|q_iq_j|}$. We can show that

$$|q_iq_j| = \left|2\sin\left(\left\lfloor\frac{j-i+1}{2}\right\rfloor\cdot\frac{\pi}{n}\right)\right|.$$

We then use trigonometric identities to bound $d(q_i,q_j)$ and $|q_iq_j|$ by simpler expressions, from which the lemma then follows. ◀

6.2 Spanners of tree-width 2

While for geometric spanners of tree-width 1, there are sets of points in convex position on which a dilation of $\Omega(n)$ is unavoidable, we can obtain some results for geometric spanners of tree-width 2 on sets of points in convex position.

As every plane graph on a set of points in convex position is outerplanar, and outerplanar graphs have tree-width ≤ 2 [15], every plane spanner on a set of points in convex position has tree-width ≤ 2. Let P be a point set with points in convex position. Since by [4], there is a plane 1.88-spanner for P, there is a 1.88-spanner of tree-width 2 for P. If the points are equally spaced on a circle, by [28] there is even a 1.4482-spanner with tree-width 2.

To show a lower bound, we build on the idea of [16]:

▶ **Lemma*** **24.** *Let P be the set of $n \geq 3$ points equally spaced on the unit circle. Every spanner on P with dilation < 2 must contain the convex hull of P.*

▶ **Lemma 25.** *Let P be the set of $n \geq 3$ points equally spaced on the unit circle. Every t-spanner for $t < 2$ of tree-width 2 on P is plane.*

Proof. Let G be a t-spanner for $t < 2$ of tree-width 2 on P. By Lemma 24, G contains the convex hull of P. For $n = 3$ the statement holds, since the K_3 is plane and has tree-width 2. So let $n \geq 4$ and suppose in the straight-line drawing of G there is an edge-crossing. Let these edges be $\{p,q\}$ and $\{p,q'\}$. On the convex hull of P there must be a path from p to p', not containing q and q', a path from p' to q, not containing p and q', a path from q to q' not containing p and p', and finally a path from q' to p, not containing p' and q. If we perform path-contractions along these paths, leaving only p, q, p' and p', the resulting graph is the K_4, which is of tree-width 3. We reach a contradiction and can conclude that G must be plane. ◀

The desired lower bound for spanners of tree-width 2 now follows from [16] combined with Lemma 25.

▶ **Proposition 26.** *Let P be a set of 23 points equally spaced on a circle. Every spanner of tree-width 2 on P has dilation at least 1.430814.*

7 Conclusion and open questions

In this paper, we give the first algorithm for computing geometric spanners with bounded tree-width for points in Euclidean space of constant dimension. Further, we show a strong dependency between the tree-width and the number of edges in connected planar graphs with few edges.

Our lower bound proves that the dilation of our spanner is worst-case optimal for tree-width k. This naturally leads to the question of optimisation: Given a set of points and a parameter k, can we compute the minimum-dilation spanner of tree-width k? Already for

$k = 1$, that is for minimum-dilation trees, the problem is NP-hard [20, 11]. This raises the question of approximation: Can we efficiently approximate the minimum-dilation spanner of tree-width k in terms of its dilation within a factor better than $\mathcal{O}(n/k^{d/d-1})$? As a bicriteria problem, the tree-width of the approximation may also be $f(k)$ for a suitable function f.

References

1 Pankaj K. Agarwal, Herbert Edelsbrunner, and Otfried Schwarzkopf. Euclidean minimum spanning trees and bichromatic closest pairs. *Discret. Comput. Geom.*, 6:407–422, 1991. doi:10.1007/BF02574698.

2 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.

3 Boris Aronov, Mark de Berg, Otfried Cheong, Joachim Gudmundsson, Herman Haverkort, Michiel Smid, and Antoine Vigneron. Sparse geometric graphs with small dilation. *Computational Geometry*, 40(3):207–219, 2008. doi:10.1016/J.COMGEO.2007.07.004.

4 Ahmad Biniaz, Mahdi Amani, Anil Maheshwari, Michiel Smid, Prosenjit Bose, and Jean-Lou De Carufel. A plane 1.88-spanner for points in convex position. *Journal of Computational Geometry*, page Vol. 7 No. 1, 2016. doi:10.20382/JOCG.V7I1A21.

5 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. Plane spanners of maximum degree six. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I 37*, pages 19–30. Springer, 2010. doi:10.1007/978-3-642-14165-2_3.

6 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.

7 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

8 Sergio Cabello. Computing the inverse geodesic length in planar graphs and graphs of bounded treewidth. *ACM Trans. Algorithms*, 18(2), March 2022. doi:10.1145/3501303.

9 Sergio Cabello and Christian Knauer. Algorithms for graphs of bounded treewidth via orthogonal range searching. *Computational Geometry*, 42(9):815–824, 2009. doi:10.1016/j.comgeo.2009.02.001.

10 Sergio Cabello and Günter Rote. Obnoxious centers in graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1713–1730, 2010. doi:10.1137/09077638X.

11 Otfried Cheong, Herman Haverkort, and Mira Lee. Computing a minimum-dilation spanning tree is NP-hard. *Computational Geometry*, 41(3):188–205, 2008. doi:10.1016/j.comgeo.2007.12.001.

12 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd Symposium on Computational Geometry*, pages 169–177, 1986. doi:10.1145/10515.10534.

13 B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77–114, 2000. doi:10.1016/s0166-218x(99)00184-5.

14 Gautam Das and Paul J Heffernan. Constructing degree-3 spanners with other sparseness properties. *International Journal of Foundations of Computer Science*, 7(02):121–135, 1996. doi:10.1007/3-540-57568-5_230.

15 Michael John Dinneen. *Bounded Combinatorial Width and Forbidden Substructures.* PhD thesis, University of Victoria, 1995.

16 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.

17 Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal of Combinatorial Theory, Series B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.

18 Joachim Gudmundsson and Christian Knauer. Dilation and detours in geometric networks. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 53–69. Chapman and Hall/CRC, 2018.

19 Iyad Kanj, Ljubomir Perkovic, and Duru Turkoglu. Degree four plane spanners: Simpler and better. *J. Comput. Geom.*, 8(2):3–31, 2017. doi:10.20382/jocg.v8i2a2.

20 Rolf Klein and Martin Kutz. Computing geometric minimum-dilation graphs is NP-hard. In *Graph Drawing: 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers 14*, pages 196–207. Springer, 2007. doi:10.1007/978-3-540-70904-6_20.

21 Hung Le and Cuong Than. Greedy spanners in euclidean spaces admit sublinear separators. *ACM Transactions on Algorithms*, 20(3):1–30, 2024. doi:10.1145/3590771.

22 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

23 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007. doi:10.1017/cbo9780511546884.

24 N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994. doi:10.1006/JCTB.1994.1073.

25 N. Robertson and P.D. Seymour. Graph minors I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35:39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

26 N. Robertson and P.D. Seymour. Graph minors II. Algorithmic aspects of tree width. *Journal of Algorithms*, 7:309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

27 Gabriel Robins and Jeffrey S. Salowe. Low-degree minimum spanning trees. *Discret. Comput. Geom.*, 14(2):151–165, 1995. doi:10.1007/BF02570700.

28 Sattar Sattari and Mohammad Izadi. An improved upper bound on dilation of regular polygons. *Computational Geometry*, 80:53–68, 2019. doi:10.1016/J.COMGEO.2019.01.009.

29 Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

30 Michiel H.M. Smid. The well-separated pair decomposition and its applications. *Handbook of approximation algorithms and metaheuristics*, 13, 2007. doi:10.1201/9781420010749.ch53.

Computing Oriented Spanners and Their Dilation

Kevin Buchin
TU Dortmund University, Germany

Antonia Kalb
TU Dortmund University, Germany

Anil Maheshwari
Carleton University, Ottawa, Canada

Saeed Odak
University of Ottawa, Canada

Carolin Rehs
TU Dortmund University, Germany

Michiel Smid
Carleton University, Ottawa, Canada

Sampson Wong
BARC, University of Copenhagen, Denmark

Abstract

Given a point set P in a metric space and a real number $t \geq 1$, an *oriented t-spanner* is an oriented graph $\overrightarrow{G} = (P, \overrightarrow{E})$, where for every pair of distinct points p and q in P, the shortest oriented closed walk in $\overrightarrow{G}$ that contains p and q is at most a factor t longer than the perimeter of the smallest triangle in P containing p and q. The *oriented dilation* of a graph $\overrightarrow{G}$ is the minimum t for which $\overrightarrow{G}$ is an oriented t-spanner.

For arbitrary point sets of size n in $\mathbb{R}^d$, where $d \geq 2$ is a constant, the only known oriented spanner construction is an oriented 2-spanner with $\binom{n}{2}$ edges. Moreover, there exists a set P of four points in the plane, for which the oriented dilation is larger than 1.46, for any oriented graph on P.

We present the first algorithm that computes, in Euclidean space, a sparse oriented spanner whose oriented dilation is bounded by a constant. More specifically, for any set of n points in $\mathbb{R}^d$, where d is a constant, we construct an oriented $(2+\varepsilon)$-spanner with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space. Our construction uses the well-separated pair decomposition and an algorithm that computes a $(1+\varepsilon)$-approximation of the minimum-perimeter triangle in P containing two given query points in $\mathcal{O}(\log n)$ time.

While our algorithm is based on first computing a suitable undirected graph and then orienting it, we show that, in general, computing the orientation of an undirected graph that minimises its oriented dilation is NP-hard, even for point sets in the Euclidean plane.

We further prove that even if the oriented graph is already given, computing its oriented dilation is APSP-hard for points in a general metric space. We complement this result with an algorithm that approximates the oriented dilation of a given graph in subcubic time for point sets in $\mathbb{R}^d$, where d is a constant.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases spanner, oriented graph, dilation, orientation, well-separated pair decomposition, minimum-perimeter triangle

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.27

Related Version *Full Version*: https://arxiv.org/abs/2412.08165

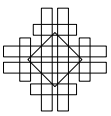
Funding *Anil Maheshwari*, *Michiel Smid*: funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Antonia Kalb: This work was supported by a fellowship of the German Academic Exchange Service (DAAD).

1 Introduction

While geometric spanners have been researched for decades (see [4, 19] for a survey), directed versions have only been considered more recently. This is surprising since, in many applications, edges may be directed or even oriented if two-way connections are not permitted. Oriented spanners were first proposed in ESA'23 [5] and have since been studied in [6] and [7].

© Kevin Buchin, Antonia Kalb, Anil Maheshwari, Saeed Odak, Carolin Rehs, Michiel Smid, and Sampson Wong;
licensed under Creative Commons License CC-BY 4.0



Formally, let P be a point set in a metric space and let $\overrightarrow{G}$ be an oriented graph with vertex set P. For a real number $t \geq 1$, we say that $\overrightarrow{G}$ is an *oriented t-spanner*, if for every pair p, q of distinct points in P, the shortest oriented closed walk in $\overrightarrow{G}$ that contains p and q is at most a factor t longer than the shortest simple cycle that contains p and q in the complete undirected graph on P. Since we consider point sets in a metric space, we observe that the latter is the minimum-perimeter triangle in P containing p and q.

Recall that a *t-spanner* is an undirected graph G with vertex set P, in which for any two points p and q in P, there exists a path between p and q in G whose length is at most t times the distance $|pq|$. Several algorithms are known that compute $(1+\varepsilon)$-spanners with $\mathcal{O}(n)$ edges for any set of n points in $\mathbb{R}^d$. Examples are the greedy spanner [19], Θ-graphs [16] and spanners based on the well-separated pair decomposition (WSPD) [22]. Moreover, it is NP-hard to compute an undirected t-spanner with at most m edges [12].

In the oriented case, while it is NP-hard to compute an oriented t-spanner with at most m edges [5], the problem of constructing sparse oriented spanners has remained open until now. For arbitrary point sets of size n in $\mathbb{R}^d$, where $d \geq 2$ is a constant, the only known oriented spanner construction is a simple greedy algorithm that computes, in $\mathcal{O}(n^3)$ time, an oriented 2-spanner with $\binom{n}{2}$ edges [5]. If P consists of three vertices of an equilateral triangle in $\mathbb{R}^2$ and a fourth point in its centre, then the smallest oriented dilation for P is $2\sqrt{3} - 2 \approx 1.46$ [5]. Thus, prior to our work, no algorithms were known that compute an oriented t-spanner with a subquadratic number of edges for any constant t.

In this paper, we introduce the first algorithm to construct sparse oriented spanners for general point sets. For any set P of n points in $\mathbb{R}^d$, where d is a constant, the algorithm computes an oriented $(2+\varepsilon)$-spanner with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

While our approach uses the WSPD [8, 22], this does not immediately yield an oriented spanner. An undirected spanner can be obtained from a WSPD by adding one edge per well-separated pair. This does not work for constructing an oriented spanner because a path in both directions needs to be considered for every well-separated pair. To construct such paths, we present a data structure that, given a pair of points, returns a point with an approximate minimum summed distance to the pair of the points, i.e. the smallest triangle in the complete undirected graph on P. Our algorithm first constructs an adequate undirected sparse graph and subsequently orients it using a greedy approach.

We also show in this paper that, in general, it is NP-hard to decide whether the edges of an arbitrary graph can be oriented in such a way that the oriented dilation of the resulting graph is below a given threshold. This holds even for Euclidean graphs.

For the problem of computing the oriented dilation of a given graph, there is a straightforward cubic time algorithm. We show a subcubic approximation algorithm.

The hardness of computing the dilation of a given graph, especially in the undirected case, is a long-standing open question, and only cubic time algorithms are known [15]. It is conjectured that computing the dilation is as hard as the all-pairs shortest paths problem (APSP). In this paper, we demonstrate that computing the oriented dilation is APSP-hard.

2 Preliminaries

Let $(P, |\cdot|)$ be a finite metric space, where the distance between any two points p and q in P is denoted by $|pq|$. When the choice between Euclidean and general metric distances is relevant for our results, we use the adjective *Euclidean* or respectively *metric graph*.

If p and q are distinct points in P, then $\Delta^*(p, q)$ denotes the shortest simple cycle containing p and q in the complete undirected graph on P; this is the triangle Δ_{pqx}, where

$$x = \arg\min\{(|px| + |qx|) \mid x \in P \setminus \{p, q\}\}.$$

We use $|\Delta^*(p, q)|$ to denote the perimeter of this triangle.

An *oriented graph* $\overrightarrow{G} = (P, \overrightarrow{E})$ is a directed graph such that for any two distinct points p and q in P, at most one of the two directed edges (p, q) and (q, p) is in $\overrightarrow{E}$. In other words, if (p, q) is in $\overrightarrow{E}$, then (q, p) is not in $\overrightarrow{E}$.

For two distinct points p and q in P, we denote by $C_{\overrightarrow{G}}(p, q)$ the *shortest closed walk* containing p and q in the oriented graph $\overrightarrow{G}$. The length of this walk is denoted by $|C_{\overrightarrow{G}}(p, q)|$; if such a walk does not exist, then $|C_{\overrightarrow{G}}(p, q)| = \infty$.

▶ **Definition 1** (oriented dilation). *Let $(P, |\cdot|)$ be a finite metric space and let $\overrightarrow{G}$ be an oriented graph with vertex set P. For two points $p, q \in P$, their* oriented dilation *is defined as*

$$\mathrm{odil}_{\overrightarrow{G}}(p, q) = \frac{|C_{\overrightarrow{G}}(p, q)|}{|\Delta^*(p, q)|}.$$

The oriented dilation of $\overrightarrow{G}$ is defined as $\mathrm{odil}(\overrightarrow{G}) = \max\{\mathrm{odil}_{\overrightarrow{G}}(p, q) \mid p, q \in P\}$.

An oriented graph with oriented dilation at most t is called an *oriented t-spanner*. Note thatthere is no oriented 1-spanner for sets of at least 4 points.

Let $G = (P, E)$ be an undirected graph. We call $\overrightarrow{G} = (P, \overrightarrow{E})$ an *orientation* of G, if for each undirected edge $\{p, q\} \in E$ it holds either $(p, q) \in \overrightarrow{E}$ or $(q, p) \in \overrightarrow{E}$.

2.1 Well-separated pair decomposition

In 1995, Callahan and Kosaraju [8] introduced the *well-separated pair decomposition* (WSPD) as a tool to solve various distance problems on multidimensional point sets.

For a real number $s > 0$, two point sets A and B form an *s-well-separated pair*, if A and B can each be covered by balls of the same radius, say ρ, such that the distance between the balls is at least $s \cdot \rho$. An *s-well-separated pair decomposition* for a set P of points is a sequence of s-well-separated pairs $\{A_i, B_i\}$, such that $A_i \cap B_i = \emptyset$ and for any two points $p, q \in P$, there is a unique index i such that $p \in A_i$ and $q \in B_i$, or $p \in B_i$ and $q \in A_i$.

The following properties are used repeatedly in this paper.

▶ **WSPD-Property 1** (Smid [22]). *Let $s > 0$ be a real number and let $\{A, B\}$ be an s-well-separated pair. For points $a, a' \in A$ and $b, b' \in B$, it holds that*

(i) $|aa'| \leq 2/s \cdot |ab|$, *and*

(ii) $|a'b'| \leq (1 + 4/s)|ab|$.

Callahan and Kosaraju [8] showed a fast WSPD-construction based on a split tree.

▶ **WSPD-Property 2** (Callahan and Kosaraju [8]). *Let P be a set of n points in $\mathbb{R}^d$, where d is a constant, and let $s > 0$ be a real number. An s-well-separated pair decomposition for P consisting of $\mathcal{O}(s^d n)$ pairs, can be computed in $\mathcal{O}(n \log n + s^d n)$ time.*

2.2 Approximate nearest neighbour queries

For approximate nearest neighbour queries, we want to preprocess a given point set P in a metric space, such that for any query point q in the metric space, we can report quickly its *approximate nearest neighbour* in P. That is, if q^* is the exact nearest neighbour of q in P, then the query algorithm returns a point q' in P such that $|qq'| \leq (1 + \varepsilon) \cdot |qq^*|$. Such a data structure is offered in [3] for the case when P is a point set in the Euclidean space $\mathbb{R}^d$:

▶ **Lemma 2** (Arya, Mount, Netanyahu, Silverman and Wu [3]). *Let P be a set of n points in $\mathbb{R}^d$, where d is a constant. In $\mathcal{O}(n \log n)$ time, a data structure of size $\mathcal{O}(n)$ can be constructed that supports the following operations:*

- *Given a real number $\varepsilon > 0$ and a query point q in $\mathbb{R}^d$, return a $(1+\varepsilon)$-approximate nearest neighbour of q in P in $\mathcal{O}(\log n)$ time.*
- *Inserting a point into P and deleting a point from P in $\mathcal{O}(\log n)$ time.*

3 Sparse oriented constant dilation spanner

This section presents the first construction for a sparse oriented $(2+\varepsilon)$-spanner for general point sets in $\mathbb{R}^d$. Our algorithm consists of the following four steps:

1. Compute the WSPD on the given point set.
2. For two points of each well-separated pair, approximate their minimum-perimeter triangle by our algorithm using approximate nearest neighbour queries presented in Section 3.1.
3. Construct an undirected graph whose edge set consists of edges of approximated triangles.
4. Orient the undirected graph using a greedy algorithm (see Lemma 3).

In Step 4, we modify a greedy algorithm from [5], which originally works as follows. Sort the $\binom{n}{3}$ triangles of the complete graph in ascending order by their perimeter. In this order, orient each triangle clockwise or anti-clockwise if possible; otherwise, we skip this triangle. At the end of the algorithm, all remaining edges are oriented arbitrarily. In [5], they show that this greedy algorithm yields an oriented 2-spanner. Note that this graph has $\binom{n}{2}$ edges.

Our modification of the greedy algorithm is to only consider the approximate minimum-perimeter triangles, rather than all $\binom{n}{3}$ triangles. Lemma 3 states that, for any two points whose triangles are $(1+\varepsilon)$-approximated in Step 2, the dilation between those points is at most $(2+2\varepsilon)$ in the resulting orientation.

▶ **Lemma 3.*** *Let P be a set of n points in $\mathbb{R}^d$, let $\varepsilon_1 > 0$ be a real number, and let L be a list of m point triples $(p,q;r)$, with p, q, and r being pairwise distinct points in P. Assume that for each $(p,q;r)$ in L,*

$$|\Delta_{pqr}| \leq (1+\varepsilon_1) \cdot |\Delta^*(p,q)|.$$

Let $G = (P,E)$ be the undirected graph whose edge set consists of all edges of the m triangles Δ_{pqr} defined by the triples $(p,q;r)$ in L. In $\mathcal{O}(m \log n)$ time, we can compute an orientation $\overrightarrow{G}$ of G, such that $\mathrm{odil}_{\overrightarrow{G}}(p,q) \leq 2+2\varepsilon_1$ for each $(p,q;r)$ in L.

▶ Remark. The above lemma only guarantees an upper bound on $\mathrm{odil}_{\overrightarrow{G}}(p,q)$ for $(p,q;r)$ in L. For such a triple, $\mathrm{odil}_{\overrightarrow{G}}(p,r)$ and $\mathrm{odil}_{\overrightarrow{G}}(q,r)$ can be arbitrarily large.

In Algorithm 1, we pick up to two points of A and up to two points of B for each well-separated pair $\{A,B\}$ and approximate the minimum-perimeter triangle for every combination of picked points, including pairs of points from the same set. The latter is necessary, to bound the size of the minimum triangle of two points in the same set (see Proof of Theorem 4). Considering all such combinations introduces a minor computational overhead, but improves the readability of the proof. Now, we show that Algorithm 1 constructs a sparse oriented $(2+\varepsilon)$-spanner for point sets in the Euclidean space $\mathbb{R}^d$:

▶ **Theorem 4.** *Given a set P of n points in $\mathbb{R}^d$ and a real number $0 < \varepsilon < 2$, an oriented $(2+\varepsilon)$-spanner for P with $\mathcal{O}(n)$ edges can be constructed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.*

* Due to space restrictions, proofs of results marked by $*$ are given in the full version of this paper.

Algorithm 1 Sparse $(2+\varepsilon)$-Spanner.

Require: Set P of n points in $\mathbb{R}^d$, positive reals $\varepsilon < 2$, ε_1, s
Ensure: $(2+\varepsilon)$-spanner for P with $\mathcal{O}(n)$ edges
  Compute an s-WSPD $\{A_i, B_i\}$ for $1 \le i \le m = \mathcal{O}(n)$
  Compute the data structure M on P to approximate nearest neighbours (see Lemma 2 [3])
  $L = \emptyset$
  for $i = 1$ **to** m **do**
    Pick $\min\{|A_i|, 2\}$ points of A_i
    Pick $\min\{|B_i|, 2\}$ points of B_i
    K = set of all picked points
    for each $\{p, q\} \in K \times K$ with $p \neq q$ **do**
      $\Delta_{pqr} = (1+\varepsilon_1)$-approximation of $\Delta^*(p, q)$ computed by Algorithm 2 given M, p, q, ε_1
      Add the point triple $(p, q; r)$ to L
  return oriented graph generated by the algorithm in Lemma 3, given P, ε_1, L

Proof. Let the constants in Algorithm 1 be equal to $\varepsilon_1 = \varepsilon/4$ and $s = 96/\varepsilon$. Let $\overrightarrow{G}$ be the graph returned by Algorithm 1.

For any two points $p, q \in P$, we prove that $|C_{\overrightarrow{G}}(p, q)| \le (2+\varepsilon) \cdot |\Delta^*(p, q)|$. The proof is by induction on the rank of $|pq|$ in the sorted sequence of all $\binom{n}{2}$ pairwise distances.

If p, q is a closest pair (for the definition see [22]), then the pair $\{\{p\}, \{q\}\}$ is in the WSPD. Therefore, a $(1+\varepsilon_1)$-approximation of $\Delta^*(p, q)$ is added to the list L of triangles, that define the edge set of $\overrightarrow{G}$, and, due to Lemma 3 and $\varepsilon_1 = \varepsilon/4$, we have $\mathrm{odil}_{\overrightarrow{G}}(p, q) \le 2 + 2\varepsilon_1 \le 2 + \varepsilon$.

Assume that p, q is not a closest pair. Let $\{A, B\}$ be the well-separated pair with $p \in A$ and $q \in B$. We distinguish cases based on whether p and/or q are picked by Algorithm 1 while processing the pair $\{A, B\}$.

Case 1. Neither p nor q are picked by Algorithm 1.

This implies $|A| \ge 3$ and $|B| \ge 3$. Let $a \in A$ and $b \in B$ be picked points. We prove

$$|C_{\overrightarrow{G}}(p, q)| \le |C_{\overrightarrow{G}}(p, a)| + |C_{\overrightarrow{G}}(a, b)| + |C_{\overrightarrow{G}}(b, q)| \le (2+\varepsilon) \cdot |\Delta^*(p, q)|.$$

A sketch of this proof is visualized in Figure 1.

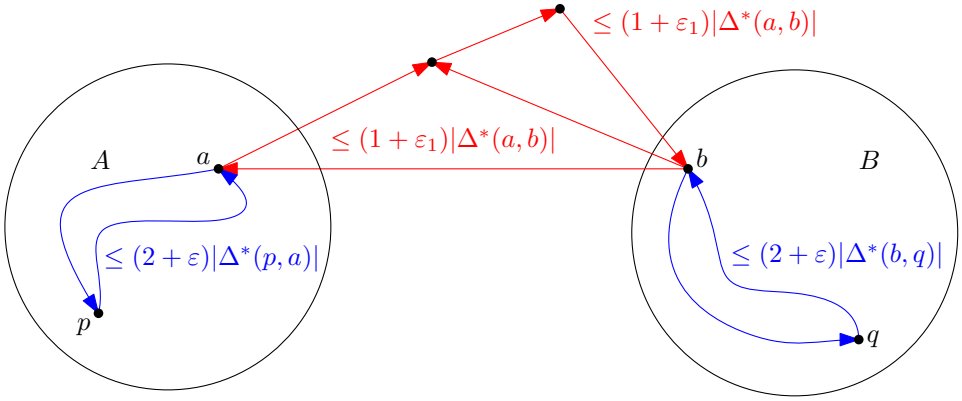


Figure 1 Visualization of Case 1: $|C_{\overrightarrow{G}}(p, q)| \le |C_{\overrightarrow{G}}(p, a)| + |C_{\overrightarrow{G}}(a, b)| + |C_{\overrightarrow{G}}(b, q)|$ in the proof of Theorem 4. $C_{\overrightarrow{G}}(a, b)$ is either a $(1+\varepsilon_1)$-approximation of $\Delta^*(a, b)$ or a union of two triangles (red), each of at most $(1+\varepsilon_1)|\Delta^*(a, b)|$. $|C_{\overrightarrow{G}}(p, a)|$ and $|C_{\overrightarrow{G}}(b, q)|$ (blue) are bounded inductively.

Since $|pa| < |pq|$ and $|bq| < |pq|$, we apply induction to bound $|C_{\overrightarrow{G}}(p,a)|$ and $|C_{\overrightarrow{G}}(b,q)|$. Since a and b are picked points, Lemma 3 applies, which gives us

$$|C_{\overrightarrow{G}}(p,q)| \leq (2+\varepsilon)|\Delta^*(p,a)| + (2+2\varepsilon_1)|\Delta^*(a,b)| + (2+\varepsilon)|\Delta^*(b,q)|.$$

Since $|A| \geq 3$, $|\Delta^*(p,a)|$ is at most the perimeter of any triangle with vertices p, a, and one other point in A. Each of the three edges of such a triangle can be bounded using WSPD-Property 1. It holds that

$$|\Delta^*(p,a)| \leq 3 \cdot 2/s \cdot |pq| \leq 3/s \cdot |\Delta^*(p,q)|, \tag{1}$$

where the last inequality $|\Delta^*(p,q)| \geq 2|pq|$ follows from the triangle inequality. Analogously, we bound $|\Delta^*(b,q)|$ by $3/s|\Delta^*(p,q)|$. By Lemma 5, we bound $|\Delta^*(a,b)|$ and achieve in total

$$|C_{\overrightarrow{G}}(p,q)| \leq 2(2+\varepsilon) \cdot 3/s|\Delta^*(p,q)| + (2+2\varepsilon_1)\,(1+8/s)\,|\Delta^*(p,q)|.$$

Since $\varepsilon_1 = \varepsilon/4$, $\varepsilon < 2$ and $s = 96/\varepsilon$, we conclude that

$$|C_{\overrightarrow{G}}(p,q)| \leq \left(2 + \frac{\varepsilon}{2} + \frac{28+10\varepsilon}{s}\right)|\Delta^*(p,q)| \leq \left(2 + \frac{\varepsilon}{2} + \frac{48}{s}\right)|\Delta^*(p,q)| = (2+\varepsilon)|\Delta^*(p,q)|.$$

Case 2. Either p or q is picked by Algorithm 1.

W.l.o.g., let p be picked for A and a distinct point b be picked for B.

Analogously to Case 1, we show that

$$|C_{\overrightarrow{G}}(p,q)| \leq |C_{\overrightarrow{G}}(p,b)| + |C_{\overrightarrow{G}}(b,q)| \leq (2+\varepsilon)|\Delta^*(p,q)|.$$

Case 3. Both p and q are picked by Algorithm 1.

An approximation of $\Delta^*(p,q)$ is added to L and then, due to Lemma 3 and $\varepsilon_1 = \frac{\varepsilon}{4}$, we have $\mathrm{odil}_{\overrightarrow{G}}(p,q) \leq 2 + 2\varepsilon_1 \leq 2 + \varepsilon$.

The initialisation is dominated by computing a WSPD in $\mathcal{O}(n \log n)$ time [8]. For each of the $\mathcal{O}(n)$ pairs, we pick at most four points and approximate the minimum triangle $\Delta^*(p,q)$ for at most six pairs $\{p,q\}$ of picked points. Therefore, computing the list L of $\mathcal{O}(n)$ triples costs $\mathcal{O}(n \log n)$ time (see Lemma 6) and orienting those costs $\mathcal{O}(n \log n)$ time (Lemma 3). Therefore, a $(2+\varepsilon)$-spanner can be computed in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space. ◀

3.1 The minimum-perimeter triangle

Let P be a set of n points in $\mathbb{R}^d$, and let p and q be two distinct points in P. The minimum triangle $\Delta^*(p,q)$ containing p and q can be computed naively in $O(n)$ time.

The following approach can be used to answer such a query in $O(n^{1-c})$ time, for some small constant $c > 0$ that depends on the dimension d. Note that computing $\Delta^*(p,q)$ is equivalent to computing the smallest real number $\rho > 0$, such that the ellipsoid

$$\sqrt{\sum_{i=1}^{d}(p_i - x_i)^2} + \sqrt{\sum_{i=1}^{d}(q_i - x_i)^2} \leq \rho$$

contains at least three points of P.

For a fixed real number ρ, we can count the number of points of P in the above ellipsoid: Using *linearization*, see Agarwal and Matousek [1], we convert P to a point set P' in a Euclidean space whose dimension depends on d. The problem then becomes that of counting

the number of points of P' in a query simplex. Chan [9] has shown that such queries can be answered in $O(n^{1-c})$ time, for some small constant $c > 0$. This result, combined with *parametric search*, see Megiddo [18], allows us to compute $\Delta^*(p, q)$ in $O(n^{1-c})$ time.

For many applications, including our $(2 + \varepsilon)$-spanner construction, an approximation of the minimum triangle is sufficient. The following lemma shows that a WSPD with respect to $s = 2/\varepsilon$ provides a $(1 + \varepsilon)$-approximation.

▶ **Lemma 5.*** *Let A and B be subsets of a point set P such that $\{A, B\}$ is an s-well-separated pair, and assume that $|B| \geq 2$.*

1. *For any point $a \in A$ and any two points $b, c \in B$, we have $|\Delta_{abc}| \leq (1 + 2/s)|\Delta^*(a, b)|$.*
2. *Assume that $|A| \geq 2$. For any two points $a, a' \in A$ and any two points $b, b' \in B$, it holds $|\Delta^*(a, b)| \leq (1 + 8/s)|\Delta^*(a', b')|$.*

Note that this works only if at least one subset of the well-separated pair contains at least two points. If both subsets have size one, we can use the following lemma to approximate the minimum triangle using approximate nearest neighbour queries.

▶ **Lemma 6.** *Let P be a set of n points in $\mathbb{R}^d$ and let $0 < \varepsilon_1 < 2$ be a real number. In $\mathcal{O}(n \log n)$ time, the set P can be preprocessed into a data structure of size $\mathcal{O}(n)$, such that, given any two distinct query points p and q in P, a $(1 + \varepsilon_1)$-approximation of the minimum triangle $\Delta^*(p, q)$ can be computed in $\mathcal{O}(\log n)$ time.*

Proof. The data structure is the approximate nearest neighbour structure of Lemma 2 [3]. The query algorithm is presented in Algorithm 2. Let the constants in this algorithm be equal to $\varepsilon_2 = \varepsilon_1/2$, $\alpha = 4/\varepsilon_1$, and $\varepsilon_3 = \frac{2}{3\sqrt{d}} \cdot \varepsilon_1$.

Let p and q be two distinct points in P. We prove that Algorithm 2 returns a $(1 + \varepsilon_1)$-approximation of $\Delta^*(p, q)$.

Algorithm 2 $(1 + \varepsilon_1)$-Approximation of the Minimum-Perimeter Triangle.

Require: Query points $p, q \in P$, positive reals $\varepsilon_1 < 2$, ε_2, ε_3, $\alpha > 2$, data structure M to approximate nearest neighbours
Ensure: Triangle Δ such that $|\Delta| \leq (1 + \varepsilon_1)|\Delta^*(p, q)|$
  $r = (1 + \varepsilon_2)$-approximate nearest neighbour of p (which is neither p nor q)
  if $|pr| > \alpha|pq|$, **then return** Δ_{pqr}
  else
    Let B be the hypercube with sides of length $3\alpha|pq|$ that is centred at p and divided into cells with sides of length of $\frac{2}{3\sqrt{d}} \cdot \varepsilon_1|pq|$
    for each cell in B **do**
      $x_c = (1 + \varepsilon_2)$-approximate nearest neighbour of the cell centre (with $x_c \neq p$, $x_c \neq q$)
    return the triangle Δ_{pqx_c} with the smallest perimeter across all cells

Throughout this proof, we denote the third point of $\Delta^*(p, q)$ by x. Let r be the $(1 + \varepsilon_2)$-approximate nearest neighbour of p in $P \setminus \{p, q\}$ that is computed by the algorithm. By the triangle inequality, it holds

$$|\Delta_{pqr}| = |pq| + |qr| + |pr| \leq 2|pq| + 2|pr|. \tag{2}$$

Case 1. $|pr| > \alpha|pq|$.

Let p^* be the exact nearest neighbour of p in the set $P \setminus \{p, q\}$. Since r is a $(1+\varepsilon_2)$-approximate nearest neighbour of p in $P \setminus \{p, q\}$, and since $|\Delta^*(p, q)| \geq 2|px|$, we have

$$|pr| \leq (1+\varepsilon_2)|pp^*| \leq (1+\varepsilon_2)|px| \leq (1+\varepsilon_2)|\Delta^*(p, q)|/2.$$

Combining this inequality with Equation (2) and the assumption that $|pr| > \alpha|pq|$, we have

$$|\Delta_{pqr}| \leq (2 + 2/\alpha)|pr| \leq (1 + 1/\alpha)(1+\varepsilon_2)|\Delta^*(p, q)|.$$

Since $\varepsilon_2 = \varepsilon_1/2$, $\alpha = 4/\varepsilon_1$ and $\varepsilon_1 < 2$, we have

$$|\Delta_{pqr}| \leq (1+\varepsilon_1/4)(1+\varepsilon_1/2)|\Delta^*(p, q)| = (1+3\varepsilon_1/4+\varepsilon_1^2/8)|\Delta^*(p, q)| \leq (1+\varepsilon_1)|\Delta^*(p, q)|.$$

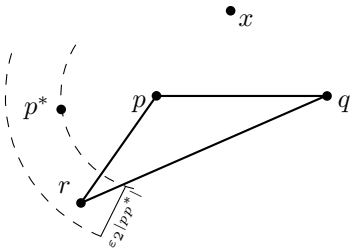


Figure 2 Visualization of Case 1 in the proof of Lemma 6. Since r is a $(1+\varepsilon_2)$-approximated nearest neighbour of p, the returned triangle Δ_{pqr} is a $(1+\varepsilon_1)$-approximation of $\Delta^*(p, q) = \Delta_{pqx}$.

Case 2. $|pr| \leq \alpha|pq|$.

Consider the hypercube B with sides of length $3\alpha|pq|$ that is centred at p.

We first prove that the third point x of $\Delta^*(p, q)$ is in B. Using Equation (2), the assumption that $|pr| \leq \alpha|pq|$, and the fact that $\alpha \geq 2$, we have

$$|\Delta^*(p, q)| \leq |\Delta_{pqr}| \leq (2 + 2\alpha)|pq| \leq 3\alpha|pq|.$$

Since $|\Delta^*(p, q)| \geq 2|px|$, it follows that $|px| \leq (3\alpha/2)|pq|$ and, therefore, $x \in B$.

Let C be the cell of B that contains x, let c be the centre of C, let c^* be the exact nearest neighbour of c in $P \setminus \{p, q\}$, and let x_c be the $(1+\varepsilon_2)$-approximate nearest neighbour of c that is computed by the algorithm. We first prove an upper bound on the distance $|xx_c|$:

$$|xx_c| \leq |xc| + |cx_c| \leq |xc| + (1+\varepsilon_2)|cc^*| \leq |xc| + (1+\varepsilon_2)|cx| = (2+\varepsilon_2)|cx| \leq 3|cx|.$$

Since x and c are in the same $\varepsilon_3|pq|$-sized cube and $\varepsilon_3 = \frac{2}{3\sqrt{d}} \cdot \varepsilon_1$, it follows that

$$|xx_c| \leq 3 \cdot \sqrt{d}/2\varepsilon_3|pq| = \varepsilon_1|pq| \leq (\varepsilon_1/2)|\Delta^*(p, q)|.$$

where the last inequality $|\Delta^*(p, q)| \geq 2|pq|$ follows from the triangle inequality.

Let $\Delta_{pqx_{c'}}$ be the triangle that is returned by the algorithm. Then

$$|\Delta_{pqx_{c'}}| \leq |\Delta_{pqx_c}| \leq |pq|+|qx|+|xx_c|+|x_cx|+|xp| = |\Delta^*(p, q)|+2|xx_c| \leq (1+\varepsilon_1)|\Delta^*(p, q)|.$$

The query algorithm performs two deletions, two insertions, and $1 + (3\alpha/\varepsilon_3)^d = \mathcal{O}(1)$ approximate nearest neighbour queries in the data structure of Lemma 2 [3]. Each such operation takes $\mathcal{O}(\log n)$ time. Therefore, the entire query algorithm takes $\mathcal{O}(\log n)$ time. ◀

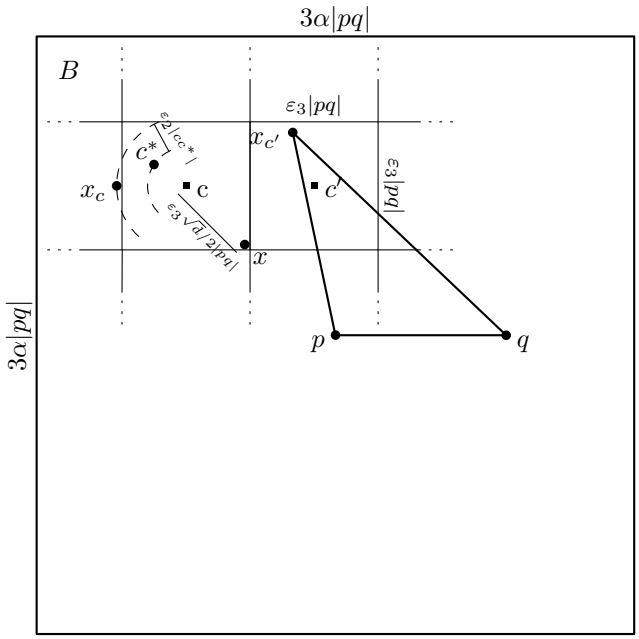


Figure 3 Visualization of Case 2 in the proof of Lemma 6. The third point x of $\Delta^*(p,q)$ lies in the cell centred at c. Since $\Delta_{pqx_{c'}}$ is returned, it holds $|\Delta_{pqx_{c'}}| \leq |\Delta_{pqx_c}|$, where $x_{c'}$ and x_c are $(1+\varepsilon_2)$-approximated nearest neighbours of c' and c.

3.2 NP-hardness

Our $(2+\varepsilon)$-spanner construction involves computing undirected $(1+\varepsilon)$-approximations of minimum-perimeter triangles, which are subsequently oriented. A similar approach is used in [5] where they prove, for convex point sets, orienting a greedy triangulation yields a plane oriented $O(1)$-spanner. Both results could be improved by optimally orienting the underlying undirected graph. The authors in [5] pose, as an open question, whether it is possible to compute, in polynomial time, an orientation of any given undirected geometric graph, that minimises the oriented dilation. We answer this question by showing NP-hardness.

▶ **Theorem 7.*** *Let P be a finite set of points in a metric space and let G be an undirected graph with vertex P. Given a real number t', it is NP-hard to decide if there exists an orientation $\overrightarrow{G}$ of G with oriented dilation $\operatorname{odil}(\overrightarrow{G}) \leq t'$. This is even true for point sets in the Euclidean plane.*

We give a proof idea, which is mainly described graphically, for details check the full version of this paper.

Proof sketch. We reduce from the NP-complete problem planar 3-SAT [17]. We start with a planar Boolean formula φ in conjunctive normal form with an incidence graph G_φ as illustrated in Figure 4. This graph can be embedded on a polynomial-size grid whose cells have side length one [13, 17]. Let $t' = 1.043$. We construct a graph G based on G_φ such that there is an orientation $\overrightarrow{G}$ of G with dilation $\operatorname{odil}(\overrightarrow{G}) \leq t'$ if and only if φ is satisfiable.

In the following, every point $p = (x, y)$ on the grid is replaced by a so-called *oriented point*, which is a pair of points $P = \{t(p), b(p)\}$ with *top* $t(p) = (x, y + \frac{\varepsilon}{2})$ and *bottom* $b(p) = (x, y - \frac{\varepsilon}{2})$, where $\varepsilon \geq 0$ is a small constant.

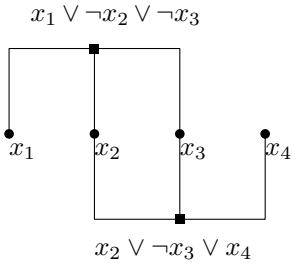


Figure 4 Example: Incidence graph of a planar 3-SAT formula embedded on a square grid.

We add an edge between $t(p)$ and $b(p)$, and its orientation encodes whether this points represent "true" or "false". W.l.o.g. we assume that an oriented edge from $b(p)$ to $t(p)$, in other words an *upwards* edge, represents "true", whereas an oriented edge from $t(p)$ to $b(p)$, in other words a *downwards* edge, represents "false".

First, we add (a polynomial number of) grid points on the edges such that all edges have length 1. Then, we replace all edges in the embedding of our formula graph G_φ by *wire* gadgets as shown in Figure 5. Note that wires propagate the orientation of oriented points – if two points next to each other on a wire have different orientations, their dilation is significantly larger than $t' = 1.043$, since the shortest oriented cycle needs to go through an additional oriented point. If they have the same orientation, their dilation is (almost) 1. To switch a signal (for a negated variable in a formula), we start the wire as in Figure 6.

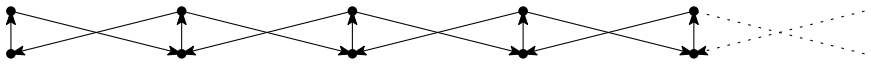


Figure 5 A wire where oriented points are oriented upwards.

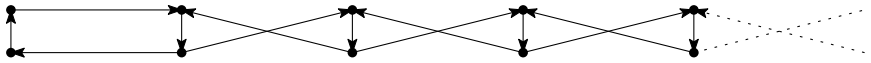


Figure 6 A wire where the orientation is switched to negate the signal.

To ensure that all clause gadgets encode the same orientation of oriented points as "true", we add a *tree of knowledge*. This is a tree with vertices on the grid shifted by $(0.5, 0.5)$ relative to the grid of G_φ and with wires as edges. The tree has two leaves per clause (see Figure 7). W.l.o.g., we assume that all oriented points of the tree of knowledge are oriented upwards (thus "true").

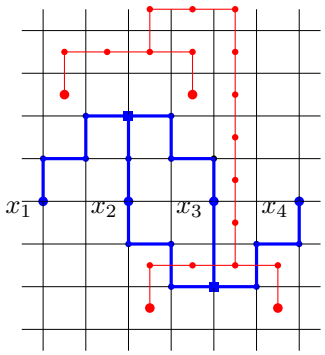


Figure 7 G_φ (blue) and its underlying grid together with a tree of knowledge (red).

All oriented points, which are not direct neighbours of a wire, are linked by a $K_{2,2}$ (compare to Figure 8). This ensures that the dilation of those points is (almost) 1. Note that the dilation of $t(p)$ and $b(p)$ is 1, regardless of whether p and p' are direct neighbours or not.

Figure 8 $K_{2,2}$ for oriented points with same orientation (left) and different orientation (right).

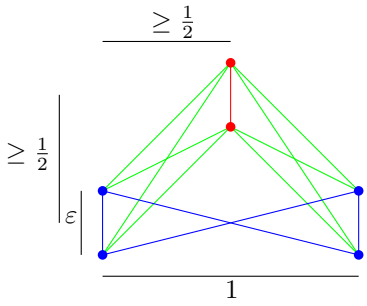


Figure 9 A wire (blue) can not be shortcut by $K_{2,2}$ gadgets (green).

Adding these $K_{2,2}$ gadgets does not affect the functionality of a wire gadget: Since any other point has at least distance $(0.5, 0.5)$ to two direct neighbours, a wire can not be shortcut by $K_{2,2}$ gadgets (compare to Figure 9).

For every clause, its two leaves in the tree of knowledge are not linked by a $K_{2,2,}$, but rather by a *clause gadget*. Figure 10 shows the two leaves and G_φ at the clause. We can assume that G_φ is embedded as shown, in particular, leaving the area directly above the clause empty. We illustrate a clause gadget in Figure 11, and in Figure 12 with more detail.

The oriented points L (left), R (right) and B (bottom) are the ends of the variable wires of a clause. They are placed such that they lie just inside an ellipse with the leaves l_1 and l_2 as foci, and without any other points in the ellipse (compare to Figures 10 and 11). As proven later, the gadget as shown in Figure 12 guarantees that, in order to obtain $\mathrm{odil}(L_1, L_2) \leq t'$, one of $\{L, R, B\}$ must lie on a shortest closed walk containing L_1, and L_2 and the orientation of that point must be the same as of L_1 and L_2 (thus, the literal is "true").

For each of $\{L, R, B\}$, we ensure there exists a *satellite* point, which is an oriented point on the variable wire, which is close but outside the ellipse. Its purpose is to ensure that the oriented dilation of L_1 (and likewise L_2) with L, B and R is below t', even if the orientation of that point is different as of L_1 and L_2, in other words, if the corresponding literal does not satisfy the clause.

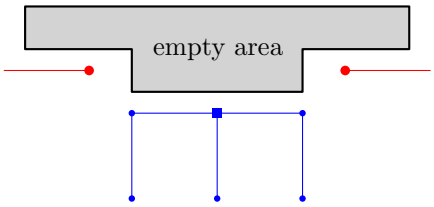


Figure 10 Embedded clause.

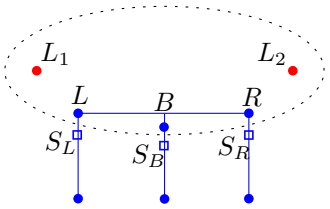


Figure 11 Clause gadget.

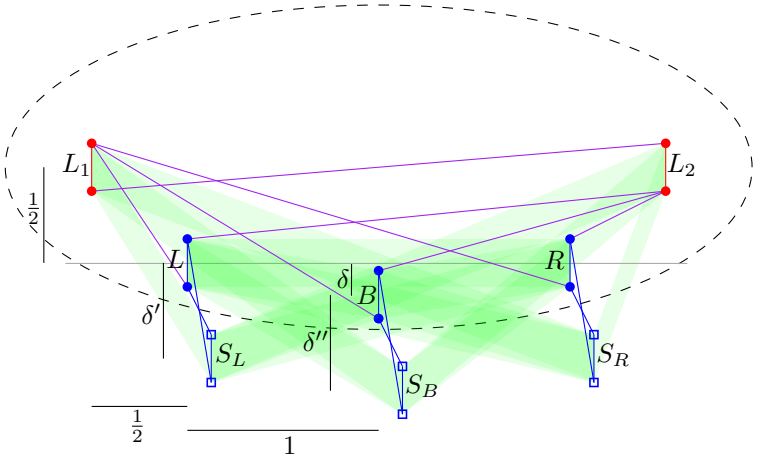


Figure 12 Detailed clause gadget. $K_{2,2}$ gadgets are indicated by green parallelograms.

By setting $\delta := 0.1335$, $\delta' := 0.0152$ and $\delta'' := 0.0304$, we obtain the following properties:

- The dilation of L_1 (and likewise L_2) with L, B and R is below t', even if the orientation of that point is different as of L_1 and L_2.
- If one of $\{L, B, R\}$ is oriented upwards, the dilation of L_1 and L_2 is smaller than $t' := 1.043$.
- If none of $\{L, B, R\}$ is oriented upwards, the dilation of L_1 and L_2 is greater than t'.

Thus, formula φ is satisfiable if and only if there exists an orientation of our constructed graph with dilation at most 1.043. ◀

Following the construction in the proof of Theorem 7, Figure 13 illustrates the graph for the formula $\varphi = (x_1 \vee \neg x_2 \vee \neq x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$ (see also Figures 4 and 7).

4 Oriented dilation

Computing the dilation of a given graph is related to the all-pairs shortest paths problem. There is a well-known cubic-time algorithm for APSP by Floyd and Warshall [11, 23]. By using their algorithm (and computing the minimum triangles naively), the oriented dilation of a given graph with n points can be computed in $\mathcal{O}(n^3)$ time.

We prove APSP-hardness for computing the oriented dilation. Since the APSP-problem is believed to need truly cubic time [20], it is unlikely that computing the oriented dilation for a given graph is possible in subcubic time.

Our hardness proof is a subcubic reduction to MinimumTriangle, which is the problem of finding a minimum weight cycle in an undirected graph with weights in $[7M, 9M]$, where the minimum weight cycle is a triangle. For two computational problems A and B, a *subcubic reduction* is a reduction from A to B where any subcubic time algorithm for B would imply a subcubic time algorithm for A (for a more thorough definition see [24]). We use the notation $A \leq_3 B$ to denote the existence of a subcubic reduction from A to B.

Vassilevska Williams and Williams [24] proved APSP-hardness for detecting if a weighted graph has a triangle of negative edge weight. Following thier proof, we prove APSP-hardness of the more specific problem MinimumTriangle:

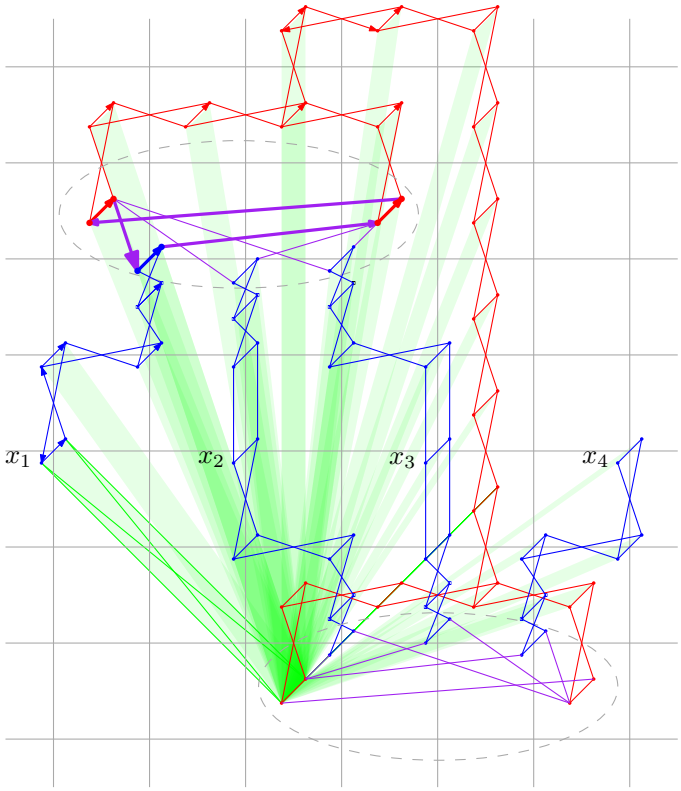


Figure 13 Graph constructed for $\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$. For visibility, oriented points are placed diagonally instead of vertically. Only for one point, all $K_{2,2}$-gadgets are indicated by green parallelograms. If the oriented point at the end of the wire from x_1 (blue) is – as indicated – oriented the same way as the tree of knowledge (red), this corresponds to setting it to true, resulting in an oriented closed walk in the clause gadget (purple) that gives a dilation smaller than 1.043.

▶ **Theorem 8.*** (APSP $\leq_3$ MinimumTriangle). *Let G be an undirected graph with weights in $[7M, 9M]$ whose minimum weight cycle is a triangle. Then, it is APSP-hard to find a minimum weight triangle in G.*

By reduction to MinimumTriangle, we prove APSP-hardness for ODIL, which is the problem of computing the oriented dilation of a given oriented graph in a metric space.

▶ **Theorem 9** (APSP $\leq_3$ Odil)**.** *Let $\overrightarrow{G}$ be an oriented metric graph. It is APSP-hard to compute the oriented dilation of $\overrightarrow{G}$.*

Proof. Due to Theorem 8, it is sufficient to prove MinimumTriangle $\leq_3$ Odil.

Let $G = (P, E)$ be an undirected graph with $|P| = n$ and weights $w : E \to [7M, 9M]$ in which the minimum weight cycle C of G is a triangle. Note that the weights of G fulfil the triangle inequality.

We define a metric space $(P', |\cdot|)$ with $P' = P \cup \{u, v\}$ by the following distances:

- $|pq| = w(p, q) + 15M$, for $\{p, q\} \in E$,
- $|pq| = 14M + 15M = 29M$, for $\{p, q\} \notin E$, and
- $|up| = |vp| = |vu| = 15M + 15M = 30M$ for $p \in P$.

So, for each missing edge of a complete graph on P, we assign its weight to be $14M$. Therefore, we obtain a metric with distances in $[7M, 14M]$. Then, we add the points u and v with distance $15M$ to each point in $p \in P$. To preserve the metric, we increase the distance of every tuple by $15M$. This construction is visualised in Figure 14.

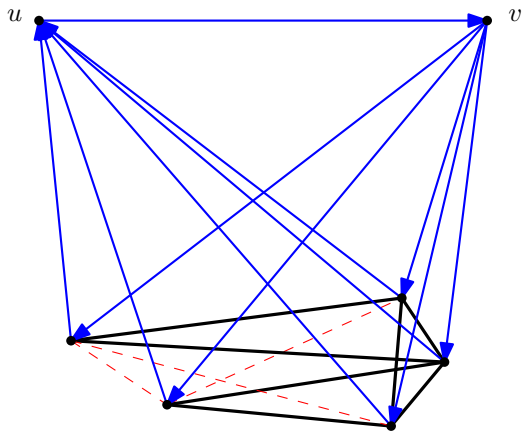


Figure 14 Graph construction in the proof of Theorem 8. Given an undirected graph on P (black), we add two points u, v with a large distance to all other points and augment it to a complete graph (red, dashed). The oriented graph $\overrightarrow{G}$ (blue) ensures that every shortest closed walk containing p and q in P visits u and v.

Let $\overrightarrow{G} = (P', \overrightarrow{E})$ be an oriented graph with $\overrightarrow{E} = \{(p,u),(v,p) \mid p \in P\} \cup \{(u,v)\}$.

We prove that the weight $w(C)$ of the minimum triangle can be computed by computing $\operatorname{odil}(\overrightarrow{G})$. In particular, we show that $w(C) = \frac{180M}{\operatorname{odil}(\overrightarrow{G})} - 45M$.

We proceed by computing the oriented dilation of $\overrightarrow{G}$. We first note that all edges of $\overrightarrow{G}$ have weight $30M$.

For every tuple of u or v and a point $p \in P$, the shortest closed walk is a triangle of length $|C_{\overrightarrow{G}}(u,p)| = |C_{\overrightarrow{G}}(v,p)| = |C_{\overrightarrow{G}}(u,v)| = 3 \cdot 30M = 90M$. Their minimum triangle contains two edges of length $30M$. Therefore, their dilation is

$$\operatorname{odil}(u,p) = \frac{|C_{\overrightarrow{G}}(u,p)|}{|\Delta^*(u,p)|} \leq \frac{90M}{2 \cdot 30M} = 1.5.$$

Analogously, it holds that $\operatorname{odil}(v,p) \leq 1.5$ for $p \in P$.

For u and v, it holds that $\operatorname{odil}(u,v) = 1$, because every triangle in P' containing u and v is an oriented closed walk in $\overrightarrow{G}$.

For two distinct points $p, q \in P$, the shortest closed walk containing p and q visits u and v and has length $|C_{\overrightarrow{G}}(p,q)| = 6 \cdot 30M = 180M$. For their minimum triangle holds $|\Delta^*(p,q)| \leq 3 \cdot 29M = 87M$. Since their dilation is larger than 2, a worst-case pair of $\overrightarrow{G}$ must be a pair in P. Therefore, the oriented dilation of the constructed graph $\overrightarrow{G}$ is

$$\operatorname{odil}(\overrightarrow{G}) = \max_{p,q \in P} \frac{|C_{\overrightarrow{G}}(p,q)|}{|\Delta^*(p,q)|} = \frac{180M}{\min\limits_{p,q,p^* \in P} w'(p,q) + w'(q,p^*) + w'(p,p^*)} > 2.$$

Since C is a triangle and $w(C) \leq 27M$, it holds that

$$\min_{p,q,p^* \in P} |pq| + |qp^*| + |p,p^*| = \min_{p,q,p^* \in P} w(p,q) + w(q,p^*) + w(p,p^*) + 3 \cdot 15M = w(C) + 45M.$$

Therefore, the oriented dilation of $\overrightarrow{G}$ is achieved by the minimum-length triangle C. Consequently, given the oriented dilation of $\overrightarrow{G}$, the length of C is $w(C) = \frac{180M}{\operatorname{odil}(\overrightarrow{G})} - 45M$.

The most expensive operation in our reduction is defining the metric on $\mathcal{O}(n^2)$ points with distances in $[22M, 30M]$. Following the definitions in [24], all instances have size $\mathcal{O}(n^2 \operatorname{polylog}(M))$ and all operations take subcubic time.

Thus, computing the oriented dilation of an oriented metric graph is at least as hard as finding a minimum weight triangle in an undirected graph. ◀

To compute the oriented dilation, testing all $\binom{n}{2}$ point tuples seems to be unavoidable. However, to approximate the oriented dilation, we show that, a linear set of tuples suffices.

▶ **Theorem 10.*** *Let P be a set of n points in $\mathbb{R}^d$, let $\overrightarrow{G}$ be an oriented graph on P, and let $\varepsilon > 0$ be a real number. Assume that, in $T(n)$ time, we can construct a data structure such that, for any two query points p and q in P, we can return, in $f(n)$ time, a k-approximation to the length of a shortest path in G between p and q. Then we can compute a value $\overline{\operatorname{odil}}$ in $\mathcal{O}(T(n) + n(\log n + f(n)))$ time, such that $(1-\varepsilon) \cdot \operatorname{odil}(\overrightarrow{G}) \leq \overline{\operatorname{odil}} \leq k \cdot \operatorname{odil}(\overrightarrow{G})$.*

Our approximation algorithm (Algorithm 3) uses the WSPD to achieve a suitable selection of $\mathcal{O}(n)$ point tuple that needs to be considered. The algorithm and its analysis work similar to our $(2+\varepsilon)$-spanner construction (compare to Algorithm 1). For each selected tuple, the algorithm approximates their oriented dilation by approximating their minimum triangle and shortest closed walk.

The precision and runtime of this approximation depend on approximating the length of a shortest path between two query points. The point-to-point shortest path problem in directed graphs is an extensively studied problem (see [14] for a survey). Alternatively, shortest path queries can be preprocessed via approximate APSP (see [10, 21, 25]). For planar directed graphs, the authors in [2] present a data structure build in $\mathcal{O}(n^2/r)$ time, such that the exact length of a shortest path between two query points can be returned in $\mathcal{O}(\sqrt{r})$ time. Setting $r = n^{2/3}$ minimizes the runtime of our Algorithm 3 to $\mathcal{O}(n^{4/3})$. It should be noted that even with an exact shortest path queries ($k = 1$) the oriented dilation remains an approximation due to the minimum-perimeter triangle approximation.

Algorithm 3 Approximate Oriented Dilation.

Require: Oriented graph $\overrightarrow{G} = (P, \overrightarrow{E})$, positive reals ε, ε_1, s, parameter k
Ensure: Value $\overline{\operatorname{odil}}$, such that $(1-\varepsilon)\operatorname{odil}(\overrightarrow{G}) \leq \overline{\operatorname{odil}} \leq k \operatorname{odil}(\overrightarrow{G})$
  Compute s-WSPD with the pairs $\{A_i, B_i\}$ for $1 \leq i \leq m$
  Compute the data structure M on P to approximate nearest neighbours (see Lemma 2 [3])
  $\overline{\operatorname{odil}} = 1$
  for $i = 1$ **to** m **do**
    $A =$ Pick $\min\{|A_i|, 2\}$ points of A_i
    $B =$ Pick $\min\{|B_i|, 2\}$ points of B_i
    for each $\{a, b\} \in A \times B$ **do**
      $\overline{d}(a,b) = k$-approximation of the length of a shortest path from a to b in $\overrightarrow{G}$
      $\overline{d}(b,a) = k$-approximation of the length of a shortest path from b to a in $\overrightarrow{G}$
      $\Delta_{abc} = (1+\varepsilon_1)$-approximation of $\Delta^*(a,b)$ by Algorithm 2 given M, a, b, ε_1
      $\overline{\operatorname{odil}} = \max\left\{\frac{\overline{d}(a,b)+\overline{d}(b,a)}{|\Delta_{abc}|}, \overline{\operatorname{odil}}\right\}$
  return $\overline{\operatorname{odil}}$

5 Conclusion and outlook

We present an algorithm for constructing a sparse oriented $(2+\varepsilon)$-spanners for multidimensional point sets. Our algorithm computes such a spanner efficiently in $O(n \log n)$ time for point sets of size n. In contrast, [5] presents a plane oriented $\mathcal{O}(1)$-spanner, but only for points in convex position. Developing algorithms for constructing plane oriented $\mathcal{O}(1)$-spanners for general two-dimensional point sets remains an open problem. Another natural open problem is to improve upon $t = 2 + \varepsilon$. For this, the question arises: Does every point set admit an oriented t-spanner with $t < 2$? Even for complete oriented graphs this is open.

Both algorithms could be improved by optimally orienting the underlying undirected graph, which is constructed first. We discard this idea by proving that, given an undirected graph, it is NP-hard to find its minimum dilation orientation. In particular, is the problem NP-hard for plane graphs or for complete graphs?

In the second part of this paper, we study the problem of computing the oriented dilation of a given graph. We prove APSP-hardness of this problem for metric graphs. We complement this by a subcubic approximation algorithm. This algorithm depends on approximating the length of a shortest path between two given points. Therefore, improving the approximation of a shortest path between two given points in directed/oriented graphs is an interesting problem. The APSP-hardness of computing the oriented dilation of metric graphs seems to be dominated by the computation of a minimum-perimeter triangle. However, for Euclidean graphs, the minimum-perimeter triangle containing two given points can be computed in $o(n)$ time. This raises the question: is computing the oriented dilation of a Euclidean graph easier than for general metric graphs?

Finally, as noted in [5], in many applications some bi-directed edges might be allowed. This opens up a whole new set of questions on the trade-off between dilation and the number of bi-directed edges. Since this is a generalisation of the oriented case, both of our hardness results also apply to such models.

References

1 Pankaj K. Agarwal and Jirí Matousek. On range searching with semialgebraic sets. *Discret. Comput. Geom.*, 11:393–418, 1994. doi:10.1007/BF02574015.

2 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 Josep Díaz and Maria J. Serna, editors, *Proc. 4th Annu. European Sympos. Algorithms (ESA)*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528. Springer-Verlag, 1996. doi:10.1007/3-540-61680-2_79.

3 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.

4 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

5 Kevin Buchin, Joachim Gudmundsson, Antonia Kalb, Aleksandr Popov, Carolin Rehs, André van Renssen, and Sampson Wong. Oriented spanners. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *Proc. 31st Annu. European Sympos. Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.26.

6 Kevin Buchin, Antonia Kalb, Guangping Li, and Carolin Rehs. Experimental analysis of oriented spanners on one-dimensional point sets. In *Proc. 36th Canad. Conf. Comput. Geom. (CCCG)*, pages 223–231, 2024.

7 Kevin Buchin, Antonia Kalb, Carolin Rehs, and André Schulz. Oriented dilation of undirected graphs. In *30th European Workshop on Computational Geometry (EuroCG)*, pages 65:1–65:8, 2024.

8 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.

9 Timothy M. Chan. Optimal partition trees. *Discret. Comput. Geom.*, 47(4):661–690, 2012. doi:10.1007/S00454-012-9410-Z.

10 Michal Dory, Sebastian Forster, Yael Kirkpatrick, Yasamin Nazari, Virginia Vassilevska Williams, and Tijn de Vos. Fast 2-approximate all-pairs shortest paths. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 4728–4757. SIAM, 2024. doi:10.1137/1.9781611977912.169.

11 Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345–345, 1962. doi:10.1145/367766.368168.

12 Panos Giannopoulos, Rolf Klein, Christian Knauer, Martin Kutz, and Dániel Marx. Computing geometric minimum-dilation graphs is NP-hard. *Internat. J. Comput. Geom. Appl.*, 20(2):147–173, 2010. doi:10.1142/S0218195910003244.

13 Michael Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions*. Dissertation, Free University Berlin, 1999. doi:10.17169/refubium-7780.

14 Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 156–165, 2005. doi:10.5555/1070432.1070455.

15 Joachim Gudmundsson and Christian Knauer. Dilation and detours in geometric networks. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 53–69. Chapman and Hall/CRC, 2018. doi:10.1201/9781420010749-62.

16 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discret. Comput. Geom.*, 7:13–28, 1992. doi:10.1007/BF02187821.

17 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

18 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983. doi:10.1145/2157.322410.

19 Giri Narasimhan and Michiel H. M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:10.1017/CBO9780511546884.

20 K. R. Udaya Kumar Reddy. A survey of the all-pairs shortest paths problem and its variants in graphs. *Acta Univ. Sapientiae Inform*, 8:16–40, 2016. doi:10.1515/ausi-2016-0002.

21 Barna Saha and Christopher Ye. Faster approximate all pairs shortest paths. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 4758–4827. SIAM, 2024. doi:10.1137/1.9781611977912.170.

22 Michiel H. M. Smid. The well-separated pair decomposition and its applications. In *Handbook of Approximation Algorithms and Metaheuristics (2)*. Chapman and Hall/CRC, 2018.

23 Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962. doi:10.1145/321105.321107.

24 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5), 2018. doi:10.1145/3186893.

25 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.

Small Triangulations of 4-Manifolds and the 4-Manifold Census

Rhuaidi Antonio Burke
School of Mathematics and Physics, The University of Queensland, Brisbane, Australia

Benjamin A. Burton
School of Mathematics and Physics, The University of Queensland, Brisbane, Australia

Jonathan Spreer
School of Mathematics and Statistics F07, The University of Sydney, Australia

Abstract

We present a framework to classify PL-types of large censuses of triangulated 4-manifolds, which we use to classify the PL-types of all triangulated 4-manifolds with up to 6 pentachora. This is successful except for triangulations homeomorphic to the 4-sphere, $\mathbb{C}P^2$, and the rational homology sphere $QS^4(2)$, where we find at most four, three, and two PL-types respectively. We conjecture that they are all standard. In addition, we look at the cases resisting classification and discuss the combinatorial structure of these triangulations – which we deem interesting in their own rights.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Theory of computation → Random search heuristics; Mathematics of computing → Mathematical software

Keywords and phrases computational low-dimensional topology, triangulations, census of triangulations, 4-manifolds, PL standard 4-sphere, Pachner graph, mathematical software, experiments in low-dimensional topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.28

Related Version *Full Version*: `https://arxiv.org/abs/2412.04768`

Supplementary Material *Software*: `https://github.com/raburke/Dim4Census` [8]
archived at `swh:1:dir:ee5ac0c76fdef9983c5de8a0be93f7684dd9a796`
Software: `https://github.com/raburke/Dim4Census` [8]
archived at `swh:1:dir:ee5ac0c76fdef9983c5de8a0be93f7684dd9a796`

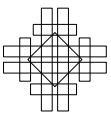
Funding *Rhuaidi Antonio Burke*: This paper was finished whilst the first author was visiting the University of Sydney (supported by the Australian Research Council's Discovery funding scheme DP190102259), and thanks the University of Sydney for their hospitality.
Jonathan Spreer: Supported by the Australian Research Council's Discovery funding scheme (project no. DP190102259).

Acknowledgements The authors thank Ryan Budney for his contributions at an early stage of this project, and the anonymous referees for their insightful comments.

1 Introduction

In the context of computational topology it is desirable to have a data set of examples on which to perform experiments and test hypotheses. As such, an exhaustive list of all 4-manifold triangulations of a certain size and type, called a *census*, serves as such a useful reference. In dimension three we currently have censuses available with up to 11 tetrahedra (containing 13 400 closed, orientable, prime, minimal triangulations). In dimension 4, where a closed 4-manifold requires an even number of pentachora, censuses are currently limited to triangulations with 2, 4, and 6 pentachora. However, despite this relative scarcity of data, even with just three tiers of the census, there are a total of 441 287 triangulations to consider.

© Rhuaidi Antonio Burke, Benjamin A. Burton, and Jonathan Spreer;
licensed under Creative Commons License CC-BY 4.0



Given a smooth manifold X, another smooth manifold X' is called *exotic (with respect to X)*, if X' is homeomorphic, but not diffeomorphic to X. In other words, X and X' represent the same topological manifold but are distinct as smooth manifolds. Dimension four is the first dimension in which exotic smooth structures appear. It is a fundamental problem in 4-manifold topology to determine the number of smooth structures a particular 4-manifold admits. The simplest and most famous instance of this problem is the *smooth 4-dimensional Poincaré conjecture* (S4PC for short):

▶ **Conjecture 1** (S4PC). *There do not exist exotic 4-spheres.*

It is typical to discuss exotic structures in relation to a "standard" or canonical smooth structure. For example, the standard structure on $\mathbb{R}^n$ is the one given by a single chart with the identity map; on $\mathbb{S}^n$ it is the one with two charts given by stereographic projection.

▶ Remark 2. The concept of a standard smooth structure is not well-defined in isolation. For instance, let $K3$ be the $K3$ *surface* with the standard smooth structure coming from being the zero-set of $w^4+x^4+y^4+x^4=0$ in $\mathbb{C}P^3$, and let $\mathbb{C}P^2$ be the standard complex projective plane. Then each of the connected sums[1] $X_1 = \#_3\mathbb{C}P^2\#_{20}\overline{\mathbb{C}P^2}$ and $X_2 = K3\#\overline{\mathbb{C}P^2}$ inherits a standard smooth structure from their summands. It is known that $X_1 \cong_{\text{TOP}} X_2$ but $X_1 \ncong_{\text{DIFF}} X_2$ [28], meaning X_1 is exotic with respect to X_2 and vice versa. The smallest triangulations of two 4-manifolds that are known to be homeomorphic, but not diffeomorphic, are *ideal triangulations* (i.e., triangulations of manifolds with non-empty boundary where the boundary is given by a neighbourhood of a vertex) with 10 pentachora [7]. The authors are not aware of any triangulations of closed 4-manifolds which are homeomorphic, but not diffeomorphic, and which are not simply formed from a connected sum of 'standard' well-known manifolds (such as X_1 and X_2 above).

Cairns [15, 16] and Whitehead [37] show that every smooth n-manifold can be triangulated, that is, admits a piecewise-linear (PL) structure. Moreover, every PL n-manifold for $n \leq 6$ admits a compatible smooth structure which is unique up to diffeomorphism [26, 32]. Hence, there is a bijective correspondence between isotopy classes of smooth and PL structures on 4-manifolds, and so we speak of smooth and PL structures on a 4-manifold interchangeably.

Related Work

The past decade has seen significant interest in the problem of enumerating and classifying triangulations, including several other attempts at classifying and simplifying triangulations of 4-manifolds, particularly 4-spheres. Most recently, Pérez [20], building on the work of Joswig et al. [27], analysed the same census we investigate in this paper.

More broadly, there has been a growing body of utilising Pachner moves to either simplify triangulations or establish PL-homeomorphisms. For example, the work Björner and Lutz [3], Lutz and Tsuruga [36], the second author [11], Altmann and the third author [2], the second and third author [14], and more recently the first author [7]. We note that whilst Björner, Lutz, Joswig et al., and Tsuruga, and (to some extent) Pérez use *simplicial complexes*, the work of the authors and Altmann instead use *generalised triangulations* (see Section 2.1). These are more "flexible" than simplicial complexes since, for example we allow two facets of the same simplex to be identified.

[1] Given two d-manifolds X and Y, their *connected sum*, written $X\#Y$ is formed by removing an open d-ball from each of X and Y and gluing them together along their resulting boundaries. We use $\#_k X$ to denote the k-fold connected sum of X with copies of itself. The operation is independent of the d-ball removed from their summands.

Our Contributions

We study the census of triangulations of closed, orientable, 4-manifolds with up to 6 pentachora. This data set contains 8 triangulations with 2, 784 triangulations with 4, and 440 495 triangulations with 6 pentachora, respectively. It includes a variety of triangulations with interesting combinatorial and topological features (see Tables 1–3 for a breakdown of numbers of triangulations by (PL-)topological types). The dataset was first generated by Budney and the second author [4] using `tricensus`, a utility of [12].

In addition, we present a new algorithm for finding PL-homeomorphisms between large numbers of 4-manifold triangulations by combining original heuristics and computational tools. This algorithm yields a near complete classification of the PL-types in the census. We note that, due to the undecidability of the 4-manifold homeomorphism problem [30], we do not expect there to be an efficient deterministic algorithm to complete this task. We also investigate the combinatorial structure of the triangulations resisting classification.

Specifically, we improve on the results of Pérez, reducing the upper bounds on the number of potential PL classes for $\mathbb{C}P^2$ and $\mathbb{S}^4$: within the set of 6-pentachoron triangulations homeomorphic to $\mathbb{C}P^2$, we reduce the number of potential PL classes from 5 down to 3; within the set of 4-pentachoron triangulations homeomorphic to $\mathbb{S}^4$, we reduce from 3 to 2; and finally for the set of 6-pentachoron triangulations homeomorphic to $\mathbb{S}^4$, we obtain at most 4 potential classes, down from 36 in [20].

2 Preliminaries

2.1 Triangulations

We refer to a 4-simplex as a *pentachoron* (plural: *pentachora*). The tetrahedral cells of a pentachoron are referred to as *facets*. We label the vertices of a pentachoron by elements of $\{0, 1, 2, 3, 4\}$, and use the convention that facet i refers to the facet opposite to the vertex labelled by i. A *(generalised, 4-dimensional) triangulation* $\mathcal{T}$ is a finite collection of n abstract pentachora, some or all of whose $5n$ facets are affinely identified ("glued") in pairs. These generalised triangulations are typically far more efficient than simplicial complexes, since we allow two facets of the same pentachoron to become identified.

The *boundary* of $\mathcal{T}$ consists of all the facets that are not identified with any other facets. We allow facets of the same pentachoron to be identified. The gluings defining $\mathcal{T}$ also have the effect of merging vertices, edges, triangles, and tetrahedra of the pentachora into equivalence classes, which we refer to as the *vertices*, *edges*, *triangles*, and *tetrahedra* of $\mathcal{T}$.

The *link* $\mathrm{lk}(v)$ of a vertex v of $\mathcal{T}$, is the "frontier" of a small regular neighbourhood of v. We treat vertex links as triangulated 3-dimensional spaces, formed by inserting a small tetrahedron into each corner of each pentachoron, and then joining together the tetrahedra from adjacent pentachora along their triangular faces. This mirrors the traditional concept of a link in a simplicial complex, but is modified to support generalised triangulations.

If v lies in the boundary of $\mathcal{T}$, its link is a space with boundary. If the link is homeomorphic to B^3, then we refer to v as a *boundary vertex*. If the link is any other space with boundary then we say v is an *invalid vertex*. On the other hand, if v does not lie in the boundary of $\mathcal{T}$, its link is a closed space. If the link is (PL-)homeomorphic to $\mathbb{S}^3$, then v is an *internal vertex*, and if it is any other closed space then it is referred to as an *ideal vertex*.

We insist that no edge of $\mathcal{T}$ is identified with itself in reverse, and that no triangle is identified with itself via a non-identity permutation. This, together with the condition that the link of every vertex is either B^3 or $\mathbb{S}^3$, guarantees that $\mathcal{T}$ triangulates a 4-manifold.

Given a triangulation $\mathcal{T}$ of a 4-manifold, the vector $f(\mathcal{T}) = (f_0, f_1, f_2, f_3, f_4)$, where f_i denotes the number of i-dimensional faces in $\mathcal{T}$, is called its *face vector*, or *f-vector* for short. Since $\mathcal{T}$ triangulates a 4-manifold, it satisfies the following equations

$$-2f_3 + 5f_4 = 0 \quad (1)$$
$$2f_1 - 3f_2 + 4f_3 - 5f_4 = 0 \quad (2)$$
$$f_0 - f_1 + f_2 - f_3 + f_4 = \chi(\mathcal{T}) \quad (3)$$

known as *generalised Dehn-Sommerville equations* [35]. Here, $\chi(\mathcal{T})$ denotes the *Euler characteristic* of $\mathcal{T}$, a topological invariant. Observe that Equation (1) implies that a triangulation of a closed 4-manifold must have an even number of pentachora.

To encode a triangulation, we give each pentachoron a label and an ordering of its five vertices. Two triangulations are *(combinatorially) isomorphic* if they are identical up to relabelling of pentachora and/or reordering of the pentachoron vertices. We can uniquely identify any isomorphism class of triangulations using an efficiently-computable string called an *isomorphism signature* [11]. Every triangulation has a unique isomorphism signature, and two triangulations have the same signature if and only if they are isomorphic.

An important tool in the study of triangulations is the *dual graph* (also known as the *face pairing graph*). Given a triangulation $\mathcal{T}$, its dual graph $\Gamma(\mathcal{T})) = (V, E)$ is the multigraph whose nodes are the pentachora in $\mathcal{T}$, and for each gluing that identifies two tetrahedral facets of p_i and p_j we add an arc between the corresponding nodes in V. By construction, $\Gamma(\mathcal{T})$ has maximum degree ≤ 5, and is 5-regular when $\mathcal{T}$ triangulates a closed 4-manifold $\Gamma(\mathcal{T})$.

The dual graph as defined above does not retain any information about the permutations used in the face identifications, and so $\mathcal{T}$ cannot be reconstructed from $\Gamma(\mathcal{T})$ alone. Nevertheless, some information about the underlying topology of the 4-manifold can still be extracted from $\Gamma(\mathcal{T})$. It also proves to be useful as a visual tool when discussing triangulations.

▶ Housekeeping 1. In order to avoid confusion, we use the terms *vertices* and *edges* exclusively when referring to triangulations, and *nodes* and *arcs* when referring to graphs.

2.2 Local Moves and the Pachner Graph

We modify our triangulations using *Pachner moves* (or *bistellar flips*), which are local moves that change a triangulation but not its underlying PL-type [33]. Informally, an (i, j) Pachner move can be thought of as taking a subcomplex of i pentachora in the boundary comlpex of the 5-simplex $\partial\Delta^5$, and replacing them with its complement in $\partial\Delta^5$ of $6 - i = j$ pentachora. This necessarily means that *(a)* the i pentachora share a common $(5 - i)$-dimensional face – which is removed from the triangulation; and *(b)* the j pentachora share a common $(i-1)$-face – which is newly inserted into the triangulation. See Figure 1 for an illustration of Pachner moves in dimension four, and see [33] for more details.

The use of Pachner moves has several key benefits: Firstly, two PL manifolds X, and X' are PL-homeomorphic if and only if there exists a sequence of Pachner moves between X and X'. This statement is known as *Pachner's theorem* and was first proven in [33]. We can therefore certify that two triangulations are PL-homeomorphic by finding a connecting sequence of Pachner moves. Even without a guarantee that this algorithm will terminate, it is effective in practice. Secondly, Pachner moves can be used to reduce the size of a triangulation (without changing the underlying PL-type). Finally, Pachner moves are already implemented and ready-to-use in several software packages. Here, we use *Regina* [12].

The *Pachner graph* of a PL manifold X, denoted by $\mathcal{P}(X)$, is used to describe and track how distinct triangulations of a 4-manifold can be related via Pachner moves. It is the (infinite) graph with nodes corresponding to isomorphism classes of triangulations of X; and two nodes of $\mathcal{P}(X)$ are joined by an arc if and only if there is a single Pachner move that takes one triangulation to the other. We refer the reader to [11] for further details.

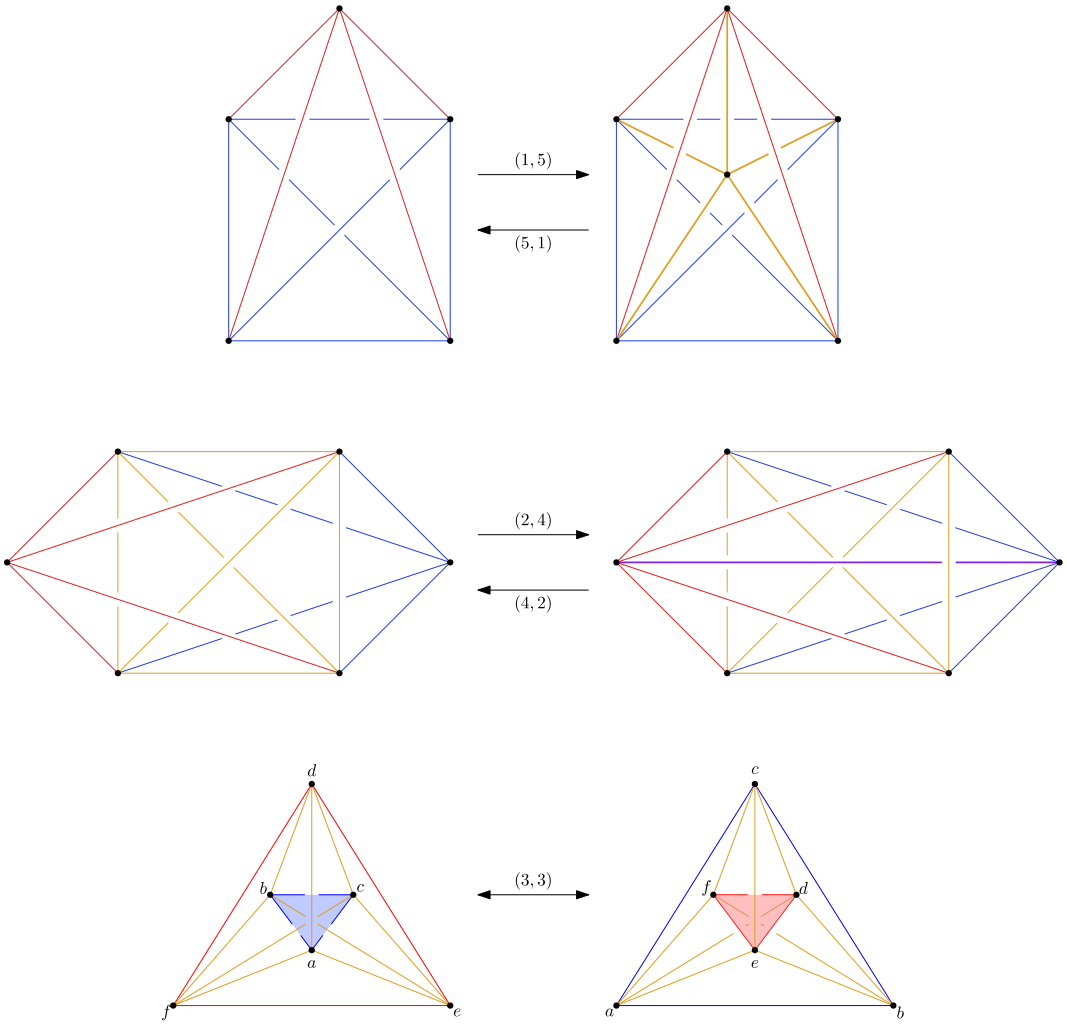


Figure 1 Pachner moves and their inverses in dimension 4.

There are many other local modifications which can be expressed by sequences of Pachner moves. When moving through the Pachner graph, using these additional modifications can be powerful, since they allow the use of "short-cuts" into different areas of the graph. In this article, we make extensive use of two of these additional moves, the 2-0-*edge move* and the 2-0-*triangle move*. The 2-0-edge move takes two pentachora, identified along three tetrahedra to form a "pillow" around a common edge, and flattens them to form two tetrahedra. The 2-0-triangle move is similar. Here, two pentachora, are identified along two tetrahedra to form a pillow around a common triangle, and flattens them to form three tetrahedra.

2.3 4-Manifolds

Let $\mathcal{T}$ be a 4-manifold triangulation. For the *ring of coefficients* $\mathbb{Z}$, the *group of p-chains*, $0 \leq p \leq 4$, denoted $C_p(\mathcal{T}, \mathbb{Z})$, of $\mathcal{T}$ is the group of formal sums of p-dimensional faces with $\mathbb{Z}$ coefficients. The *boundary operator* is a linear operator $\partial_p : C_p(\mathcal{T}, \mathbb{Z}) \to C_{p-1}(\mathcal{T}, \mathbb{Z})$ such that $\partial_p \sigma = \partial_p\{v_0, \cdots, v_p\} = \sum_{j=0}^{p}\{v_0, \cdots, \widehat{v_j}, \cdots, v_p\}$, where σ is a face of $\mathcal{T}$, $\{v_0, \ldots, v_p\}$ represents σ as a face of a pentachoron of $\mathcal{T}$ in local vertices $v_0, \ldots, v_p$, and $\widehat{v_j}$ means v_j is deleted from the list. Denote by $Z_p(\mathcal{T}, \mathbb{Z})$ and $B_{p-1}(\mathcal{T}, \mathbb{Z})$ the kernel and the image of ∂_p respectively. Observing $\partial_p \circ \partial_{p+1} = 0$, we define the *p-th homology group* $H_p(\mathcal{T}, \mathbb{Z})$ of $\mathcal{T}$ by the quotient $H_p(\mathcal{T}, \mathbb{Z}) = Z_p(\mathcal{T}, \mathbb{Z})/B_p(\mathcal{T}, \mathbb{Z})$. Each homology group H_p is a finitely generated $\mathbb{Z}$-module, and is a topological invariant of the manifold triangulated by $\mathcal{T}$. We write $\beta_p = \dim(H_p(\mathcal{T}, \mathbb{Z}))$ for the dimension of (the free part of) H_p. Informally, $H_p(\mathcal{T}, \mathbb{Z})$, $0 \leq p \leq 4$, of a triangulation $\mathcal{T}$ counts the number of "p-dimensional holes" in $\mathcal{T}$. For a more thorough introduction to homology theory see [25].

Let X be a closed, oriented 4-manifold. Representatives of classes in $H_2(X; \mathbb{Z})$ generically intersect in a finite number of points (possibly after isotoping them into transverse position). The *intersection form* Q_X of X, is the symmetric, unimodular, bilinear form defined by

$$Q_X : H_2(X; \mathbb{Z}) \times H_2(X; \mathbb{Z}) \to \mathbb{Z}, \quad Q_X(\alpha, \beta) = S_\alpha \cdot S_\beta := \sum_{S_\alpha \cap S_\beta} \pm 1,$$

where S_α, and S_β are 2-chains representing the classes $\alpha, \beta \in H_2(X; \mathbb{Z})$. If X is smooth, S_α, and S_β can be chosen to be oriented surfaces embedded in X [23, Proposition 1.2.3].

A landmark result of 4-manifold topology is the following classification result for simply connected (i.e. having trivial fundamental group[2]) topological 4-manifolds due to Freedman.

▶ **Theorem 3** (Freedman [21]). *For every symmetric, unimodular, bilinear form Q, there exists a closed simply connected topological 4-manifold X such that $Q_X = Q$. If Q is even, this manifold is unique up to homeomorphism. If Q is odd then are exactly two different homeomorphism types of manifolds with the given intersection form, however only at most one of these admits a PL structure.*

Shortly after Freedman's result, Donaldson showed the following equally important result.

▶ **Theorem 4** (Donaldson [19]). *The symmetric unimodular bilinear form $\oplus_m[+1]$ is the only positive definite form that can be realised as the intersection form of a PL 4-manifold.*

The results by Freedman and Donaldson (together with Serre's algebraic classification of indefinite forms) imply the following key result.

▶ **Theorem 5.** *Two simply connected PL 4-manifolds are homeomorphic if and only if their intersection forms have the same rank, signature, and parity.*

▶ Housekeeping 2. All manifolds are assumed to be PL/smooth, closed, connected, and orientable, unless explicitly stated otherwise.

▶ **Example 6.** The 4-sphere $\mathbb{S}^4$ has no 2-homology and so $Q_{\mathbb{S}^4} = \varnothing$. The complex projective plane, $\mathbb{C}P^2$, has $Q_{\mathbb{C}P^2} = [+1]$, and the oppositely oriented manifold $\overline{\mathbb{C}P^2}$ has $Q_{\overline{\mathbb{C}P^2}} = [-1]$. $S^2 \times S^2$ has intersection form $Q_{S^2 \times S^2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

[2] The *fundamental group* of a manifold X, denoted $\pi_1(X)$, describes closed paths in X up to homotopy, with the group operation of path concatenation.

2.4 Handle Decompositions and Kirby Diagrams

In the smooth setting, we primarily work with 4-manifolds via *handle decompositions*. Let X, X' be two smooth 4-manifolds. We say X is obtained from X' by attaching a (4-dimensional) k-handle, denoted $X = X' \cup_\varphi h^k$, if there is an embedding $\varphi : S^{k-1} \times D^{4-k} \to \partial X'$ such that X is of the form $X = \left[X' \sqcup D^k \times D^{4-k}\right] / \varphi(x) \sim x$, where D^k denotes the closed k-disk ($k \in \{1, 2, 3, 4\}$). There always exists a Morse function $f : X \to \mathbb{R}$ inducing a handle decomposition $X = \bigcup_{i=0}^{n} X_i$ with $\varnothing = X_{-1} \subset X_0 \subset \cdots \subset X_n = X$ in which X_i is obtained from X_{i-1} by attaching i-handles [31]. Since X is closed and connected, it can be assumed that there is a single 0- and 4-handle. Hence a closed 4-manifold X is obtained from B^4 by attaching 1-, 2-, and 3-handles and finally capping off with another B^4. Given such a handle decomposition, we depict X by drawing the attaching regions of the handles in the boundary of the 0-handle ($\partial B^4 \cong \mathbb{S}^3 \cong \mathbb{R}^3 \cup \{*\}$). By a result of Laudenbach and Poenaru [29], 3- and 4-handles attach uniquely up to diffeomorphism, and so it suffices to understand how the 1- and 2-handles attach. We depict 1-handles by "dotted" unknots (see Section 5.4 of [23] for details of this notation). 2-handles are attached via a map of the form $S^1 \times D^2 \to S^3$. These maps are determined up to isotopy by (i) an embedding $S^1 \times \{0\} \to S^3$ (i.e. a *knot*) and (ii) a choice of normal vector field on the knot. Classes of such vector fields are in bijection with the integers [23]. As such, we draw a 2-handle as a knot decorated with an integer. A decorated link diagram of this form – dotted unknots and integer decorated links – is called a *Kirby diagram* and gives a combinatorial encoding of a closed 4-manifold up to diffeomorphism. Figure 2 is an example of a typical Kirby diagram.

3 Topological Classification

We describe a classification of the triangulations in the census up to topological homeomorphism. This serves as a stepping board to then carry out the PL classification. We start by grouping the triangulations of the census by their homology groups. Since all manifolds under consideration are closed and orientable, we omit $H_0(X; \mathbb{Z}) \cong H_4(X; \mathbb{Z}) \cong \mathbb{Z}$ from the homology vector, i.e. we simply write $(H_1(X; \mathbb{Z}), H_2(X; \mathbb{Z}), H_3(X; \mathbb{Z}))$. In a second step, and in the case of simply connected triangulations, we use *Regina*'s built-in intersection form routine and Theorem 5 to split these groups further.

For the remaining triangulations, we simply guess their topological types, build triangulations of these manifolds (using *Katie*, see Section 4.4, and built-in functionality of *Regina*), and compare them to the census manifolds using Pachner moves (that is, we establish a PL homeomorphism). Altogether, this leads to the following classification, summarised in Tables 1, 2, and 3 (note that the content of the fourth column refers to work done in Section 4). In Table 2, we begin to see the appearance of S^1-bundles over *lens spaces*, denoted $L(p, q)$, which are important class of 3-manifold obtained by gluing two solid tori together (see [34] for more).

Table 1 Topological classification of the closed orientable 2-pentachoron census.

# Pentachora	4-Manifold	# Triangulations	# PL Classes
2	$\mathbb{S}^4$	6	1
	$S^3 \times S^1$	2	1

Table 2 Topological classification of the closed orientable 4-pentachoron census.

# Pentachora	4-Manifold	# Triangulations	# PL Classes
4	$\mathbb{S}^4$	647	≤ 2
	$S^3 \times S^1$	126	1
	$\mathbb{C}P^2$	4	1
	$S^3 \times S^1 \# \mathbb{C}P^2$	3	1
	$\mathbb{R}P^3 \times S^1$	1	1
	$L(3,1) \times S^1$	1	1
	$L(3,1) \tilde{\times} S^1$	2	1

Rational Homology 4-Spheres with Finite Fundamental Group

Before giving the table for 6-pentachora, we first define a particular family of 4-manifolds. Let $R(n)$ denote the 4-manifold given by the Kirby diagram in Figure 2, in which there are n strands wrapping around the 1-handle.

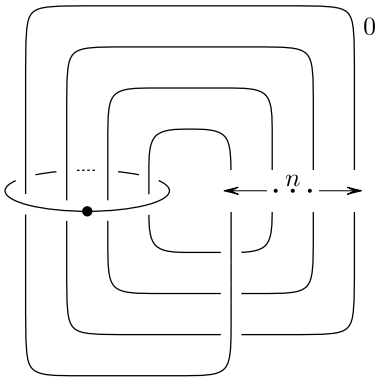


Figure 2 Kirby diagram of the rational ball $R(n)$.

$R(n)$ is a *rational homology ball*. Let $QS^4(n)$ denote the double of $R(n)$, that is $QS^4(n) := DR(n) = R(n) \cup_{\mathrm{id}} R(n)$. $QS^4(n)$ is then a *rational homology 4-sphere* with $\pi_1(QS^4(n)) \cong \mathbb{Z}_n$ and homology vector $(\mathbb{Z}_n, \mathbb{Z}_n, 0)$. By using a combination of *Katie*, Up-Side-Down-Simplify (see Section 4), and *Regina*, we obtain representative triangulations of $QS^4(n)$ for $n \in \{2, 3\}$, each with six pentachora respectively.

▶ Remark 7. We note that the topological type of $QS^4(n)$ was not identified in [20].

4 The PL classification algorithm

In this section we present our search heuristic to establish piecewise linear homeomorphisms between large quantities of triangulated manifolds conjectured to be in the same PL class. We first start by going over some subroutines before sketching the algorithm as a whole.

4.1 Important subroutines

Up-Side-Down-Simplify (USDS): This subroutine is the heart of our algorithm. It is relatively easy to describe, but details are very important. We point out that it is difficult to design a method which is efficient for both triangulations that can trivially be merged

Table 3 Topological classification of the closed orientable 6-pentachoron census.

# Pentachora	4-Manifold	# Triangulations	# PL Classes
6	$\mathbb{S}^4$	405 188	≤ 4
	$S^3 \times S^1$	29 124	1
	$\mathbb{C}P^2$	4 423	≤ 3
	$S^2 \times S^2$	5	1
	$S^2 \tilde{\times} S^2$	7	1
	$\#_2\mathbb{C}P^2$	8	1
	$S^3 \times S^1 \# \mathbb{C}P^2$	1 477	1
	$S^3 \times S^1 \#_2 \mathbb{C}P^2$	6	1
	$\mathbb{R}P^3 \times S^1$	42	1
	$L(3,1) \times S^1$	55	1
	$L(3,1) \tilde{\times} S^1$	64	1
	$L(4,1) \times S^1$	3	1
	$L(4,1) \tilde{\times} S^1$	3	1
	$L(5,2) \times S^1$	1	1
	$L(5,2) \tilde{\times} S^1$	1	1
	$QS^4(2)$	84	≤ 2
	$QS^4(3)$	4	1

with a different class, as well as pathological cases needing millions, if not billions of moves to escape a local area of the Pachner graph (apart from the triangulation presented in Section 5 resisting classification altogether, there are four more 4-pentachoron triangulations requiring a large number of moves before being classified as standard). Extensive research has been done to find effective strategies to compare PL homeomorphism types for triangulated 4-manifolds, see Section 1 for a detailed discussion. At least in the case of the census of small 4-manifold triangulations, the below strategy yields the best results on a consistent basis.

Our heuristic is a slight variation of a standard biased Markov chain-style random walk through the Pachner graph of a triangulation. We bias the sizes of the triangulations visited by the random walk around a fixed target size of $\hat{n}$ pentachora (in our calculations, we used $8 \leq \hat{n} \leq 12$). If the current state of the random walk has more than $\hat{n}$ pentachora, an exponential penalty is imposed on choosing a Pachner move further increasing the size of the state. The same is true for a current state of smaller size than $\hat{n}$ and moves further reducing its size.

The core idea of our approach lies in the type of moves we choose to increase and decrease the sizes of our triangulations: we use 2-4-Pachner moves to increase the size of a triangulation, but 2-0- edge- and triangle-moves to reduce the size of a triangulation. Moreover, we use standard 3-3-Pachner moves to change triangulations while keeping their f-vector constant.

Empirical evidence suggest, that this approach mixes triangulations much faster than by just using standard Pachner moves. This approach has already been used very successfully by the first author [7] in the search of minimal triangulations. The method used there is even simpler in structure than the one presented below.

Our base method has four input parameters that stay fixed for the duration the method is run: *(a)* A set probability $x \in [0,1]$ to decide whether a 3-3-move or some other move is performed, *(b)* a target size $\hat{n}$ for triangulations to be visited in the random walk, *(c)* a parameter $\alpha \in \mathbb{R}$ determining the severity of the penalty for sampling a triangulation of size away from $\hat{n}$, and *(d)* a number of steps $s \in \mathbb{Z}$.

Heuristic 1

Setup:

- $x \in [0,1]$, $\alpha \in \mathbb{R}$, $\hat{n} \in \mathbb{Z}$, $s \in \mathbb{Z}$

Main Loop:

- Set $\mathcal{T} = \mathcal{T}_0$
- For each step $1 \leq i \leq s$ and while $|\mathcal{T}| \neq n$:
  1. Update $\mathcal{T} = \mathcal{T}'$
  2. Update $\beta = \frac{e^{\alpha(\hat{n}-|\mathcal{T}|)}}{1+e^{\alpha(\hat{n}-|\mathcal{T}|)}} \in [0,1]$
  3. Sample $u \in \mathcal{U}([0,1])$
  4. Case $u > x$: if available, perform 3-3-move on $\mathcal{T}$ to return triangulation $\mathcal{T}'$
  5. Sample $v \in \mathcal{U}([0,1])$
  6. If $v > \beta$: if available, perform 2-0-edge- or triangle-move on $\mathcal{T}$ to return triangulation $\mathcal{T}'$
  7. Perform 2-4-move on $\mathcal{T}$ to return triangulation $\mathcal{T}'$
- If $\mathcal{T}$ and $\mathcal{T}_0$ are in different classes, merge classes

Adjust-Vertex-Number: This subroutine takes a triangulated 4-manifold $\mathcal{T}_0$ as input, and outputs a PL homeomorphic triangulation with v vertices.

If $\mathcal{T}_0$ has fewer than v vertices, perform 1-5-moves $v - f_0(\mathcal{T}_0)$ times to obtain a v-vertex triangulation. If, on the other hand, the input triangulation has more than v vertices, we randomly perform edge collapses. If not enough edge collapses are available, we run USDS in between until they become available. It is worthwhile noting that the latter case has no guarantee to terminate in general, but does very quickly in practice.

4.2 The algorithm

Our implementation is based on establishing PL-homeomorphisms using Pachner's theorem [33], i.e. by describing sequences of Pachner moves transforming one triangulation into another. Our computations are organised in a Union-Find structure: at any step of our calculations, every triangulation is associated to a class of triangulations for which pairwise PL homeomorphisms have already been established. Each class has a unique representative triangulation. If a sequence of Pachner moves is found turning a triangulation from one class into a triangulation from another class, both classes are merged with the representative of the latter class, and we continue. This way, if the task is to classify N triangulations, we only need to establish $O(N)$, rather than $O(N^2)$ PL homeomorphisms.

We start with a large number of triangulations $\mathcal{T}_1, \ldots, \mathcal{T}_N$. Here, we assume that all triangulations have the same number of n pentachora, but this is not necessary for the algorithm to work. We furthermore assume that we have already established that all triangulations $\mathcal{T}_i$, $1 \leq i \leq N$, are homeomorphic. Again, this is not a necessary requirement for the algorithm to work, but we require, however, that all triangulations have already been tested to have the same Euler characteristic m.

For n-pentachora triangulations of Euler characteristic m, it is a consequence of Equations (1)–(3) that their f-vector is determined by their number of vertices. We have the following algorithm.

Main Algorithm

1. Initialise the UNION-FIND structure with N classes $\{\mathcal{T}_i\}$, $1 \leq i \leq N$ of size 1.
2. Fix $v = \min\{f_0(\mathcal{T}_i) | 1 \leq i \leq N\}$, the smallest number of vertices found in our list.
3. For all $1 \leq i \leq N$:
   a. Use ADJUST-VERTEX-NUMBER to produce a v-vertex triangulation $\mathcal{T}'$
   b. Use USDS to connect $\mathcal{T}'$ to $\mathcal{T}_j$, $f_0(\mathcal{T}_j) = v$, $f_4(\mathcal{T}_j) = n$, from the list
   Once this step is complete, every PL class has a v-vertex representative.
4. For all classes, run USDS to connect its representative to a new class. Use a moderate time-out threshold to skip hard cases.
5. As long as there are still more than K classes, goto **6.**
6. For remaining classes, run USDS with relaxed parameters.

4.3 Implementation, timings, and remarks

Our implementation of the algorithm described in this section is available at `https://github.com/raburke/Dim4Census`.

Our guiding principle for the implementation is to minimise human intervention. We point out that we can modify arbitrarily many triangulations in parallel until they can be merged with an existing component of the UNION-FIND structure. This parallelises the bottleneck of the computation for large censuses of largely easy-to-handle triangulations. When dealing with pathologically difficult combinatorial structures, running different random walks on the same triangulation may lead to similar speed-ups. However, for the 6-pentachoron census with its $\approx 400,000$ triangulations, parallelisation is not crucial and we hence defer its implementation to future work.

For some indications of running times, we ran our algorithm on a laptop with an 11th Gen Intel i7 processor and 32GB of RAM, with input the 405 188 6-pentachoron triangulations homeomorphic to S^4. We obtained the following timings for first running ADJUST-VERTEX-NUMBER (referred to as *Step 1*), then running USDS until only 10 connected components are left over (*Step 2*), and then the times to merge any additional component until we reduce to 6 connected components, which is when we stopped our calculations.

Table 4 Sample running times of the Main Algorithm with input 405 188 6-pentachoron 4-spheres.

Step 1 (s)	Step 2 (s)	9 comp (s)	8 comp (s)	7 comp (s)	6 comp (s)	Total time (s)
13 686	10 752	88	4 787	11 008	4 987	45 308
14 061	10 306	373	1 097	335	1 143	27 315
14 170	9 264	365	451	464	3 754	28 468
14 182	9 359	1 116	1 772	99	3 772	30 300
14 142	9 791	405	811	1 598	7 961	34 708
13 212	9 342	302	48	3 100	3 155	29 159

Achieving 5 connected components usually does not take much longer (with 1 611 and 18 929 additional seconds in two of the six runs summarised above). Achieving four connected components takes around a week, as we were able to observe on multiple occasions. The algorithm never achieved 3 connected components. Running the complete algorithm in parallel on 100+ cores may produce additional results. Alternatively, exhaustive enumeration on a larger machine with more memory may lead to additional merges of connected components.

Table 5 Sample running times of PL classification for selected manifolds.

Manifold	Step 1 (s)	Step 2 (s)	1 comp. (s)	Total time (s)
$S^3 \times S^1$	2480	119	51	2650
$S^2 \tilde{\times} S^2$	115	0	4 514	4 629
$\#_2\mathbb{C}P^2$	27	0	1 531	1 558
$S^3 \times S^1 \# \mathbb{C}P^2$	1 060	140	42	1 242
$S^3 \times S^1 \#_2 \mathbb{C}P^2$	0	0	62 022	62 022

PL classification of other topological types was achieved through a combination of exhaustive enumeration and our main algorithm. Relevant timings are summarised below.

4.4 *Katie* and PL classification results

As discussed in Section 4, our PL classification algorithm determines whether two given triangulations are PL homeomorphic. What is missing from this algorithm is a reference triangulation for which its PL homeomorphism type is known.

For this we use the software tool *Katie* [6, 7], developed by the first author. *Katie* is based on an algorithm due to Casali and Cristofori [18]. It takes as input a Kirby diagram and produces a triangulation of the associated PL 4-manifold. For each of the topological types in Tables 1–3, we start with the Kirby diagram representing its canonical PL-type, and compare the resulting triangulation to the triangulations in the census.

For all but three topological types, all triangulations homeomorphic to a given 4-manifold, are pairwise PL-homeomorphic. The three exceptions are $\mathbb{S}^4$, $\mathbb{C}P^2$, and the $\mathbb{Q}$-homology sphere $QS^4(2)$, cf. Table 3 for which we find 4, 3, and 2 components respectively.

As already documented in Section 4.3, our algorithm reliably takes all 405 188 triangulations homeomorphic to $\mathbb{S}^4$ from the 6-pentachoron census and classifies them into around 5 PL classes within a day of computation time. Some of the remaining classes have remarkable combinatorial properties, which are discussed in more detail in Section 5. These remaining classes are certified to be impossible to connect to other classes using an exhaustive enumeration and traversal of the Pachner graph with an excess height of 6 (that is, only triangulations with number of pentachora up to $6 + 6 = 12$ are considered).

In some contrast to the triangulations homeomorphic to $\mathbb{S}^4$, we can connect all but four of the 29 124 triangulations homeomorphic to $S^3 \times S^1$ using exhaustive simplification and traversal of the Pachner graph, with an excess height of at most four. The remaining four can be confirmed to be standard $S^3 \times S^1$s by (i) retriangulating to show they were all PL-homeomorphic to each other, and then (ii) traversing the Pachner graph with an excess height of at most 6. Alternatively, using our algorithm, all 29 124 triangulations homeomorphic to $S^3 \times S^1$ can be connected within an hour of computation time, see Section 4.3.

This difference in behaviour is noteworthy since $\mathbb{S}^4$ and $S^3 \times S^1$ are the only two 4-manifolds which can be triangulated using only 2 pentachora.

5 Pathological triangulations and future work

5.1 Combinatorial obstructions

Many of the triangulations resisting classification contain a particular 2-pentachoron subcomplex which we discuss briefly in this section. The complex in question, denoted C, has isomorphism signature `cHIbbb0bRbpb`, f-vector $(1, 1, 5, 6, 2)$, and has boundary a 2-tetrahedron solid torus. Considered in isolation, C is not a triangulation of a manifold (its

vertex link is a pinched genus-4 handlebody, and the link of its edge is a thrice-punctured sphere). However, C can be uniquely closed into an ideal triangulation of a 4-manifold: the unique[3] 2-pentachoron triangulation of a Cappell–Shaneson 2-knot complement [5]. This is a triangulation of the complement of a "knotted" 2-sphere in the 4-sphere, $S^4 - \nu S^2$.

The "cut-open" Cappell–Shaneson complex C embeds in a topological 4-ball B^4_C, where B^4_C is obtained from the unique 4-pentachoron 4-sphere still resisting classification (`eAMPcaabcddd+aoa+aAa8aQara`) by ungluing a single facet identification. We denote this 4-sphere by S^4_C. The isomorphism signature of B^4_C is `eGzMkabcdddcaGa8aAa0awa`. Its fundamental group presentation, as simplified by *Regina* [12] or GAP's `SimplifiedFpGroup` [24], is $\langle a, b \mid a^3b^3a^{-2}b^{-2}, a^{-3}b^{-1}a^5b^2\rangle$; moreover, GAP certifies that this is a presentation of the trivial group. It seems natural to assume that S^4_C relates – in some way – to the Cappell–Shaneson spheres, formerly conjectured to be exotic [17], and shown to be standard in [1, 22].

It is worthwhile to note that, when B^4_C is attached to a larger triangulation, the B^4_C subcomplex remains intact through very large flip sequences. In this sense, C, or B^4_C, is a potential combinatorial obstruction to simplify triangulations it appears within (similar to the Dunce hat being a combinatorial obstruction to collapsing a contractible space). Additionally, the presentation of the fundamental group of B^4_C is also challenging to simplify, but possible using the simplification heuristic for finitely presented groups implemented in GAP [24].

We have the following results and conjectures concerning B^4_C and S^4_C.

▶ **Conjecture 8.** *B^4_C is PL-homeomorphic to the standard 4-ball.*

We conjecture that B^4_C supports the standard PL structure because of its small size. Another reason is its relation to a Cappell-Shaneson sphere that is known to be standard [1, 22], see above.

▶ **Proposition 9.** *If there exists a sequence of Pachner moves connecting S^4_C to a triangulation of the standard PL 4-sphere, then this sequence must contain at least one triangulation with at least 12 pentachora.*

Proof. Running an exhaustive enumeration of all local move sequences starting from S^4_C and visiting triangulations up to 10 pentachora (i.e., an excess height of 6) does not connect S^4_C to any other 4-pentachoron triangulation of the 4-sphere. ◀

To conclude this article, we detail how to extend the observations made in this section and, more broadly, in this article to two research directions for future work.

5.2 The 8-Pentachoron Census

Generating a census (of any type) with n simplicies consists of two stages: First, enumerate all possible dual graphs with n nodes; then, for each graph, test all possible $5n/2$ facet gluings and retain those that yield valid triangulations. For 2, 4, and 6 pentachora there are 3, 26, and 638 such graphs, leading to 8, 784, and 440 495 valid 4-manifold triangulations, respectively.

We can deduce from this that only a very small fraction of possible gluings give rise to a triangulation of a 4-manifold, implying that various optimisations are possible (and needed) to build the census even only up to 6 pentachora (cf. [9, 10, 13] for an overview

[3] Verifiable via a very short exhaustive search of the ideal 2-pentachoron census.

of such optimisations in the 3-dimensional setting). In the case of 8 pentachora however, existing optimisations are no longer sufficient. Consequently, entirely new algorithms are in development to complete this next step of the 4-dimensional census. A completed 8-pentachoron census will come with new challenges for classifying their PL-types. Undoubtedly, more interesting pathological triangulations will emerge and in greater numbers. This is work in progress.

5.3 2-Knot Complements

The Cappell – Shaneson subcomplex C, discussed at the beginning of this section is only one of many such topological objects that can be described using only a small number of pentachora. In theory, each of them can lead to difficult triangulations of 4-balls and 4-spheres, related to B^4_C and S^4_C. Such examples are very useful for constructions in low-dimensional topology, or to benchmark future iterations of search methods. Given such a collection of examples, the major challenge is to either rigorously quantify how difficult they are to simplify, or to relate them to former or present potential counterexamples to S4PC.

References

1 S. Akbulut. Cappell-Shaneson homotopy spheres are standard. *Ann. of Math. (2)*, 171(3):2171–2175, 2010. doi:10.4007/annals.2010.171.2171.

2 E. G. Altmann and J. Spreer. Sampling triangulations of manifolds using Monte Carlo methods. arXiv:2310.07372, 2024. Preprint, to appear in Exp. Math. Acceptance date: 8 Nov 2024.

3 A. Björner and F. H. Lutz. Simplicial manifolds, bistellar flips and a 16-vertex triangulation of the Poincaré homology 3-sphere. *Experiment. Math.*, 9(2):275–289, 2000. doi:10.1080/10586458.2000.10504652.

4 R. Budney and B. A. Burton. A census of small triangulated 4-manifolds, 2025+. In preparation.

5 R. Budney, B. A. Burton, and J. Hillman. Triangulating a Cappell-Shaneson knot complement. *Math. Res. Lett.*, 19(5):1117–1126, 2012. doi:10.4310/MRL.2012.v19.n5.a12.

6 R. A. Burke. Katie, 2024. Version 2.0. URL: https://github.com/raburke/Comp4Top.

7 R. A. Burke. Practical Software for Triangulating and Simplifying 4-Manifolds. 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 29:1–29:23, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.29.

8 Rhuaidi Antonio Burke, Benjamin A. Burton, and Jonathan Spreer. raburke/Dim4Census. Software, swhId: swh:1:dir:ee5ac0c76fdef9983c5de8a0be93f7684dd9a796 (visited on 2025-05-30). URL: https://github.com/raburke/Dim4Census, doi:10.4230/artifacts.23281.

9 B. A. Burton. Efficient enumeration of 3-manifold triangulations. *Austral. Math. Soc. Gaz.*, 31(2):108–114, 2004.

10 B. A. Burton. Enumeration of non-orientable 3-manifolds using face-pairing graphs and union-find. *Discrete Comput. Geom.*, 38(3):527–571, 2007. doi:10.1007/s00454-007-1307-x.

11 B. A. Burton. The Pachner graph and the simplification of 3-sphere triangulations. In *Proc. 27th ACM Symp. Comput. Geom. (SoCG 2011), Paris, France, June 13–15, 2011*, pages 153–162. ACM, New York, 2011. doi:10.1145/1998196.1998220.

12 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2022. Version 7.2. URL: https://regina-normal.github.io.

13 B. A. Burton and W. Pettersson. An Edge-Based Framework for Enumerating 3-Manifold Triangulations. 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 270–284, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SOCG.2015.270.

14 B. A. Burton and J. Spreer. Computationally proving triangulated 4-manifolds to be diffeomorphic, 2014. arXiv:1403.2780.

15 S. S. Cairns. Triangulation of the manifold of class one. *Bull. Amer. Math. Soc.*, 41(8):549–552, 1935. doi:10.1090/S0002-9904-1935-06140-3.

16 S. S. Cairns. A simple triangulation method for smooth manifolds. *Bull. Amer. Math. Soc.*, 67:389–390, 1961. doi:10.1090/S0002-9904-1961-10631-9.

17 S. E. Cappell and J. L. Shaneson. Some new four-manifolds. *Ann. of Math. (2)*, 104(1):61–72, 1976. doi:10.2307/1971056.

18 M. R. Casali and P. Cristofori. Kirby diagrams and 5-colored graphs representing compact 4-manifolds. *Rev. Mat. Complut.*, 36(3):899–931, 2023. doi:10.1007/s13163-022-00438-x.

19 S. K. Donaldson. An application of gauge theory to four-dimensional topology. *J. Differential Geom.*, 18(2):279–315, 1983. URL: http://projecteuclid.org/euclid.jdg/1214437665.

20 A. Pérez-Cerezo Flohr. Inspection of a census of 4-manifolds, 2024. Master's Thesis.

21 M. H. Freedman. The topology of four-dimensional manifolds. *J. Differential Geometry*, 17(3):357–453, 1982. URL: http://projecteuclid.org/euclid.jdg/1214437136.

22 R. E. Gompf. More Cappell-Shaneson spheres are standard. *Algebr. Geom. Topol.*, 10(3):1665–1681, 2010. doi:10.2140/agt.2010.10.1665.

23 R. E. Gompf and A. I. Stipsicz. *4-manifolds and Kirby calculus*, volume 20 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1999. doi:10.1090/gsm/020.

24 The GAP Group. Gap – groups, algorithms, and programming, version 4.13.1, 2024. URL: https://www.gap-system.org.

25 A. Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

26 M. W. Hirsch and B. Mazur. *Smoothings of piecewise linear manifolds*, volume No. 80 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ; University of Tokyo Press, Tokyo, 1974.

27 M. Joswig, D. Lofano, F. H. Lutz, and M. Tsuruga. Frontiers of sphere recognition in practice. *J. Appl. Comput. Topol.*, 6(4):503–527, 2022. doi:10.1007/s41468-022-00092-8.

28 P. B. Kronheimer and T. S. Mrowka. Recurrence relations and asymptotics for four-manifold invariants. *Bull. Amer. Math. Soc. (N.S.)*, 30(2):215–221, 1994. doi:10.1090/S0273-0979-1994-00492-6.

29 F. Laudenbach and V. Poénaru. A note on 4-dimensional handlebodies. *Bull. Soc. Math. France*, 100:337–344, 1972. URL: http://www.numdam.org/item?id=BSMF_1972__100__337_0.

30 A. Markov. The insolubility of the problem of homeomorphy. *Dokl. Akad. Nauk SSSR*, 121:218–220, 1958.

31 J. Milnor, M. Spivak, and R. Wells. *Morse Theory. (AM-51), Volume 51*. Princeton University Press, 1969. URL: http://www.jstor.org/stable/j.ctv3f8rb6.

32 J. Munkres. Obstructions to the smoothing of piecewise-differentiable homeomorphisms. *Ann. of Math. (2)*, 72:521–554, 1960. doi:10.2307/1970228.

33 U. Pachner. Konstruktionsmethoden und das kombinatorische Homöomorphieproblem für Triangulationen kompakter semilinearer Mannigfaltigkeiten. *Abh. Math. Sem. Univ. Hamburg*, 57:69–86, 1987. doi:10.1007/BF02941601.

34 N. Saveliev. *Lectures on the Topology of 3-Manifolds: An Introduction to the Casson Invariant*. De Gruyter Textbook. Walter de Gruyter & Co., Berlin, 2nd edition, 2012. doi:10.1515/9783110250367.

35 D. M. Y. Sommerville. The relations connecting the angle-sums and volume of a polytope in space of n dimensions. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(770):103–119, 1927. URL: http://www.jstor.org/stable/94871.

36 M. Tsuruga and F. H. Lutz. Constructing complicated spheres, 2013. arXiv:1302.6856.

37 J. H. C. Whitehead. On C^1-complexes. *Ann. of Math. (2)*, 41:809–824, 1940. doi:10.2307/1968861.

Sparsification of the Generalized Persistence Diagrams for Scalability Through Gradient Descent

Mathieu Carrière
Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France

Seunghyun Kim
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Woojin Kim
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Abstract

The generalized persistence diagram (GPD) is a natural extension of the classical persistence barcode to the setting of multi-parameter persistence and beyond. The GPD is defined as an integer-valued function whose domain is the set of intervals in the indexing poset of a persistence module, and is known to be able to capture richer topological information than its single-parameter counterpart. However, computing the GPD is computationally prohibitive due to the sheer size of the interval set. Restricting the GPD to a subset of intervals provides a way to manage this complexity, compromising discriminating power to some extent. However, identifying and computing an effective restriction of the domain that minimizes the loss of discriminating power remains an open challenge.

In this work, we introduce a novel method for optimizing the domain of the GPD through gradient descent optimization. To achieve this, we introduce a loss function tailored to optimize the selection of intervals, balancing computational efficiency and discriminative accuracy. The design of the loss function is based on the known erosion stability property of the GPD. We showcase the efficiency of our sparsification method for dataset classification in supervised machine learning. Experimental results demonstrate that our sparsification method significantly reduces the time required for computing the GPDs associated to several datasets, while maintaining classification accuracies comparable to those achieved using full GPDs. Our method thus opens the way for the use of GPD-based methods to applications at an unprecedented scale.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Nonconvex optimization

Keywords and phrases Multi-parameter persistent homology, Generalized persistence diagram, Generalized rank invariant, Non-convex optimization, Gradient descent

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.29

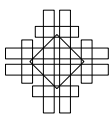
Related Version *Full Version*: `https://arxiv.org/abs/2412.05900` [11]

Supplementary Material *Software (Source Code)*: `https://github.com/L-ebesgue/sparse_GPDs` archived at `swh:1:dir:2147b0ae5a14e4c082811cb721798cb8974fb0b9`

Funding *Mathieu Carrière*: Partially supported by ANR grant "TopModel", ANR-23-CE23-0014, and supported by the French government, through the 3IA Cote d'Azur Investments in the project managed by the National Research Agency (ANR) with the reference number ANR-23-IACL-0001. *Woojin Kim*: Partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (RS-2025-00515946).

Acknowledgements This work stemmed from a conversation between M.C. and W.K. during the Dagstuhl Seminar on "Applied and Combinatorial Topology" (24092), held from February 26 to March 1, 2024. M.C. and W.K. thank the organizers of the Dagstuhl Seminar and appreciate the hospitality of Schloss Dagstuhl – Leibniz Center for Informatics. The authors are also grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

© Mathieu Carrière, Seunghyun Kim, and Woojin Kim;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Persistent homology, a central tool in topological data analysis (TDA), enables the study of topological features in datasets through algebraic invariants. In the classical one-parameter setting, the persistence barcode (or equivalently, the persistence diagram) serves as a complete, discrete, and computationally tractable invariant of a persistence module. Persistent homology can be extended to multi-parameter persistent homology, which provides tools for capturing the topological features of datasets using multiple filtrations instead of just one. However, the transition to multi-parameter persistent homology introduces significant complexity into the algebraic structure of the associated persistence modules [3, 6, 9].

Nevertheless, the generalized persistence diagram (GPD) extends the notion of persistence diagram from the one-parameter to the multi-parameter setting in a natural way [20, 33]. Although the GPD has been extensively studied in terms of stability, discriminating power, computation, and generalizations (see, e.g., [1, 13, 16, 17, 20, 22, 23, 24]), the computational complexity of GPDs remains a major obstacle [19]. The primary challenge arises from the size of their domain: the domain of the complete GPD is either $\mathrm{Int}(\mathbb{R}^d)$, the set of all intervals in $\mathbb{R}^d$, or any appropriately chosen, finite subset of $\mathrm{Int}(\mathbb{R}^d)$ – however, in this case, domains still tend to be enormous to avoid sacrificing the GPD discriminating power [13]. Nevertheless, GPDs are flexible, in the sense that they are still well-defined on *any finite* subdomain $\mathcal{I} \subset \mathrm{Int}(\mathbb{R}^d)$. Even if the subdomain has small size, as the GPD over $\mathcal{I}$ is simply defined as the Möbius inversion of the generalized rank invariant (GRI) over $\mathcal{I}$ (see Definition 1). This allows to control the aforementioned complexity of GPD computation by picking a small, or *sparse*, subdomain $\mathcal{I}$. Moreover, the topological information loss due to using such sparse subdomains can be mitigated by looking for *relevant* intervals, i.e., intervals that are rich in topological content; indeed, even with a substantially small subdomain $\mathcal{I} \subset \mathrm{Int}(\mathbb{R}^d)$, the GPD over $\mathcal{I}$ can still be finer than other traditional invariants of multi-parameter persistence modules, such as the rank invariant (RI) [9]; see [13] for details. However, how to design these subdomains in the "best" way so as to reduce computational cost while maintaining the discriminating power of the GPD is a question that has not been much explored so far.

Sparsification of the GPD via gradient descent. Motivated by the computational challenges of the GPD and the flexibility of this invariant upon selecting subsets of intervals as its domain, we propose a method for *automatically sparsifying GPDs* computed from $\mathbb{R}^2$-persistence modules based on gradient descent. Namely, we consider the following scenario.

Suppose we aim at classifying instances of a given dataset based on their topological features, and we have already computed a set of corresponding persistence modules $\{M_i : \mathbb{R}^2 \to \mathrm{vect}_k\}_{1\leq i\leq t}$ from the dataset, where each persistence module corresponds to an individual data point. We consider the set $\{\mathrm{dgm}^{\mathcal{I}}_{M_i}\}_{1\leq i\leq t}$ of GPDs of M_i over a large set $\mathcal{I}$ of intervals in $\mathbb{R}^2$ (cf. Definition 1). We refer to these as the *full GPDs*.

Let $n \gg 1$ be the cardinality of $\mathcal{I}$, and let m be a sparsification parameter $m \in \mathbb{N}^* = \{1, 2, \ldots\}$, which is typically significantly smaller than n. Let $\binom{\mathrm{Int}(\mathbb{R}^2)}{m}$ denote the set of m-subsets of $\mathrm{Int}(\mathbb{R}^2)$, i.e., $\{\mathcal{J} \subset \mathrm{Int}(\mathbb{R}^2) : |\mathcal{J}| = m\}$. In order to identify an m-subset of intervals in $\mathbb{R}^2$ over which the new GPDs are computed (and subsequently used to classify the persistence modules $\{M_i\}_{1\leq i\leq t}$), we proceed as follows. **Firstly,** we identify a loss function *defined on* $\binom{\mathrm{Int}(\mathbb{R}^2)}{m}$:

$$\begin{array}{rccl} \mathcal{L}_{d_\ell, m, \{M_i\}_{1\leq i\leq t}} : & \binom{\mathrm{Int}(\mathbb{R}^2)}{m} & \to & \mathbb{R} \\ & \mathcal{J} & \mapsto & \sum_{i=1}^{t} d_\ell(\mathrm{dgm}^{\mathcal{J}}_{M_i}, \mathrm{dgm}^{\mathcal{I}}_{M_i}), \end{array} \tag{1}$$

where d_ℓ is an appropriate dissimilarity function. **Secondly,** we search for a minimizer of the loss function. The goal of this search is to identify a subset $\mathcal{J}^*$ of m intervals such that the GPDs of the $\{M_i\}_i$ over this sparse subset $\{\mathrm{dgm}_{M_i}^{\mathcal{J}^*}\}_{1\leq i\leq t}$, the *sparse GPDs*, best approximate their corresponding full counterparts overall. One natural way of searching for a (local) minimizer is through **gradient descent**, starting from a randomly chosen m-subset $\mathcal{J}_{\mathrm{init}}$ of $\mathcal{I}$. To achieve this, the following requirements either must be met or highly desirable:

- **(I)** (Distance) A suitable distance or dissimilarity function d_ℓ, utilized in constructing the loss function above, ideally satisfying certain stability guarantees w.r.t. the *interleaving distance* between persistence modules [12, 25],
- **(II)** (Vectorization) A certain representation of the loss function $\mathcal{L}_{d_\ell,m,\{M_i\}_{1\leq i\leq t}}$ as a map defined on some subset $\mathcal{D}$ of Euclidean space,
- **(III)** (Convexity) Convexity of the subset $\mathcal{D}$,
- **(IV)** (Loss regularity) Lipschitz stability and differentiability of $\mathcal{L}_{d_\ell,m,\{M_i\}_{1\leq i\leq t}}$, and
- **(V)** (Feasibility) Computational feasibility of $\mathcal{L}_{d_\ell,m,\{M_i\}_{1\leq i\leq t}}$ for practical implementation.

Our contributions can be listed according to Items (I)-(V) mentioned above.

Summary of contributions.

- A natural choice for d_ℓ in Item (I) would be to use the *erosion distance* d_{E} [13, 33], a standard metric between GPDs,[1] that is known to be stable under perturbations of input persistence modules w.r.t. the interleaving distance. However, the use of the erosion distance requires GPDs to be defined on the *same* set of intervals (Definition 3) that is *closed under thickening*, which implies the domain must be *infinite*. All these make it difficult to directly utilize d_{E}. Hence, **we introduce the sparse erosion distance $\hat{d}_{\mathbf{E}}$ between GPDs relative to (possibly) *different* interval sets, as an adaptation of $d_{\mathbf{E}}$ (Definition 4, Proposition 6 (ii), and Corollary 7)**.[2]
- Regarding Item (V), we restrict our focus to intervals in $\mathbb{R}^2$ with a small number of minimal and maximal points (Remark 10). Then, if the sparse erosion distance $\hat{d}_{\mathrm{E}}$ is computed between GPDs of the *same* persistence module M, **we prove that $\hat{d}_{\mathbf{E}}$ depends solely on the domains $\mathcal{I},\mathcal{J}$, and *not* on the input persistence module, i.e. $\hat{d}_{\mathbf{E}}((\mathbf{dgm}_M,\mathcal{I}),(\mathbf{dgm}_M,\mathcal{J})) = \hat{d}(\mathcal{I},\mathcal{J})$ (Proposition 6 (iii))**. This fact significantly enhances the tractability of gradient descent as it allows to avoid recomputing the GPDs at every iteration. Moreover, **we derive a closed-form formula for the computation of $\hat{d}(\mathcal{I},\mathcal{J})$ (Theorem 9 (i)).**
- For achieving Items (II)–(IV), we only consider intervals with at most two minimal points and one maximal point. This ensures the existence of natural embeddings of these intervals into Euclidean space $\mathbb{R}^6$, which can be obtained by stacking up the coordinates of the interval middle points (x,y) together with the lengths $a,b,c,d\geq 0$ needed to define the interval lower boundaries (Figure 1). The main advantages of this vectorization method (w.r.t. the other natural ones) are that (1) it allows for **a simple formulation of the loss function (Theorem 9 (ii))**, and (2) its variables are all independent from each other, which makes the implementation of gradient descent easier (Remarks 10

[1] Note that d_{E} is also referred to as a metric between generalized rank invariants (GRIs), e.g., in [13, 21]. Since the GPD and the GRI determine one another (cf. Remark 2 (i) and (iii)), d_{E} can thus also be viewed as a metric between GPDs, as in [33, 36].

[2] Another possibility would be to use bottleneck and Wasserstein distances, however we show in Remark 17 in the full version [11] that they fail to ensure continuity of the loss function.

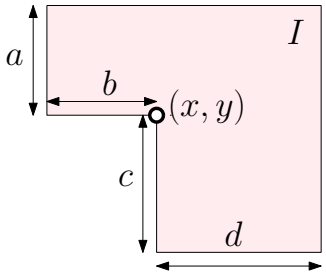


Figure 1 Any $(2,1)$-interval I of $\mathbb{R}^2$, as depicted above, is represented by $\mathbf{v}_I = (x, y, a, b, c, d) \in \mathbb{R}^6$. Any $(1,1)$-interval can also be represented by $(x, y, a, b, c, d) \in \mathbb{R}^6$ with $b = c = 0$.

and 11).[3] Using this vectorization method, **we also prove that $\mathcal{D}$ not only forms a convex subset of Euclidean space $\mathbb{R}^{6m}$ (Proposition 13), but also ensures the Lipschitz stability and almost everywhere differentiability of the loss function $\mathcal{L}_{d_\ell, m, \{M_i\}_{1\leq i\leq t}}$ (Theorem 14 and Proposition 16)**. In fact, we prove that the loss function can actually be realized as

$$\begin{array}{rccl} \mathcal{L}_{\hat{d_{\mathrm{E}}},m}: & \mathcal{D}(\subset \mathbb{R}^{6m}) & \to & \mathbb{R} \\ & \mathbb{v}_{\mathcal{J}} & \mapsto & t \cdot \hat{d}(\mathcal{I}, \mathcal{J}), \end{array} \tag{2}$$

where $\mathbb{v}_{\mathcal{J}}$ is our $6m$-dimensional embedding of $\mathcal{J}$. Since t (the size of the dataset) is a constant, the (local) minimizers of $\mathcal{L}_{\hat{d_{\mathrm{E}}},m}$ coincide with those of the map $t^{-1} \cdot \mathcal{L}_{\hat{d_{\mathrm{E}}},m} : \mathbb{v}_{\mathcal{J}} \mapsto \hat{d}(\mathcal{I}, \mathcal{J})$. Hence, searching for a (local) minimizer of $\mathcal{L}_{\hat{d_{\mathrm{E}}},m}$ is essentially searching for the (locally) best m intervals that represent the domain $\mathcal{I}$ of the full GPDs.

Finally, regarding Item (V) again, and in order to showcase the efficiency of our proposed sparsification method, **we provide numerical experiments on topology-based time series classification with supervised machine learning (Section 5 and Figure 2)**, in which we show that the sparse GPDs $\{\mathrm{dgm}_{M_i}^{\mathcal{J}^*}\}_{1\leq i\leq t}$ can be computed in much faster time than the full GPDs, while maintaining similar or better classification performances from random forests models. Our code is fully available at sparse GPDs.

Comparison with other works. While there exist many existing works that utilize multi-parameter persistent homology to enhance the performance of machine learning models [10, 14, 27, 29, 32, 34, 35, 37], our work takes a different approach as it focuses on methods to mitigate the computational overhead associated with multi-parameter persistence descriptors.

The works most closely related to ours are [29] and [37], which use the RI or GRI for multi-parameter persistence modules in a machine learning context. Firstly, [29] is based on an equivalent representation of the GPDs (computed from rectangle intervals[4]) as *signed measures* [7], which allows to compare GPDs with optimal transport distances, as well as to deploy known vectorization techniques intended for general measures. Secondly, the approach proposed in [37] involves vectorizing the GRIs of 2-parameter persistence modules by evaluating them on intervals with specific shapes called *worms*. Our goal is different: we rather aim at vectorizing and sparsifying the *domains* of the GPDs in order to achieve good

[3] Alternatively, we can consider intervals with one minimal points and at most two maximal points. Either choice ensures that our restricted GPDs are not a weaker invariant than the rank invariant or the signed barcode [7]; see [13, Section 4].

[4] This types of GPDs are often called signed barcodes.

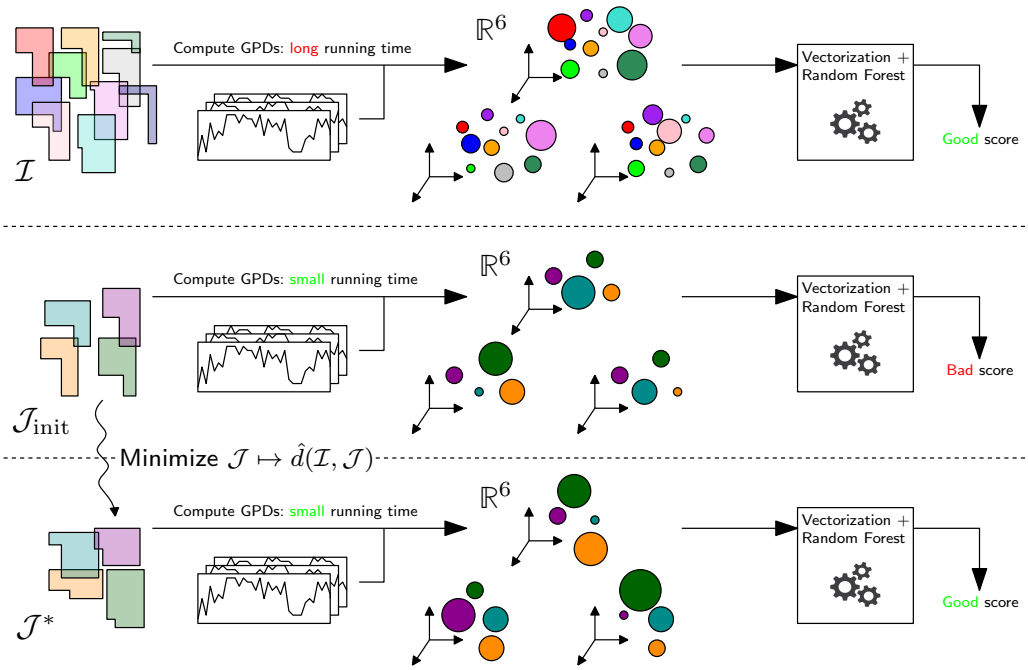


Figure 2 Our pipeline for sparsifying GPDs in the context of time series classification. The sizes of the GPD points in $\mathbb{R}^6$ are proportional to their multiplicities.

performance scores. In fact, the sparsification process that we propose can actually be used complementarily to both [29] and [37] by first sparsifying the set of rectangles or worms before applying their vectorization methods. Note that differentiability properties of both of these approaches *w.r.t. the multi-parameter filtrations* were recently established [32, 34]; in contrast, our work deals with the differentiability of a GPD-based loss function *w.r.t. the interval domains* (while keeping the multi-parameter filtrations fixed).

Organization. Section 2 reviews basic properties of the GPD and GRI. Section 3 introduces the *sparse* erosion distance, clarify its relation to the erosion distance given in [13], and presents its closed-form formula, which is specialized and useful in our setting. Section 4 establishes the Lipschitz stability and differentiability of our loss function. Section 5 presents our numerical experiments. Finally, Section 6 discusses future research directions.

2 Preliminaries

In this article, $P = (P, \leq)$ stands for a poset, regarded as the category whose objects are the elements of P, and for any pair $p, q \in P$, there exists a unique morphism $p \to q$ if and only if $p \leq q$. All vector spaces in this article are over a fixed field k. Let vect_k denote the category of finite-dimensional vector spaces and linear maps over k. A functor $P \to \text{vect}_k$ will be referred to as a **(P-)persistence module**. The **direct sum** of any two P-persistence modules is defined pointwise. Any $M : P \to \text{vect}_k$ is **trivial** if $M(x) = 0$ for all $x \in P$. If a

nontrivial M is not isomorphic to a direct sum of any two nontrivial persistence modules, M is **indecomposable**. Every persistence module is decomposed into a direct sum of indecomposable modules, uniquely determined up to isomorphism [2, 4].

An **interval** I of P is a subset $I \subseteq P$ such that: (i) I is nonempty. (ii) If $p, q \in I$ and $p \leq r \leq q$, then $r \in I$. (iii) I is **connected**, i.e. for any $p, q \in I$, there is a sequence $p = p_0, p_1, \cdots, p_\ell = q$ of elements of I with p_i and p_{i+1} comparable for $0 \leq i \leq \ell - 1$. By $\mathrm{Int}(P)$ we denote the set of all intervals of P.

Given any $I \in \mathrm{Int}(P)$, the **interval module** k_I is the P-persistence module, with

$$(k_I)(p) := \begin{cases} k & \text{if } p \in I \\ 0 & \text{otherwise.} \end{cases}, \qquad k_I(p \leq q) := \begin{cases} \mathrm{id}_k & \text{if } p \leq q \in I \\ 0 & \text{otherwise.} \end{cases}$$

Every interval module is indecomposable [5, Proposition 2.2]. A P-persistence module M is **interval-decomposable** if it is isomorphic to a direct sum $\bigoplus_{j \in J} k_{I_j}$ of interval modules. In this case, the **barcode** of M is defined as the multiset $\mathrm{barc}(M) := \{k_{I_j} : j \in J\}$.

For $p \in P$, let $p^\uparrow$ denote the set of points $q \in P$ such that $p \leq q$. Clearly, $p^\uparrow$ belongs to $\mathrm{Int}(P)$. A P-persistence module is **finitely presentable** if it is isomorphic to the cokernel of a morphism $\bigoplus_{a \in A} k_{a^\uparrow} \to \bigoplus_{b \in B} k_{b^\uparrow}$, where A and B are finite multisets of elements of P.

When P is a connected poset, the **(generalized) rank** of M, denoted by $\mathrm{rank}(M)$, is defined as the rank of the canonical linear map from the limit of M to the colimit of M, which is a nonnegative integer [20]. This isomorphism invariant of P-persistence modules, which takes a single integer value, can be refined into an integer-valued function as follows. Let $\mathcal{I}$ be any nonempty subset of $\mathrm{Int}(P)$. The **generalized rank invariant (GRI) of M over $\mathcal{I}$** is the map $\mathrm{rk}_M^{\mathcal{I}} : \mathcal{I} \to \mathbb{Z}_{\geq 0}$ given by $I \mapsto \mathrm{rank}(M|_I)$, where $M|_I$ is the restriction of M to I [20]. When $\mathcal{I} = \mathrm{Int}(P)$, we denote $\mathrm{rk}_M^{\mathcal{I}}$ simply as rk_M.

The generalized persistence diagram (GPD) of M over $\mathcal{I}$ captures the changes of the GRI values when $I \in \mathcal{I}$ varies. Its formal definition follows.

▶ **Definition 1** ([13]). *The **generalized persistence diagram (GPD) of M over $\mathcal{I}$** is defined as the function $\mathrm{dgm}_M^{\mathcal{I}} : \mathcal{I} \to \mathbb{Z}$ that satisfies:*[5]

$$\text{for all } I \in \mathcal{I}, \quad \mathrm{rk}_M^{\mathcal{I}}(I) = \sum_{\substack{J \in \mathcal{I} \\ J \supseteq I}} \mathrm{dgm}_M^{\mathcal{I}}(J). \tag{3}$$

▶ Remark 2 ([13, Sections 2 and 3] and [7]).

(i) If $\mathcal{I}$ is finite, then $\mathrm{dgm}_M^{\mathcal{I}}$ exists.

(ii) If $\mathrm{dgm}_M^{\mathcal{I}}$ exists, then it is unique.

(iii) If $\mathrm{dgm}_M^{\mathcal{I}}$ exists, then $\mathrm{dgm}_M^{\mathcal{I}}$ and $\mathrm{rk}_M^{\mathcal{I}}$ determine one another.

(iv) (Monotonicity) $\mathrm{rk}_M^{\mathcal{I}}(I) \leq \mathrm{rk}_M^{\mathcal{I}}(J)$ for any pair $I \supseteq J$ in $\mathcal{I}$.

(v) (The GPD generalizes the barcode) Let $\mathcal{I} = \mathrm{Int}(P)$. If M is interval decomposable, then $\mathrm{dgm}_M^{\mathcal{I}}$ exists. In this case, for any $I \in \mathrm{Int}(P)$, $\mathrm{dgm}_M^{\mathcal{I}}(I)$ coincides the multiplicity of I in $\mathrm{barc}(M)$. Also, $\mathrm{dgm}_M^{\mathcal{I}}$ often exists even when M is not interval decomposable.

(vi) If M is a *finitely presentable* $\mathbb{R}^d$-persistence module, then the GPD over $\mathrm{Int}(\mathbb{R}^d)$ exists [13, Theorem C(iii)].

In the rest of the article, every $\mathbb{R}^d$-persistence module M is assumed to be finitely presentable, thus its GPD over $\mathrm{Int}(\mathbb{R}^d)$ exists, and is denoted by dgm_M.

[5] The condition in Equation (3) is a generalization of *the fundamental lemma of persistent homology* [18].

3 Sparse erosion distance between GPDs relative to sampled intervals

We adapt the notion of erosion distance to define a distance between *GPDs relative to (possibly different) sampled intervals*. When comparing the same GPD *with two different sampled intervals*, this distance simplifies to a distance between sampled intervals. This is relevant in our case, as we compare the full and sparse GPDs of the *same* persistence module.

3.1 Sparse erosion distance

In this section, we review the definition of the erosion distance and adapt it to define the sparse erosion distance between *GPDs relative to sampled intervals*.

For $\epsilon \in \mathbb{R}$, the vector $\epsilon(1, \dots, 1) \in \mathbb{R}^d$ will be simply denoted by ϵ whenever there is no risk of confusion. For $I \in \mathrm{Int}(\mathbb{R}^d)$ and $\epsilon \in \mathbb{R}_{\geq 0}$, we consider the *$\epsilon$-thickening* $I^\epsilon := \bigcup_{p \in I} B_\epsilon^\square(p)$ of I, where $B_\epsilon^\square(p)$ stands for the closed ϵ-ball around p w.r.t. the supremum distance, i.e. $B_\epsilon^\square(p) = \{q \in \mathbb{R}^d \mid p - \epsilon \leq q \leq p + \epsilon\}$. See Figure 3. A subset $\mathcal{I} \subset \mathrm{Int}(\mathbb{R}^d)$ is said to be **closed under thickening** if for all $I \in \mathcal{I}$ and for all $\epsilon \in \mathbb{R}_{\geq 0}$, the interval I^ϵ belongs to $\mathcal{I}$.

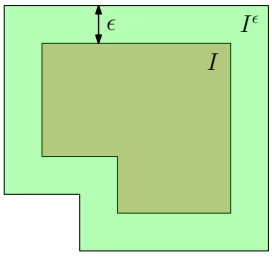


Figure 3 ϵ-thickening I^ϵ.

▶ **Definition 3** ([13, Definition 5.2]). *Let M and N be $\mathbb{R}^d$-persistence modules. Let $\mathcal{I}$ be any subset of* $\mathrm{Int}(\mathbb{R}^d)$ that is closed under thickening. *The* ***erosion distance*** *between* $\mathrm{dgm}_M^{\mathcal{I}}$ *and* $\mathrm{dgm}_N^{\mathcal{I}}$ *(and equivalently between* $\mathrm{rk}_M^{\mathcal{I}}$ *and* $\mathrm{rk}_N^{\mathcal{I}}$ *by Remark 2 (ii)) is*

$$d_{\mathrm{E}}(\mathrm{dgm}_M^{\mathcal{I}}, \mathrm{dgm}_N^{\mathcal{I}}) := \inf(\epsilon > 0 : \textit{for all } I \in \mathcal{I},\ \mathrm{rk}_N(I^\epsilon) \leq \mathrm{rk}_M(I) \textit{ and } \mathrm{rk}_M(I^\epsilon) \leq \mathrm{rk}_N(I)).$$

A **correspondence** between nonempty sets A and B is a subset $\mathcal{R} \subset A \times B$ satisfying the following: (1) for each $a \in A$, there exists $b \in B$ such that $(a, b) \in \mathcal{R}$, and (2) for each $b \in B$, there exists $a \in A$ such that $(a, b) \in \mathcal{R}$. For $\epsilon \in \mathbb{R}_{\geq 0}$, an ***$\epsilon$*-correspondence** $\mathcal{R}$ between nonempty $\mathcal{I}, \mathcal{J} \subset \mathrm{Int}(\mathbb{R}^d)$ is a correspondence $\mathcal{R} \subset \mathcal{I} \times \mathcal{J}$ such that for all $(I, J) \in \mathcal{R}$, $J \subset I^\epsilon$ and $I \subset J^\epsilon$. Blending the ideas of the Hausdorff and erosion distances, we obtain our new distance between *GPDs relative to sampled intervals*:

▶ **Definition 4** (Sparse erosion distance between GPDs relative to sampled intervals). *For any $M, N : \mathbb{R}^d \to \mathrm{vect}_k$ and any nonempty $\mathcal{I}, \mathcal{J} \subset \mathrm{Int}(\mathbb{R}^d)$, the* ***sparse erosion distance*** *between pairs* $(\mathrm{dgm}_M, \mathcal{I})$ *and* $(\mathrm{dgm}_N, \mathcal{J})$ *is*

$$\hat{d}_{\mathrm{E}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J})) := \inf(\epsilon > 0 :\ \textit{there exists an } \epsilon\text{-correspondence } \mathcal{R} \subset \mathcal{I} \times \mathcal{J}$$
$$\textit{s.t.} \forall (I, J) \in \mathcal{R},\ \forall \delta \in \mathbb{R}_{\geq 0},\ \mathrm{rk}_N(J^{\epsilon+\delta}) \leq \mathrm{rk}_M(I^\delta) \textit{ and } \mathrm{rk}_M(I^{\epsilon+\delta}) \leq \mathrm{rk}_N(J^\delta)).$$

We remark that $\hat{d}_{\mathrm{E}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J}))$ captures not only (1) the algebraic difference between $[M$ with respect to $\mathcal{I}]$ and $[N$ with respect to $\mathcal{J}]$, but also (2) the geometric difference between the domains $\mathcal{I}$ and $\mathcal{J}$. To see (1), consider, for example, the $\mathbb{R}$-persistence modules

$M = k_{[0,1]}$ and $N = k_{[0,2]}$, and let both $\mathcal{I}$ and $\mathcal{J}$ be the singleton set $\{[3,4]\}$. Then, one can see that $\hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J})) = 0$ from the fact that $\mathrm{rk}_M^{\mathcal{I}} = \mathrm{rk}_N^{\mathcal{J}} = 0$ and the monotonicity of rk_M and rk_N. To see (2), let M and N be any isomorphic $\mathbb{R}$-persistence modules, and set $\mathcal{I} := \{[0, \frac{1}{2}]\}$ and $\mathcal{J} := \{[0,1]\}$. Then, we obtain $\hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J})) \geq 1/2$, solely due to the difference between $\mathcal{I}$ and $\mathcal{J}$.

▶ **Proposition 5.** *$\hat{d_{\mathrm{E}}}$ is an extended pseudometric. (See the full version [11] for the proof.)*

For any nonempty $\mathcal{I}, \mathcal{J} \subset \mathrm{Int}(\mathbb{R}^d)$, let

$$\hat{d}(\mathcal{I}, \mathcal{J}) := \inf(\epsilon > 0 : \text{ there exists an } \epsilon\text{-correspondence between } \mathcal{I} \text{ and } \mathcal{J}). \tag{4}$$

We clarify the relationship among d_{E}, $\hat{d_{\mathrm{E}}}$, and $\hat{d}$:

▶ **Proposition 6.** *Let $M, N : \mathbb{R}^d \to \mathrm{vect}_k$ and let $\mathcal{I}, \mathcal{J} \subset \mathrm{Int}(\mathbb{R}^d)$ be nonempty. We have:*

(i) $\hat{d}(\mathcal{I}, \mathcal{J}) \leq \hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J}))$.

(ii) *If $\mathcal{I} = \mathcal{J}$ are closed under thickening, then*

$$\hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J})) \leq d_{\mathrm{E}}(\mathrm{dgm}_M^{\mathcal{I}}, \mathrm{dgm}_N^{\mathcal{I}}). \tag{5}$$

(iii) *If $M \cong N$ or $\mathrm{dgm}_M = \mathrm{dgm}_N$, then*

$$\hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J})) = \hat{d}(\mathcal{I}, \mathcal{J}). \tag{6}$$

(See the full version [11] for the proof.)

We remark that the inequality given in Item (i) can be strict. For instance, let $d = 1$, $\mathcal{I} = \mathcal{J} = \mathrm{Int}(\mathbb{R}^d)$, $M = 0$, and $N = k_{[0,1)}$. Then, $0 = \hat{d}(\mathcal{I}, \mathcal{J}) < \hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{J}))$.

By Proposition 6 (ii) and the prior stability result of d_{E} [13, Theorem H], we have:

▶ **Corollary 7.** *For any $M, N : \mathbb{R}^d \to \mathrm{vect}_k$ and for any $\mathcal{I} \subset \mathrm{Int}(\mathbb{R}^d)$, we have*

$$\hat{d_{\mathrm{E}}}((\mathrm{dgm}_M, \mathcal{I}), (\mathrm{dgm}_N, \mathcal{I})) \leq d_{\mathrm{I}}(M, N),$$

where the right-hand side is the interleaving distance between M and N.

3.2 Closed-form formula for the sparse erosion distance

In this section, we find a closed-form formula for the right-hand side of Equation (6), when each interval in $\mathcal{I}$ and $\mathcal{J}$ has only finitely many minimal and maximal points (Theorem 9). This result is essential when studying our loss function in Section 4 and optimizing the domains of the GPDs in Section 5.

For $p, q \in \mathbb{N}^*$, an interval $I \in \mathrm{Int}(\mathbb{R}^2)$ is a (p,q)-interval if I has exactly p minimal points and exactly q maximal points. For any interval $I \in \mathrm{Int}(\mathbb{R}^2)$, let $\min(I)$ (resp. $\max(I)$) be the set of all minimal (resp. maximal) elements of I. For $(x,y) \in \mathbb{R}^2$, define

$$\delta(x,y) := \begin{cases} 1 & \text{if} \quad x \leq y \\ 0 & \text{if} \quad x > y. \end{cases} \tag{7}$$

▶ **Lemma 8.** *For $p_r, q_r, p'_s, q'_s \in \mathbb{N}^*$, let $I_r, J_s \in \mathrm{Int}(\mathbb{R}^2)$ be (p_r, q_r)- and (p'_s, q'_s)-intervals, respectively. Then, for any $\epsilon \in \mathbb{R}_{\geq 0}$, $J_s \subset I_r^{\epsilon}$ and $I_r \subset J_s^{\epsilon}$ if and only if*

$$\epsilon \geq \max\left(\max_k a_k, \max_l b_l, \max_i a'_i, \max_j b'_j\right), \quad \textit{where}$$

- *i, j, k, l range from 1 to p_r, q_r, p'_s, q'_s respectively; a_k and b_l are defined as (8) and (9) respectively; a'_i and b'_j are obtained by interchanging the roles of i and k in (8), and those of j and l in (9), respectively;*
- $\min(I_r) =: \{(x_i^r, y_i^r) : 1 \le i \le p_r\}$, $\min(J_s) =: \{(x_k^s, y_k^s) : 1 \le k \le p'_s\}$,
  $\max(I_r) =: \{(X_j^r, Y_j^r) : 1 \le j \le q_r\}$, $\max(J_s) =: \{(X_l^s, Y_l^s) : 1 \le l \le q'_s\}$.

$$\min_i \Big(\max\Big((1 - \delta(x_i^r, x_k^s))|x_k^s - x_i^r|, (1 - \delta(y_i^r, y_k^s))|y_k^s - y_i^r| \Big) \Big) \tag{8}$$

$$\min_j \Big(\max\Big(\delta(X_j^r, X_l^s)|X_l^s - X_j^r|, \delta(Y_j^r, Y_l^s)|Y_l^s - Y_j^r| \Big) \Big) \tag{9}$$

(See the full version [11] for the proof.) Let $\mathcal{I} := \{I_r\}_{r=1}^n$ and $\mathcal{J} := \{J_s\}_{s=1}^m$ be sets of intervals of $\mathbb{R}^2$ with only finitely many minimal and maximal points. Let $\mathcal{E}$ the $(n \times m)$-matrix (ϵ_{rs}) where ϵ_{rs} is the RHS of the inequality given in Lemma 8, i.e.

$$\epsilon_{rs} = \max(\max_k a_k, \max_l b_l, \max_i a'_i, \max_j b'_j) \tag{10}$$

and $i, j, k, l, a_k, b_l, a'_i, b'_j$ as defined in Lemma 8. Next, we show that $\hat{d}(\mathcal{I}, \mathcal{J})$, as defined in Equation (4), equals the largest of the smallest elements across all rows and columns of $\mathcal{E}$. In particular, the following variables and functions are useful for describing the closed form formula for $\hat{d}(\mathcal{I}, \mathcal{J})$, when each of $\mathcal{I}$ and $\mathcal{J}$ consists solely of $(1,1)$- or $(2,1)$-intervals.

When I_r and J_s are $(2,1)$-intervals, as depicted in Figure 4, let

$$\begin{aligned}
x_1 &:= x_2^r, & y_1 &:= y_1^r, & a &:= Y_1^r - y_1^r, & b &:= x_2^r - x_1^r,\\
c &:= y_1^r - y_2^r, & d &:= X_1^r - x_2^r, & x_2 &:= x_2^s, & y_2 &:= y_1^s,\\
e &:= Y_1^s - y_1^s, & f &:= x_2^s - x_1^s, & g &:= y_1^s - y_2^s, & h &:= X_1^s - x_2^s.
\end{aligned}$$

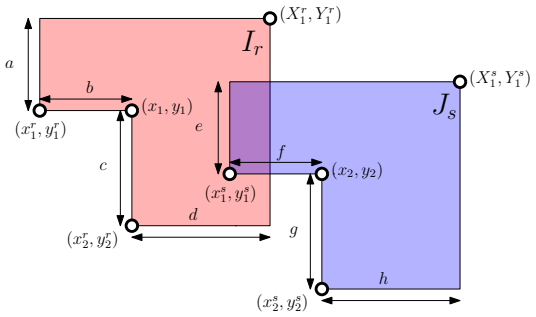


Figure 4 The parametrization of I_r and J_s.

When I_r (resp. J_s) is a $(1,1)$-interval, set $x_1^r = x_2^r$, $y_1^r = y_2^r$ and thus $b = c = 0$ (resp. $x_1^s = x_2^s$, $y_1^s = y_2^s$ and thus $f = g = 0$). Also, let

$$\begin{aligned}
F(w_1, w_2, w_3, w_4) &:= \max\big(\delta(w_1, w_2)|w_2 - w_1|, \delta(w_3, w_4)|w_4 - w_3|\big),\\
G(m_1, m_2, m_3, m_4, m_5) &:= \min(F(m_1, m_2, m_3, m_4), m_5),\\
H(o_1, o_2, o_3, o_4) &:= \max\big(\min(o_1, o_2), \min(o_3, o_4)\big),
\end{aligned}$$

where all input variables of F, G and H are real numbers.

▶ **Theorem 9.** *Let $\mathcal{I} := \{I_r\}_{r=1}^n$ and $\mathcal{J} := \{J_s\}_{s=1}^m$ be sets of intervals of $\mathbb{R}^2$ with only finitely many minimal and maximal points.*

(i) *We have $\hat{d}(\mathcal{I}, \mathcal{J}) = \max\Big(\max_r(\min_s \epsilon_{rs}), \max_s(\min_r \epsilon_{rs})\Big)$ where ϵ_{rs} is as defined in Equation (10) and I_r (resp. J_s) is a (p_r, q_r)-interval (resp. (p'_s, q'_s)-interval).*

(ii) *If $\mathcal{I}$ and $\mathcal{J}$ consist solely of $(1,1)$- or $(2,1)$-intervals, then ϵ_{rs} is the maximum of (11)-(14).*

$$H\Big(F(x_2 - f, x_1 - b, y_2, y_1), F(x_2 - f, x_1, y_2, y_1 - c), \\ F(x_2, x_1 - b, y_2 - g, y_1), F(x_2, x_1, y_2 - g, y_1 - c)\Big) \tag{11}$$

$$F(x_1 + d, x_2 + h, y_1 + a, y_2 + e) \tag{12}$$

$$H\Big(F(x_1 - b, x_2 - f, y_1, y_2), F(x_1 - b, x_2, y_1, y_2 - g), \\ F(x_1, x_2 - f, y_1 - c, y_2), F(x_1, x_2, y_1 - c, y_2 - g)\Big) \tag{13}$$

$$F(x_2 + h, x_1 + d, y_2 + e, y_1 + a) \tag{14}$$

(See the full version [11] for the proof.)

▶ Remark 10. In Theorem 9 (i), as p_r, q_r, p'_s, and q'_s increase, the complexity of computing ϵ_{rs} increases. This is because, the ranges over which the maxima on the RHS of Equation (10) increase, thereby increasing the computational complexity of $\hat{d}(\mathcal{I}, \mathcal{J})$ as well.

4 Lipschitz stability and differentiability of the loss function

The goal of this section is to establish Lipchitz continuity/stability and (almost everywhere) differentiability of our loss function (cf. Equation (2)). From Figure 1, recall that we denote the embedding of a $(1,1)$-or $(2,1)$-interval I of $\mathbb{R}^2$ into $\mathbb{R}^6$ by $\mathbf{v}_I$.

▶ Remark 11 (Independence of variables). Although there are multiple ways to embed (p,q)-intervals of $\mathbb{R}^2$ into Euclidean space, for any $p, q \in \mathbb{N}$, some vectorization methods are more efficient than others in our context. For instance, suppose that we represent the $(2,1)$-interval I in Figure 1 by simply concatenating the coordinates of its minimal and maximal points. Namely, we represent I with the vector $(x_1, y_1, x_2, y_2, X, Y) \in \mathbb{R}^6$, where $x_1 = x - b$, $y_1 = y$, $x_2 = x$, $y_2 = y - c$, $X = x + d$, $Y = y + a$. A difficulty with this embedding in the context of gradient descent is that the variables are *not* independent: one must have $(x_1, y_1) \leq (X, Y)$, $(x_2, y_2) \leq (X, Y)$, and $(x_2, y_2) \not\leq (x_1, y_1)$. Ensuring such relations between variables is not trivial in non-convex optimization (without using refined techniques such as projected gradient descent). On the other hand, using our proposed embedding is more practical, as all variables are independent, and our only requirements are that a, b, c, d must be nonnegative, which can easily be imposed using, e.g., exponential or ReLU functions.

Let $\mathcal{I} := \{I_r\}_{r=1}^n$ and $\mathcal{J} := \{J_s\}_{s=1}^m$ consist solely of $(1,1)$- or $(2,1)$-intervals of $\mathbb{R}^2$.

▶ **Lemma 12.** *For $I_r \in \mathcal{I}$ and $J_s \in \mathcal{J}$, consider $\mathbf{v}_{I_r} = (x_1, y_1, a, b, c, d)$ and $\mathbf{v}_{J_s} = (x_2, y_2, e, f, g, h)$ defined as in Figure 1. If $||\mathbf{v}_{I_r} - \mathbf{v}_{J_s}||_\infty \leq \epsilon$, then $\epsilon_{rs} \leq 2\epsilon$ where ϵ_{rs} is described as in Theorem 9 (ii). (See the full version [11] for the proof.)*

We define $\mathbb{v}_{\mathcal{I}}$ as the concatenation of $\mathbf{v}_{I_r}$ for all $I_r \in \mathcal{I}$, i.e., $\mathbb{v}_{\mathcal{I}} := (\mathbf{v}_{I_1} | \dots | \mathbf{v}_{I_n}) \in \mathbb{R}^{6n}$. Let $\mathfrak{I}$ be the collection of all ordered n-sets $\mathcal{I}$ of $(1,1)$-or $(2,1)$-intervals of $\mathbb{R}^2$. Then, define the function $V : \mathfrak{I} \to \mathbb{R}^{6n}$ by $\mathcal{I} \mapsto \mathbb{v}_{\mathcal{I}}$.

▶ **Proposition 13.** *The image of $\mathfrak{I}$ via the function V is a convex subset of $\mathbb{R}^{6n}$.*

Proof. Let S be the collection of all $(1,1)$- and all $(2,1)$-intervals of $\mathbb{R}^2$. Consider the map from S to $\mathbb{R}\times\mathbb{R}\times\mathbb{R}_{>0}\times\mathbb{R}_{\geq 0}\times\mathbb{R}_{\geq 0}\times\mathbb{R}_{>0}$ depicted in Figure 1. Clearly, this map is a surjection. It follows that the image of S via the map $I\mapsto(x,y,a,b,c,d)$ depicted in Figure 1 is a convex subset of $\mathbb{R}^6$. Now, observe that the image of $\mathfrak{I}$ via the map V is equal to the n-fold Cartesian product of the image of S, which is a subset of $\mathbb{R}^{6n}$. The fact that any Cartesian product of convex sets is convex implies our claim. ◀

▶ **Theorem 14** (Lipschitz stability of loss function)**.** *Let $\mathcal{K}$ be any finite set of $(1,1)$- or $(2,1)$-intervals, and let $\mathcal{J}_1,\mathcal{J}_2$ be any two n-sets of $(1,1)$- or $(2,1)$-intervals for some $n\in\mathbb{N}^*$. Then, we have $|\hat{d}(\mathcal{K},\mathcal{J}_1)-\hat{d}(\mathcal{K},\mathcal{J}_2)|\leq 2\cdot\min_\pi||\mathbb{v}_{\pi(\mathcal{J}_1)}-\mathbb{v}_{\mathcal{J}_2}||_\infty$, where the minimum is taken over all permutations on $\mathcal{J}_1$.*

Proof. By the triangle inequality, we have $|\hat{d}(\mathcal{K},\mathcal{J}_1)-\hat{d}(\mathcal{K},\mathcal{J}_2)|\leq\hat{d}(\mathcal{J}_1,\mathcal{J}_2)$ and thus it suffices to show that $\hat{d}(\mathcal{J}_1,\mathcal{J}_2)\leq 2\cdot\min_\pi||\mathbb{v}_{\pi(\mathcal{J}_1)}-\mathbb{v}_{\mathcal{J}_2}||_\infty$. Let π_0 be a permutation on $\mathcal{J}_1$ that attains the minimum. Let $\mathcal{I}:=\pi_0(\mathcal{J}_1)$ and $\mathcal{J}:=\mathcal{J}_2$. By Lemma 12, ϵ_{rr} (resp. ϵ_{ss}) $\leq 2\epsilon$ for all $r,s\in\{1,\dots,n\}$ and hence $\min_s\epsilon_{rs}$ (resp. $\min_r\epsilon_{rs}$) $\leq 2\epsilon$ for each r (resp. s). Thus, we have $\max_r(\min_s\epsilon_{rs})\leq 2\epsilon$ and $\max_s(\min_r\epsilon_{rs})\leq 2\epsilon$. By Theorem 9 (i), we are done. ◀

▶ Remark 15. The opposite direction of Theorem 14 does not hold, i.e. there does not exist $c>0$ such that $|\hat{d}(\mathcal{K},\mathcal{J}_1)-\hat{d}(\mathcal{K},\mathcal{J}_2)|\geq c\cdot\min_\pi||\mathbb{v}_{\pi(\mathcal{J}_1)}-\mathbb{v}_{\mathcal{J}_2}||_\infty$. However, the non-existence of such $c>0$ is *not* an issue in our work as what we require is the Lipschitz stability and almost everywhere differentiability of the loss function (which we establish in the next proposition), in order to prevent erratic and oscillating gradient descent iterations.

▶ **Proposition 16.** *The loss function given in Equation (2) is differentiable almost everywhere.*

Proof. The three maps $\mathbb{R}^2\to\mathbb{R}$ given by $(x,y)\mapsto\delta(x,y)|y-x|$ (cf. Equation (7)), $(x,y)\mapsto\max\{x,y\}$, and $(x,y)\mapsto\min\{x,y\}$ are finitely segmented piecewise linear. Therefore, the functions given in Equations (11)–(14) are all finitely segmented piecewise linear on Euclidean spaces, and thus so are ϵ_{rs} given in Theorem 9 (ii). Hence, trivially, these functions are differentiable almost everywhere. Now, Proposition 6 (iii) implies our claim. ◀

5 Numerical Experiments on Sparsification

In this section, we make use of the results proved above to provide a method for *sparsifying* GPDs. Indeed, computing a single GPD $\mathrm{dgm}_M^{\mathcal{I}}$ on a persistence module M coming from a simplicial complex S requires computing zigzag persistence modules on all intervals of $\mathcal{I}$, as described in [16], and then computing the corresponding Möbius inversion, yielding a time complexity of $O(n^3)$, where n is the number of intervals in $\mathcal{I}$, due to the need to compute the Möbius function value for all possible pairs of intervals, each requiring $O(n)$ operations to iterate over each interval. Thus, the complexity of computing a single GPD is $O(n\times N_s^{2.376}+n^3)$, where N_s is the number of simplices [30]. Hence, computing all GPDs from a dataset of persistence modules becomes intractable when n is large.

Therefore, as described in the introduction, our goal in this section is to design a *sparse* subset of $(2,1)$-intervals $\mathcal{J}^*$ of size $m\ll n$ that minimizes the loss function in Equation (2) with gradient descent by treating every interval in $\mathcal{J}$ as a parameter to optimize.

Time-series datasets. The datasets we consider are taken from the UCR repository [15], and correspond to classification tasks that have already been studied with persistent homology before [29, Section 4.2]. More precisely, instances in these datasets take the form of labelled time series, that we pre-process with time-delay embedding in order to turn them into point clouds. Specifically, each labelled time series $T = \{f(t_1), \dots, f(t_n)\}$ of length n is transformed into a point cloud $X_T \subset \mathbb{R}^3$ of cardinality $n-2$ with

$$X_T := \{(f(t_1), f(t_2), f(t_3))^T, \dots, (f(t_{n-2}), f(t_{n-1}), f(t_n))^T\}.$$

Then, we compute both the Vietoris-Rips filtration and the sublevel set filtration induced by a Gaussian kernel density estimator (with bandwidth $\sigma = 0.1 \cdot \mathrm{diam}(X_T)$, where $\mathrm{diam}(X_T) := \max_{x,y \in X_T} \|x - y\|_2$ is the diameter of the point cloud) using the `PointCloud2FilteredComplex` function of the `multipers` library [28] (a tutorial is available in [26]). Both filtrations are then normalized so that their ranges become equal to the unit interval $[0, 1]$.

Loss function. In order to compute GPDs out of these 2-parameter filtrations and modules, one needs a subset of intervals. As explained in the introduction, we then minimize

$$\mathcal{L}_{\hat{d_{\mathrm{E}}},m} : \mathbb{V}_{\mathcal{J}} \mapsto \hat{d}(\mathcal{I}, \mathcal{J}),$$

where the full domain $\mathcal{I}$ (resp. the sparse domain $\mathcal{J}$) is comprised of $n = 1,600$ (resp. $m = 400$) $(2,1)$-intervals obtained from a grid in $\mathbb{R}^6$ computed by evenly sampling 10 (resp. 5) values for x and y and 2 values for a, b, c, d within their corresponding filtration ranges.[6] Note that while the $(2,1)$-intervals in $\mathcal{J}$ are treated as parameters to optimize, the $(2,1)$-intervals in $\mathcal{I}$ are fixed throughout the optimization process. Moreover, the formula that we provided in Theorem 9 (ii) can be readily implemented in any library that uses auto-differentiation, such as `pytorch`. In particular, we run stochastic gradient descent with momentum 0.9 on $\mathcal{L}$ for 750 epochs using learning rate $\eta = 0.001$ with exponential decay of factor 0.99 to achieve convergence, and obtain our sparse subset $\mathcal{J}^*$. See Figure 5 for a visualization of the loss decrease. Note how the Lipschitz stability proved in Theorem 14 translates into a smooth decrease with small oscillations.

Accuracy scores. In order to quantify the running time improvements as well as the information loss (if any) when switching from the full domain $\mathcal{I}$ to the optimized sparse one $\mathcal{J}^*$, we computed the full and sparse GPDs in homology dimensions 0 and 1 with the zigzag persistence diagram implementation in `dionysus` [31], and then we trained random forest classifiers on these GPDs to predict the time series labels.[7] In order to achieve this, we first turned both the full and optimized GPDs into Euclidean vectors by first binning every GPD (seen as a point cloud in $\mathbb{R}^6$) with a fixed 6-dimensional histogram, and then convolving this histogram with a 6-dimensional Gaussian kernel in order to smooth its values, with a procedure similar to the one described in [29, Section 3.2.1]. Both the histogram bins and the kernel bandwidths were found with 3-fold cross-validation on the training set (see the provided code for hyperparameter values). Then, random forest classifiers were trained on these smoothed histograms to predict labels; in Table 1 we report the accuracy scores

[6] We also tried random uniform sampling for the initial distributions of x, y, a, b, c, d, but this resulted in slower convergence and lower performances.

[7] Recall that optimizing the sparse domain can be performed without computing GPDs, so the optimization process described in the previous paragraph is done once and for all, before computing the GPDs.

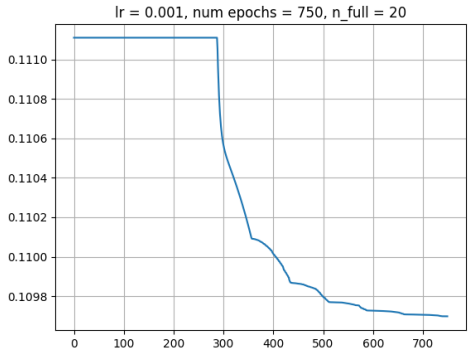


Figure 5 Loss decrease across gradient descent iterations. Note that the loss value stays on a plateau for the first ∼300 iterations; this is due to the fact that during these first iterations, the parameters in $\mathcal{J}$ that are updated with gradient descent are not yet the ones achieving the maxima and minima in the closed-form formula provided in Theorem 9 (ii). Moreover, in this example, gradient descent converges to a local minimum of the loss that is only ∼0.01 lower than the initial value, partially because we are using the small learning rate 0.001.

of these classifiers for the initial (before optimization) sparse domain $\mathcal{J}_{\text{init}}$, the optimized sparse domain $\mathcal{J}^*$, and the full domain $\mathcal{I}$. Moreover, in Table 2, we report the running time needed to compute all GPDs using either $\mathcal{J}_{\text{init}}$, $\mathcal{J}^*$, or $\mathcal{I}$, as well as the improvement when switching from $\mathcal{J}^*$ to $\mathcal{I}$. Note that our goal is not to improve on the state-of-the-art for time series classification, but rather to assess whether optimizing the loss $\mathcal{L}_{\hat{d_{\text{E}}},m}$ based on our upper bound is indeed beneficial for improving topology-based models. See Figure 2 for a schematic overview of our full pipeline.

Discussion on results. As one can see from Table 1, there is either a clear improvement or a comparable performance in accuracy scores after optimizing $\mathcal{J}$. Indeed, by minimizing $\mathcal{L}_{\hat{d_{\text{E}}},m}$, one forces the sparse domain $\mathcal{J}$ to be as close as possible to the full domain $\mathcal{I}$, and thus to retain as much topological information as possible. Hence, either the full GPDs are more efficient than the initial sparse GPDs, in which case the optimized GPDs perform much better than the initial sparse ones, or the full GPDs are less efficient than the initial sparse GPDs,[8] in which case the optimized GPDs have comparable or slightly worse performances than the initial sparse ones thanks to the small sizes of their domains (except for the PC and IPD datasets, for which the optimized GPDs still perform interestingly better). In all cases, we emphasize that optimized GPDs provide the best solution: *they maintain scores at levels that are either comparable or better than the best solution between the initial sparse and full GPDs, while avoiding to force users to choose interval domains* (as their domains are obtained automatically with gradient descent). Note in particular that whenever optimized GPDs do not achieve the best scores, their performances remain of the same order as those of their competitors (see, e.g., DPC or SC), whereas when optimized GPDs are the best, the performances of the other two competitors are sometimes far behind (see, e.g., GPA).

[8] Recall that the accuracy score is only an *indirect* measure: While the full GPDs are *always* richer in topological information, we hypothesize that their scores might still be lower due to a large number of intervals on which the full GPDs vanish – a question that we leave for further study.

As for running times, we observe a slight increase from the running times associated to the initial sparse GPDs (except for the IPD dataset), which we hypothesize is due to the use of optimized intervals that intersect more with each other, and thus induce more poset relations when computing Möbius inversions, and a strong improvement over the running times associated to the full GPDs, with ratios ranging between 5 and 15 times faster. *Our optimized GPDs thus achieve the best of both worlds: they are strinkingly fast to compute while keeping high accuracy scores.* Our code was run on a `2x Xeon SP Gold 5115 @ 2.40GHz`, and is fully available at sparse GPDs.

Table 1 Accuracy scores (%) of random forest classifiers trained on several UCR datasets. Underline indicates best score between the initial and optimized sparse domains, while bold font indicates best score overall. See the full version [11] for the dataset full names.

	C	DPA	DPC	DPT	PPA	PPC	PPT	ECG	IPD
Init.	0.750	**0.705**	**0.746**	**0.561**	**0.790**	0.718	0.693	**0.790**	0.677
Optim.	0.786	0.691	0.721	**0.561**	0.785	**0.742**	**0.737**	**0.790**	**0.690**
Full	**0.857**	0.669	0.743	0.554	0.780	0.729	0.707	0.740	0.651
	MI	P	SL	GP	GPA	GPM	GPO	PC	SC
Init.	0.534	**0.924**	**0.565**	**0.900**	0.835	0.946	0.987	0.789	0.447
Optim.	**0.588**	0.886	0.546	0.893	**0.959**	**0.959**	0.956	**0.800**	0.487
Full	0.545	0.838	0.531	0.847	0.864	0.946	**0.990**	0.778	**0.503**

Table 2 Running times (seconds) needed for computing all GPDs on several UCR datasets. Underline indicates best running time between the initial and optimized sparse domains, while bold font indicates best running time overall. See the full version [11] for the dataset full names.

	C	DPA	DPC	DPT	PPA	PPC	PPT	ECG	IPD
Init.	**350**	**1076**	**1902**	**1178**	**1182**	**1660**	**1147**	**556**	**1262**
Optim.	421	1172	2054	1265	1257	1749	1232	621	1328
Full	2304	11672	20163	13007	12745	18427	12844	5098	19860
Improv.	5.47x	9.95x	9.82x	10.28x	10.13x	10.53x	10.42x	8.20x	15.74x
	MI	P	SL	GP	GPA	GPM	GPO	PC	SC
Init.	**2989**	**792**	**3478**	**703**	**1576**	**1658**	**1573**	**1339**	**1240**
Optim.	3399	909	3919	883	1955	2061	1925	1550	1324
Full	29074	6280	31285	5773	12913	13523	12849	10730	13089
Improv.	8.55x	6.91x	7.98x	6.54x	6.60x	6.56x	6.67x	6.92x	9.88x

6 Conclusion

Our sparsification method demonstrates the practicality of approximating full GPDs with sparse ones: while significantly reducing computational costs, our appropriate loss function also ensures that their discriminative power is not too compromised. We thus believe that our work paves the way for efficiently deploying multi-parameter topological data analysis to large-scale applications, that are currently out of reach for most multi-parameter topological invariants from the literature. In what follows, we outline several potential future directions.

Optimization for finer GPDs. In our experiments, we optimized the GPDs over intervals with at most two minimal points and exactly one maximal point. Allowing more complex intervals, as well as dataset-dependent terms in the loss function, could improve performance,

but finding suitable embeddings of such intervals into Euclidean space remains a challenge (cf. Remark 11) and would increase the computational cost (cf. Remark 10). It would also be interesting to quantify the discriminating power of GPDs with measures that are more direct than the score of a machine learning classifier (or at least to further study the dependencies between GPD sparsification and classifier scores), and to investigate on heuristics (based on drops in the loss values) for deciding whether sparse GPDs are sufficiently good so that optimization can be stopped.

Experimental validation for other GRI-based descriptors. While we only focused on sparsifying GPDs in this work, it would be interesting to measure the extent to which our interval domain optimization adapts to other descriptors based on the GRI from the TDA literature; of particular interest are the GPDs/signed barcodes coming from *rank exact decompositions*, which have recently proved to be stable [8] (see also Section 8.2 in [7] which discusses the influence of the choice of the interval domains on the resulting invariants), as well as GRIL, which focuses on specific intervals called *worms* [37].

References

1 Hideto Asashiba, Etienne Gauthier, and Enhao Liu. Interval replacements of persistence modules. *arXiv preprint*, 2024. arXiv:2403.08308.

2 Gorô Azumaya. Corrections and supplementaries to my paper concerning Krull-Remak-Schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

3 Ulrich Bauer, Magnus Botnan, Steffen Oppermann, and Johan Steen. Cotorsion torsion triples and the representation theory of filtered hierarchical clustering. *Advances in Mathematics*, 369:107171, 2020.

4 Magnus Botnan and William Crawley-Boevey. Decomposition of persistence modules. *Proceedings of the American Mathematical Society*, 148(11):4581–4596, 2020.

5 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & Geometric topology*, 18(6):3133–3204, 2018.

6 Magnus Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of Algebras and Related Structures*, pages 77–150. CoRR, 2023.

7 Magnus Botnan, Steffen Oppermann, and Steve Oudot. Signed barcodes for multi-parameter persistence via rank decompositions and rank-exact resolutions. *Foundations of Computational Mathematics*, pages 1–60, 2024.

8 Magnus Botnan, Steffen Oppermann, Steve Oudot, and Luis Scoccola. On the bottleneck stability of rank decompositions of multi-parameter persistence modules. *Advances in Mathematics*, 451:109780, 2024.

9 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.

10 Mathieu Carrière and Andrew Blumberg. Multiparameter persistence image for topological machine learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 22432–22444. Curran Associates, Inc., 2020.

11 Mathieu Carriere, Seunghyun Kim, and Woojin Kim. Sparsification of the generalized persistence diagrams for scalability through gradient descent. *arXiv preprint arXiv:2412.05900*, 2024. doi:10.48550/arXiv.2412.05900.

12 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.

13 Nate Clause, Woojin Kim, and Facundo Memoli. The generalized rank invariant: Möbius invertibility, discriminating power, and connection to other invariants. *arXiv preprint*, 2024. arXiv:2207.11591v5.

14 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.

15 Hoang-Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ratanamahatana, and Eamonn Keogh. The UCR time series archive. *arXiv*, 2018. arXiv:1810.07758.

16 Tamal Dey, Woojin Kim, and Facundo Mémoli. Computing generalized rank invariant for 2-parameter persistence modules via zigzag persistence and its applications. In *38th International Symposium on Computational Geometry (SoCG 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.34.

17 Tamal Dey, Aman Timalsina, and Cheng Xin. Computing generalized ranks of persistence modules via unfolding to zigzag modules. *arXiv preprint*, 2024. doi:10.48550/arXiv.2403.08110.

18 Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2008.

19 Donghan Kim, Woojin Kim, and Wonjun Lee. Super-polynomial growth of the generalized persistence diagram, 2024. doi:10.48550/arXiv.2412.04889.

20 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *Journal of Applied and Computational Topology*, 5(4):533–581, 2021. doi:10.1007/S41468-021-00075-1.

21 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.

22 Woojin Kim and Facundo Mémoli. Persistence over posets. *Notices of the American Mathematical Society*, 70(08), 2023.

23 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.

24 Woojin Kim and Samantha Moore. Bigraded Betti numbers and generalized persistence diagrams. *Journal of Applied and Computatioal Topology*, 2024. doi:10.1007/s41468-024-00180-x.

25 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. doi:10.1007/S10208-015-9255-Y.

26 David Loiseaux. Time series classification. Accessed: 2025-03-24. URL: https://davidlapous.github.io/multipers/notebooks/time_series_classification.html.

27 David Loiseaux, Mathieu Carrière, and Andrew Blumberg. A framework for fast and stable representations of multiparameter persistent homology decompositions. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, Inc., 2023.

28 David Loiseaux and Hannah Schreiber. multipers: Multiparameter persistence for machine learning. *Journal of Open Source Software*, 9(103):6773, 2024. doi:10.21105/JOSS.06773.

29 David Loiseaux, Luis Scoccola, Mathieu Carrière, Magnus Botnan, and Steve Oudot. Stable vectorization of multiparameter persistent homology using signed barcodes as measures. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, Inc., 2023.

30 Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, pages 216–225, 2011. doi:10.1145/1998196.1998229.

31 Dmitriy Morozov. Dionysus 2. Accessed: 2025-03-24. URL: https://mrzv.org/software/dionysus2/.

32 Soham Mukherjee, Shreyas Samaga, Cheng Xin, Steve Oudot, and Tamal Dey. D-gril: End-to-end topological learning with 2-parameter persistence. *arXiv*, 2024. arXiv:2406.07100.

33 Amit Patel. Generalized persistence diagrams. *Journal of Applied and Computational Topology*, 1(3):397–419, 2018. doi:10.1007/S41468-018-0012-6.

34 Luis Scoccola, Siddharth Setlur, David Loiseaux, Mathieu Carrière, and Steve Oudot. Differentiability and optimization of multiparameter persistent homology. In *41st International Conference on Machine Learning (ICML 2024)*, volume 235, pages 43986–44011. PMLR, 2024.

35 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. URL: https://jmlr.org/papers/v21/19-054.html.

36 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.

37 Cheng Xin, Soham Mukherjee, Shreyas Samaga, and Tamal Dey. GRIL: A 2-parameter persistence based vectorization for machine learning. In *2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning*. OpenReviews.net, 2023.

Computation of Toroidal Schnyder Woods Made Simple and Fast: From Theory to Practice

Luca Castelli Aleardi
LIX, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Eric Fusy
LIGM, Université Gustave Eiffel, Champs-sur-Marne, France

Jyh-Chwen Ko
LIX, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Razvan-Stefan Puscasu
École polytechnique fédérale de Lausanne (EPFL), Switzerland

Abstract

We consider the problem of computing Schnyder woods for graphs embedded on the torus. We design simple linear-time algorithms based on canonical orderings that compute toroidal Schnyder woods for simple toroidal triangulations. The Schnyder woods computed by one of our algorithm are crossing and satisfy an additional structural property: at least two of the mono-chromatic components of the Schnyder wood are connected. We also exhibit experimental results empirically confirming three conjectures involving the structure of toroidal and higher genus Schnyder woods.

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

Keywords and phrases Schnyder woods, toroidal triangulations, canonical ordering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.30

Related Version *Full Version*: `https://hal.science/hal-05027238` [13]

Supplementary Material
Software: `https://www.lix.polytechnique.fr/~amturing/software.html`

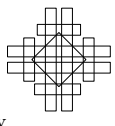
Funding *Luca Castelli Aleardi*: this work is supported by ANR grant "3DMaps" ANR-20-CE48-0018.
Eric Fusy: this work is supported by ANR grant "3DMaps" ANR-20-CE48-0018.

1 Introduction

Graphs embedded on surfaces exhibit many significant structural properties. In the planar case, one of the most elegant characterizations is provided by Schnyder woods, a profound combinatorial structure introduced by W. Schnyder to define a new planarity criterion [31] and to design an efficient and elegant straight-line grid drawing algorithm [32]. Schnyder woods lead to a global characterization of planar graphs in terms of spanning trees and have inspired numerous generalizations and applications in various fields, including graph drawing [17, 18, 19], graph encoding [10, 11], enumerative combinatorics [2, 20], random sampling [29] and plane spanners [4].

The first generalizations [5, 14, 15] of Schnyder woods to non planar graphs were motivated by applications involving graph encoding problems, aiming to preserve their global spanning properties. Since then the study of Schnyder woods and related structures for graphs embedded on toroidal [3, 6, 12, 16, 21, 22, 27] and higher genus [1, 14, 15, 25, 33] surfaces have attracted a lot of attention. This is particularly true in the toroidal case, for which some elegant generalizations lead to applications ranging from graph drawing [22] to bijective encodings [3, 16, 21].

© Luca Castelli Aleardi, Eric Fusy, Jyh-Chwen Ko, and Razvan-Stefan Puscasu;
licensed under Creative Commons License CC-BY 4.0



Toroidal Schnyder woods

A key property of *toroidal triangulations* is that the number of edges is exactly three times the number of vertices, by Euler's formula. Gonçalves and Lévêque [22] show how to endow a simple toroidal triangulation with an edge orientation and coloring with three colors [1] $\{0, 1, 2\}$ where every vertex has outgoing degree 3 and satisfies a local Schnyder rule (as in the plane). It turns out that the i-cycles [2] are directed, non-contractible and homotopic, and they do not intersect each other. Moreover, any toroidal triangulation admits a *crossing Schnyder wood* for which every pair of i-cycles and j-cycles (for $i \neq j$) are crossing, which is particularly relevant for graph drawing applications. Indeed, the crossing condition is crucial for obtaining a periodic straight-line drawing [22] of toroidal maps on a grid of polynomial resolution, which extends the face-counting approach of planar Schnyder drawings [32, 18, 19].

1.1 Contribution

Addressing two questions left open in [22, 27], we design and implement a simple algorithm for computing (crossing) toroidal Schnyder woods: our approach is based on the vertex shelling paradigm involving canonical orderings [3] and allows us to get mono-chromatic components which are connected, for at least two of the three colors. Our main result is stated below:

▶ **Theorem 1.** *Given a simple toroidal triangulation $\mathcal{T}$ it is possible to compute in linear time a crossing Schnyder wood of $\mathcal{T}$ such that the mono-chromatic components of color* 0 *and of color* 1 *are connected.*

As far as we know, our approach leads to the first implementation for computing toroidal Schnyder woods: previous solutions [22, 27] rely on a very involved case analysis or on unpublished results (requiring more than linear time) and, to our knowledge, they have not been implemented, even in the triangulated case (see Section 1.2 for a more detailed discussion), nor do they provide a guarantee on connectness of any mono-chromatic component.

We also conduct experiments providing an empirical answer to conjectures stated in [1, 22] regarding the existence of Schnyder woods for higher genus surfaces.

1.2 Related works: computing crossing toroidal Schnyder woods

Via edge contractions. One can compute crossing Schnyder woods for essentially 3-connected toroidal maps making use of the approach based on edge contractions [22]. The main idea is to perform a sequence of edge contractions until we are left with few vertices: the Schnyder wood can be recovered by assigning colors after the edge expansions (in reverse order). The validity of this approach relies on a very involved case analysis [22, 27]: this results in a polynomial time algorithm, but it is not known how to get linear time while preserving the crossing condition, which is a non-local property.

Via three paths meeting at a vertex. For simple triangulations another way [22] to compute toroidal Schnyder consists in cutting a toroidal triangulation $\mathcal{T}$ along three non-contractible (and non-homotopic) cycles $\Gamma_0, \Gamma_1, \Gamma_2$ whose intersection is a common single vertex v. As

[1] In our pictures we use red for color 0, blue for color 1 and black for color 2.
[2] An i-cycle, for $i \in \{0, 1, 2\}$, is a mono-chromatic cycle whose edges are of color i.
[3] In the planar case there exists a linear time algorithm for computing Schnyder woods via canonical orderings [8, 26] performing vertex shellings: the fact that such an algorithm also exists in the toroidal case has been left as an open question in [27].

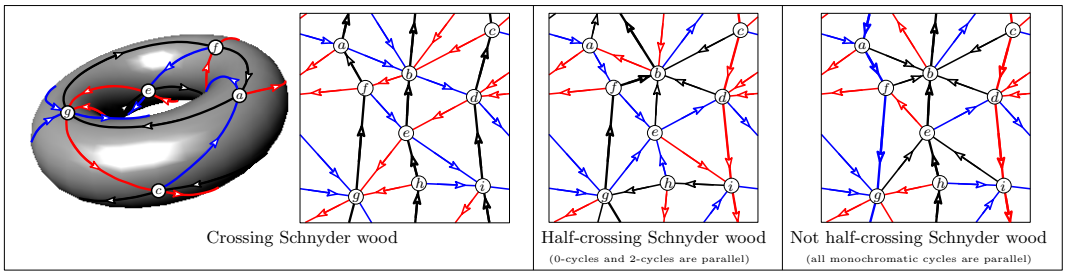


Figure 1 A toroidal simple triangulation of size 9, endowed with three different Schnyder woods.

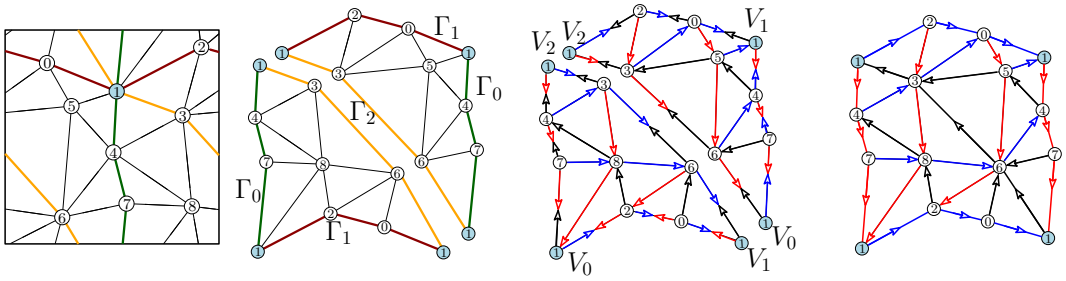


Figure 2 Computing a toroidal Schnyder wood via 3 non-contractible cycles crossing at v_1.

result one obtains a partition of $\mathcal{T}$ into two quasi-triangulations G_1 and G_2, which are internally 3-connected and can be endowed with a planar Schnyder wood (refer to [18, 19]). Then computing Schnyder woods for G_1 and G_2 and gluing them yields a crossing Schnyder wood for $\mathcal{T}$ (see Fig. 2). The Schnyder wood satisfies the following 3-*cycles property*: there are (at least) three mono-chromatic cycles, one for each color, crossing at a single vertex and which are disjoint elsewhere. It is worth noting that the construction described in [22] does not guarantee that the mono-chromatic components are connected (the Schnyder wood could have several disjoint mono-chromatic i-cycles). Unfortunately, as pointed out in [27], it is not clear how to get a linear-time algorithm: the pre-processing phase relies on an unpublished result by Fijavz (the computation of three non-homotopic and non-contractible cycles whose intersection is a common vertex) which involves a result on disjoint paths in [30].

Via reversal of oriented cycles. In the triangulated case, the basic approach using edge contraction (without maintaining crossing property) leads to an algorithm for computing balanced toroidal Schnyder woods in linear time (Thm 10 in [27]). As shown in [27] (Thm 11) it is possible to turn a balanced Schnyder wood into a half-crossing one by performing the reversal of oriented cycles. Then one can perform another cycle reversal phase in order to make the Schnyder wood crossing (Thm 13 in [27]). All these phases can be executed in linear time, but we are not aware of any existing implementation. [4]

[4] For a fast implementation one needs to efficiently process contractible edges and to maintain the map data structure under local updates: this is not trivial in practice, even in the (planar) triangulated case.

2 Preliminaries

Toroidal and cylindric triangulations

In this work we deal with *toroidal triangulations* which are maps on the torus with only triangular faces (such maps are *simple* if they have no loops nor multiple edges). We will denote by n the total number of vertices. In our drawings we will represent such maps in the planar model of the flat torus. We also consider (simple) *cylindric triangulations*, which are planar maps with a marked inner face of simple contour denoted C_{in} (the outer face contour being denoted C_{ext}) such that all other inner faces have degree 3. G is *non-degenerate* if C_{in} and C_{ext} are disjoint (not sharing any vertex). We will denote by n the number of vertices of $G \setminus C_{in}$, and will also use notation $\partial^i G$ for C_{in} and $\partial^e G$ for C_{ext}. We will draw cylindric triangulations either in the annular representation or in the flat model of the cylinder.

Toroidal Schnyder woods (for triangulations)

▶ **Definition 2** ([22, 27]). *Let $\mathcal{T}$ be a toroidal triangulation. A* Schnyder wood *of $\mathcal{T}$ is an orientation and labeling, with colors in $\{0, 1, 2\}$ of the edges of $\mathcal{T}$ so as to satisfy the following* ***local Schnyder condition (for all vertices):*** *for each vertex v, the edges incident to v in counterclockwise (*ccw*) order are: one outgoing edge colored 2, zero or more incoming edges colored 1, one outgoing edge colored 0, zero or more incoming edges colored 2, one outgoing edge colored 1, and zero or more incoming edges colored 0.*

A Schnyder wood is half-crossing *if there are two distinct colors $i, j \in \{0, 1, 2\}$ such that there is an i-cycle crossing a j-cycle (in which case all i-cycles cross all j-cycles). A Schnyder wood is* crossing *if for each pair of distinct colors $i, j \in \{0, 1, 2\}$ there is an i-cycle crossing a j-cycle).*

The pictures in Fig. 1 show a toroidal triangulation endowed with three Schnyder woods. Definition 2 is particularly elegant since it preserves the symmetry of the three colors, but in general the global spanning property of Schnyder woods is lost: the graph induced by any given color could consist of several disjoint connected components, each component being made of a cycle with trees attached to it (as every vertex has outdegree 1 in a given color).

Canonical orderings for cylindric triangulations

As shown in [12], it is possible to extend the notion of canonical orderings [23] to the cylindric case as follows. A *cylindric canonical ordering* is an ordering $\pi = \{v_1, \dots v_n\}$ of the vertices of $G \setminus C_{in}$ such that, for every $k \geq 0$, the map G_k induced by $C_{in} \cup \{v_1, \dots, v_k\}$ is a cylindric (possibly degenerate) triangulation, and such that, for every $k \geq 1$, the vertex v_k belongs to the outer boundary cycle of G_k, denoted $\partial^e G_k$, and its neighbors in G_{k-1} are consecutive boundary vertices on $\partial^e G_{k-1}$. Such an ordering can be computed in linear time via an incremental vertex shelling procedure, where vertices together with their incident edges and faces are removed in reverse order from the boundary $\partial^e G_k$ ($n \geq k \geq 1$). In order to be removed, a vertex $v \in \partial^e G_k \setminus C_{in}$ must be *free*: v cannot be incident to a chord of $\partial^e G_k$. The existence of free vertices on $\partial^e G_k$ relies on an inductive argument similar to the planar case (we refer to [12] for more details).

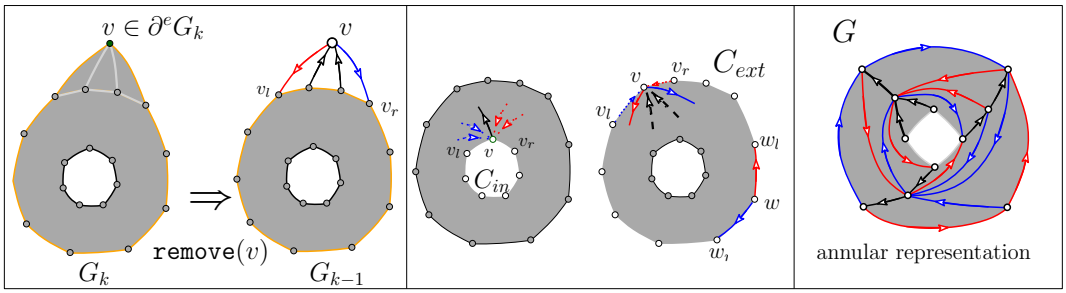


Figure 3 Left: coloring rule associated to the `remove` operator. Center: local conditions for vertices on C_{in} and C_{ext}. Right: a cylindric Schnyder wood of a cylindric triangulation.

3 From cylindric Schnyder woods to toroidal Schnyder woods

For the simple triangulated case, we show how to derive toroidal Schnyder woods from the notion of cylindric canonical orderings [12]. We first planarize the input triangulation $\mathcal{T}$ cutting along a chordless non-contractible cycle Γ, getting a non-degenerate cylindric triangulation G (a non-contractible cycle can be computed in linear time [9]).

Algorithm 1 Computation of a toroidal Schnyder wood via a cylindric canonical ordering.

Data: a simple toroidal triangulation $\mathcal{T}$, a non-contractible chordless cycle Γ
Result: a toroidal Schnyder wood $S(\mathcal{T})$
cut $\mathcal{T}$ along Γ: the result is a cylindric triangulation G with boundaries C_{in}, C_{ext};
$k \leftarrow n$;
$G_k \leftarrow G$, $\partial^e G_k \leftarrow C_{ext}$;
while $\partial^e G_k \neq C_{in}$ **do**
  $v \leftarrow$ any free vertex on $\partial^e G_k$;
  `remove`(v) from $\partial^e G_k$; `/* Orient the edges incident to` v `in` G_k `*/`
  $k \leftarrow k - 1$;
end
identify edges of C_{in} and C_{ext} ;

Then we compute a cylindric canonical ordering of G and we proceed as in the planar case by defining a vertex removal operation: at a given shelling step k, `remove`(v) updates the outer boundary of G_k by removing vertex v while assigning colors and orientations to edges incident to v in G_k, as illustrated in the left picture of Fig. 3: the left and right edges (v, v_{left}) and (v, v_{right}) become outgoing from v (of color 0 and 1), and all remaining edges get oriented toward v (of color 2). This shelling procedure is described in detail in [12], the pseudocode is provided in Algorithm 1. As a result we get a *cylindric Schnyder wood* $S(G)$, i.e., a coloring (in 0, 1 or 2) and an orientation for all edges in $G \setminus C_{in}$ satisfying the three conditions below (depicted in Fig. 3):

- **inner vertices** satisfy the local Schnyder condition as in Definition 2.
- **local condition for** C_{ext}: each vertex $v \in C_{ext}$ has exactly two outgoing edges (of color 0 and 1). When traversing the neighbors of $v \in C_{ext}$ in ccw order around v we encounter: a (possibly empty) set of incoming edges (of color 1), an outgoing edge of color 0, a (possibly empty) set of incoming edges (of color 2), an outgoing edge of color 1, and the remaining edges (if any) are incoming of color 0.

- **local condition for** C_{in}: each vertex $v \in C_{in}$ has one outgoing incident edge e of color 2; all remaining incident edges are incoming at v, being of color 1 if they are at the left of e, and of color 0 otherwise.

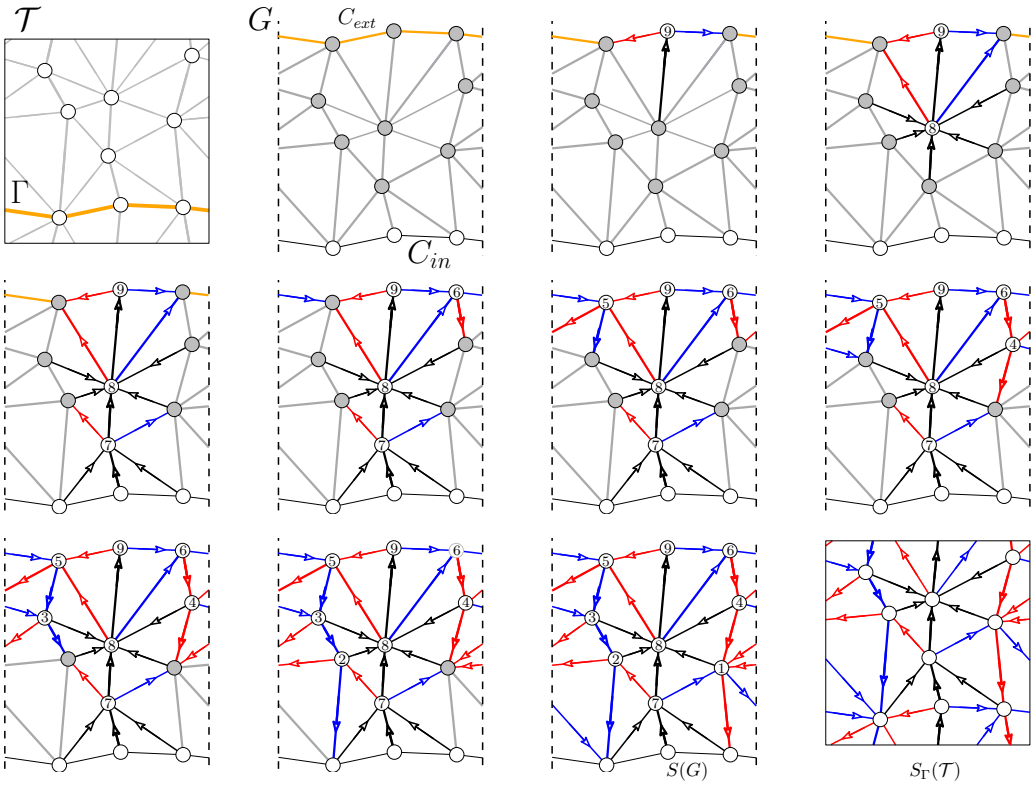


Figure 4 Illustration of Algorithm 1. The toroidal triangulation $\mathcal{T}$ is cut along a non-contractible and chordless cycle Γ. The resulting cylindric triangulation G is endowed with a cylindric canonical ordering (vertices are labeled accordingly), yielding a cylindric Schnyder wood $S(G)$: after gluing C_{in} and C_{ext} we get a toroidal Schnyder wood $S_\Gamma(\mathcal{T})$.

After identification of the cycles C_{ext} and C_{in} we get back the toroidal triangulation $\mathcal{T}$ endowed with a toroidal Schnyder wood $S_\Gamma(\mathcal{T})$ satisfying the additional properties expressed in the Lemma below (the proof is provided in [13]).

▶ **Lemma 3.** *Let $\mathcal{T}$ be a simple toroidal triangulation and Γ a chordless non-contractible cycle. The Schnyder woods $S(G)$ and $S_\Gamma(\mathcal{T})$ computed as above satisfy (see Fig. 4):*

1. *in $S(G)$ the edges of color 2 define a spanning forest of $G \setminus C_{ext}$, whose (directed) trees are rooted on a vertex of C_{ext}; Similarly, the edges of color 0 (resp. 1) define a spanning forest of $G \setminus C_{in}$, whose (directed) trees are rooted on a vertex of C_{in};*
2. *in $S_\Gamma(\mathcal{T})$ all i-cycles cross the cycle Γ and are not homotopic to Γ (for $i \in \{0, 1, 2\}$);*
3. *if $S_\Gamma(\mathcal{T})$ is half-crossing then the 0-cycles and 1-cycles are pairwise crossing.*

Balanced Schnyder woods

As illustrated in Fig. 4 the Schnyder woods computed by Algoritm 1 may fail to be crossing or half-crossing (we will remedy this in the next sections), but as we show here they satisfy the important property of being balanced. A Schnyder wood is called *balanced* if every

non-contractible (undirected) cycle C has the same number of outgoing edges on each side. It is actually sufficient (Lemma 34 in [27]) to have the condition satisfied by two non-contractible cycles that are not homotopic (see [13] for the proof of the statement below).

▶ **Corollary 4.** *The Schnyder woods computed by Algorithm 1 are balanced.*

Balanced Schnyder woods are very instrumental in the bijective encoding [16, 21] of simple toroidal triangulations by decorated unicellular maps (which yields an asymptotically optimal encoding procedure). Indeed the first step of the construction is to compute the so-called *minimal* balanced Schnyder wood of the simple toroidal triangulation $\mathcal{T}$ to be encoded. It turns out that this can easily be done in linear time (by an adaptation of the planar-case arguments given in [7], see also [24, Lemma 3.2]) starting from an arbitrary balanced Schnyder wood of $\mathcal{T}$, as provided by Algorithm 1.

3.1 Constrained cylindric Schnyder woods

Assume we are given a cylindric triangulation G with two boundary cycles C_{in} and C_{ext} (with no chord at C_{in}) and a chordless path $P = \{u_0, u_1, \dots u_{t-1}\}$ originating from vertex $u_0 \in C_{in}$ and ending at $u_{t-1} \in C_{ext}$, such that P intersects the boundary cycles of G only at u_0 and u_{t-1}. We explain here how to adapt the shelling procedure so as to obtain a cylindric Schnyder wood of G such that P is a directed path in color 2.

▶ **Definition 5.** *Let $P = \{u_0, u_1, \dots u_{t-1}\}$ be a path satisfying the conditions above. A cylindric Schnyder wood of G is called* P-constrained *if every edge $(u_i, u_{i+1}) \in P$ is oriented toward u_{i+1} and has color 2.*

We say that a boundary vertex $v \in \partial^e G_k \setminus P$ (for $k \geq 1$) is *strongly free (with respect to P)* if v is free and not adjacent to any vertex of P_k, where $P_k = P \cap (G_k \setminus \partial^e G_k)$ is the restriction of P to the interior of G_k. The vertex $u \in P \cap \partial^e G_k$ is strongly free if it does not have any incident chord at $\partial^e G_k$. We modify the shelling procedure described in the previous section by removing at each shelling step only vertices on $\partial^e G_k$ which are strongly free, as illustrated in Fig. 5 (see [13] for the proof of the Lemma below).

▶ **Lemma 6.** *Given a cylindric triangulation G and a path P defined as above, the adapted shelling procedure outputs a P-constrained Schnyder wood of G, and runs in linear time.*

4 Rivers

We consider here a particular type of "thin" cylindric triangulations called *rivers*, for which we compute a so-called *rigthmost* (resp. *leftmost*) P-constrained Schnyder wood. Rivers, and properties of their rightmost (resp. leftmost) Schnyder wood, are key to obtain the half-crossing (resp. crossing) property, as presented in Section 5 (resp. Section 6).

4.1 Definition of a river

Let us define a *river* R as a non-degenerate simple cylindric triangulation whose outer and inner boundary cycles, denoted $\partial^i R = \{x_0, x_1, \dots, x_{r-1}\}$ and $\partial^e R = \{z_0, z_1, z_2, \dots, z_{s-1}\}$, satisfy the following conditions: $\partial^i R$ and $\partial^e R$ are chordless cycles such that every vertex $z \in \partial^e R$ has an incident edge $(z, \bar{x})$, where $\bar{x} \in \partial^i R$, and every vertex $x \in \partial^i R$ has an incident edge $(x, \bar{z})$, where $\bar{z} \in \partial^e R$ (such edges are called *non-trivial chords* of R).

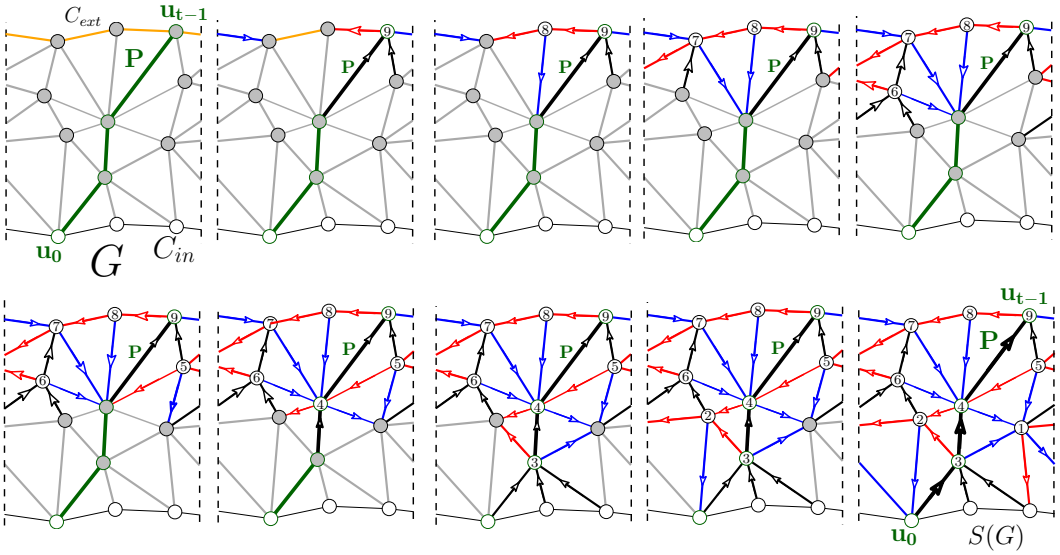


Figure 5 Computation of a P-constrained Schnyder wood (Lemma 6): only vertices which are strongly free (with respect to the path $P = \{u_0, \ldots, u_{t-1}\}$) can be removed at each step.

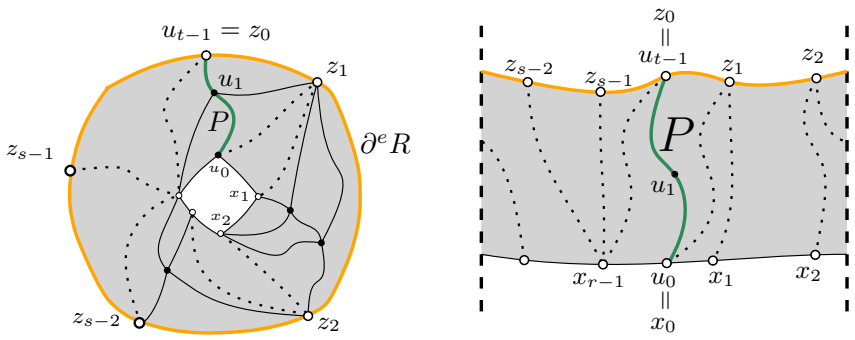


Figure 6 (Left) annular representation of a river R with a chordless path P crossing only once its boundary cycles. Right: only non-trivial chords are drawn, as dotted arcs.

4.2 Right-most (left-most) cylindric canonical ordering of a river

Let us assume that R is provided with a chordless path $P = \{u_0, \ldots, u_{t-1}\}$ intersecting its boundaries only once, at $u_0 = x_0$ and $u_{t-1} = z_0$ (refer to Fig. 6). Observe that the non-trivial chords (u, y), where u is either u_0 or u_{t-1} and $y \in (\partial^i R \cup \partial^e R) \setminus P$, can be partitioned into two (possibly empty) sets, depending on whether (u, y) is at the right or at the left of the path P. We classify the vertex u_{t-1} according to four types (see Fig. 7):

(A): u_{t-1} has chords both at the left and at the right of P;

(B): u_{t-1} has no chords at the left nor at the right of P, so that (u_0, u_{t-1}) is the only chord at u_{t-1};

(C1): u_{t-1} has chords at the right of P, but not at the left of P;

(C2): u_{t-1} has chords at the left of P, but not at the right of P.

Let us now define the *right-most traversal* of R with respect to P (yielding a P-constrained cylindric Schnyder wood) as given by Algorithm 2 below (refer also to Fig. 7 for an illustration):

Algorithm 2 Right-most traversal of a river.

Data: a river R, a path $P = \{u_0, u_1, \ldots, u_{t-1} = z_0\}$
Result: a P-constrained cylindric Schnyder wood of R
if u_{t-1} *is of* ***type (A)*** **then**
  $v \leftarrow z_1$;
else
  $v \leftarrow z_0$;
end
$x_c \leftarrow$ extremity of the left-most chord of v ;
remove v from the outer boundary $\partial^e R$;
while $\partial^e R \neq \partial^i R$ **do**
  $v \leftarrow$ the strongly free vertex on $\partial^e R$ which is closest to x_c at its right;
  remove v from the outer boundary $\partial^e R$;
end

One crucial property is that, after computing a P-constrained Schnyder wood $S_P(R)$ according to a right-most traversal, if u_{t-1} is of type (A), (B) or (C1) then every vertex z on $\partial^e R$ lies on a path of color 1 ending at P. If u_{t-1} is of type (C2) this property may fail, but the right-most traversal leads to the existence of a ccw-oriented contractible cycle enclosed in R. These properties are a consequence of the statement below (see [13] for the proof).

▶ **Lemma 7.** *Let us consider a river R and a chordless path P as defined above. Let $S_P(R)$ be the P-constrained cylindric Schnyder wood computed by Algorithm 2. Then we have:*

1. *if u_{t-1} is of type (A), then for every vertex $z_i \in \partial^e R$ with $1 < i < s$, the path of color 1 starting from z_i passes by u_{t-1};*
2. *if u_{t-1} is of type (B) or (C1), then for every vertex $z \in \partial^e R$, the path of color 1 starting from z ends at u_0;*
3. *if u_{t-1} is of type (C2): then all vertices z_i are lying on a path of color 1 ending either at P or at x_c. Moreover, there exists at least one vertex z_i lying on a ccw-oriented cycle (enclosed in R) defined by $(x_c, u_{t-1}, z_{s-1}, \ldots, z_i)$.*

Left-most traversal. One can define left-most traversals proceeding in a symmetric way with respect to Algorithm 2: at the beginning we set $v = z_{s-1}$ if u_{t-1} is of type (A) and $v = z_0$ otherwise; x_c is set to be the extremity of the right-most chord of v; finally, at each shelling step one removes the strongly free vertex which is closest at the left of x_c.

Similarly to the properties stated in Lemma 7 one can show that if u_{t-1} is of type (A), (B) or (C2) then each vertex z on $\partial^e R$ lies on a path of color 0 ending at P.

5 Algorithm for half-crossing toroidal Schnyder woods

We describe in this section an algorithm to compute a half-crossing toroidal Schnyder wood for any simple toroidal triangulation, with the further property that the mono-chromatic component in color 0 or in color 1 is connected.

5.1 Existence of a river in cylindric triangulations

A key ingredient of the algorithm is the computation of a river in the cylindric triangulation that is obtained by cutting along a chordless non-contractible cycle.

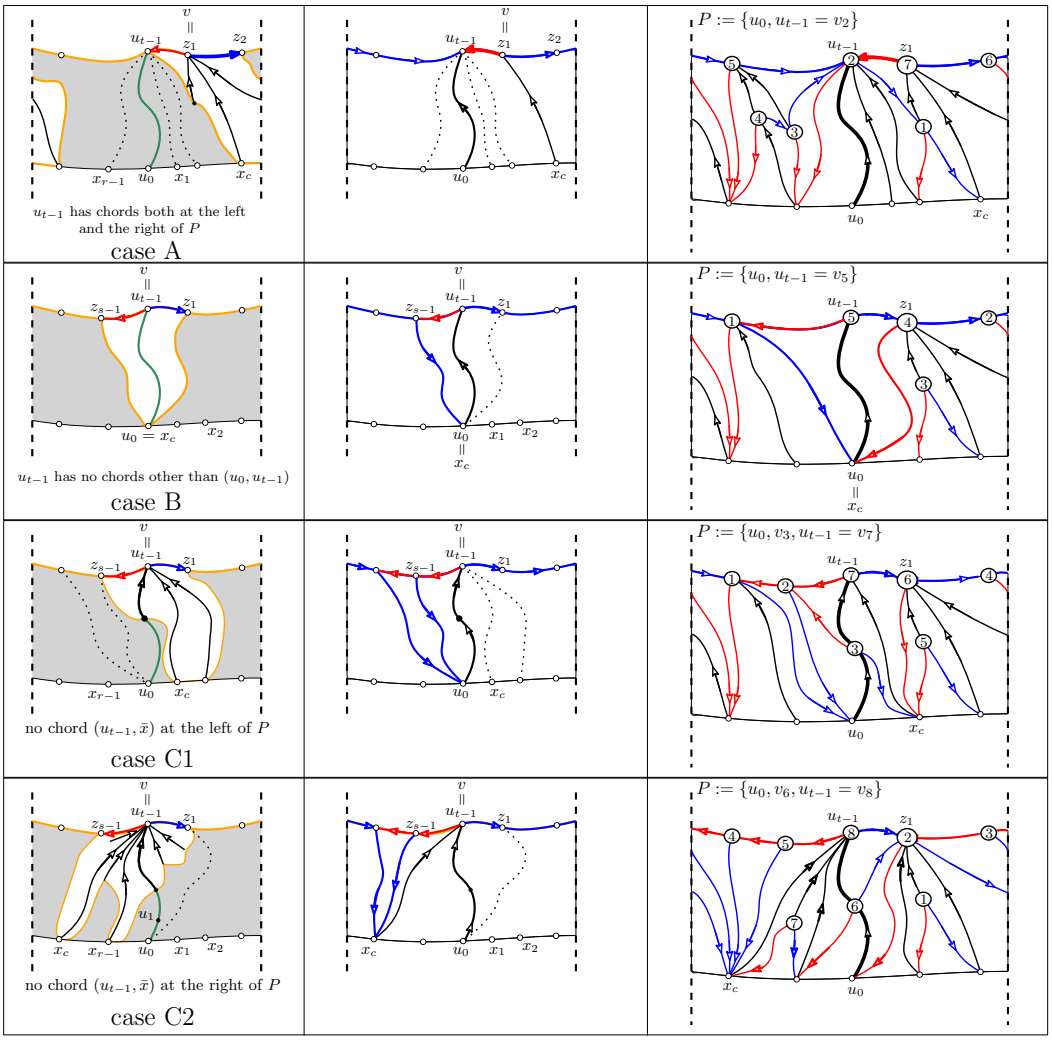


Figure 7 Illustration of Algorithm 2 and Lemma 7: we distinguish four cases for the right-most traversal of a river R. Left pictures: removal of the first vertex on $\partial^e R$. Center pictures: paths of color 1 obtained via a right-most traversal of R. Right pictures: complete execution of Algorithm 2 on four examples (vertex labels are assigned according to the cylindric canonical ordering).

▶ **Lemma 8.** *Given a non-degenerate cylindric triangulation G with no chord at C_{in} or C_{ext}, it is possible to compute in linear time a river R contained in G, whose inner and outer boundaries are homotopic to $\partial^i G$ (the proof is provided in [13]).*

5.2 Algorithm

Let $\mathcal{T}$ be a simple toroidal triangulation. We now describe the algorithm computing a half-crossing Schnyder woods of $\mathcal{T}$; the complete pseudo-code is provided in Algorithm 3, see also Figure 8 for an illustration on an example. After cutting $\mathcal{T}$ along a non-contractible and chordless cycle Γ (as in Section 3), we obtain a non-degenerate cylindric triangulation G, and compute a river R in G (Lemma 8). As a result, the faces of G are partitioned into three regions: G_{top} (a possibly degenerate cylindric triangulation), the river R, and G_{bottom} (a possibly degenerate cylindric triangulation).

Algorithm 3 Half-crossing Schnyder woods (with a connected mono-chromatic component).

Data: a simple toroidal triangulation $\mathcal{T}$, a non-contractible chordless cycle Γ
Result: a half-crossing Schnyder wood
// Pre-processing step
cut $\mathcal{T}$along Γ: let G be the resulting cylindric triangulation ;
compute a river R and the partition $G = G_{top} \cup R \cup G_{bottom}$;
// First pass
compute a Schnyder wood $S(G_{top})$ of G_{top} ;
choose an arbitrary non trivial chord $e = (x, z)$ of R ;
$P \leftarrow \{x, z\}$;
if z *has* ***type (A), (B) or (C1)*** **then**
  run the right-most P-constrained traversal of (R, P);
  $r \leftarrow 1$;
else
  run the left-most P-constrained traversal of (R, P) ;
  $r \leftarrow 0$;
end
compute a Schnyder wood $S(G_{bottom})$ of G_{bottom} ;
glue boundary cycles together and let $S(\mathcal{T}) = S(G_{bottom}) \cup S(R) \cup S(G_{top})$;
if *the r-cycle and 2-cycles are crossing in* $S(\mathcal{T})$ **then**
  return $S(\mathcal{T})$;
end
// Run a second pass on R
$\gamma_2 \leftarrow$ any 2-cycle of $S(\mathcal{T})$; // Remark: the r-cycle and 2-cycles are parallel
$P_2 \leftarrow \gamma_2 \cap R$; // restriction of γ_2 to the river R
$u \leftarrow \partial^e R \cap P_2$;
if u *has* ***type (A), (B) or (C1)*** **then**
  run the right-most P-constrained traversal of (R, P_2);
  $r \leftarrow 1$;
else
  run the left-most P-constrained traversal of (R, P_2) ;
  $r \leftarrow 0$;
end
// Remark: $S(G_{bottom})$ and $S(G_{top})$ are P_2-constrained
glue boundary cycles together and let $S(\mathcal{T}) = S(G_{bottom}) \cup S(R) \cup S(G_{top})$;
return $S(\mathcal{T})$;

The algorithm relies on two passes (the second one not always necessary). In the first pass, we let P be an arbitrary non-trivial chordal edge $\{x, z\}$ for the river R. If z is not of type C2, we perform a rightmost traversal of R, otherwise a leftmost traversal. We also compute in an independent way two (arbitrary) Schnyder woods of G_{top} and G_{bottom}: after boundary identifications we obtain a Schnyder wood $S(T)$ of $\mathcal{T}$. It follows from Lemma 7 (see the proof of Proposition 9) that if z is not (resp. is) of type C2, then there is a single monochromatic cycle C of color 1 (resp. color 0), and moreover it winds only once around Γ (it may share edges with Γ but reaches it only once from the left). If C crosses any cycle of color 2, then we output $S(\mathcal{T})$. Otherwise we need a second pass. Let γ_2 be an arbitrary

2-cycle. Note that γ_2 winds once around Γ since it is homotopic to C. Due to edge colors and local Schnyder conditions, it intersects γ_2 at a single vertex (as opposed to C it can not share edges with Γ). Hence, if we cut γ_2 along Γ, we obtain a path $P_2 \subset G$ which is chordless and meets the boundary cycles only at its extremities. We can then adjust the Schnyder wood in R as follows: if the upper end of P_2 is not of type C2, we perform a rightmost traversal of R with respect to P_2, otherwise a leftmost traversal. This updates $S(\mathcal{T})$ into a half-crossing Schnyder wood, such that in the first (resp. second) case there is a single cycle of color 1 (resp. 0) and it the cycle γ_2 of color 2 (as expressed by Proposition 9).

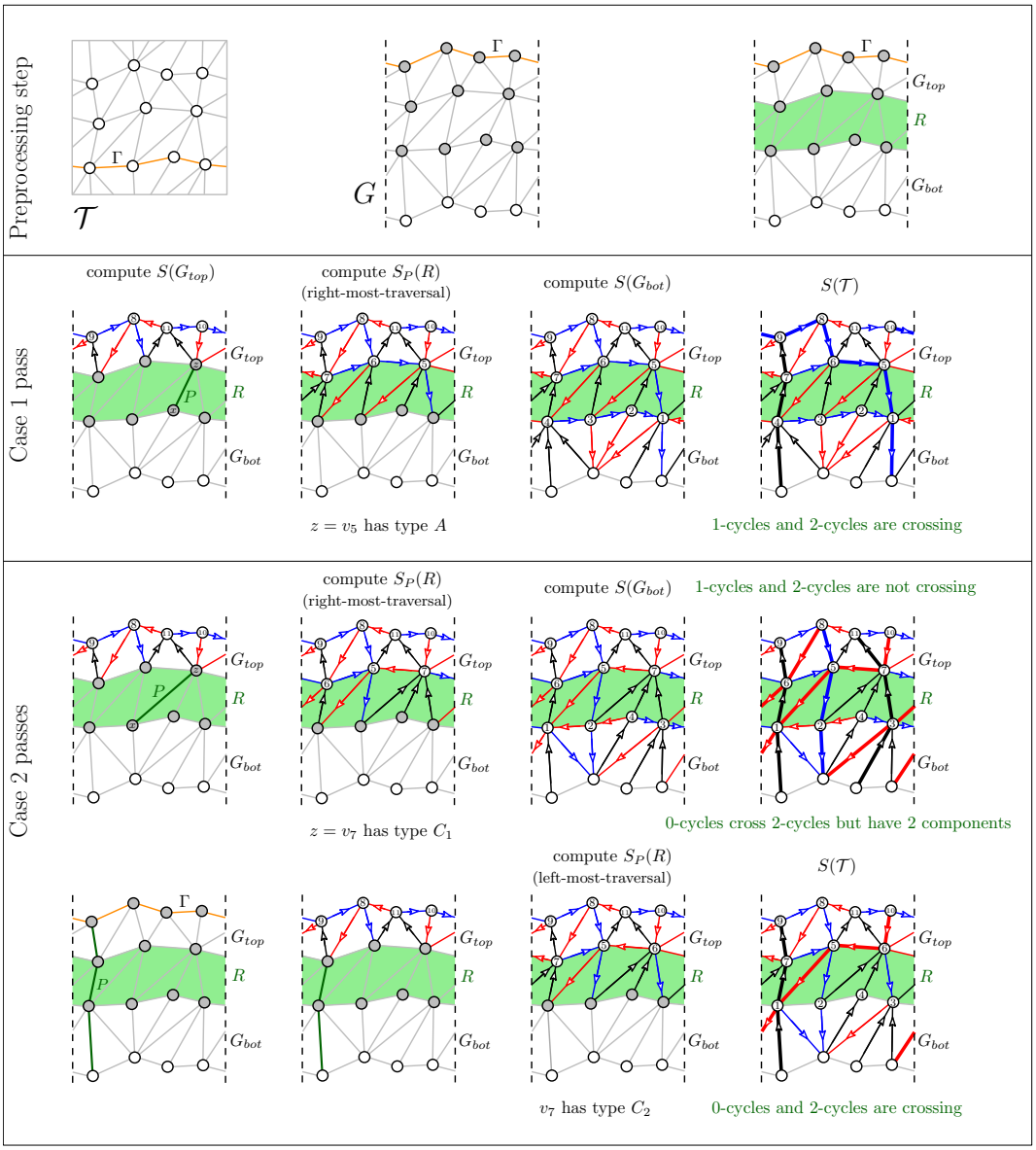


Figure 8 Execution of Algorithm 3 on a small example.

▶ **Proposition 9.** *Given a simple toroidal triangulation $\mathcal{T}$, Algorithm 3 computes in linear time a half-crossing Schnyder wood of $\mathcal{T}$ via vertex shellings. More precisely, there is $r \in \{0, 1\}$ such that the edges of color r define a single connected component, and the unique r-cycle crosses the 2-cycles (see [13] for the proof).*

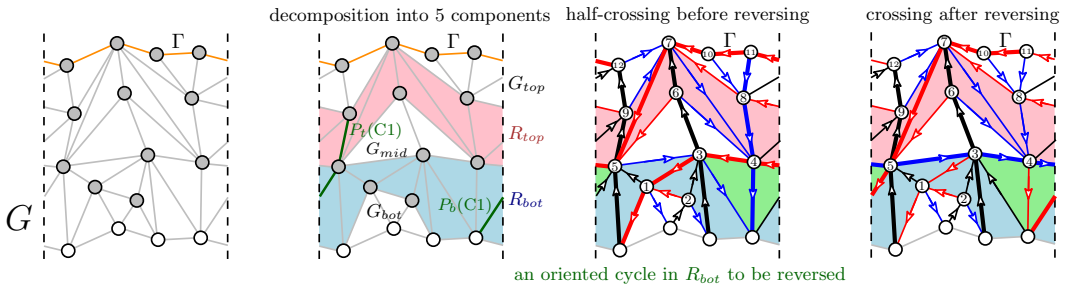


Figure 9 Illustration of the algorithm of Section 6.2 for computing crossing Schnyder woods. At the end of the first phase the Schnyder wood is half-crossing but not crossing (the 1-cycle is parallel to the 2-cycles). But the bottom river contains a ccw-oriented contractible cycle C (enclosing the shaded green region): the reversal of C leads to a 1-cycle parallel to Γ and thus crossing the 2-cycles, making the Schnyder wood crossing. The edges of color 0 and 1 define a connected component.

6 Algorithm for crossing toroidal Schnyder woods

Building on the ideas of Section 5, we give in this section a linear-time algorithm for computing a crossing toroidal Schnyder wood, with the additional property that the mono-chromatic components in color 0 and in color 1 are connected, as stated in Theorem 1.

6.1 Existence of two non-overlapping rivers in cylindric triangulations

Compared to the procedure for half-crossing Schnyder woods, the algorithm for crossing Schnyder woods now requires the computation of two parallel and non-overlapping rivers R_t and R_b in the cylindric triangulation obtained after cutting along a non-contractible and chordless cycle. The existence of such a pair of rivers is guaranteed by the next statement.

▶ **Lemma 10.** *Given a non-degenerate cylindric triangulation G with no chord at C_{in} or C_{ext}, nor between C_{in} and C_{ext}, it is possible in linear time to compute a pair of non-overlapping rivers R_t (top river) and R_b (bottom river) contained in G, whose inner and outer boundaries are homotopic to $\partial^i G$ (see [13] for the proof).*

6.2 Algorithm

Given a toroidal simple triangulation $\mathcal{T}$, as before the first step consists in cutting $\mathcal{T}$ along a chordless non-contractible cycle Γ, yielding a cylindric triangulation G. Since Γ is chordless, the triangulation G has no chord at C_{in} or at C_{ext}, nor chords from C_{in} to C_{ext} (such an edge would correspond to a chord of Γ starting at one side and ending at the other side). Hence we can apply Lemma 10, which yields a decomposition of G into five components (from top to bottom): a top cylindric triangulation G_{top}, the river R_t, a middle cylindric triangulation G_{middle}, the river R_b, and a bottom cylindric triangulation G_{bottom}. Note that G_{top}, G_{middle} and G_{bottom} are possibly degenerate. We then follow the pipeline used in Algorithm 3: one river (resp. the other river) is used to guarantee that color 0 (resp. color 1) is crossing color 2, and that the mono-chromatic component is connected.

Precisely, in a first pass, we choose a chord e_t in R_t that is the leftmost chord at its top vertex, and let $P_t = e_t$ be the path according to which we perform a rightmost traversal of R_t (note that (R_t, P_t) is of type B or C_1). Similarly, we choose a chord e_b in R_b that is the rightmost chord at its top vertex, and we let $P_b = e_b$ be the path according to which we

perform a leftmost traversal of R_b (note that (R_b, P_b) is of type B or C_2). We also compute an arbitrary cylindric Schnyder wood for each of the three components G_{top}, G_{middle}, and G_{bottom}. Then we glue together the Schnyder woods for the five components. Identifying C_{in} with C_{ext} we obtain a toroidal Schnyder wood of $\mathcal{T}$. From Lemma 7 and arguments already used in Section 5, the mono-chromatic component of color 1 (resp. color 0) is connected, thanks to the top-river (resp. bottom river). If the Schnyder wood is crossing we return it as the output.

If it is not crossing we need a second pass, similarly to Section 5. By the 3rd item in Lemma 3 we must have either color 1 parallel to color 2, or color 0 parallel to color 2. In both cases, due to the rivers, the non-contractible cycles of color 2 must wind only once (they cross Γ at a single vertex), so in G such a cycle γ_2 gives a path P_2 from C_{in} to C_{ext}. Keeping the Schnyder woods for G_{top}, G_{middle}, and G_{bottom} unchanged, we need a second pass to adjust those for R_t and R_b in order to obtain the crossing property. Precisely, let P_t (resp. P_b) be the restriction of P_2 to R_t (resp. R_b). If (R_t, P_t) and (R_b, P_b) are not both of type C2 or are not both of type C1, then we can choose one of them, denote it by (R', P'), not to be of type C2, and the other one, denote it by (R'', P''), not to be of type C1. Then we perform a right-most traversal for (R', P'), and a left-most traversal for (R'', P''). After these updates on the Schnyder woods for the two rivers, and by the arguments of Section 5, the toroidal Schnyder wood of $\mathcal{T}$ now has the property that cycles of color 1 (resp. 0) cross cycles of color 2 thanks to R' (resp. R''), and that the monochromatic component of color 1 (resp. 0) are still connected.

It remains to treat the case where at the second pass we have (R_t, P_t) and (R_b, P_b) both of type C1 or both of type C2. Let us assume they are both of type C2. Then we adjust the Schnyder woods in R_t and R_b by performing the leftmost P_t-constrained traversal of R_t, and the rightmost P_b-constrained traversal of R_b. At this point we have the crossing of color 0 with color 2, thanks to the top river and the 2nd item in Lemma 3. In color 1 the crossing property may however fail. But we can take advantage of the 3rd item in Lemma 7, which guarantees the existence of a specific directed cycle C in R_b. Reversing this cycle makes $\partial^e R_b$ a cycle in color 1 that is parallel to Γ.

▷ Claim 11. If in the second phase the rivers (R_t, P_t) and (R_b, P_b) are of type C2, then after reversing the cycle C, the toroidal Schnyder wood of $\mathcal{T}$ is crossing, and the mono-chromatic components of color 0 and of color 1 are connected (see [13] for the proof).

The case where (R_t, P_t) and (R_b, P_b) are both of type C1 can be handled very similarly, by performing the rightmost P_t-constrained traversal of R_t, the leftmost P_b-constrained traversal of R_b, and reversing the specific directed cycle in R_b that is guaranteed by the mirror-statement of the 3rd item in Lemma 3.

This concludes the proof of our main result, as stated in Theorem 1.

▶ Remark 12. An alternative to our algorithm is given by Theorem 13 in [27], which allows to obtain a crossing Schnyder wood from a half-crossing one (such as computed by Proposition 9), however without guarantee of connectedness for the mono-chromatic components. It relies on the repeated reversal of carefully chosen directed cycles. As a comparison, our algorithm needs to reverse at most one directed cycle, and it is easy to implement in linear time (even though the overall proof of correctness is a bit involved).

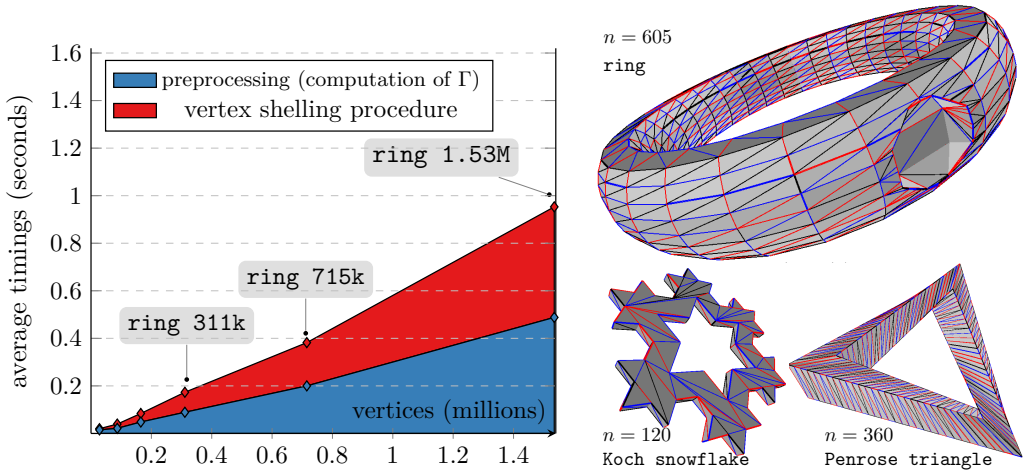


Figure 10 (Left) runtime performances of Algorithm 1: average timings (computed over several runs) are expressed in seconds as a function of the size (millions of vertices). (Right) examples of genus 1 triangle meshes endowed with a toroidal Schnyder wood.

7 Experimental results

7.1 Implementation and experiments

We have written `Java` implementations [5] of our algorithms and performed tests on genus 1 triangle meshes of various sizes (mostly taken from *Thingi10K* repository). Our algorithms for computing toroidal Schnyder woods are significantly simpler to analyze (in particular regarding the linear time complexity) and implement than previous solutions [22]. As confirmed by the runtime plots of Fig. 10, the approach based on vertex shelling leads to a very fast implementation: we can process more than $1M$ vertices per second (our experiments are run on a single core of a `core i7-9700` cpu, allocating $8GB$ memory for the JVM, running `Java 1.8` under Ubuntu Linux).

7.2 Conjectures on higher genus Schnyder woods: empirical results

We have conducted experiments providing an empirical confirmation of three conjectures involving Schnyder woods for triangulations in genus 1 and genus 2. We wrote a `Java` program that exhaustively generates all possible 3-orientations (see definition below in genus more than 1) and Schnyder woods for triangulations of genus $g \leq 2$. Our generator is fast enough to process all tiny triangulations (generated using `surftri` [34], see Table 1) in a reasonable amount of time, allowing us to compute some statistics and test some conjectures.

Connectedness of the three mono-chromatic graphs in genus 1

We have checked that all toroidal simple triangulations of size at most 11 admit a Schnyder wood for which the edges of the same color define a connected graph (having thus 3 mono-chromatic cycles, one for each color), providing an empirical answer to an open question stated in [22] (our algorithm of Section 6 guarantees a positive answer in two of the three colors). We also consider a second open question asked in [22] concerning the existence of

[5] Datasets, runnable programs and source code (pure Java) are made available at: `www.lix.polytechnique.fr/~amturing/software.html`.

Table 1 Simple triangulations of genus 1 and 2 (enumerated using `surftri` [34]).

g=1	irreducible triang.	all triangulations
n=7	1	1
n=8	4	7
n=9	15	112
n=10	1	2109
n=11	-	37867

g=2	irreducible triang.	all triangulations
n=7	-	-
n=8	-	-
n=9	-	-
n=10	865	865
n=11	26276	113506

crossing Schnyder woods satisfying a stronger requirement (referred to as *connected* 3*-cycles property*): the Schnyder wood is crossing, the three mono-chomatic components consist of a single connected component and the intersection of the three non trivial cycles is a single vertex [6]. Our experiments provide an empirical positive answer to this question (for $n \leq 11$): we report in Figure 11 some statistics and the list of irreducible [7] triangulations of genus 1, each endowed with a crossing Schnyder wood satisfying this stronger condition.

Existence of Schnyder woods in genus 2

For a simple triangulation of genus $g \geq 1$, a *3-orientation* is an orientation such that for every vertex v the out-degree is divisible by 3 and at least 3 (no sinks). In genus $g \geq 2$ some vertices must have degree more than 3, since $\sum_v(\deg(v) - 3) = 6g - 6$ according to the Euler relation. It is shown in [1] that any simple triangulation of genus $g \geq 2$ admits a 3-orientation. The following natural open question (asked in [27]) then arises: for every simple triangulation of genus $g \geq 2$, is it always possible to find a 3-orientation for which the edges can be colored so as to satisfy a *multiple* local Schnyder condition at every vertex (as depicted in Fig. 12)? In his PhD dissertation J. Suagee [33] shows that the answer is positive for every simple triangulation of genus $g \geq 2$ having edge-width at least $40(2^g - 1)$. We have checked that the answer is actually positive for every simple triangulations of genus 2 and $n \leq 11$, suggesting that the assumption on the large edge-width is maybe unnecessary.

8 Concluding remarks

We expect that the construction of Section 3 can be extended to 3-connected toroidal maps using the notion of cylindric toroidal canonical ordering defined for the 3-connected case in [12], leading to a fast linear-time algorithm. As in the triangulated case, the obtained Schnyder woods should have the property of being balanced, which is instrumental in the bijective encoding of 3-connected toroidal maps [3] (indeed it relies on the computation of the *minimal* balanced Schnyder wood of the map, which can easily be obtained in linear time starting from an arbitrary balanced Schnyder wood).

[6] Observe that Thm. 9 in [22] guarantees the existence of three edge-disjoint mono-chromatic cycles crossing at a single vertex, but each mono-chromatic graph may consist of several disjoint components.
[7] A triangulation is irreducible if no edge can be contracted without producing multiple edges.

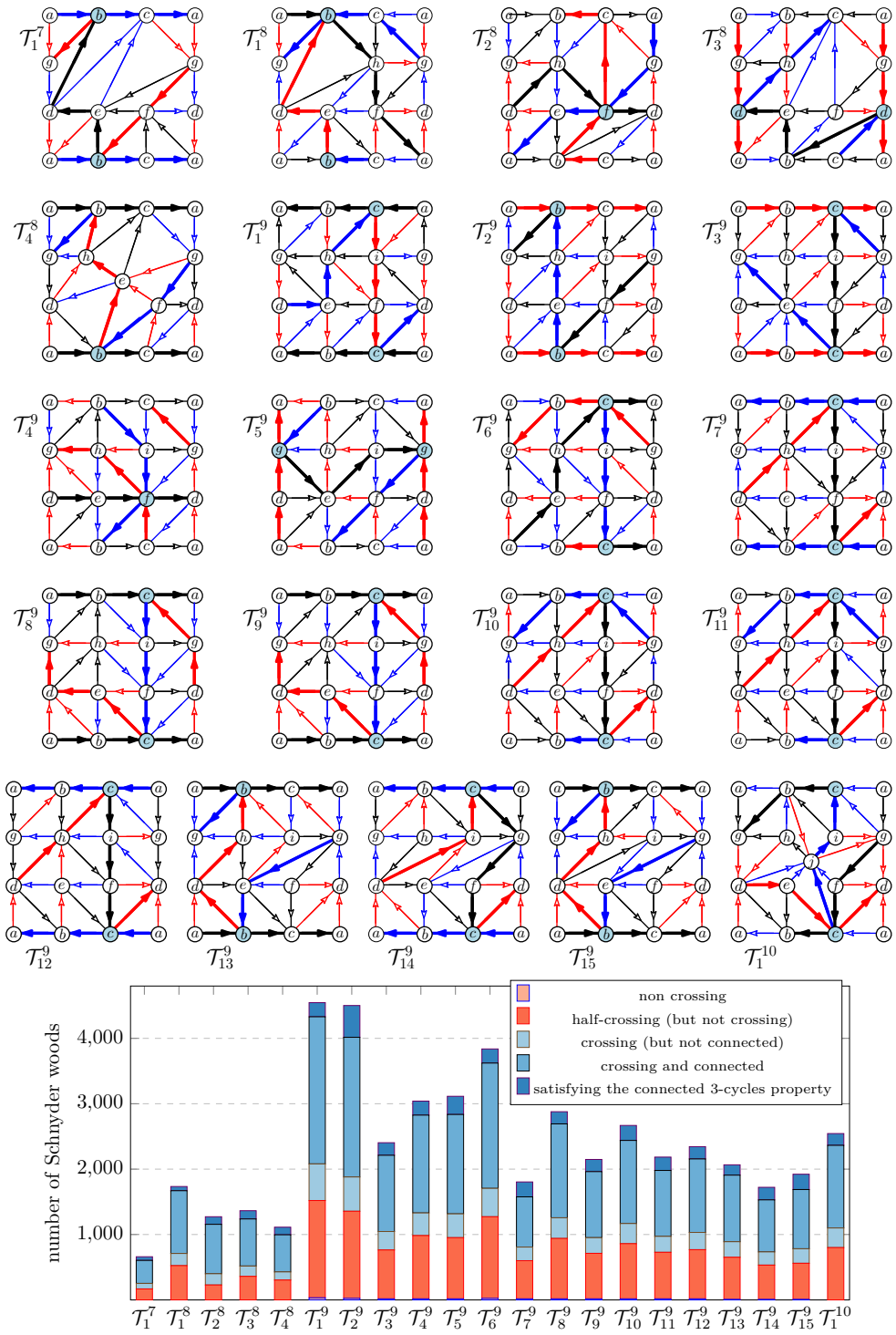


Figure 11 List of genus 1 irreducible triangulations (the embeddings are provided in [28]). Each triangulation is endowed with a Schnyder wood satisfying the connected 3-cycles property computed by our generator. The bottom plots report some statistics computed by our generator.

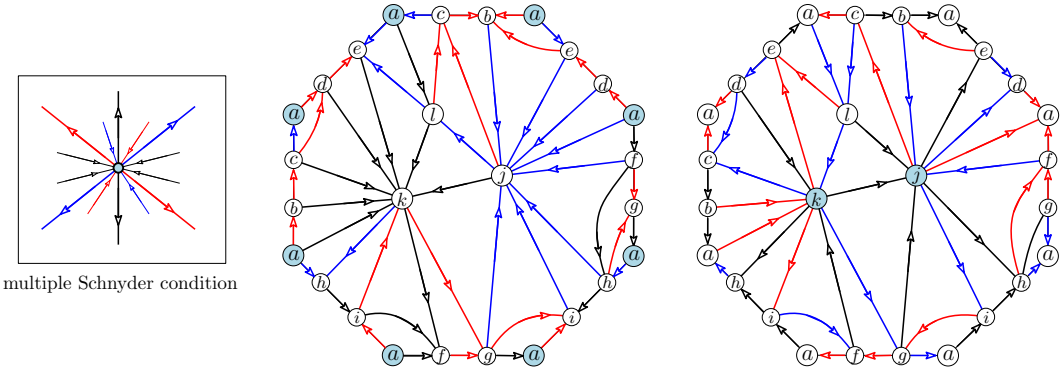


Figure 12 Planar layout of a genus 2 triangulation: it is endowed with two Schnyder woods satisfying the multiple Schnyder condition computed by our generator. In the left case vertex a has outgoing degree 9, while in the right one vertices j and k have outgoing degree 6. This triangulation admits 16.421.904 3-orientations and 509844 valid Schnyder woods.

References

1 Boris Albar, Daniel Gonçalves, and Kolja Knauer. Orienting triangulations. *J. Graph Theory*, 83(4):392–405, 2016. doi:10.1002/jgt.22005.

2 Olivier Bernardi and Nicolas Bonichon. Catalan's intervals and realizers of triangulations. *Journal of Combinatorial Theory, Series A*, 116(1):55–75, 2009. 22 pages. URL: https://hal.archives-ouvertes.fr/hal-00143870.

3 Nicolas Bonichon, Éric Fusy, and Benjamin Lévêque. A bijection for essentially 3-connected toroidal maps. *Eur. J. Comb.*, 95:103290, 2021. doi:10.1016/J.EJC.2020.103290.

4 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perkovic. Plane spanners of maximum degree six. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010*, pages 19–30, 2010. doi:10.1007/978-3-642-14165-2_3.

5 Nicolas Bonichon, Cyril Gavoille, and Arnaud Labourel. Edge partition of toroidal graphs into forests in linear time. In *ICGT*, volume 22, pages 421–425, 2005. doi:10.1016/J.ENDM.2005.06.065.

6 Nicolas Bonichon and Benjamin Lévêque. A bijection for essentially 4-connected toroidal triangulations. *Electron. J. Comb.*, 26(1):1, 2019. doi:10.37236/7897.

7 U. Brandes and D. Wagner. A linear time algorithm for the arc disjoint menger problem in planar directed graphs. *Algorithmica*, 28(1):16–36, 2000. doi:10.1007/s004530010029.

8 Enno Brehm. 3-orientations and Schnyder 3-tree-decompositions. *Master's Thesis, FB Mathematik und Informatik, Freie Universität Berlin*, 2000.

9 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding cycles with topological properties in embedded graphs. *SIAM J. Discret. Math.*, 25(4):1600–1614, 2011. doi:10.1137/100810794.

10 Luca Castelli Aleardi and Olivier Devillers. Array-based compact data structures for triangulations: Practical solutions with theoretical guarantees. *JoCG*, 9(1):247–289, 2018. doi:10.20382/jocg.v9i1a8.

11 Luca Castelli Aleardi and Olivier Devillers. SCARST: Schnyder compact and regularity sensitive triangulation data structure. In *40th Intern. Symp. on Computational Geometry (SoCG 2024)*, volume 293 of *LIPIcs*, pages 32:1–32:19, 2024. doi:10.4230/LIPICS.SOCG.2024.32.

12 Luca Castelli Aleardi, Olivier Devillers, and Éric Fusy. Canonical ordering for graphs on the cylinder, with applications to periodic straight-line drawings on the flat cyclinder and torus. *J. Comput. Geom.*, 9(1):391–429, 2018. doi:10.20382/jocg.v9i1a14.

13 Luca Castelli Aleardi, Éric Fusy, Jyh-Chwen Ko, and Razvan-Stefan Puscasu. Computation of toroidal schnyder woods made simple and fast: from theory to practice. preprint, 2025. URL: https://hal.science/hal-05027238.

14 Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. Schnyder woods for higher genus triangulated surfaces. In *Proc. of the 24th ACM Symp. on Comput. Geom. (SoCG)*, pages 311–319. ACM, 2008. doi:10.1145/1377676.1377730.

15 Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. Schnyder woods for higher genus triangulated surfaces, with applications to encoding. *Discrete & Computational Geometry*, 42(3):489–516, 2009. doi:10.1007/s00454-009-9169-z.

16 Vincent Despré, Daniel Gonçalves, and Benjamin Lévêque. Encoding toroidal triangulations. *Discrete & Computational Geometry*, 57(3):507–544, 2017. doi:10.1007/s00454-016-9832-0.

17 Raghavan Dhandapani. Greedy drawings of triangulations. *Discrete & Computational Geometry*, 43(2):375–392, 2010. doi:10.1007/s00454-009-9235-6.

18 Stefan Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18(1):19–37, 2001. doi:10.1023/A:1010604726900.

19 Stefan Felsner. Geodesic embeddings and planar graphs. *Order*, 20(2):135–150, 2003. doi:10.1023/B:ORDE.0000009251.68514.8b.

20 Stefan Felsner and Florian Zickfeld. On the number of planar orientations with prescribed degrees. *Electr. J. Comb.*, 15(1), 2008. URL: http://www.combinatorics.org/Volume_15/Abstracts/v15i1r77.html.

21 Éric Fusy and Benjamin Lévêque. Orientations and bijections for toroidal maps with prescribed face-degrees and essential girth. *J. Comb. Theory A*, 175:105270, 2020. doi:10.1016/J.JCTA.2020.105270.

22 Daniel Gonçalves and Benjamin Lévêque. Toroidal maps: Schnyder woods, orthogonal surfaces and straight-line representations. *Discrete & Computational Geometry*, 51(1):67–131, 2014. doi:10.1007/s00454-013-9552-7.

23 Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. doi:10.1007/BF02086606.

24 Samir Khuller, Joseph Naor, and Philip N. Klein. The lattice structure of flow in planar graphs. *SIAM J. Discret. Math.*, 6(3):477–490, 1993. doi:10.1137/0406038.

25 Kolja Knauer, Daniel Gonçalves, and Benjamin Lévêque. On the structure of Schnyder woods on orientable surfaces. *J. Comput. Geom.*, 10(1):127–163, 2019. doi:10.20382/jocg.v10i1a5.

26 Stephen G. Kobourov. Canonical orders and schnyder realizers. In *Encyclopedia of Algorithms*, pages 277–283. Springer, 2016. doi:10.1007/978-1-4939-2864-4_650.

27 Benjamin Lévêque. *Generalization of Schnyder woods to orientable surfaces and applications*. Hdr thesis, Université Grenoble Alpes, 2016. URL: https://tel.archives-ouvertes.fr/tel-01488943.

28 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins, 2001.

29 Dominique Poulalhon and Gilles Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46(3-4):505–527, 2006. doi:10.1007/s00453-006-0114-8.

30 Neil Robertson and Paul D. Seymour. Graph minors. VI. disjoint paths across a disc. *J. Comb. Theory, Ser. B*, 41(1):115–138, 1986. doi:10.1016/0095-8956(86)90031-6.

31 Walter Schnyder. Planar graphs and poset dimension. *Order*, pages 323–343, 1989.

32 Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 90, pages 138–148, 1990. URL: http://departamento.us.es/dma1euita/PAIX/Referencias/schnyder.pdf.

33 Jason Suagee. *Existence and Construction of Schnyder Orientations over a Large Class of Higher Genus Surface Triangulations*. PhD thesis, George Washington University, 2021.

34 Thom Sulanke. Generating maps on surfaces. *Discret. Comput. Geom.*, 57(2):335–356, 2017. doi:10.1007/s00454-016-9853-8.

A Linear Time Algorithm for the Maximum Overlap of Two Convex Polygons Under Translation

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Isaac M. Hair
Department of Computer Science, University of California, Santa Barbara, CA, USA

Abstract

Given two convex polygons P and Q with n and m edges, the *maximum overlap* problem is to find a translation of P that maximizes the area of its intersection with Q. We give the first randomized algorithm for this problem with linear running time. Our result improves the previous two-and-a-half-decades-old algorithm by de Berg, Cheong, Devillers, van Kreveld, and Teillaud (1998), which ran in $O((n+m)\log(n+m))$ time, as well as multiple recent algorithms given for special cases of the problem.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Convex polygons, shape matching, prune-and-search, parametric search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.31

Funding *Timothy M. Chan*: Work supported by NSF Grant CCF-2224271.

1 Introduction

Problems related to convex polygons are widely studied in computational geometry, as they are fundamental and give insights to more complex problems. We consider one of the most basic problems in this class:

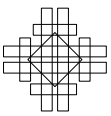
▶ **Problem 1.** *Given two convex polygons P and Q in the plane, with n and m edges respectively, find a vector $t \in \mathbb{R}^2$ that maximizes the area of $(P+t) \cap Q$, where $P+t$ denotes P translated by t.*

This problem for convex polygons was first explicitly posed as an open question by Mount, Silverman, and Wu [34]. In 1998, de Berg, Cheong, Devillers, van Kreveld, and Teillaud [21] presented the first efficient algorithm which solves the problem in $O((n+m)\log(n+m))$ time. As many problems have $n \log n$ complexity, this would appear to be the end of the story for the above problem – or is it?

Motivation and Background. The computational geometry literature is replete with various types of problems about polygons [36], e.g., polygon containment [14] (which has seen exciting development even in the convex polygon case, with translations and rotations, as recently as last year's SoCG [13]). The maximum overlap problem may be viewed as an outgrowth of polygon containment problems.

The maximum overlap problem may also be viewed as a formulation of *shape matching* [7], which has numerous applications and has long been a popular topic in computational geometry. Many notions of distance between shapes have been used in the past (e.g., Hausdorff distance, Fréchet distance, etc.), and the overlap area is another very natural measure (the larger it is, the closer the two shapes are) and has been considered in numerous papers for different classes of objects (e.g., two convex polytopes in higher dimensions [3, 5, 4], two arbitrary polygons in the plane [18, 25], two unions of balls [10], one convex polygon vs. a discrete

© Timothy M. Chan and Isaac M. Hair;
licensed under Creative Commons License CC-BY 4.0



point set[1] [9, 1], etc.) as well as different types of motions allowed (translations, rotations, and scaling – with translations-only being the most often studied). Problem 1 is perhaps the simplest and most basic version, but is already quite fascinating from the theoretical or technical perspective, as we will see.

A Superlinear Barrier? Due to the concavity of (the square root of) the objective function [21], it is not difficult to obtain an $O((n+m)\,\mathrm{polylog}(n+m))$-time algorithm for the problem by applying the well-known *parametric search* technique [30, 2] (or more precisely, a multidimensional version of parametric search [38]). De Berg, Cheong, Deviller, van Kreveld, and Teillaud [21] took more care in eliminating extra logarithmic factors to obtain their $O((n+m)\log(n+m))$-time algorithm, by avoiding parametric search and instead using more elementary binary search and matrix search techniques [24]. But any form of binary search would seem to generate at least one logarithmic factor, and so it is tempting to believe that their time bound might be the best possible.

On the other hand, in the geometric optimization literature, there is another well-known technique that can give rise to *linear*-time algorithms for various problems: namely, *prune-and-search* (pioneered by Megiddo and Dyer [31, 22, 32]). For prune-and-search to be applicable, one needs a way to prune a fraction of the input elements at each iteration, but our problem is sufficiently complex that it is unclear how one could throw away any vertex from the input polygons and preserve the answer.

Perhaps because of these reasons, no improved algorithms have been reported since de Berg et al.'s 1998 work. This is not because of lack of interest in finding faster algorithms. For example, a (lesser known) paper by Kim, Choi, and Ahn [28] considered an unbalanced case of Problem 1 and gave an algorithm with $O(n+m^2\log^3 n)$ running time, which is linear when $m \ll \sqrt{n/\log^3 n}$. Ahn et al. [6] investigated *approximation* algorithms for the problem and obtain logarithmic running time for approximation factor arbitrarily close to 1, while Har-Peled and Roy [25] more generally gave a linear-time algorithm to approximate the entire objective function, which was then used to obtain a linear-time approximation algorithm for an extension of the problem to polygons that are decomposable into $O(1)$ convex pieces.

Our Result. We obtain a randomized (exact) algorithm for Problem 1, running in *linear* (i.e., $O(n+m)$) expected time! This is the first improvement to de Berg et al.'s result [21] in 26 years, and is clearly optimal, and so fully settles the complexity of Problem 1.

A New Kind of Prune-and-Search? Besides our result itself, the way we obtain linear time is also interesting. As mentioned, if one does not care about extra $\log(n+m)$ factors, one can use (multidimensional) parametric search to solve the problem. Specifically, parametric search allows us to reduce the optimization problem to the implementation of a *line oracle* (determining which side of a given line the optimal point t^* lies in), which in turn can be reduced to a *point oracle* (evaluating the area of $(P+t)\cap Q$ for a given point t). A point oracle can be implemented in linear time by computing the intersection $(P+t)\cap Q$ explicitly. (In de Berg et al.'s work [21], they directly implemented the line oracle in linear time.)

Various linear-time prune-and-search algorithms (e.g., for low-dimensional linear programming [31, 22, 32], ham-sandwich cuts [33], centerpoints [26], Euclidean 1-center [23], rectilinear 1-median [39, 35], etc.) also exploit line or point oracles; using (in modern parlance)

[1] In the discrete case, we are maximizing the number of points inside the translated polygon.

cuttings [16], such oracle calls let them refine the search space and eliminate a fraction of the input at each iteration. Unfortunately, for our problem, we can't technically eliminate any part of the input polygons P and Q. Instead, our new idea is to reduce the *cost* of the oracles iteratively as the search space is refined, which effectively is as good as reducing the input complexity itself. As the algorithm progresses, oracles get cheaper and cheaper, and we will bound the total running time by a geometric series.

To execute this strategy, one should think of oracles as data structures with sublinear query time (i.e., sublinear in the original input size $n + m$). Our key idea to designing such data structures is to divide the input polygons into *blocks* of a certain size based on angles/slopes. This idea is inspired by the recent work of Chan and Hair [13] (they studied the convex polygon containment problem with both translations and rotations, and although our problem without rotations might seem different, their divide-and-conquer algorithms also crucially relied on division of the input convex polygons by angles/slopes). We show that at each iteration, as we refine the search space, we can use the current "block data structure" to obtain a better "block data structure" with a larger block size. (The refinement process from one block structure to another block structure shares some general similarities with the idea of *fractional cascading* [17].)

We are not aware of any prior algorithms in the computational geometry literature that work in the same way (which makes our techniques interesting). A linear-time algorithm by Chan [12] for a matrix searching problem (reporting all elements in a Monge matrix less than given value) has a recurrence similar to the one we obtain here, but this appears to be a coincidence. An algorithm for computing so-called "ε-approximations" in range spaces by Matoušek [29] also achieved linear time (for constant ε) by iteratively increasing block sizes, but the algorithm still operates in the realm of traditional prune-and-search (reducing actual input size). Chazelle's infamous linear-time algorithm for triangulating simple polygons [15] also iteratively refines a "granularity" parameter analogous to our block size, and as one of many steps, also requires the implementation of oracles (for ray shooting, in his case) whose cost similarly varies as a function of the granularity (at least from a superficial reading of his paper) – fortunately, our algorithm here will not be as complicated!

Remarks. Recently, Jung, Kang, and Ahn [27] have announced a *linear*-time algorithm for a related problem: given two convex polygons P and Q in the plane, find a vector $t \in \mathbb{R}^2$ that minimizes the area of the convex hull $(P + t) \cup Q$. Although the problem may look similar to our problem on the surface, it is in many ways different: in the minimum convex hull problem, the optimal point t^* is known to lie on one of $O(n + m)$ candidate lines (formed by an edge of one polygon and a corresponding extreme vertex of the other polygon), whereas in the maximum overlap problem (Problem 1), t^* lies on one of $O((n + m)^2)$ candidate lines (defined by pairs of edges of the two polygons). Thus a more traditional prune-and-search approach can be used to solve the minimum convex hull problem, but not the maximum overlap problem. Indeed, Jung et al.'s paper [27] considered the maximum overlap problem, but they do not give an improvement over de Berg et al.'s $O((n + m) \log(n + m))$ time algorithm for the two-dimensional case. Instead, they give some new results for the problem in higher dimensions d, but their time bound is $O(n^{2\lfloor d/2 \rfloor})$, which is much larger than linear.

We observe that, in the specific case of maximum overlap for convex polytopes in dimension $d = 3$, one can obtain a significantly faster $O((n + m) \operatorname{polylog}(n + m))$-time algorithm by directly using multi-dimensional parametric search [38], since a point oracle (computing the area of the intersection of two convex polytopes in $\mathbb{R}^3$) can be done in $O(n+m)$ time by explicitly computing the intersection [15, 11], and this is parallelizable with

$O((n+m)\operatorname{polylog}(n+m))$ work and $O(\operatorname{polylog}(n+m))$ steps by known parallel algorithms for intersection of halfspaces in $\mathbb{R}^3$, i.e., convex hull in $\mathbb{R}^3$ in the dual (e.g., see [8]). This observation seems to have not appeared in literature before (including [40, 27]).

2 Preliminaries

We may assume that P and Q have no (adjacent) parallel edges, as these can be merged. Without loss of generality, P and Q both contain the origin as an interior point. Throughout, we assume that the edges of P and Q are given in counterclockwise order. We define $N := n + m$, and we use $[x]$ to denote the set $\{1, \ldots, x\}$. If x is not an interger, then we use $[x]$ to denote the set $\{1, \ldots, \lceil x \rceil\}$, i.e., x is implicitly rounded up to the nearest integer. "*With high probability*" means "with probability at least $1 - n^{-\Omega(1)}$." The *interior* of a triangle refers to all points strictly inside of its bounding edges, and the *interior* of a line segment refers to all points strictly between its bounding points.

The Objective Function. We want to find a translation maximizing

$$\operatorname{Area}((P + t) \cap Q).$$

De Berg, Cheong, Devillers, van Kreveld, and Teillaud showed several interesting properties of the area as a function of t [21]. For our purposes, we only need to import the following.

▶ **Lemma 1.** *$\sqrt{\operatorname{Area}((P + t) \cap Q)}$ is downward concave over all values of t that yield nonzero overlap of $P + t$ with Q.*

The restriction on t is important, as $A(t)$ suddenly transitions from a convex function to a constant function $A(t) = 0$ when t is outside of the specified region. For our algorithm, however, this technicality will have little impact.

Cuttings. First, we give a definition of the type of cuttings we use.

▶ **Definition 2** (ε-Cuttings). *A cutting for a set X of n hyperplanes in $\mathbb{R}^d$, for any $d \geq 1$, is a partition of $\mathbb{R}^d$ into simplices such that the interior of every simplex intersects at most $\varepsilon \cdot n$ hyperplanes of X.*

Endowed with the ability to sample random hyperplanes, we can actually produce such a cutting with very high probability in sublinear time, as per Clarkson and Shor [19, 20] (roughly, we can take logarithmically many random hyperplanes, and return a canonical triangulation of their arrangement).

▶ **Lemma 3.** *Let X be a set of n hyperplanes in $\mathbb{R}^d$, for any $d \geq 1$. Assume we can sample a uniformly random member of X in expected time $O(n^{0.1})$. Let ε be any constant. Then in $O(n^{0.1+o(1)})$ expected time, we can find a set of $O(1)$ simplices that constitutes an ε-cutting of X with high probability.*

Area Prefix Sums. Let $v_1, \ldots, v_n$ be the vertices of P listed in counterclockwise order, and let $v'_1, \ldots, v'_m$ be the vertices of Q listed in counterclockwise order. Let $P[v_x : v_y]$ denote the convex hull of the set of points containing: (i) the origin, (ii) the vertices v_x and v_y, and (iii) all vertices encountered when traversing the boundary of P in counterclockwise order from v_x to v_y. We define $Q[v'_x : v'_y]$ analogously.

By a standard application of prefix sums, we can produce a data structure to report $\text{Area}(P[v_x : v_y])$ and $\text{Area}(Q[v'_x : v'_y])$ with $O(N)$ preprocessing time and $O(1)$ query time. This is since (taking $\text{Area}(P[v_x : v_y])$ as an example):

$$\text{Area}(P[v_x : v_y]) = \text{Area}(P[v_1 : v_y]) - \text{Area}(P[v_1 : v_x]) \text{ when } y > x, \text{ and}$$
$$\text{Area}(P[v_x : v_y]) = \text{Area}(P) - (\text{Area}(P[v_1 : v_x]) - \text{Area}(P[v_1 : v_y])) \text{ when } y < x.$$

Throughout, we refer to the above data structure as the *area prefix sums for P and Q*.

The Configuration Space. Consider two convex polygons A and B, or more generally, two polygonal chains A and B. Let e be an edge of A and let e' be an edge of B. These two edges define a parallelogram $\pi(e, e')$ in configuration space such that, for any vector $t \in \mathbb{R}^2$, we have that $e + t$ intersects e' iff $t \in \pi(e, e')$. Let $\partial\pi(e, e')$ denote the set of four line segments defining $\pi(e, e')$. We define the set of all such line segments as

$$\Pi(A, B) := \bigcup_{\text{edge } e \text{ of } A, \text{ edge } e' \text{ of } B} \partial\pi(e, e'). \tag{1}$$

We define $\overleftrightarrow{\Pi}(A, B)$ to be $\Pi(A, B)$ where each line segment is extended to a line.

Angles and Angle Ranges. The *angle* of an edge e on the boundary of an arbitrary convex polygon A, denoted $\theta_A(e)$, is the angle (measured counterclockwise) starting from a horizontal rightward ray and ending with a ray coincident with e that has A on its left side. We always have that $\theta_A(e) \in [0, 2\pi)$. For any subset X of the edges of A, let $\Lambda_A(X)$ denote the minimal interval in $[0, 2\pi)$ such that $\theta_A(e) \in \Lambda_A(X)$ for all $e \in X$. We call $\Lambda_A(X)$ the *angle range* for the subset X with respect to A.

3 Blocking Scheme

Given a parameter $b \in \mathbb{N}$, referred to as the *block size*, we define a partition[2] of P and Q into *blocks* $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ as follows. First (implicitly) sort the set[3]

$$\{\theta_P(e) : e \text{ is an edge of } P\} \cup \{\theta_Q(e) : e \text{ is an edge of } Q\}$$

in increasing order and then partition the resulting sequence into consecutive subsequences $S_1, \ldots S_{\lceil N/b \rceil}$ of length b each (except possibly the last one). We then define P_i as the convex hull of the origin and the set of edges[4] $\{e : e \text{ is an edge of } P, \text{ and } \theta_P(e) \in S_i\}$. We define Q_j similarly. See Figure 1 for an example.

Now, b uniquely defines the partition, so we refer to $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ as the *blocks determined by parameter b*. This scheme is useful because the angle ranges $\Lambda_P(E_P(P_i))$ and $\Lambda_Q(E_Q(Q_j))$ are strictly disjoint whenever $i \neq j$, where $E_P(P_i)$ is the set of edges that P_i shares with P, and $E_Q(Q_j)$ is the set of edges that Q_j shares with Q. We call $E_P(P_i)$ the *restriction of P_i's edges to P*, and similarly for $E_Q(Q_j)$.

[2] Actually, it isn't quite a partition because the boundaries may overlap, but these have zero area.
[3] We assume that no angles are shared between the edges of P and Q, as this case can be handled easily but requires a clumsier definition.
[4] The set of edges may be empty, in which case the block is just a single point (the origin).

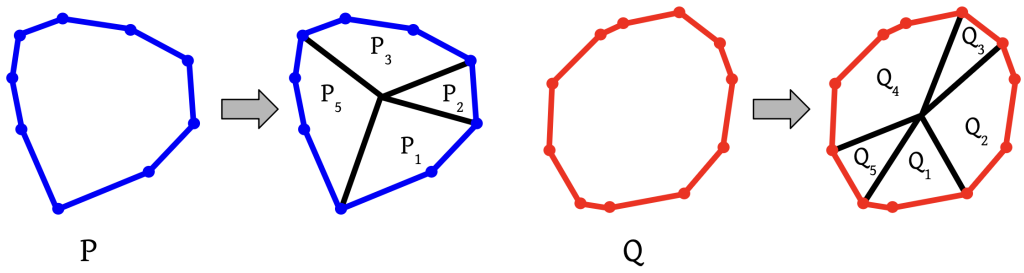


Figure 1 A partition of example polygons P and Q into blocks, with $b = 4$. Note that the spatial *sizes* of blocks may vary greatly, and the number of edges in each P_i and Q_j may differ. In fact, for some indices i, one of P_i or Q_j may contain just the origin.

3.1 A Data Structure Using Blocks

Towards producing a prune-and-search style algorithm, we want some way to reduce the time required to compute the vector t^* which maximizes $\text{Area}((P + t^*) \cap Q)$. Instead of reducing the complexity of P or Q directly, we maintain and refine a specific data structure that contains information about P and Q and allows us to rapidly search for optimal placements.

▶ **Definition 4** ($(\mu, b, \mathcal{T})$-Block Structure). *A $(\mu, b, \mathcal{T})$-block structure is a data structure containing the following:*

1. *Access to the vertices of P and Q in counterclockwise order with $O(1)$ query time, and access to the area prefix sums for P and Q with $O(1)$ query time.*
2. *A block size parameter b, which implies blocks $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$.*
3. *A region $\mathcal{T} \subset \mathbb{R}^2$ that is either (i) a triangle, (ii) a line segment, or (iii) a point.*
4. *A partition of $S := [N/b]$ into subsets S_{Good} and S_{Bad}, with the requirement $|S_{\text{Bad}}| \leq \mu$.*
5. *For every $i \in S_{\text{Good}}$, access in time $O(1)$ to a constant-complexity quadratic function f_i such that $f_i(t) = \text{Area}((P_i + t) \cap Q_i)$ for all $t \in \mathcal{T}$.*

Throughout, we assume that S_{Good} and S_{Bad} are sorted, since we can apply radix sort in time $O(|S_{\text{Good}}| + |S_{\text{Bad}}|)$. When using the block structure, we aim to *decrease* the size of $\mathcal{T}$ while *increasing* the block size b. If this is done carefully enough, we get a surprising result: a sublinear time oracle to report $\text{Area}((P + t) \cap Q)$ for any translation $t \in \mathcal{T}$.

4 The Linear Time Algorithm

We will reduce Problem 1 to a few different subroutines/oracles. Roughly speaking, these allow us to produce a highly refined block structure in linear time, which can be used along with multidimensional parametric search [38] to find the translation of P maximizing its area of intersection with Q in time $O(N/\log N)$.

▶ **Problem 2** (MaxRegion). *Given a $(\mu, b, \mathcal{T})$-block structure for P and Q, find*

$$\max_{t \in \mathcal{T}} \text{Area}((P + t) \cap Q)$$

along with the translation $t^* \in \mathbb{R}^2$ realizing this maximum.[5]

[5] During some calls to MaxRegion, $\mathcal{T}$ may *not* contain the global optimum.

▶ **Problem 3** (CASCADE). *Given a $(\mu, b, \mathcal{T})$-block structure, parameters μ' and b' such that b divides b', and a triangle or line segment $\mathcal{T}' \subseteq \mathcal{T}$, either produce a $(\mu', b', \mathcal{T}')$-block structure, or report failure. Failure may be reported only if the interior of $\mathcal{T}'$ intersects more than $\mu'/2$ lines from the set*

$$\mathcal{S} := \bigcup_{k \in \{0,\ldots,\lceil N/b' \rceil - 1\}} \left(\bigcup_{(i,j) \in [\frac{b'}{b}] \times [\frac{b'}{b}]} \overleftrightarrow{\Pi}(P_{k \cdot \frac{b'}{b}+i}, Q_{k \cdot \frac{b'}{b}+j}) \right), \tag{2}$$

where $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ are the blocks of P and Q determined by b.

The set $\mathcal{S}$ may appear mysterious, but it simply corresponds to the line extensions for the edges that could "cause trouble" for the cascading procedure. The only observation necessary right now is that $|\mathcal{S}| = O(Nb')$, i.e., $\mathcal{S}$ is near-linear in size for all $b' = N^{o(1)}$.

We write the expected time complexity of the fastest algorithm for MAXREGION as $T_{d\text{-MAX}}(N, b, \mu)$, where d is the dimension of the region $\mathcal{T}$. We write the expected time complexity of the fastest algorithm for CASCADE as $T_{\text{CASCADE}}(N, b, b', \mu)$. The latter will not depend on μ' or the dimensions of $\mathcal{T}$ and $\mathcal{T}'$.

The next lemmas will be proved in Sections 5 and 6. The main idea behind proving each is to group most of the blocks of P and Q into larger convex polygons that intersect in at most a constant number of locations, and then to process the rest via brute force.

▶ **Lemma 5** (Point Oracle). *For all $2 \le b \le N$, for all μ,*

$$T_{0\text{-MAX}}(N, b, \mu) = O\left(N \cdot \frac{\log^2 b}{b} + \mu b\right).$$

▶ **Lemma 6** (Cascading Subroutine). *For all $2 \le b, b' \le N$ such that b divides b', for all μ,*

$$T_{\text{CASCADE}}(N, b, b', \mu) = O\left(N \cdot \frac{\log b' \log b}{b} + \mu b^2\right).$$

4.1 Reducing to the Point Oracle and Cascading Subroutine

We now give an algorithm for MAXREGION when the input $(\mu, b, \mathcal{T})$-block structure has $\mathcal{T}$ being a triangle or line segment. The algorithm works by invoking MAXREGION on regions with lower dimension to iteratively shrink $\mathcal{T}$.

▶ **Lemma 7.** *For all $2 \le b, b' \le N^{o(1)}$ such that b divides b', for all μ, μ' such that $\mu' = N^{1-o(1)}$,*

$$\begin{aligned} T_{2\text{-MAX}}(N, b, \mu) &= O(\log(Nb'/\mu')) \cdot T_{1\text{-MAX}}(N, b, \mu) + O(N^{0.1+o(1)}) + \\ &\quad O(1) \cdot T_{\text{CASCADE}}(N, b, b', \mu) + T_{2\text{-MAX}}(N, b', \mu') \\ T_{1\text{-MAX}}(N, b, \mu) &= O(\log(Nb'/\mu')) \cdot T_{0\text{-MAX}}(N, b, \mu) + O(N^{0.1+o(1)}) + \\ &\quad O(1) \cdot T_{\text{CASCADE}}(N, b, b', \mu) + T_{1\text{-MAX}}(N, b', \mu'). \end{aligned}$$

Proof. The algorithm for when $\mathcal{T}$ is a line segment is nearly identical to the algorithm for when $\mathcal{T}$ is a triangle, so for brevity we only focus on the triangle case. Let $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ be the blocks of P and Q determined by block size parameter b.

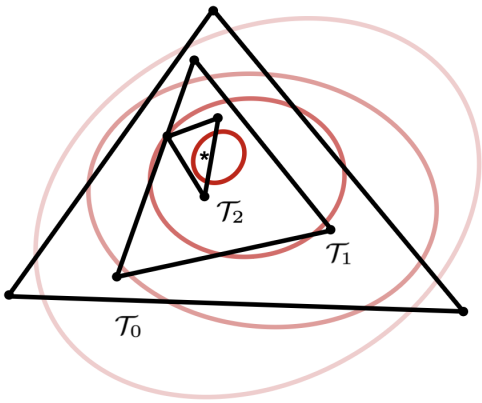


Figure 2 A series of triangles $\mathcal{T}_0$, $\mathcal{T}_1$, and $\mathcal{T}_2$ as produced by the algorithm. The (unknown) optimal placement t^* is denoted by an asterisk. The red curves indicate contour lines of the objective function, with darker circles indicating larger values of $\text{Area}((P+t) \cap Q)$.

Define $\mathcal{T}_0 := \mathcal{T}$. We will produce a series of triangles $\mathcal{T}_0 \supset \mathcal{T}_1 \supset \mathcal{T}_2 \supset \ldots$ such that each triangle is guaranteed to contain the optimal translation t^* for the original region $\mathcal{T}$, and then take the final triangle in this sequence as $\mathcal{T}'$. Once we have $\mathcal{T}'$, we invoke Cascade to produce a $(\mu', b', \mathcal{T}')$-block structure, and then find the optimal translation within $\mathcal{T}'$ (along with the area for this translation) by invoking MaxRegion on this new block structure.

In the subsequent analysis, we show that, with high probability, the interior of $\mathcal{T}'$ produced as per our sequence will intersect at most $\mu'/2$ lines of the set $\mathcal{S}$. If this event occurs, then Cascade will be *forced* to produce a $(\mu', b', \mathcal{T}')$-block structure. If this event does not occur, and Cascade returns failure, we can re-run the randomized algorithm to produce a new $\mathcal{T}'$ and try again. The expected number of attempts is $O(1)$, and so all of the steps (except for producing $\mathcal{T}'$) have a time overhead of $O(1) \cdot T_{\text{Cascade}}(N, b, b', \mu) + T_{2\text{-Max}}(N, b', \mu')$.

Producing $\mathcal{T}'$. As per the above, the expected number of times we need to make $\mathcal{T}'$ is $O(1)$, so we only focus on the time complexity to make $\mathcal{T}'$ once. Consider producing a triangle $\mathcal{T}_{x+1}$ from a triangle $\mathcal{T}_x$ in the sequence. As a first step, we use Lemma 3 to make, with high probability, a $\frac{1}{2}$-cutting for the lines of $\mathcal{S}$ that intersect the interior of $\mathcal{T}_x$.

To apply this lemma, we need an efficient sampler. We argue that naive rejection sampling will suffice. Observe that if rejection sampling fails to find a random line of $\mathcal{S}$ that intersects the interior of $\mathcal{T}_x$ within time $O(N^{0.1})$, then with high probability only $N^{1-\Omega(1)}$ lines of $\mathcal{S}$ intersect $\mathcal{T}_x$. By assumption $\mu'/2 = N^{1-o(1)}$, so we can directly take $\mathcal{T}' = \mathcal{T}_x$. If rejection sampling is fast enough, then we can produce the required $\frac{1}{2}$-cutting with high probability in time $O(|\mathcal{S}|^{0.1+o(1)}) = O(N^{0.1+o(1)})$. Then, we intersect this cutting with $\mathcal{T}_x$ and re-triangulate the resulting cells, which takes $O(1)$ time in total.

The final step is to locate the cell of this triangulation that contains t^*. This cell will be $\mathcal{T}_{x+1}$. To do this, we check all $O(1)$ triangles by recursing in the dimension. We assume that t^* is strictly contained within some cell, as the case that it is on the boundary can be detected similarly. Consider a single triangle $\mathcal{X}$. Let $\mathcal{X}^-$ be an infinitesimally smaller copy of $\mathcal{X}$ that is strictly inscribed within $\mathcal{X}$.[6] All $O(1)$ bounding line segments for $\mathcal{X}$ and $\mathcal{X}^-$

[6] Infinitesimal shifts are a standard trick, and all of the algorithms in this paper can handle them without modification as long as they are implemented carefully enough.

are contained within $\mathcal{T}$. Therefore, we can use the input $(\mu, b, \mathcal{T})$-block structure as a block structure for these line segments, and find the optimal translation (along with the area of intersection) restricted to each line segment in time $O(1) \cdot T_{\text{1-MAX}}(N, b, \mu)$. If one of the line segments for $\mathcal{X}^-$ has a translation realizing a larger area than all of the line segments for $\mathcal{X}$, then by Lemma 1 we know that $t^* \in \mathcal{X}$. Otherwise, $\mathcal{X}$ does not contain t^*.[7]

With high probability, the interior of $\mathcal{T}_{x+1}$ will intersect at most half as many lines of $\mathcal{S}$ as $\mathcal{T}_x$. By construction and the fact that b divides b' (and hence $b \leq b'$), we have $|\mathcal{S}| = O(Nb')$. Thus there are at most $O(\log \frac{|\mathcal{S}|}{\mu'}) = O(\log(Nb'/\mu'))$ iterations before we produce a triangle that intersects at most $\mu'/2$ lines of $\mathcal{S}$ with high probability and hence can be taken as $\mathcal{T}'$. The time for all iterations is $O(\log(Nb'/\mu')) \cdot (T_{\text{1-MAX}}(N, b, \mu) + O(N^{0.1+o(1)})) = O(\log(Nb'/\mu')) \cdot T_{\text{1-MAX}}(N, b, \mu) + O(N^{0.1+o(1)})$. ◀

4.2 Putting it all Together

We now present our linear time algorithm. This amounts to solving a curious recurrence, and we actually have significant freedom in choosing the values for b and b'.

▶ **Theorem 8.** *There is an algorithm for Problem 1 running in expected time $O(n + m)$.*

Proof. We will always take $\mu := 20 \cdot N/b^3$ and $\mu' := 20 \cdot N/(b')^3$, so we can drop the μ and μ' arguments from the time complexity of each subroutine/oracle. We begin by reducing from Problem 1 to an instance of MaxRegion over a $(20 \cdot N/2^3, 2, \mathcal{T})$-block structure, which requires no work beyond computing the area prefix sums because we can take $S_{\text{Bad}} = [\lceil N/2 \rceil]$ and $S_{\text{Good}} = \emptyset$. Thus the total expected running time for Problem 1 will be $O(N) + T_{\text{2-MAX}}(N, 2)$.

Before analyzing $T_{\text{2-MAX}}(N, 2)$, we find a closed-form expression for $T_{\text{1-MAX}}(N, b)$ when $2 \leq b \leq N^{o(1)}$. As per Lemmas 5, 6, and 7, we have the following for any $2 \leq b, b' \leq N^{o(1)}$ such that b divides b':

$$T_{\text{1-MAX}}(N, b) = O\left(N \cdot \frac{\log b' \log^2 b}{b}\right) + T_{\text{1-MAX}}(N, b').$$

We can efficiently parallelize the algorithm for zero-dimensional MaxRegion given in the proof of Lemma 5. So via a standard application of multidimensional parametric search (see Theorem 4.4 in the full version of [38]) that uses zero-dimensional MaxRegion for the decision problem, $T_{\text{1-MAX}}(N, b) = O(N/\log N)$ when $b \geq (\log N)^{C_1}$ for some constant C_1. If we take $b' = 2b$ (for example) and solve the recurrence, we get

$$T_{\text{1-MAX}}(N, b) = O\left(\frac{N}{\log N} + \sum_{a=\log b}^{O(\log \log N)} N \cdot \frac{\log^3(2^a)}{2^a}\right) = O\left(N \cdot \frac{\log^3 b}{b} + \frac{N}{\log N}\right).$$

Using the above along with Lemmas 5, 6, and 7, we have the following for any $2 \leq b, b' \leq N^{o(1)}$ such that b divides b':

$$T_{\text{2-MAX}}(N, b) = O\left(N \cdot \frac{\log b' \log^3 b}{b} + N \cdot \frac{\log b'}{\log N}\right) + T_{\text{2-MAX}}(N, b').$$

[7] All invocations might return a maximum area of zero, but we can handle this in $O(1)$ time by checking whether an arbitrary translation that causes P and Q to overlap is located within $\mathcal{X}$.

If we again take $b' = 2b$ and stop to apply multidimensional parametric search when $b' = (\log N)^{C_2}$ for some constant C_2, we have $O(\log\log N)$ recursion depth, and $\log b'$ never exceeds $O(\log\log N)$. Solving the recurrence in the same way as for $T_{1\text{-MAX}}(N, b)$ gives:

$$T_{2\text{-MAX}}(N, b) = O\left(N \cdot \frac{\log^4 b}{b} + N \cdot \frac{\log^2 \log N}{\log N}\right).$$

Clearly $T_2(N, 2) = O(N)$, and the theorem follows. ◀

5 Constructing the Point Oracle

In this section, we prove the Point Oracle Lemma, restated below:

▶ **Lemma 5** (Point Oracle). *For all $2 \le b \le N$, for all μ,*

$$T_{0\text{-MAX}}(N, b, \mu) = O\left(N \cdot \frac{\log^2 b}{b} + \mu b\right).$$

In other words, we need to use a given $(\mu, b, \mathcal{T})$-block structure (where $\mathcal{T}$ is just a single point t^*) to calculate $\text{Area}((P + t^*) \cap Q)$. We can decompose the area function as

$$\text{Area}((P + t^*) \cap Q) = \sum_i \text{Area}((P_i + t^*) \cap Q_i) + \sum_{i \neq j} \text{Area}((P_i + t^*) \cap Q_j)$$

and then solve for each individual summation rapidly. The first summation is quite easy to compute by simply reading through the lists S_{Good} and S_{Bad} that are provided.

▶ **Lemma 9.** *There is an algorithm running in time $O(N/b + \mu b)$ such that, given a $(\mu, b, \mathcal{T})$-block structure, where $2 \le b \le N^{o(1)}$ and $\mathcal{T}$ consists of just a single point t^*, it will compute*

$$\sum_i \text{Area}((P_i + t^*) \cap Q_i),$$

where $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ are the blocks of P and Q determined by parameter b.

Proof. By definition of the block structure, we have a partition of $S = [N/b]$ into two subsets S_{Good} and S_{Bad} such that $|S_{\text{Bad}}| \le \mu$, and for each $i \in S_{\text{Good}}$ we have a constant complexity quadratic function $f_i(t) = \text{Area}((P_i + t) \cap Q_i)$ for all $t \in \mathcal{T}$.

We can thus compute

$$\sum_{i \in S_{\text{Good}}} \text{Area}((P_i + t^*) \cap Q_i)$$

in time $O(N/b)$ by simply evaluating each $f_i(t^*)$. Since $|S_{\text{Bad}}| \le \mu$, we can compute

$$\sum_{i \in S_{\text{Bad}}} \text{Area}((P_i + t^*) \cap Q_i)$$

by evaluating the area for each $(P_i + t^*) \cap Q_i$ directly. The time required is $O(\mu b)$ using known algorithms for intersecting two convex polygons whose edges are in sorted order. ◀

We now turn our attention to the summation $\sum_{i \neq j} \text{Area}((P_i + t^*) \cap Q_j)$. Because every term amounts to computing the intersection for two blocks with *different* angle ranges, we could use binary searches to evaluate each term in $O(\text{polylog}(b))$ time. However, this is too slow for our purposes, as there are a near quadratic number of terms. Instead, we take advantage of additional structure to decompose the summation into $O(N/b)$ intersections of *unions of blocks*, such that each intersection can still be computed via binary search.

As a tool, we need a way to efficiently intersect convex chains that meet certain conditions.

▶ **Lemma 10.** *Let A and B be any two convex polygons, and let X and Y be any two polygonal chains whose edges are a subset of the edges of A and B, respectively. If $\Lambda_A(X) \cap \Lambda_B(Y) = \emptyset$, then X and Y intersect at most twice, and we can find the intersection point(s) in time $O(\log|X| \log|Y|)$.*

Proof. Because the angle ranges for X and Y are disjoint, they act as pseudo-disks, and hence can intersect at most twice. We can find the intersection point(s), if they exist, using a standard nested binary search in $O(\log|X| \log|Y|)$ time (or a more clever binary search [37] in $O(\log|X| + \log|Y|)$ time, though we don't need this improvement). ◀

Now we can show how to compute $\sum_{i \neq j} \text{Area}((P_i + t^*) \cap Q_j)$.

▶ **Lemma 11.** *There is an algorithm running in time $O(N \cdot \frac{\log^2 b}{b})$ such that, given a $(\mu, b, \mathcal{T})$-block structure, where $\mathcal{T}$ consists of just a single point t^*, it will compute*

$$\sum_{i \neq j} \text{Area}((P_i + t^*) \cap Q_j),$$

where $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ are the blocks of P and Q determined by parameter b.

Proof. We give a recursive algorithm for the problem. As a subroutine, we consider the problem of computing, given two integers $x, y \in [N/b]$ with $x > y$,

$$\sum_{(i,j) \in W(x,y)} \text{Area}((P_i + t^*) \cap Q_j), \text{ with } W(x,y) := \{(i,j) \in [N/b] \times [N/b] : i \neq j, x \leq i, j \leq y\}.$$

Let $T(y - x, b)$ be the time complexity of this subroutine. Since the original summation $\sum_{i \neq j} \text{Area}((P_i + t^*) \cap Q_j)$ is equivalent to $\sum_{(i,j) \in W(1, \lceil N/b \rceil)} \text{Area}((P_i + t) \cap Q_j)$, the lemma amounts to showing that $T(\lceil N/b \rceil - 1, b) = O(N \cdot \frac{\log^2 b}{b})$, or more generally, $T(z, b) = O(z \log^2 b)$.

Consider an instance of this subroutine with any parameters $y > x$. Let $m := \lceil \frac{x+y}{2} \rceil$ and $z := y - x$. We can decompose the target summation as follows:

$$\begin{aligned}\sum_{(i,j) \in W(x,y)} \text{Area}((P_i + t^*) \cap Q_j) \;=\; & \\ \sum_{(i,j) \in W(x,m-1)} \text{Area}((P_i + t^*) \cap Q_j) \;+\; & \sum_{(i,j) \in W(m,y)} \text{Area}((P_i + t^*) \cap Q_j) \;+ \\ \sum_{x \leq i < m \leq j \leq y} \text{Area}((P_i + t^*) \cap Q_j) \;+\; & \sum_{x \leq j < m \leq i \leq y} \text{Area}((P_i + t^*) \cap Q_j).\end{aligned}$$

The first two summations can be evaluated in time at most $2T(\lceil \frac{y-x}{2} \rceil, b)$ via recursion (unless $x - y = 1$, in which case they are automatically zero). The last two summations can actually be evaluated directly. We consider the case of the very last summation; the remaining one can be evaluated via a symmetric process. First note that

$$\sum_{x \leq j < m \leq i \leq y} \text{Area}((P_i + t^*) \cap Q_j) \;=\; \text{Area}\left(\left(\bigcup_{m \leq i \leq y} P_i + t^*\right) \cap \left(\bigcup_{x \leq j < m} Q_j\right)\right).$$

Let $A = \bigcup_{x \leq i < m} P_i$ and $B = \bigcup_{m \leq j \leq y} Q_j$. Observe that A and B are both convex polygons, since each is the union of consecutive blocks. By construction, the edges that A shares with P and the edges that B shares with Q form two convex polygonal chains that

have disjoint angle ranges, i.e., $\Lambda_P(E_P(A)) \cap \Lambda_Q(E_Q(B)) = \emptyset$. These chains account for all but two edges of A and two edges of B. Thus we can decompose A and B into $O(1)$ convex polygonal chains satisfying the preconditions of Lemma 10. This implies that A and B intersect in at most a constant number of locations, and furthermore we can find the intersection points in time $O(\log^2(bz))$.

By decomposing A and B into $O(1)$ pieces based on the intersection locations, we can use the area prefix sums[8], along with a direct computation on the $O(1)$ intersecting pairs of edges, to find the area of intersection in $O(1)$ additional time. So the recurrence is $T(z, b) = 2T(\lceil z/2 \rceil, b) + O(\log^2(bz))$ when $z > 1$ and $T(1, b) = O(\log^2 b)$. This solves to $T(z, b) = O(z \log^2 b)$. ◀

6 Constructing the Cascading Subroutine

The last step is to prove the Cascading Subroutine Lemma, restated below:

▶ **Lemma 6** (Cascading Subroutine). *For all $2 \le b, b' \le N$ such that b divides b', for all μ,*

$$T_{\textsc{Cascade}}(N, b, b', \mu) = O\left(N \cdot \frac{\log b' \log b}{b} + \mu b^2\right).$$

In other words, given a $(\mu, b, \mathcal{T})$-block structure, parameters μ' and b', and a triangle or line segment $\mathcal{T}'$, we want to either produce a valid $(\mu', b', \mathcal{T}')$-block structure, or report that more than $\mu'/2$ lines of the set $\mathcal{S}$ intersect $\mathcal{T}'$. Throughout, let $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots, Q_{\lceil N/b \rceil}$ be the blocks of P and Q determined by block size parameter b, and let $P'_1, \ldots, P'_{\lceil N/b' \rceil}$ and $Q'_1, \ldots, Q'_{\lceil N/b' \rceil}$ be the blocks of P and Q determined by block size parameter b'.

First, we show that constant-complexity quadratic functions may actually be used to summarize the area function for certain block pairs.

▶ **Lemma 12.** *Let A and B be any convex polygons, and let $\mathcal{T}'$ be any triangle or line segment contained in one cell of $\Pi(A, B)$.*[9] *Then there exists a constant-complexity quadratic function f such that $f(t) = \text{Area}((A + t) \cap B)$ for all $t \in \mathcal{T}'$.*

Proof. This follows from known observations by de Berg et al. [21]. ◀

Before describing how to perform cascading, we need a tool to help classify the new blocks based on their relationship with $\mathcal{T}'$.

▶ **Lemma 13.** *Let A and B be any two convex polygons, and let X and Y be any two polygonal chains whose edges are a subset of the edges of A and B, respectively. Let $\mathcal{T}'$ be any triangle or line segment. If $\Lambda_A(X) \cap \Lambda_B(Y) = \emptyset$, then we can determine if some line segment in $\Pi(X, Y)$ intersects the interior of $\mathcal{T}'$ in time $O(\log |X| \log |Y|)$.*

Proof. Let the set of vertices for $\mathcal{T}'$ be $V(\mathcal{T}')$. For each $t \in V(\mathcal{T}')$, we know that, by Lemma 10, $X + t$ and Y can intersect in at most two points, and we can find the two pairs of edges that contain these intersection(s) in time $O(\log |X| \log |Y|)$.[10] Let the set of all $O(1)$ edge pairs (e_X, e_Y) for all vertices in $V(\mathcal{T}')$ be L. We can determine if $\mathcal{T}' \subseteq \pi(e_X, e_Y)$ for

[8] The area prefix sums for P and Q also act as ares prefix sums for A and B.

[9] It is permissible for $\mathcal{T}'$ to touch the boundary of one cell, as long as it doesn't strictly intersect.

[10] There is an edge case where at least one of the intersections occurs at a vertex of X or a vertex of Y, but this can be handled with slightly more work without increasing the asymptotic running time.

all $(e_X, e_Y) \in L$ via brute force in time $O(1)$. If this is *not* the case, then some line segment of $\Pi(X, Y)$ intersects the interior of $\mathcal{T}'$, and we are done. If this *is* the case, then because $\mathcal{T}'$ is convex and each $\pi(e_X, e_Y)$ is convex, we have that each translation $t \in \mathcal{T}'$ realizes a set of intersecting pairs that is a (possibly strict) superset of L.

It remains to determine if any translations $t \in \mathcal{T}'$ realize an intersecting pair not in L, because this case (and only this case) would mean that a line segment in $\Pi(X, Y)$ intersects the interior of $\mathcal{T}'$. Let X^- be X with its edges participating in L deleted, and let Y^- be Y with its edges participating in L deleted. In time $O(\log|X|\log|Y|)$ we can use techniques similar to the ones given above to determine if the interior of the (implicitly generated) Minkowski sum of $\mathcal{T}'$ and X^- has any intersection with Y, and then if the interior of the (implicitly generated) Minkowski sum of $\mathcal{T}'$ and X has any intersection with Y^-. ◀

We are now ready to prove Lemma 6. The main observation is that the new quadratic functions stored in this data structure only need to apply when $t \in \mathcal{T}'$, not the more general case that $t \in \mathcal{T}$, so many of the previously difficult blocks can now be handled. We will find a quadratic function for almost all indices $k \in [N/b']$ in two steps:

1. Scan through the information provided in the input $(\mu, b, \mathcal{T})$-block structure, and use this to recover part of the function $\text{Area}((P'_k + t) \cap Q'_k)$.
2. Recover the missing information for $\text{Area}((P'_k + t) \cap Q'_k)$ by intersecting $O(\frac{b'}{b})$ pairs of convex polygons with disjoint angle ranges. We evaluate these using nested binary searches (Lemma 10 or 13) and area prefix sums.

Proof. For all $k \in [N/b']$, note that because b divides b', we have a set of indices $C_k = \{(k-1)\cdot\frac{b'}{b}+1, \ldots, k\cdot\frac{b'}{b}\}$ such that $P'_k = \bigcup_{i\in C_k} P_i$ and $Q'_k = \bigcup_{j\in C_k} Q_j$.[11] If for all $(i,j) \in C_k \times C_k$ and for all $t \in \mathcal{T}'$ we can summarize $\text{Area}((P_i+t)\cap Q_j)$ with a constant-complexity quadratic function, then we can simply add these to get a new function $f_k(t) = \text{Area}((P'_k + t) \cap Q'_k)$.

We first argue that each quadratic function $f_k(t)$ can be found efficiently, if it exists. Consider just a single value of k. We use the decomposition

$$f_k(t) = \sum_{(i,j)\in C_k\times C_k} \text{Area}((P_i+t)\cap Q_j) = \sum_{i\in C_k} \text{Area}((P_i+t)\cap Q_i) + \sum_{\substack{(i,j)\in C_k\times C_k:\\ i\neq j}} \text{Area}((P_i+t)\cap Q_j).$$

For the first summation on the right hand side, we scan through the list S_{Good} from the original $(\mu, b, \mathcal{T})$-block structure. Each function $\text{Area}((P_i + t) \cap Q_i)$ with $i \in S_{\text{Good}}$ already has a quadratic function, and reading these takes time $O(\frac{b'}{b})$ for each k, or $O(N/b)$ time for all k. Globally, there are at most μ indices i not in S_{Good}. For each of these, we can check whether $\mathcal{T}'$ lies in a single cell of $\Pi(P_i, Q_i)$ via directly examining each line segment in $\Pi(P_i, Q_i)$ in time $O(b^2)$. If this is the case, then by Lemma 12, $\text{Area}((P_i + t) \cap Q_i)$ can be summarized by a quadratic function for all $t \in \mathcal{T}'$, and we find the function via directly examining each parallelogram in $\Pi(P_i, Q_i)$ in time $O(b^2)$; the total additional time is $O(\mu b^2)$. (If this fails for some $i \in C_k$, we will add the index k to S'_{Bad}.)

Now consider the second summation on the right hand side. We can rewrite this as

$$\sum_{\substack{(i,j)\in C_k\times C_k:\\ i\neq j}} \text{Area}((P_i + t) \cap Q_j) = \sum_{i\in C_k} \left(\text{Area}((P_i + t) \cap Q^-_{i,k}) + \text{Area}((P_i + t) \cap Q^+_{i,k})\right),$$

where $Q^-_{i,k}$ is the union of all Q_j's in the original sum with $j < i$, and $Q^+_{i,k}$ is the union of all Q_j's in the original sum with $j > i$. We evaluate each of the terms individually.

[11] The set may be smaller for the very last value of k, but this does not impact the proof.

Consider the case of $\text{Area}((P_i+t)\cap Q^-_{i,k})$, as the case of $\text{Area}((P_i+t)\cap Q^+_{i,k})$ is symmetric. Now, $Q^-_{i,k}$ is a convex polygon, because it is the union of consecutive blocks. By construction, $\Lambda_P(E_P(P_i)) \cap \Lambda_Q(E_Q(Q^-_{i,k})) = \emptyset$. Therefore, we can break $P_i + t$ and $Q^-_{i,k}$ into $O(1)$ pieces that satisfy the preconditions of Lemma 10, showing that $P_i + t$ and $Q^-_{i,k}$ can only intersect at a constant number of locations for any translation $t \in \mathbb{R}^2$. These same pieces allow us to apply Lemma 13 to determine if $\mathcal{T}'$ lies in a single cell of $\Pi(P_i, Q^-_{i,k})$, which takes time $O(\log|P_i|\log|Q^-_{i,k}|) = O(\log b \log b')$. If this is the case, then by Lemma 12 there exists a quadratic function equal to $\text{Area}((P_i+t)\cap Q^-_{i,k})$ for all $t \in \mathcal{T}'$. The nonconstant components of this quadratic function only depend on the $O(1)$ intersecting pairs of edges, and we have already found these via Lemma 13. Thus we can break P_i and $Q^-_{i,k}$ into $O(1)$ pieces based on the intersection points, and use the area prefix sums from the $(\mu, b, \mathcal{T})$-block structure (which also serve as area prefix sums for P_i and $Q^-_{i,k}$), to find the quadratic function equal to $\text{Area}((P_i + t) \cap Q^-_{i,k})$ in $O(1)$ additional time. (If this fails for some $i \in C_k$, we will add the index k to S'_{Bad}.)

Adding the time across all terms for all $k \in [N/b']$ gives an overall time of $O(N \cdot \frac{\log b \log b'}{b})$.

Each index k for which we found a function $f_k(t)$ is added to S'_{Good}, and we store its function. The remaining indices are added to S'_{Bad}. The last step is to argue that $|S'_{\text{Bad}}| \le \mu'$, meaning we have constructed a valid $(\mu', b', \mathcal{T}')$-block structure.

By construction, note that every $k \in S'_{\text{Bad}}$ corresponds to an intersection between $\bigcup_{(i,j)\in C_k\times C_k} \Pi(P_i, Q_j)$ and the interior of $\mathcal{T}'$. Also by construction (see Equation 2), each line $\overleftrightarrow{\mathcal{L}}$ in $\mathcal{S}$ has at most two indices k such $\overleftrightarrow{\mathcal{L}}$ is the line extension of some line segment in $\bigcup_{(i,j)\in C_k\times C_k} \Pi(P_i, Q_j)$. Therefore we can charge each $k \in S'_{\text{Bad}}$ to different lines of $\mathcal{S}$ in such a way that each line of $\mathcal{S}$ will be charged at most twice. Thus if $|S'_{\text{Bad}}| > \mu'$, this implies that more than $\mu'/2$ lines of $\mathcal{S}$ intersect the interior of $\mathcal{T}'$, so the algorithm can just return failure in this case. ◀

References

1. Pankaj K. Agarwal, Torben Hagerup, Rahul Ray, Micha Sharir, Michiel H. M. Smid, and Emo Welzl. Translating a planar object to maximize point containment. In *Proc. 10th Annual European Symposium on Algorithms (ESA)*, volume 2461 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2002. doi:10.1007/3-540-45749-6_8.
2. Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30(4):412–458, 1998. doi:10.1145/299917.299918.
3. Hee-Kap Ahn, Peter Brass, and Chan-Su Shin. Maximum overlap and minimum convex hull of two convex polyhedra under translations. *Comput. Geom.*, 40(2):171–177, 2008. doi:10.1016/J.COMGEO.2007.08.001.
4. Hee-Kap Ahn, Siu-Wing Cheng, Hyuk Jun Kweon, and Juyoung Yon. Overlap of convex polytopes under rigid motion. *Comput. Geom.*, 47(1):15–24, 2014. doi:10.1016/J.COMGEO.2013.08.001.
5. Hee-Kap Ahn, Siu-Wing Cheng, and Iris Reinbacher. Maximum overlap of convex polytopes under translation. *Comput. Geom.*, 46(5):552–565, 2013. doi:10.1016/J.COMGEO.2011.11.003.
6. Hee-Kap Ahn, Otfried Cheong, Chong-Dae Park, Chan-Su Shin, and Antoine Vigneron. Maximizing the overlap of two planar convex sets under rigid motions. *Comput. Geom.*, 37(1):3–15, 2007. doi:10.1016/J.COMGEO.2006.01.005.
7. 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.

8 Nancy M. Amato and Franco P. Preparata. A time-optimal parallel algorithm for three-dimensional convex hulls. *Algorithmica*, 14(2):169–182, 1995. doi:10.1007/BF01293667.

9 Gill Barequet, Matthew T. Dickerson, and Petru Pau. Translating a convex polygon to contain a maximum number of points. *Comput. Geom.*, 8:167–179, 1997. doi:10.1016/S0925-7721(96)00011-9.

10 Sergio Cabello, Mark de Berg, Panos Giannopoulos, Christian Knauer, René van Oostrum, and Remco C. Veltkamp. Maximizing the area of overlap of two unions of disks under rigid motion. *Int. J. Comput. Geom. Appl.*, 19(6):533–556, 2009. doi:10.1142/S0218195909003118.

11 Timothy M. Chan. A simpler linear-time algorithm for intersecting two convex polyhedra in three dimensions. *Discret. Comput. Geom.*, 56(4):860–865, 2016. doi:10.1007/S00454-016-9785-3.

12 Timothy M. Chan. (Near-)linear-time randomized algorithms for row minima in Monge partial matrices and related problems. In *Proc. of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1465–1482, 2021. doi:10.1137/1.9781611976465.88.

13 Timothy M. Chan and Isaac M. Hair. Convex polygon containment: Improving quadratic to near linear time. In *Proc. 40th International Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 34:1–34:15, 2024. doi:10.4230/LIPICS.SOCG.2024.34.

14 Bernard Chazelle. The polygon containment problem. *Advances in Computing Research*, 1(1):1–33, 1983. URL: https://www.cs.princeton.edu/~chazelle/pubs/PolygContainmentProb.pdf.

15 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discret. Comput. Geom.*, 6:485–524, 1991. doi:10.1007/BF02574703.

16 Bernard Chazelle. Cuttings. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035179.CH25.

17 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. doi:10.1007/BF01840440.

18 Siu-Wing Cheng and Chi-Kit Lam. Shape matching under rigid motion. *Comput. Geom.*, 46(6):591–603, 2013. doi:10.1016/J.COMGEO.2013.01.002.

19 Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discret. Comput. Geom.*, 2:195–222, 1987. doi:10.1007/BF02187879.

20 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.

21 Mark de Berg, Otfried Cheong, Olivier Devillers, Marc J. van Kreveld, and Monique Teillaud. Computing the maximum overlap of two convex polygons under translations. *Theory Comput. Syst.*, 31(5):613–628, 1998. doi:10.1007/PL00005845.

22 Martin E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13(1):31–45, 1984. doi:10.1137/0213003.

23 Martin E. Dyer. On a multidimensional search technique and its application to the euclidean one-centre problem. *SIAM J. Comput.*, 15(3):725–738, 1986. doi:10.1137/0215052.

24 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.

25 Sariel Har-Peled and Subhro Roy. Approximating the maximum overlap of polygons under translation. *Algorithmica*, 78(1):147–165, 2017. doi:10.1007/S00453-016-0152-9.

26 Shreesh Jadhav and Asish Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discret. Comput. Geom.*, 12:291–312, 1994. doi:10.1007/BF02574382.

27 Mook Kwon Jung, Seokyun Kang, and Hee-Kap Ahn. Minimum convex hull and maximum overlap of two convex polytopes. In *Proc. 36th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025. To appear.

28 Garam Kim, Jongmin Choi, and Hee-Kap Ahn. Maximum overlap of two convex polygons. *KIISE Transactions on Computing Practices*, 27(8):400–405, 2021. doi:10.5626/KTCP.2021.27.8.400.

29 Jirí Matoušek. Approximations and optimal geometric divide-an-conquer. *J. Comput. Syst. Sci.*, 50(2):203–208, 1995. doi:10.1006/JCSS.1995.1018.

30 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983. doi:10.1145/2157.322410.

31 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12(4):759–776, 1983. doi:10.1137/0212052.

32 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984. doi:10.1145/2422.322418.

33 Nimrod Megiddo. Partitioning with two lines in the plane. *J. Algorithms*, 6(3):430–433, 1985. doi:10.1016/0196-6774(85)90011-2.

34 David M. Mount, Ruth Silverman, and Angela Y. Wu. On the area of overlap of translated polygons. *Comput. Vis. Image Underst.*, 64(1):53–61, 1996. doi:10.1006/CVIU.1996.0045.

35 Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Inf. Process. Lett.*, 85(3):117–122, 2003. doi:10.1016/S0020-0190(02)00370-8.

36 Joseph O'Rourke, Subhash Suri, and Csaba D. Tóth. Polygons. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, pages 787–810. Chapman and Hall/CRC, 3rd edition, 2017. URL: https://www.csun.edu/~ctoth/Handbook/chap30.pdf.

37 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.

38 Sivan Toledo. Maximizing non-linear concave functions in fixed dimension. In *Proc. 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 676–685, 1992. doi:10.1109/SFCS.1992.267783.

39 Eitan Zemel. An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems. *Inf. Process. Lett.*, 18(3):123–128, 1984. doi:10.1016/0020-0190(84)90014-0.

40 Honglin Zhu and Hyuk Jun Kweon. Maximum overlap area of a convex polyhedron and a convex polygon under translation. In *Proc. 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 61:1–61:16, 2023. doi:10.4230/LIPICS.SOCG.2023.61.

Faster Algorithms for Reverse Shortest Path in Unit-Disk Graphs and Related Geometric Optimization Problems: Improving the Shrink-And-Bifurcate Technique

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Zhengcheng Huang
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

In a series of papers, Avraham, Filtser, Kaplan, Katz, and Sharir (SoCG'14), Kaplan, Katz, Saban, and Sharir (ESA'23), and Katz, Saban, and Sharir (ESA'24) studied a class of geometric optimization problems – including reverse shortest path in unweighted and weighted unit-disk graphs, discrete Fréchet distance with one-sided shortcuts, and reverse shortest path in visibility graphs on 1.5-dimensional terrains – for which standard parametric search does not work well due to a lack of efficient parallel algorithms for the corresponding decision problems. The best currently known algorithms for all the above problems run in $O^*(n^{6/5}) = O^*(n^{1.2})$ time (ignoring subpolynomial factors), and they were obtained using a technique called *shrink-and-bifurcate*. We improve the running time to $\tilde{O}(n^{8/7}) \approx O(n^{1.143})$ for these problems. Furthermore, specifically for reverse shortest path in unweighted unit-disk graphs, we improve the running time further to $\tilde{O}(n^{9/8}) = \tilde{O}(n^{1.125})$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric optimization problems, parametric search, shortest path, disk graphs, Fréchet distance, visibility, distance selection, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.32

Related Version *Full Version*: https://arxiv.org/abs/2504.06434

Funding *Timothy M. Chan*: Work supported by NSF Grant CCF-2224271.

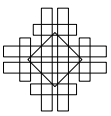
1 Introduction

Parametric search is one of the most well-known and powerful techniques for solving geometric optimization problems [5]. First introduced by Megiddo [26] in the 80s, the technique has since found countless applications in computational geometry and beyond. To search for the optimal value r^* of an optimization problem, we first solve the decision problem: given an input value r, decide whether $r^* \leq r$. The idea is to simulate an algorithm $\mathcal{A}$ for the decision problem[1] with the input value being the unknown r^* itself; whenever $\mathcal{A}$ makes a comparison with the unknown r^*, we resolve the comparison by calling the decision algorithm. Megiddo showed that if $\mathcal{A}$ is parallelizable with polylogarithmic number of rounds, then the simulation can be done efficiently with only a polylogarithmic factor increase in the total running time.

Over the years, a number of refinements and alternatives to parametric search have been proposed (e.g., Cole's trick [14], matrix searching [18, 19], expander-based approaches [24], Chan's randomized optimization technique [8] and implicit linear programming technique [10, 11]), but these alternatives are more specialized and are mainly about lowering the extra polylogarithmic factors in the running time.

[1] The algorithm $\mathcal{A}$ that we are simulating need not solve the decision problem, although it often is the case, so long as the optimal value r^* is one of its "critical values".

© Timothy M. Chan and Zhengcheng Huang;
licensed under Creative Commons License CC-BY 4.0



In the last couple of years, Avraham, Filtser, Kaplan, Katz, and Sharir [6], Wang and Zhao [29], Kaplan, Katz, Saban, and Sharir [21], Katz, Saban, and Sharir [23], and Agarwal, Kaplan, Katz, and Sharir [3] have identified and explored a natural class of geometric optimization problems (listed below) where parametric search does not do too well. For these problems, efficient near-linear-time decision algorithms exist, but they do not seem efficiently parallelizable (the decision version of these problems are related to single-source shortest paths, but breadth-first search (BFS) and Dijkstra's algorithm are inherently sequential). For many of the problems in this class, the search spaces are actually Euclidean distances defined by pairs of input points, and so parametric search could be replaced by binary search using an oracle for *distance selection*, but unfortunately, known distance selection algorithms requires near $O(n^{4/3})$ time in the plane [1, 24, 28, 13].

To overcome these issues, Avraham *et al.* [6] introduced a new clever variant of the parametric search technique they termed *shrink-and-bifurcate*, which was subsequently adopted by Kaplan *et al.* [21], Katz *et al.* [23], and Agarwal *et al.* [3]. The technique is able to solve most of these problems with an unusual running time between near-linear and $n^{4/3}$: namely, $O^*(n^{6/5})$.[2]

Although Avraham *et al.*'s improvement may seem incremental, this $O^*(n^{6/5})$ bound has remained stagnant (Avraham *et al.*'s paper appeared over 10 years ago), despite more applications and more appearances of this bound in recent papers [21, 23, 3]; the ideal goal of obtaining near-linear time at the moment seems far out of reach. In this work, we want to understand this class of problems better, and to see if this mysterious exponent 6/5 can be improved.

In the following, we state the specific problems considered in this class and review their background.

Reverse shortest path in (unweighted) unit-disk graphs. We start with the simplest of these problems: *reverse shortest path* for unit-disk graphs. For every set of points P in the plane and positive real number r, let $G_r(P)$ denote the graph with vertex set P, where we put an edge between two points iff their Euclidean distance is at most r; equivalently, $G_r(P)$ is the intersection graph of the disks with centers P and radius $r/2$. Note that $G_r(P)$ is a *unit-disk graph* (since we can rescale the radius to 1). In the reverse shortest path problem, one is given a set of points P in the plane, two vertices $s, t \in P$, and a positive integer λ, and is asked to compute the minimum real number r^* such that $G_{r^*}(P)$ contains an s-to-t path with at most λ edges.

This problem is quite natural and can be viewed equivalently as finding the minimum bottleneck path from s to t that uses at most λ links (or "hops"), where minimizing *bottleneck* refers to minimizing the longest edge in the path. (As is well known, computing minimum bottleneck paths without constraints on the number of links reduces to minimum spanning trees; *e.g.*, see [20]. It is reasonable to add restrictions on the number of links.)

The decision version of this problem takes an extra parameter r, and asks one to determine whether $G_r(P)$ contains an s-to-t-path with at most λ edges. The decision problem (or computing the minimum-link s-to-t path in a unit disk graph) can be solved in $O(n \log n)$ time, as shown by Cabello and Jejčič [7] using the Delaunay triangulation, or alternatively by Chan and Skrepetos [12] using a grid-based approach, or even earlier by Efrat, Itai, and Katz [17] (who solved a very similar problem as a subroutine for computing bichromatic maximum matchings). Now, since r^* is always the Euclidean distance between two input

[2] In this paper, we use the O^* and $\tilde{O}$ notation to hide $n^{o(1)}$ and $(\log n)^{O(1)}$ factors respectively.

points, it was immediately noted by Cabello and Jejčič that reverse shortest path can be solved in $\tilde{O}(n^{4/3})$ time by binary search, using an $\tilde{O}(n^{4/3})$-time distance selection algorithm (as we have mentioned earlier) to guide the search.

The first algorithm to beat $n^{4/3}$ for reverse shortest path was discovered by Wang and Zhao [29], using a grid-based approach. Their algorithm runs in $\tilde{O}(n^{5/4})$ time. Subsequently, Kaplan *et al.* [21] presented a different, faster randomized algorithm running in $O^*(n^{6/5})$ time, by applying the aforementioned shrink-and-bifurcate technique.

Reverse shortest path in weighted unit-disk graphs. In a *weighted* unit-disk graph, each edge is weighted by the Euclidean distance between the centers of the two disks. The reverse shortest path problem can be defined in a similar way, except that λ is now a cap on the sum of the weights along the s-to-t path.

Wang and Xue [27] gave an $O(n \log^2 n)$-time algorithm for the decision problem in the weighted case. Wang and Zhao [29] extended their $\tilde{O}(n^{5/4})$-time algorithm for reverse shortest path to weighted unit-disk graphs (with a slightly larger polylogarithmic factor). Kaplan *et al.* [21] similarly extended their $O^*(n^{6/5})$-time randomized algorithm to the weighted case, again via shrink-and-bifurcate.

Reverse shortest path in unweighted and weighted disk graphs. The reverse shortest path problem can be generalized to intersection graphs of disks with arbitrary radii. The input is a set S of disks in the plane, and $G_r(S)$ is defined as the graph with S as vertices, where we put an edge between two disks iff their distance is at most r (this is sometimes referred to as a *proximity graph*); equivalently, $G_r(S)$ is the intersection graph of the disks with the same centers as S but radii increased additively by $r/2$. In the weighted case, each edge is weighted by the Euclidean distance between the centers of the two disks (or alternatively, the Euclidean distance of the centers minus the sum of the two radii).

As shown by Kaplan *et al.* [21], the decision problem can be solved in $\tilde{O}(n)$ time by a careful implementation of BFS (in the unweighted case) or Dijkstra's algorithm (in the weighted case), using known dynamic geometric data structures [22] (which have large hidden polylogarithmic factors).

Wang and Zhao's grid-based approach to reverse shortest path does not extend to arbitrary disk graphs. However, the shrink-and-bifurcate approach is still applicable, and yields an $O^*(n^{5/4})$-time randomized algorithm as shown by Kaplan *et al.* [21].

Reverse shortest path in segment proximity graphs. Another generalization is to the case of line segments. Given a set S of *disjoint* line segments in the plane, define the segment proximity graph $G_r(S)$ to be the graph with S as vertices, where we put an edge between two segments iff their distance is at most r. The reverse shortest path problem for (unweighted) line segments can then be defined in the same way.

This problem was first considered by Agarwal *et al.* [4], who gave a randomized $O^*(n^{4/3})$-time algorithm. Later, Agarwal *et al.* [3] solved the corresponding decision problem in $O(n \log^2 n)$-time. They then applied the shrink-and-bifurcate technique to obtain an improved randomized algorithm for the optimization problem running in $O^*(n^{6/5})$ time, assuming that the spread (the maximum-to-minimum distance ratio) is polynomially bounded.

Discrete Fréchet distance with one-sided shortcuts. The *discrete Fréchet distance* is a popular measure of closeness between two polygonal chains. Given polygonal chains $P = \langle p_1, \ldots, p_{|P|} \rangle$ and $Q = \langle q_1, \ldots, q_{|Q|} \rangle$ with $|P| + |Q| = n$, consider two frogs, a P-frog

and a Q-frog, starting at points p_1 and q_1 respectively. In each move, one frog leap from its current vertex to the next vertex on their respective polygonal chain, while the other frog stays at its current vertex. The discrete Fréchet distance between P and Q is defined as the minimum distance r^* for which the two frogs can arrive at $p_{|P|}$ and $q_{|Q|}$, such that the Euclidean distance between the two frogs is always at most r^* before and after every leap.

The discrete Fréchet distance with *one-sided shortcuts* is defined similarly, except that we allow one frog, say the P-frog, to leap to any vertex ahead during each move. In other words, the one-sided discrete Fréchet distance between P and Q is the minimum discrete Fréchet distance between P' and Q, over all subsequences P' of P that start at p_1 and end at $p_{|P|}$. The problem of computing the discrete Fréchet distance with one-sided shortcuts is what first prompted Avraham *et al.* [6] to introduce their shrink-and-bifurcate technique. They noted a linear-time algorithm for the decision problem and obtained an $O^*(n^{6/5})$-time randomized algorithm for the optimization problem.

Reverse shortest path in visibility graphs over 1.5-dimensional terrains. A *terrain* T in 1.5 dimensions refers to an x-monotone polygonal chain in the plane. Two points p, q on or above the terrain are said to *see* each other if the line segment $\overline{pq}$ does not intersect T. Given a 1.5-dimensional terrain T and a set of points P on or above T with $|P| + |T| = n$, the visibility graph $G(P, T)$ is a graph with vertices P and edges indicating which pairs of points see each other.

In the reverse shortest path problem over a terrain T, we are given a set P of points on T, and we want to erect a tower at each point in P. In particular, given a terrain T, a set of points P on T, a pair of points $s, t \in P$, and a positive integer λ, we are asked to find the minimum real number h^* such that the visibility graph $G(P(h^*), T)$ contains an s-t path with at most λ edges, where $P(h^*)$ is the tips of the towers of height h^* erected at the points of P.

This problem was first introduced by Agarwal *et al.* [4], who gave a randomized $O^*(n^{4/3})$-time algorithm. Katz *et al.* [23] solved the decision problem in $\tilde{O}(n)$ time by a careful implementation of BFS on the visibility graph, with clever uses of range searching data structures. Katz *et al.* then applied the shrink-and-bifurcate technique to obtain a randomized $O^*(n^{6/5})$-time algorithm for the optimization problem.

1.1 The Shrink-and-Bifurcate Technique

As noted, the current fastest algorithms for all of the above problems are obtained via the shrink-and-bifurcate technique introduced by Avraham *et al.* [6]. The setting requires that critical values (values that the decision algorithm may compare the input value against) are distances defined by pairs of input objects for some distance function. The strategy consists of two stages, *interval shrinking* and *bifurcation*.

1. The interval shrinking stage is a relaxed version of binary search – instead of insisting on finding the answer r^* immediately, we seek an interval $(r^-, r^+]$ that contains r^* and that contains at most L critical values for a given parameter L. In the case of Euclidean distances of points in the plane, Avraham *et al.* showed that this stage can be done in $O^*(n^{4/3}/L^{1/3} + D(n))$ time (using randomization), where $D(n)$ is the running time of the decision algorithm. This is potentially better than the $\tilde{O}(n^{4/3})$ time bound for distance selection.

2. The bifurcation stage then uses this interval $(r^-, r^+]$ to simulate the decision algorithm at the unknown r^* more efficiently than standard parametric search when the decision algorithm is not parallelizable. Avraham et al. showed that this simulation can be done in $\tilde{O}(L^{1/2}D(n))$ time.

When $D(n) = \tilde{O}(n)$, choosing $L = n^{2/5}$ to balance the costs of the two stages gives an overall running time of $O^*(n^{6/5})$.

Some of the applications use distance functions other than Euclidean distances of points in the plane, but the $O^*(n^{4/3}/L^{1/3} + D(n))$ bound for interval shrinking is still applicable. An exception is the reverse shortest path problem for general disk graphs. Here, the corresponding distance selection problem requires $O^*(n^{3/2})$ time, and the interval shrinking subproblem is solved in $O^*(n^{3/2}/L^{1/2} + D(n))$ time instead. When $D(n) = \tilde{O}(n)$, choosing $L = n^{1/2}$ to balance the costs of the two stages gives an overall running time of $O^*(n^{5/4})$ instead.

1.2 New Results

The new contributions in this paper are twofold:

1. We improve all the previous $O^*(n^{6/5})$ time bounds – for reverse shortest path in unit-disk graphs, weighted unit-disk graphs, segment proximity graphs, visibility graphs over 1.5-dimensional terrains, and discrete Fréchet distance with one-sided shortcuts – to $\tilde{O}(n^{8/7})$. (In particular, for discrete Fréchet distance with one-sided shortcuts, this is the first improvement on the exponent in 10 years.)

   For reverse shortest path in arbitrary disk graphs (unweighted or weighted), we improve the previous $O^*(n^{5/4})$ time bound to $O^*(n^{6/5})$.

   Although the improvements are incremental, they are all obtained from one simple idea for improving interval shrinking (we keep the bifurcation part unchanged) – the simplicity and generality of our idea make it easy to apply to existing (and potentially future) applications of the shrink-and-bifurcate approach, and are the main takeaways of the paper. More precisely, we show how to lower the $O^*(n^{4/3}/L^{1/3} + D(n))$ time bound for interval shrinking subproblem to $\tilde{O}(n^{4/3}/L^{2/3} + D(n))$ (or in the arbitrary disk graph case, lower $O^*(n^{3/2}/L^{1/2} + D(n))$ to $O^*(n^{3/2}/L^{3/4} + D(n))$). The original interval shrinking method by Avraham *et al.* needs to modify range searching techniques for distance selection, constructing "partial biclique covers" and then applying random sampling. Our new method, in contrast, can be described in just a few lines, and simply uses a known distance selection algorithm *as a black box* on random subsets (see Section 2).

2. Next, we focus more closely on the simplest of this class of problems, namely, reverse shortest path in (unweighted) unit-disk graphs. For this specific problem, we are able to further improve the running time to $\tilde{O}(n^{9/8})$. The improvement of the exponent from $6/5 = 1.2$ to $9/8 = 1.125$ is a bigger increment (for example, comparable in magnitude to Wang and Zhao's original improvement [29] from $4/3 \approx 1.333$ to $5/4 = 1.25$, or Kaplan *et al.*'s improvement [21] from $5/4 = 1.25$ to $6/5 = 1.2$).

   Interestingly, we obtain our result by combining the shrink-and-bifurcate technique with Wang and Zhao's original grid-based approach! Although it may be natural to consider combining the two approaches, how to do so is not obvious at all. We need to put together several nontrivial technical ideas – this is the most intricate part of the paper (see Section 3).

2 Interval Shrinking

In this section, we formally define the *interval shrinking* subproblem in an abstract, self-contained setting, and then describe our new randomized method for this subproblem.

▶ **Subproblem 1** (Interval shrinking). *Suppose we have an unknown value r^*. Suppose there is a* decision oracle *that can decide whether $r^* \le r$ for an input value r in $D(n)$ time where $D(n) \ge n$.*

Let $\delta(\cdot,\cdot)$ be a function over pairs of objects. Suppose there is a selection oracle *that, given two sets P and Q of objects and a number k, can compute the k-th smallest value in $\delta(P,Q) := \{\delta(p,q) \mid (p,q) \in P \times Q\}$ in $K(|P|,|Q|) = \tilde{O}(|P|^\alpha |Q|^\alpha + |P| + |Q|)$ time for some constant $\alpha < 1$.*

Given two sets P and Q of objects with $|P|+|Q| = n$ and a number L, we want to compute an interval $(r^-, r^+]$ that contains r^* and that contains at most $\tilde{O}(L)$ values in $\delta(P,Q)$.

If δ is the Euclidean distance function for points in the plane (which is the case for the applications to reverse shortest path in unit-disk graphs or discrete Fréchet distance with one-sided shortcuts), known distance selection algorithms [1, 24, 28, 13] between an m-point set and an n-point set run in $K(m,n) = \tilde{O}(m^{2/3}n^{2/3} + m + n)$ time, and thus we have $\alpha = 2/3$. In this planar Euclidean case, Avraham *et al.* [6] solved the above interval shrinking problem in $O^*(n^{4/3}/L^{1/3} + D(n))$ time with high probability. We will describe an improved method.

It suffices to describe how to find an $r^+ > r^*$ such that $[r^*, r^+]$ contains at most $\tilde{O}(L)$ values in $\delta(P,Q)$, since finding r^- is analogous. Also, it suffices to design an algorithm which is correct with probability $\Omega(1)$, since we can get an algorithm which is correct w.h.p.[3] by repeating for logarithmically many trials and returning the minimum r^+ found.

2.1 First Version

Our first interval shrinking algorithm is simple, as presented in Algorithm 1.

Algorithm 1 Basic interval shrinking.

1. Take random subsets $\widehat{P} \subseteq P$ and $\widehat{Q} \subseteq Q$ where each element is chosen independently with probability $\frac{1}{\sqrt{L}}$
2. **return** the successor r^+ of r^* in $\delta(\widehat{P},Q) \cup \delta(P,\widehat{Q})$, which can be computed by binary searches in $\delta(\widehat{P},Q)$ and $\delta(P,\widehat{Q})$ using the selection and decision oracles

Correctness. Let $E \subseteq P \times Q$ be the set of L pairs that have the L smallest $\delta(\cdot,\cdot)$ values after r^*. Think of E as a bipartite graph between P and Q. Then E has at least $\sqrt{L}$ non-isolated vertices in P or at least $\sqrt{L}$ non-isolated vertices in Q (because otherwise, E would have fewer than L edges). W.l.o.g., assume the former. Then $\widehat{P} \times Q$ hits E with probability at least $1 - (1 - \frac{1}{\sqrt{L}})^{\sqrt{L}} = \Omega(1)$. This implies that there are at most L values in $\delta(\widehat{P},Q)$ between r^* and r^+ with probability $\Omega(1)$.

[3] With high probability, i.e., probability $1 - O(1/n^c)$ for an arbitrarily large constant c.

Running time. Note that $|\widehat{P}|, |\widehat{Q}| = \tilde{O}(\frac{n}{\sqrt{L}})$ w.h.p. Using the selection and decision oracles, w.h.p., the binary searches can be done in time

$$\tilde{O}\left(\left(\frac{n}{\sqrt{L}}\right)^{\alpha} n^{\alpha} + n^{\alpha}\left(\frac{n}{\sqrt{L}}\right)^{\alpha} + D(n)\right) = \tilde{O}(n^{2\alpha}/L^{\alpha/2} + D(n)).$$

For planar Euclidean distances with $\alpha = 2/3$, we have thus recovered the $O^*(n^{4/3}/L^{1/3} + D(n))$ result of Avraham *et al.* [6]. However, our algorithm is arguably simpler, because, unlike Avraham *et al.*, we invoke distance selection as a black box, without needing to modify the internal mechanisms of distance selection.

We remark that the idea of randomly sampling *pairs* of input objects is more common and has been used before in the context of slope selection or distance selection [25, 16, 9] (and in Avraham *et al.*'s previous method [6]). But here we are sampling the input objects themselves (which in some sense is even simpler, though the analysis is slightly trickier).

2.2 Improved Version

Upon closer examination, one may notice that the basic algorithm has room for improvement. In particular, if E has $\Theta(\sqrt{L})$ non-isolated vertices in P and $\Theta(\sqrt{L})$ non-isolated vertices in Q, then clearly our samples $\widehat{P} \times Q$ and $P \times \widehat{Q}$ are wasteful. Indeed, we can eliminate such redundancy by sampling in P and Q simultaneously with different rates. We present the details in Algorithm 2.

Algorithm 2 Improved interval shrinking.

for $i = 1, \ldots, \log n$ **do**
  Take a random subset $\widehat{P}_i \subseteq P$ where each element is chosen independently with probability $\frac{1}{\lceil L/2^i \rceil}$
  Take a random subset $\widehat{Q}_i \subseteq Q$ where each element is chosen independently with probability $\frac{1}{2^i}$
end
return the successor r^+ of r^* in $\bigcup_{i=1}^{\log n} \delta(\widehat{P}_i, \widehat{Q}_i)$, which can be computed by binary search in each $\delta(\widehat{P}_i, \widehat{Q}_i)$ using the selection and decision oracles

Correctness. Let $E \subseteq P \times Q$ be the set of $L \log n$ pairs that have the $L \log n$ smallest $\delta(\cdot, \cdot)$ values after r^*. Consider the bipartite graph $H = (P \sqcup Q, E)$. Let $M_i \subseteq P$ be the vertices with degree in the range $[2^{i-1}, 2^i)$ in H. Since $\sum_{i=1}^{\log n} 2^i |M_i| \geq L \log n$, there must exist an index i_0 for which $|M_{i_0}| \geq \lceil L/2^{i_0} \rceil$. Observe that $\widehat{P}_{i_0} \times \widehat{Q}_{i_0}$ hits E if

(i) $\widehat{P}_{i_0}$ hits M_{i_0}, and

(ii) for some $p \in \widehat{P}_{i_0} \cap M_{i_0}$, $\widehat{Q}_{i_0}$ hits the neighborhood of p in E.

Now, (i) holds with probability at least $1 - (1 - \frac{1}{\lceil L/2^{i_0} \rceil})^{\lceil L/2^{i_0} \rceil} = \Omega(1)$. For a fixed $p \in \widehat{P}_{i_0} \cap M_{i_0}$, the probability that $\widehat{Q}_{i_0}$ contains at least one neighbor of p is at least $1 - (1 - \frac{1}{2^{i_0}})^{2^{i_0 - 1}} = \Omega(1)$. Thus, conditioned on (i), the probability of (ii) is $\Omega(1)$. Hence, $\widehat{P}_{i_0} \times \widehat{Q}_{i_0}$ hits E with probability $\Omega(1)$. This implies that there are at most $L \log n$ values in $\delta(P, Q)$ between r^* and r^+ with probability $\Omega(1)$.

Running time. Note that $|\widehat{P}_i| = \tilde{O}(\frac{n}{\lceil L/2^i \rceil})$ and $|\widehat{Q}_i| = \tilde{O}(\frac{n}{2^i})$ w.h.p. Using the selection and decision oracles, w.h.p. the binary searches can be done in time

$$\tilde{O}\left(\sum_{i=1}^{\log n}\left(\left(\frac{n}{\lceil L/2^i \rceil}\right)^{\alpha}\left(\frac{n}{2^i}\right)^{\alpha} + D(n)\right)\right) = \tilde{O}(n^{2\alpha}/L^{\alpha} + D(n)).$$

▶ **Theorem 2.** *Subproblem 1 (interval shrinking) can be solved in $\tilde{O}(n^{2\alpha}/L^{\alpha} + D(n))$ time w.h.p.*

For example, for planar Euclidean distances, the bound is $\tilde{O}(n^{4/3}/L^{2/3} + D(n))$, which is noticeably better than $\tilde{O}(n^{4/3}/L^{1/3} + D(n))$.

2.3 Combining with Bifurcation

After interval shrinking, Avraham *et al.* [6] devised a clever variant of parametric search via an idea called *bifurcation* (also redescribed in more generality in Kaplan *et al.*'s subsequent paper [21]) which accomplishes the following:

▶ **Lemma 3** (Bifurcation [6])**.** *Suppose we have an unknown value r^*. Suppose there is a* decision oracle *that can decide whether $r^* \le r$ for an input value r in $D(n)$ time.*

Suppose we have an algorithm $\mathcal{A}$ that has one input parameter and runs in no more than $O(D(n))$ time, and during the execution of the algorithm, it can only access its input parameter via binary comparisons. The values that $\mathcal{A}$ can compare its input parameter with are called the critical values*.*

Lastly, suppose that we are also given an interval $(r^-, r^+]$ that is guaranteed to contain r^* and that contains at most $\tilde{O}(L)$ critical values.

Then we can simulate $\mathcal{A}$ on input r^* in $\tilde{O}(L^{1/2}D(n))$ time.

Typically, in applications of the technique, the algorithm $\mathcal{A}$ we want to simulate is the decision algorithm itself; here, the simulation produces an interval containing r^* such that all input values lying in the interval have the same computation path and thus the same answer as r^* (namely, "yes"), and so the left endpoint of this interval gives us precisely the value of r^*.

We will not redescribe the proof of the above lemma, as we will only use it as a black box. (The rough idea in bifurcation is to simulate multiple branches of $\mathcal{A}$ at the same time, until we have accumulated sufficiently many comparisons and will then resolve all of them as a batch using the decision oracle; comparisons with values outside the interval $(r^-, r^+]$ are free.) Combining our improved interval shrinking method with bifurcation, we immediately obtain better results for all optimization problems stated in the introduction:

▶ **Theorem 4.**

(a) *The reverse shortest path problem in unweighted and weighted unit-disk graphs, segment proximity graphs (assuming polynomial spread), and visibility graphs over 1.5-dimensional terrains, and the discrete Fréchet distance problem with one-sided shortcuts, as defined in the introduction, can all be solved in $\tilde{O}(n^{8/7})$ time w.h.p. by (Las Vegas) randomized algorithms.*

(b) *The reverse shortest path problem in unweighted and weighted disk graphs can be solved in $O^*(n^{6/5})$ time w.h.p. by (Las Vegas) randomized algorithms.*

Proof. As mentioned in the introduction, all these problems have known decision algorithms running in $D(n) = \tilde{O}(n)$ time [6, 21, 23, 3]. In all these decision algorithms, critical values are indeed obtained from some function δ over pairs of input objects.

For all the applications in (a), the function δ satisfies $\alpha = 2/3$. (Some care is needed for reverse shortest path in visibility graphs in 1.5-dimensional terrains; see the full version.)

Combining Theorem 2 with Lemma 3, we get total running time

$$\tilde{O}\left(\frac{n^{4/3}}{L^{1/3}} + L^{1/2}n\right).$$

Choosing $L = n^{2/7}$ yields $\tilde{O}(n^{8/7})$.

For the applications in (b), we have $\alpha = 3/4 + o(1)$ [21]. The total running time becomes

$$O^*\left(\frac{n^{3/2}}{L^{3/4}} + L^{1/2}n\right).$$

Choosing $L = n^{2/5}$ yields $O^*(n^{6/5})$. ◀

3 Reverse Shortest Path in Unweighted Unit-Disk Graphs

For reverse shortest path in unweighted unit-disk graphs, we show how to improve the running time even further.

In this section, $\delta(p, q)$ denotes the Euclidean distance between points p and q, and $\delta(P, Q) := \{\delta(p, q) \mid (p, q) \in P \times Q\}$ for two point sets P and Q. Let $d_G(s, t)$ denote the shortest-path distance between s and t in a graph G.

We are given an input set P of n points in the plane, $s, t \in P$, and a number λ. Let $G_r(P)$ be the (unweighted) graph with vertex set P, where we put an edge between p and q iff $\delta(p, q) \leq r$. We want to compute the smallest r^* for which $d_{r^*}(s, t) \leq \lambda$, where we use $d_r(s, t)$ as a shorthand for $d_{G_r(P)}(s, t)$.

3.1 Recap of Chan and Skrepetos' Decision Algorithm

Our algorithms rely on the details of the known $\tilde{O}(n)$-time decision algorithm by Chan and Skrepetos [12], so we will first give a quick recap of their algorithm.

Given an input value r, we want to compute $d_r(s, t)$. We first create a uniform grid Ψ_r of side length $r/\sqrt{2}$. Chan and Skrepetos' algorithm is a careful implementation of BFS using the grid. We start from s and generate each level of the BFS tree (the "frontier") one at a time. The key observation is that each grid cell only needs to be visited a constant number of times during the BFS, because (i) points in the same cell have the roughly same shortest-path distance ± 1 from s in $G_r(P)$ and so each cell "appears" in only $O(1)$ levels of the BFS tree, and (ii) each cell has only $O(1)$ neighboring cells – we say that two cells are *neighboring* if the minimum Euclidean distance between the two cells is at most r. Given one level of the BFS tree, we can generate the next level (i.e., "expand" the frontier) by solving the following subproblems between cells in the current level and their neighboring cells:

▶ **Subproblem 5.** *Given an input value r, a subset of n_r red points in one grid cell σ and a subset of n_b blue points in a neighboring grid cell σ', determine for each blue point whether there is a red point at distance at most r from it.*

Chan and Skrepetos solved the above subproblem in $O(n_r + n_b)$ time, assuming that the input points are pre-sorted, by using envelopes of pseudo-lines. As a result, the total running time of their algorithm is linear after pre-sorting.

The above algorithm is not efficiently parallelizable (since the number of levels in the BFS may be large). Instead, we can apply the bifurcation technique to simulate the algorithm on r^*. However, one issue arises: the envelope computations requires critical values defined

by triples of input elements, not pairs. As noted in other papers [21, 29], this can be fixed by switching to a different solution to Subproblem 5 which is slightly slower by a logarithmic factor but is simpler: namely, we compute the nearest red neighbor to each blue point by point location in the Voronoi diagram of the red points in $O((n_r + n_b)\log(n_r + n_b))$ time [15]; then for each blue point, we just compare r with its nearest neighbor distance. This way, the critical values are all Euclidean distances of pairs of input points.

Another issue is the construction of the grid Ψ_r, i.e., the assignment of input points to grid cells in Ψ_r. This part is easily parallelizable, and so we can apply standard parametric search to construct the grid Ψ_{r^*} in $\tilde{O}(n)$ time, all done as preprocessing. (In fact, the time bound for constructing Ψ_{r^*} is $O(n \log n)$, as shown by Wang and Zhao [29].)

3.2 Heavy vs. Light Cells

Applying the shrink-and-bifurcate technique to the preceding decision algorithm using our improved interval shrinking method yields an overall time bound of $\tilde{O}(n^{8/7})$, as we have already shown in Theorem 4(a). To obtain our best result, we will combine with another idea that was used in Wang and Zhao's original paper [29]: heavy vs. light cells.

Recall that we have already constructed the grid Ψ_{r^*}. Let Δ be a parameter to be chosen later. Call a grid cell σ *heavy* if $|P \cap \sigma| \geq \Delta$, or *light* otherwise. The number of heavy cells is clearly at most $O(n/\Delta)$. Furthermore, a pair of neighboring cells (σ, σ') is called a *light pair* if one of σ and σ' is a light cell (the other may be light or heavy).

Our new strategy is as follows. First, we perform interval shrinking, but only with respect to distances of points from light pairs. Intuitively, there are fewer such distances and so interval shrinking should be doable faster. This allows us to simulate Chan and Skrepetos' algorithm at r^* on a modified graph where heavy cells are eliminated – or more precisely, points in a common heavy cell are contracted into a single vertex. When we "uncontract" the heavy cells, this causes an $O(n/\Delta)$ additive error to all the shortest-path distances in $G_{r^*}(P)$. As a final step, we show how to use a dynamic program to recover the value r^*. In the next three subsections, we provide the details of all these steps.

3.3 Faster Interval Shrinking for Light Pairs

In the following, we show how to solve the interval shrinking problem faster for distances from light pairs. More precisely, define $\delta_{\text{light}}(p, q) = \delta(p, q)$ if $p \in \sigma$ and $q \in \sigma'$ for some light pair (σ, σ'), and $\delta_{\text{light}}(p, q) = \infty$ otherwise. We apply Algorithm 2 to do interval shrinking with respect to δ_{light}.

To analyze the running time, we need an efficient implementation of the selection oracle for

$$\delta_{\text{light}}(\widehat{P}_i, \widehat{Q}_i) \;=\; \bigcup_{\text{light pair } (\sigma,\sigma')} \delta(\widehat{P}_i \cap \sigma, \widehat{Q}_i \cap \sigma') \;\cup\; \{\infty\}.$$

Let $n_\sigma := |P \cap \sigma|$ for each cell σ. Note that $|\widehat{P}_i \cap \sigma| = \tilde{O}(\frac{n_\sigma}{\lceil L/2^i \rceil})$ and $|\widehat{Q}_i \cap \sigma'| = \tilde{O}(\frac{n_{\sigma'}}{2^i})$ for every (σ, σ') w.h.p. We already have a selection oracle for each $\delta(\widehat{P}_i \cap \sigma, \widehat{Q}_i \cap \sigma')$ that w.h.p. runs in time

$$\tilde{O}\left(\left(\frac{n_\sigma}{\lceil L/2^i \rceil}\right)^{2/3}\left(\frac{n_{\sigma'}}{2^i}\right)^{2/3} + n_\sigma + n_{\sigma'}\right) \;=\; \tilde{O}\left(\frac{n_\sigma^{2/3} n_{\sigma'}^{2/3}}{L^{2/3}} + n_\sigma + n_{\sigma'}\right).$$

To obtain a selection oracle for $\delta_{\text{light}}(\widehat{P}_i, \widehat{Q}_i)$, we face a problem analogous to selection in a union of multiple sorted arrays, except that the arrays are implicitly given. Frederickson and Johnson [18] gave an algorithm for selection in k sorted arrays of size n running in $O(k \log n)$ time; the algorithm proceeds in $O(\log n)$ rounds, and in each round, it looks up only $O(1)$ elements in each array. In our scenario, each array lookup corresponds to a call to the selection oracle for $\delta(\widehat{P}_i \cap \sigma, \widehat{Q}_i \cap \sigma')$ for a light pair (σ, σ'). Thus, by Frederickson and Johnson's algorithm, selection in the union $\delta_{\text{light}}(\widehat{P}_i, \widehat{Q}_i)$ can be done in total time

$$\tilde{O}\left(\sum_{\text{light pair } (\sigma,\sigma')} \left(\frac{n_\sigma^{2/3} n_{\sigma'}^{2/3}}{L^{2/3}} + n_\sigma + n_{\sigma'}\right)\right)$$
$$\leq \tilde{O}\left(\sum_{\text{light pair } (\sigma,\sigma')} \frac{\Delta^{1/3}(n_\sigma + n_{\sigma'})}{L^{2/3}} + n_\sigma + n_{\sigma'}\right) = \tilde{O}\left(\frac{\Delta^{1/3} n}{L^{2/3}} + n\right),$$

since $\min\{n_\sigma, n_{\sigma'}\} \leq \Delta$ for each light pair (σ, σ'), and $\sum_{\text{neighboring pair } (\sigma,\sigma')}(n_\sigma + n_{\sigma'}) = O(n)$. Therefore, the total time for interval shrinking with respect to δ_{light} is $\tilde{O}(\Delta^{1/3} n / L^{2/3} + D(n))$ w.h.p.

3.4 Computing Distances with $O(n/\Delta)$ Additive Error

Let H_r be the graph obtained from $G_r(P)$ by contracting the points inside each heavy cell of Ψ_r into a single vertex. Since there are $O(n/\Delta)$ contracted vertices in H_r, we have $d_r(s,p) - k \leq d_{H_r}(s,p) \leq d_r(s,p)$ with $k = O(n/\Delta)$, for every $p \in P$.

It is straightforward to modify Chan and Skrepetos' decision algorithm to do BFS on this modified graph H_r. Expanding the frontier again reduces to solving Subproblem 5 for pairs of neighboring cells. Note that for two neighboring cells that are both heavy, we can determine whether there is an edge between them by pre-computing the bichromatic closest pair between the points in the neighboring cells (via Voronoi diagrams) and testing r against this closest pair distance – call this a *special* distance. This pre-computation takes $O(n \log n)$ total time.

After performing interval shrinking as described in previous subsection, we apply the bifurcation technique (Lemma 3) to simulate BFS on H_{r^*}. The critical values indeed are all Euclidean distances from light pairs only, except for the $O(n)$ special distances above, but initially before the simulation starts, we can resolve the comparisons with all $O(n)$ special distances by binary search using $O(\log n)$ calls to the decision algorithm in $\tilde{O}(n)$ time.

At the end of the simulation, we will then have computed $d_{H_{r^*}}(s,p)$ for all $p \in P$, which yield estimates to $d_{r^*}(s,p)$ with additive error $O(n/\Delta)$. To summarize:

▶ **Lemma 6.** *In time $\tilde{O}(\Delta^{1/3} n / L^{2/3} + L^{1/2} n)$, one can compute for every $p \in P$ an estimate $\tilde{d}_p$ with $d_{r^*}(s,p) - k \leq \tilde{d}_p \leq d_{r^*}(s,p)$, where $k = O(n/\Delta)$.*

3.5 Recovering r^*

Finally, we will compute r^* using the estimates obtained in Lemma 6. This final step is done via a dynamic program, and does not require any knowledge of the details of the earlier steps.

▶ **Lemma 7.** *Given a number k and estimates $\tilde{d}_p$ with $d_{r^*}(s,p) - k \leq \tilde{d}_p \leq d_{r^*}(s,p)$ for all $p \in P$, one can compute r^* in $\tilde{O}(kn)$ time.*

To prove Lemma 7, it is helpful to view the problem as computing a *minimum bottleneck path* from s to t with at most λ links (as mentioned in the introduction), i.e., a path from s to t with at most λ links that minimizes the maximum Euclidean length of the edges along the path.

There is a straightforward dynamic programming solution to this problem: Let $D_i(p)$ denote the minimum bottleneck value over all paths from s to p with i links. We want $r^* = \min_{i \le \lambda} D_i(t)$. We have the following recursive formula: for each $p \in P$,

$$D_i(p) = \min_{q \in P} \max\{D_{i-1}(q), \delta(p,q)\}, \tag{1}$$

with $D_0(s) = 0$ and $D_0(p) = \infty$ for all $p \in P - \{s\}$ as base cases. Unfortunately, in this dynamic program, the number of table entries is $O(n\lambda)$, which is $O(n^2)$ in the worst case.

We use the given estimates $\tilde{d}_p$ to speed up the dynamic program. The main observation is that it suffices to generate $D_i(p)$ when $i - k \le \tilde{d}_p \le i$. The number of such table entries is thus reduced to $O(kn)$.

More precisely, define a subset $P_i = \{p \in P : i - k \le \tilde{d}_p \le i\}$ for each i. Since a point belongs to $O(k)$ of these subsets, $\sum_i |P_i| = O(kn)$. For each $p \in P_i$, define

$$\widehat{D}_i(p) = \min_{q \in P_{i-1}} \max\{\widehat{D}_{i-1}(q), \delta(p,q)\}, \tag{2}$$

with $\widehat{D}_0(s) = 0$. (All unspecified $\widehat{D}_i(p)$ values are implicitly set to ∞.)

We claim that $r^* = \min_{i \le \lambda} \widehat{D}_i(t)$.

Correctness. It is obvious that $\widehat{D}_i(p) \ge D_i(p)$ for all p and i. Thus, $\min_{i \le \lambda} \widehat{D}_i(t) \ge \min_{i \le \lambda} D_i(t) = r^*$.

For the reverse direction, take a shortest path $\langle u_0, u_1, \ldots, u_i \rangle$ from s to t in $G_{r^*}(P)$, with $u_0 = s$, $u_i = t$, and $i \le \lambda$. Note that $d_{r^*}(s, u_j) = j$ for every $j = 0, \ldots, i$. So, $j - k \le \tilde{d}_{u_j} \le j$, i.e., $u_j \in P_j$ for every $j = 0, \ldots, i$. It follow from (2) that $\widehat{D}_0(u_0) = 0$, $\widehat{D}_1(u_1) \le r^*$, $\widehat{D}_2(u_2) \le r^*$, $\ldots$, $\widehat{D}_i(u_i) \le r^*$ (since $\delta(u_0, u_1), \ldots, \delta(u_{i-1}, u_i) \le r^*$). Thus, $\min_{i \le \lambda} \widehat{D}_i(t) \le r^*$.

Running time. For an efficient implementation of the dynamic program, we need a data structure for the following variant of nearest neighbor queries:

▶ **Lemma 8.** *Given a set Q of n points in the plane where each point q has a weight w_q, we can preprocess Q in $O(n \log^2 n)$ time so that given any query point p, we can find $q \in Q$ minimizing $\max\{\delta(p,q), w_q\}$ in $O(\log^2 n)$ time, where $\delta(\cdot,\cdot)$ denotes Euclidean distance.*

Proof. We apply standard multi-level geometric data structuring techniques [2]: Order Q by increasing weights. Split Q into the left half Q_L and the right half Q_R. Store a point location structure for the Voronoi diagram of Q_L [15]. Recursively build data structures for Q_L and for Q_R. The preprocessing time satisfies the recurrence $P(n) = 2P(n/2) + O(n \log n)$.

Given a query point p, we first find its nearest neighbor q_L in Q_L in $O(\log n)$ time.

- Case 1: $\delta(p, q_L) \le w_{q_L}$. Then $\min_{q \in Q_R} \max\{\delta(p,q), w_q\} \ge w_{q_L} = \max\{\delta(p, q_L), w_{q_L}\}$, and so it suffices to recurse in Q_L.
- Case 2: $\delta(p, q_L) > w_{q_L}$. Then $\min_{q \in Q_L} \max\{\delta(p,q), w_q\} = \delta(p, q_L)$, and so it suffices to recurse in Q_R.

The query time satisfies the recurrence $Q(n) = Q(n/2) + O(\log n)$, and thus we have $Q(n) = O(\log^2 n)$. ◀

Having computed $\widehat{D}_{i-1}(\cdot)$, we can evaluate $\widehat{D}_i(\cdot)$ according to (2) by preprocessing P_{i-1} in the data structure from Lemma 8 (with weights $w_q := \widehat{D}_{i-1}(q)$) and answering $|P_i|$ queries. The running time is $\tilde{O}(|P_{i-1}| + |P_i|)$. Summing over all i yields total running time $\tilde{O}(kn)$. This completes the proof of Lemma 7.

▶ **Theorem 9.** *The reverse shortest path problem in (unweighted) unit-disk graphs can be solved in $\tilde{O}(n^{9/8})$ time w.h.p. by a (Las Vegas) randomized algorithm.*

Proof. By combining Lemma 6 and Lemma 7, we obtain an algorithm that runs in time

$$\tilde{O}\left(\frac{\Delta^{1/3}n}{L^{2/3}} + L^{1/2}n + \frac{n^2}{\Delta}\right).$$

Choosing $L = n^{1/4}$ and $\Delta = n^{7/8}$ yields the desired result. ◀

References

1. Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting distances in the plane. *Algorithmica*, 9(5):495–514, 1993. doi:10.1007/BF01187037.
2. Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. AMS Press, 1999.
3. Pankaj K. Agarwal, Haim Kaplan, Matthew J. Katz, and Micha Sharir. Segment proximity graphs and nearest neighbor queries amid disjoint segments. In *Proc. 32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 7:1–7:20, 2024. doi:10.4230/LIPICS.ESA.2024.7.
4. Pankaj K. Agarwal, Matthew J. Katz, and Micha Sharir. On reverse shortest paths in geometric proximity graphs. *Comput. Geom.*, 117:102053, 2024. Preliminary version in ISAAC 2022. doi:10.1016/J.COMGEO.2023.102053.
5. Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30(4):412–458, 1998. doi:10.1145/299917.299918.
6. Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Trans. Algorithms*, 11(4):29:1–29:29, 2015. Preliminary version in SoCG 2014. doi:10.1145/2700222.
7. Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Comput. Geom.*, 48(4):360–367, 2015. doi:10.1016/J.COMGEO.2014.12.003.
8. Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.
9. Timothy M. Chan. On enumerating and selecting distances. *Int. J. Comput. Geom. Appl.*, 11(3):291–304, 2001. doi:10.1142/S0218195901000511.
10. Timothy M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 430–436, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982853.
11. Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. Optimal algorithms for geometric centers and depth. *SIAM J. Comput.*, 51(3):627–663, 2022. doi:10.1137/21M1423324.
12. Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proc. 27th International Symposium on Algorithms and Computation (ISAAC)*, volume 64 of *LIPIcs*, pages 24:1–24:13, 2016. doi:10.4230/LIPICS.ISAAC.2016.24.
13. Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log* shaving, two-dimensional fractional cascading, and decision trees. *ACM Trans. Algorithms*, 20(3):24, 2024. doi:10.1145/3591357.

14 Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34(1):200–208, 1987. doi:10.1145/7531.7537.

15 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: https://www.worldcat.org/oclc/227584184, doi:10.1007/978-3-540-77974-2.

16 Michael B. Dillencourt, David M. Mount, and Nathan S. Netanyahu. A randomized algorithm for slope selection. *Int. J. Comput. Geom. Appl.*, 2(1):1–27, 1992. doi:10.1142/S0218195992000020.

17 Alon Efrat, Alon Itai, and Matthew J Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001. doi:10.1007/s00453-001-0016-8.

18 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.

19 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.

20 Te C. Hu. The maximum capacity route problem. *Operations Research*, 9(6):898–900, 1961.

21 Haim Kaplan, Matthew J. Katz, Rachel Saban, and Micha Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. In *Proc. 31st Annual European Symposium on Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 67:1–67:14, 2023. doi:10.4230/LIPICS.ESA.2023.67.

22 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.

23 Matthew J. Katz, Rachel Saban, and Micha Sharir. Near-linear algorithms for visibility graphs over a 1.5-dimensional terrain. In *Proc. 32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 77:1–77:17, 2024. doi:10.4230/LIPICS.ESA.2024.77.

24 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM J. Comput.*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.

25 Jirí Matoušek. Randomized optimal algorithm for slope selection. *Inf. Process. Lett.*, 39(4):183–187, 1991. doi:10.1016/0020-0190(91)90177-J.

26 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983. doi:10.1145/2157.322410.

27 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discret. Comput. Geom.*, 64(4):1141–1166, 2020. doi:10.1007/S00454-020-00219-7.

28 Haitao Wang and Yiming Zhao. Improved algorithms for distance selection and related problems. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 101:1–101:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.101.

29 Haitao Wang and Yiming Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023. Preliminary versions in WADS 2021 and WALCOMM 2022. doi:10.20382/JOCG.V14I1A2.

On Zarankiewicz's Problem for Intersection Hypergraphs of Geometric Objects

Timothy M. Chan
Siebel School of Computing and Data Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

Chaya Keller
School of Computer Science, Ariel University, Israel

Shakhar Smorodinsky
Department of Computer Science, Ben-Gurion University of the NEGEV, Be'er Sheva, Israel

Abstract

In this paper we study the hypergraph Zarankiewicz's problem in a geometric setting – for r-partite intersection hypergraphs of families of geometric objects. Our main results are essentially sharp bounds for families of axis-parallel boxes in $\mathbb{R}^d$ and families of pseudo-discs. For axis-parallel boxes, we obtain the sharp bound $O_{d,t}(n^{r-1}(\frac{\log n}{\log\log n})^{d-1})$. The best previous bound was larger by a factor of about $(\log n)^{d(2^{r-1}-2)}$. For pseudo-discs, we obtain the bound $O_t(n^{r-1}(\log n)^{r-2})$, which is sharp up to logarithmic factors. As this hypergraph has no algebraic structure, no improvement of Erdős' 60-year-old $O(n^{r-(1/t^{r-1})})$ bound was known for this setting. Futhermore, even in the special case of discs for which the semialgebraic structure can be used, our result improves the best known result by a factor of $\tilde{\Omega}(n^{\frac{2r-2}{3r-2}})$.

To obtain our results, we use the recently improved results for the graph Zarankiewicz's problem in the corresponding settings, along with a variety of combinatorial and geometric techniques, including shallow cuttings, biclique covers, transversals, and planarity.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Zarankiewicz's Problem, hypergraphs, intersection graphs, axis-parallel boxes, pseudo-discs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.33

Related Version *Full Version*: https://arxiv.org/pdf/2412.06490 [4]

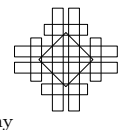
Funding *Timothy M. Chan*: Supported by NSF Grant CCF-2224271.
Chaya Keller: Partially supported by Grant 1065/20 from the Israel Science Foundation.
Shakhar Smorodinsky: Partially supported by Grant 1065/20 from the Israel Science Foundation.

1 Introduction

1.1 Background

Zarankiewicz's problem for graphs. A central research area in extremal combinatorics is *Turán-type questions*, which ask for the maximum number of edges in a graph on n vertices that does not contain a copy of a fixed graph H. This research direction was initiated in 1941 by Turán, who showed that the maximum number of edges in a K_r-free graph on n vertices is $(1-\frac{1}{r-1}+o(1))\frac{n^2}{2}$. Soon after, Erdős, Stone and Simonovits solved the problem for all non-bipartite graphs H. They showed that the maximum number is $(1-\frac{1}{\chi(H)-1}+o(1))\frac{n^2}{2}$, where $\chi(H)$ is the chromatic number of H.

© Timothy M. Chan, Chaya Keller, and Shakhar Smorodinsky;
licensed under Creative Commons License CC-BY 4.0



The bipartite case turned out to be significantly harder. In 1951, Zarankiewicz raised the following Turán-type question for complete bipartite graphs: Given $n, t \in \mathbb{N}$, what is the maximum number of edges in a *bipartite*[1] graph G on n vertices that does not contain a copy of the complete bipartite graph $K_{t,t}$? This question, known as "Zarankiewicz's problem", has become one of the central open problems in extremal graph theory (see [18]). In one of the cornerstone results of extremal graph theory, Kővári, Sós and Turán [15] proved an upper bound of $O(n^{2-\frac{1}{t}})$. This bound is sharp for $t = 2, 3$ and known matching lower bound constructions use geometric bipartite intersection graphs [3]. The question whether this bound is tight for $t \geq 4$ is widely open.

In recent years, numerous works obtained improved bounds on the number of edges in various algebraic and geometric settings (e.g., [2, 5, 9, 11, 12, 13, 14, 17, 20, 21]). Several of these works studied Zarankiewicz's problem for *intersection graphs* of two families of geometric objects. In such a graph $G(A_1, A_2)$, the vertices are two families A_1, A_2 of geometric objects, and for $a_1 \in A_1$ and $a_2 \in A_2$, (a_1, a_2) is an edge of G if $a_1 \cap a_2 \neq \emptyset$. In particular, Chan and Har-Peled [5] obtained an $O(tn(\frac{\log n}{\log \log n})^{d-1})$ bound for intersection graphs of points vs. axis-parallel boxes in $\mathbb{R}^d$, improving upon results of [2, 21], and observed that a matching lower bound construction appears in a classical paper of Chazelle [6]. They also obtained an $O(tn \log \log n)$ bound for intersection graphs of points vs. pseudo-discs in the plane, and bounds for points vs. halfspaces, balls, shapes with "low union complexity", and more. Keller and Smorodinsky [14] proved a bound of $O_t(n\frac{\log n}{\log \log n})$ for intersection graphs of two families of axis-parallel rectangles in the plane and a bound of $O_t(n)$ for intersection graphs of two families of pseudo-discs. Both bounds are sharp, up to the dependence on t.

Zarankiewicz's problem for hypergraphs. The "hypergraph analogue" of Zarankiewicz's problem asks for the maximum number of hyperedges in an r-uniform hypergraph on n vertices that does not contain $K^r_{t,t,\ldots,t}$ (i.e., the complete r-partite r-uniform hypergraph with all parts having size t) as a subhypergraph.[2] This question was raised in 1964 by Erdős [10], who obtained an upper bound of $O(n^{r-\frac{1}{t^{r-1}}})$ and an almost matching lower bound of $\Omega(n^{r-\frac{c}{t^{r-1}}})$ for general hypergraphs. Note that $\Omega(n^{r-1})$ is a trivial lower bound for this problem, as the complete r-partite graph $K^r_{n,n,\ldots,n,t-1}$ has $(t-1)n^{r-1}$ edges and is clearly $K^r_{t,t,\ldots,t}$-free.

In recent years, a number of works obtained improvements of the bound of Erdős under various algebraic assumptions on the hypergraph. Do [8] obtained a bound of the form $O_{D,d,t,r}(n^{r-\alpha})$ for semialgebraic hypergraphs in $\mathbb{R}^d$, where $\alpha = \alpha(r, d) < 1$ and D is the description complexity. (Here and in the sequel, the asymptotic notation $g = O_x(f)$ means that the dependence on the parameter x, which is assumed to be constant, is not specified). Improved bounds in the same setting were later obtained by Do [9] and by Tidor and Yu [19]. In particular, the results of [19] yield a bound of the form $O_{t,r}(n^{r-\frac{r}{3r-2}})$ for r-partite intersection hypergraphs of r families of discs in the plane. Tong [22] generalized the results of [8] to classes of hypergraphs that satisfy the distal regularity lemma. Basit, Chernikov, Starchenko, Tao, and Tran [2] obtained an improved bound for semilinear hypergraphs. In particular, their technique yields the bound $O_{d,t,r}(n^{r-1}(\log n)^{d(2^{r-1}-1)})$, for r-partite intersection hypergraphs of r families of axis-parallel boxes in $\mathbb{R}^d$ (see definition below).

---

[1] Note that asking the question for bipartite graphs G like Zarankiewicz did, leads to the same order of magnitude of the size of the extremal graph like in Turán's original question which considers general graphs G, as any graph G with e edges contains a bipartite subgraph with at least $e/2$ edges.

[2] Like in the case of graphs, asking the question for r-partite r-uniform hypergraphs leads to the same order of magnitude of the size of the extremal hypergraph, as any r-uniform hypergraph H with e hyperedges contains an r-partite sub-hypergraph with at least $e \cdot \frac{r!}{r^r}$ hyperedges.

All these improved bounds use in a crucial way the algebraic structure of the hypergraph. No improvements of the bound of Erdős in non-algebraic settings are known.

1.2 Our results

In this paper we study Zarankiewicz's problem in intersection hypergraphs of families of geometric objects. In such an r-uniform r-partite hypergraph $H(A_1, A_2, \ldots, A_r)$, the vertices are r families $A_1, A_2, \ldots, A_r$ of geometric objects, and for $a_1 \in A_1, \ldots, a_r \in A_r$, $(a_1, a_2, \ldots, a_r)$ is a hyperedge of H if $a_1 \cap \ldots \cap a_r \neq \emptyset$. The number of hyperedges in H is denoted by $\mathcal{E}(H)$.

Intersection hypergraphs of axis-parallel boxes. Our first main result is the following upper bound for Zarankiewicz's problem for r-uniform intersection hypergraphs of axis-parallel boxes in $\mathbb{R}^d$. For convenience, we assume that all boxes involved are in a general position, meaning that no two ℓ-dimensional faces in the same directions lie on the same ℓ-flat.

▶ **Theorem 1.** *Let $r, t, d \geq 2$, and let H be the intersection hypergraph of families $A_1, \ldots, A_r$ of n axis-parallel boxes in $\mathbb{R}^d$, such that $A_1 \cup \ldots \cup A_r$ is in a general position. If H is $K^r_{t,\ldots,t}$-free, then*

$$|\mathcal{E}(H)| = O_{r,d}\left(t^2 n^{r-1} \left(\frac{\log n}{\log\log n}\right)^{d-1}\right).$$

This result, which improves the aforementioned result of Basit et al. [2] by a factor of about $(\log n)^{d(2^{r-1}-2)}$, is sharp, up to a factor of $O_{r,d}(t)$. Indeed, as was mentioned above, Chan and Har-Peled [5, Appendix B] showed that for $r = 2$, a lower bound of $\Omega_d(tn(\frac{\log n}{\log\log n})^{d-1})$ for the intersection graph of n points and n axis-parallel boxes in $\mathbb{R}^d$ follows from an old result of Chazelle [6]. This bound clearly holds also for the intersection graph of two families of n axis-parallel boxes in $\mathbb{R}^d$, and thus, it trivially yields an $\Omega_d(tn^{r-1}(\frac{\log n}{\log\log n})^{d-1})$ lower bound for r families, by adding $r-2$ families of n large boxes that contain all the boxes of the two first families.

Our proof exploits geometric properties of axis-parallel boxes and builds upon the upper bounds for Zarankiewicz's problem for intersection graphs of two families of axis-parallel boxes, obtained in [5] for intersections of points and boxes in $\mathbb{R}^d$, and in [14] for intersections of two families of rectangles in $\mathbb{R}^2$. In fact, a direct application of the result of [14] yields a weaker bound with t^6 instead of t^2. We improve the bound to t^2, and show that this is the best bound that can be obtained with the general strategy of [14]. However, we believe that the right dependence on t is linear.

It is somewhat surprising that the power of $\log n$ in the bound does not depend on r. Showing this is the most complex part of our proof, which uses biclique covers [7] and a non-standard inductive argument.

Intersection hypergraphs of pseudo-discs. Our second main result concerns intersection hypergraphs of pseudo-discs. A family of simple Jordan regions in the plane is called *a family of pseudo-discs* if the boundaries of every two regions intersect at most twice. For technical reasons, it is convenient to assume that the pseudo-discs are y-monotone, namely, that the intersection of any vertical line with a region from the family is either empty or an interval. In addition, we assume that the pseudo-discs are in general position, namely, no three boundaries intersect in a point.

▶ **Theorem 2.** *Let $r, t \geq 2$, and let H be the intersection hypergraph of families $A_1, \ldots, A_r$ of n y-monotone pseudo-discs in general position in the plane. If H is $K^r_{t,\ldots,t}$-free, then $|\mathcal{E}(H)| = O_r(t^6 n^{r-1} (\log n)^{r-2})$.*

This result is obviously sharp up to a factor of $O(t^5 (\log n)^{r-2})$. Our proof builds upon the upper bound for Zarankiewicz's problem for intersection graphs of two families of pseudo-discs obtained in [14], and uses geometric properties of pseudo-discs and shallow cuttings [16]. As general intersection hypergraphs of pseudo-discs have no algebraic structure, no improvement of the 60-year-old $O(n^{r-\frac{1}{t^{r-1}}})$ bound of Erdős [10] was known in this setting. Furthermore, in the special case of discs whose semialgebraic structure allows applying the previous algebraic works, our result improves over the best known previous result of Tidor and Yu [19] by a factor of $\tilde{\Omega}(n^{\frac{2r-2}{3r-2}})$.

Organization of the paper. In Section 2 we present the proof of Theorem 1. In Section 3 we present the proof of Theorem 2. In the appendices we prove several lemmas that are used in the proofs of the theorems.

2 Intersection Hypergraphs of Axis-Parallel Boxes

In this section we prove our new bound for Zarankiewicz's problem for intersection hypergraphs of axis-parallel boxes in $\mathbb{R}^d$ – namely, Theorem 1. First, in Section 2.1 we present a bound for the *lopsided* Zarankiewicz problem for intersection *graphs* of two families of boxes. Then, in Section 2.2 we use the lopsided version to handle r-partite intersection hypergraphs of boxes.

2.1 Lopsided version of Zarankiewicz's problem for two families of boxes

In this subsection we bound the number of edges in a $K_{t,gt}$-free intersection graph of two families of axis-parallel boxes in $\mathbb{R}^d$. We prove the following:

▶ **Theorem 3.** *Let A, B be two multisets of boxes in $\mathbb{R}^d$ where $|A| = n, |B| = m$ and $m = \text{poly}(n)$. If their intersection graph $G_{A,B}$ is $K_{t,gt}$-free (t on the side of A), then*

$$|E(G_{A,B})| \leq O_d(gtnf_{d,t}(n) + mf_{d,t}(m)),$$

where $f_{d,t}(n) = t(\frac{\log n}{\log \log n})^{d-1}$.

Theorem 3 is a generalization and an improvement of the bound on intersections between two families of axis-parallel rectangles in $\mathbb{R}^2$ proved in [14, Theorem 1.7]. Compared to the result of [14], Theorem 3 applies in $\mathbb{R}^d$ for all $d \geq 2$, is not restricted to the symmetric case of $K_{t,t}$, and has a better dependence on t.

▶ Remark 4. We state the theorem in terms of the auxiliary function $f_{d,t}(n)$, as in the sequel it will be used in a blackbox manner. We conjecture that $f_{d,t}(n)$ can be replaced with $(\frac{\log n}{\log \log n})^{d-1}$, which would lead to a bound of $O_{d,r}(tn^{r-1}(\frac{\log n}{\log \log n})^{d-1})$ for Zarankiewicz's problem for r-partite intersection hypergraphs of axis-parallel boxes in $\mathbb{R}^d$, which is sharp in terms of both n and t.

The proof of Theorem 3 is divided into two main cases, based on the following observation. Any intersection between two axis-parallel boxes belongs to one of two types:

1. *Vertex containment intersections*, in which a vertex of one box is contained in the other box;

2. *Facet intersections*, in which a facet of one box intersects a facet of the other box.

We bound each type of intersections with a different divide-and-conquer argument, as presented below.

2.1.1 Bounding vertex containment intersections

Due to the lack of symmetry between the two sides, we have to consider separately two types of intersections:

- Intersections in which a vertex of a box in A is contained in a box in B;
- Intersections in which a vertex of a box in B is contained in a box in A.

For each type, the problem reduces to bounding the number of edges in $K_{t,gt}$-free intersection graphs of points and axis-parallel boxes. Indeed, for the first case we can bound the number of intersections by $|E(G_{A',B})|$, where A' is the set of vertices of the boxes in A. The second case can be handled similarly. We prove the following proposition, which is a lopsided version of the bound of [5, Theorem 4.5] on the number of edges in $K_{t,t}$-free bipartite intersection graphs of points and axis-parallel boxes in $\mathbb{R}^d$ (though, with a weaker dependence on t).

▶ **Proposition 5.** *Let A be a multiset of points in $\mathbb{R}^d$ and let B be a multiset of axis-parallel boxes, where $|A| = n$ and $|B| = m$. Then for any $\epsilon > 0$,*

1. *If the bipartite intersection graph $G_{A,B}$ is $K_{t,gt}$-free (t on the side of A) then it has $O_\epsilon\left(gt^2 n(\frac{\log n}{\log\log n})^{d-1} + tm(\frac{\log n}{\log\log n})^{d-2+\epsilon}\right)$ edges.*
2. *If the bipartite intersection graph of $G_{A,B}$ is $K_{gt,t}$-free (t on the side of B) then it has $O_\epsilon\left(tn(\frac{\log n}{\log\log n})^{d-1} + gt^2 m(\frac{\log n}{\log\log n})^{d-2+\epsilon}\right)$ edges.*

As we show below, this provides a sufficiently strong bound on the vertex containment intersections, for the needs of Theorem 3.

Due to space constraints, the proof of Proposition 5 is presented in the full version of the paper [4].

2.1.2 Bounding facet intersections

We claim that for a large n, the number of facet intersections is significantly smaller than the number of vertex containment intersections.

▶ **Proposition 6.** *Let A, B be multisets of axis-parallel boxes in $\mathbb{R}^d$, where $|A| = n$ and $|B| = m$. If the bipartite intersection graph $G_{A,B}$ is $K_{t,gt}$-free (t on the side of A) then the number of facet intersections between a box in A and a box in B is*

$$O_d\left(gt^2 n(\log n)^{d-2} + tm(\log n)^{d-2}\right).$$

We use the following lemma.

▶ **Proposition 7.** *Let A be a multiset of horizontal segments in $\mathbb{R}^2$ and let B be a multiset of vertical segments in $\mathbb{R}^2$, where $|A| = n$ and $|B| = m$. Let $g > 0$. If the bipartite intersection graph $G_{A,B}$ is $K_{t,gt}$-free (where the t is on the side of A) then it has $O(gt^2 n + tm)$ edges.*

Due to space constraints, the proof of Proposition 7 is presented in the full version of the paper [4].

Proof of Proposition 6. The proof is by induction on d. The base case $d = 2$ amounts to bounding the number of edges in a $K_{2t,2gt}$-free bipartite intersection graph of a multiset A of horizontal segments in $\mathbb{R}^2$ and a multiset B of vertical segments in $\mathbb{R}^2$. Indeed, each "facet

intersection" in the plane is either an intersection between a horizontal edge of a rectangle in A with a vertical edge of a rectangle in B, or vice versa. We can bound the number of facet intersections of the first type by $|E(G_{A',B'})|$, where A' is the multiset of horizontal edges of rectangles in A and B' is the multiset of vertical edges of rectangles in B. Note that this intersection graph is $K_{2t,2gt}$-free, as otherwise, $G_{A,B}$ would contain $K_{t,gt}$. The second type can be handled similarly. Hence, Proposition 7 yields a bound of $O(gt^2n + tm)$ in this setting, which proves the induction basis.

In the induction step, we assume that for $(d-1)$-dimensional boxes, the number of facet intersections is bounded by

$$O_d(gt^2n(\log n)^{d-3} + tm(\log n)^{d-3}).$$

We bound the number of facet intersections between $x \in A$ and $y \in B$, such that the intersecting facets are othogonal to the $(d-1)$'th and the d'th axis, respectively. A bound on the total number of facet intersections clearly follows by multiplying with d^2. In the rest of the proof, we call such special intersections "good facet intersections", or just "intersections" for brevity. We make sure that in the dimension reductions in the induction process presented below, every time the first coordinate is the one that is removed, and thus, the good facet intersections are not affected.

We use the following auxiliary notion. Let $I_d(n, m)$ be the maximum number of good facet intersections between multisets A, B of axis-parallel boxes inside a "vertical strip" $\mathcal{U} = \{x \in \mathbb{R}^d : u_L < x_1 < u_R\}$ of $\mathbb{R}^d$, where:

1. Each box in A, B has either half of its vertices or all of its vertices in $\mathcal{U}$;
2. The total number of vertices of boxes in A (resp., boxes in B) inside $\mathcal{U}$ is $n \cdot 2^{d-1}$ (resp., $m \cdot 2^{d-1}$);
3. The bipartite intersection graph of A, B is $K_{t,gt}$-free.

We shall prove that

$$I_d(n, m) \leq O_d(gt^2n(\log n)^{d-2} + tm(\log n)^{d-2}).$$

This clearly implies the assertion of the proposition, as by considering a vertical strip $\mathcal{U}$ that fully contains all boxes in A and B (where $|A| = n$ and $|B| = m$), we get that the number of good facet intersections between A and B is at most $I_d(2n, 2m)$. (Note that we neglect factors of $O_d(1)$).

Let A, B be multisets of boxes that satisfy assumptions (1)–(3) with respect to a strip $\mathcal{U} \subset \mathbb{R}^d$. We divide $\mathcal{U}$ into two vertical sub-strips $\sigma_1 = \{x \in \mathbb{R}^d : u_L < x_1 < u'\}$ and $\sigma_2 = \{x \in \mathbb{R}^d : u' < x_1 < u_R\}$, such that each sub-strip contains $n \cdot 2^{d-2}$ vertices of boxes in A. For $i = 1, 2$, we denote by A_i (resp., B_i) the boxes in A (resp., B) that have at least one vertex in σ_i, and by A'_i (resp., B'_i) the boxes in A (resp., B) that intersect σ_i but do not have vertices in it. Note that $A'_1 \subset A_2, A'_2 \subset A_1$, and similarly for B. We also denote the number of vertices of boxes in B contained in σ_i by $2^{d-1} \cdot m_i$.

The number of good facet intersections in $\mathcal{U}$ between boxes in A and boxes in B is at most

$$I(A_1, B_1) + I(A_2, B_2) + I(A_1, B'_1) + I(A'_2, B_2) + I(A_2, B'_2) + I(A'_1, B_1), \tag{1}$$

where $I(X, Y)$ denotes the number of good facet intersections between an element of X and an element of Y.

Indeed, $I(A_1, B_1)$ (resp., $I(A_2, B_2)$) counts intersections between pairs of boxes that have at least one vertex in σ_1 (resp., σ_2). $I(A_1, B_1') + I(A_2', B_2)$ upper bounds the number of intersections between a box in A that has a vertex in σ_1 and a box in B that has a vertex in σ_2. $I(A_2, B_2') + I(A_1', B_1)$ upper bounds the number of intersections between a box in A that has a vertex in σ_2 and a box in B that has a vertex in σ_1.

By the definitions, we have $I(A_1, B_1) \leq I_d(\frac{n}{2}, m_1)$ (where σ_1 is taken as the vertical strip instead of $\mathcal{U}$). Similarly, $I(A_2, B_2) \leq I_d(\frac{n}{2}, m_2)$.

To handle the other types of intersections, we observe that a box in A_1 intersects a box in B_1' if and only if their projections on the hyperplane $\mathcal{H} = \{x \in \mathbb{R}^d : x_1 = u'\}$ (which are $(d-1)$-dimenstional boxes) intersect. Let $\bar{A}_1$ and $\bar{B}_1'$ denote the corresponding families of projections. We have $|\bar{A}_1| = |A_1| = \frac{n}{2}$ and $|\bar{B}_1'| = |B_1'| \leq m_2$. The multisets $\bar{A}_1$ and $\bar{B}_1'$ are multisets of axis-parallel boxes in $\mathbb{R}^{d-1}$ whose bipartite intersection graph is $K_{t,gt}$-free. Hence, by the induction hypothesis we have

$$\begin{aligned} I(A_1, B_1') = I(\bar{A}_1, \bar{B}_1') &\leq O_d\left(gt^2 \tfrac{n}{2} \left(\log \tfrac{n}{2}\right)^{d-3} + tm_2 \left(\log \tfrac{n}{2}\right)^{d-3}\right) \\ &\leq O_d\left(gt^2 \tfrac{n}{2} (\log n)^{d-3} + tm_2 (\log n)^{d-3}\right), \end{aligned}$$

Applying the same argument to $I(A_2', B_2), I(A_1', B_1)$ and $I(A_2, B_2')$, we get

$$\begin{aligned} &I(A_1, B_1') + I(A_2', B_2) + I(A_2, B_2') + I(A_1', B_1) \\ &\quad \leq O_d(4gt^2 \tfrac{n}{2} (\log n)^{d-3} + 2tm_2 (\log n)^{d-3} + 2tm_1 (\log n)^{d-3}) \\ &\quad = O_d(2gt^2 n (\log n)^{d-3} + 2tm (\log n)^{d-3}). \end{aligned}$$

Combining this with the bounds on $I(A_1, B_1)$ and $I(A_2, B_2)$ and substituting to (1), we obtain the recursive formula

$$I_d(n, m) \leq \max_{m_1 + m_2 = m} \left(I_d(\tfrac{n}{2}, m_1) + I_d(\tfrac{n}{2}, m_2) + O_d(2gt^2 n (\log n)^{d-3} + 2tm (\log n)^{d-3})\right),$$

which solves to

$$I_d(n, m) \leq \log n \cdot O_d(gt^2 n (\log n)^{d-3} + tm (\log n)^{d-3}) \leq O_d(gt^2 n (\log n)^{d-2} + tm (\log n)^{d-2}).$$

This completes the proof. ◀

2.1.3 Completing the proof of Theorem 3

Now we are ready to wrap up the proof of Theorem 3.

Proof of Theorem 3. Let A, B be multisets of axis-parallel boxes in $\mathbb{R}^d$ that satisfy the assumptions of the theorem. As written above, the intersections between A and B can be divided into *vertex containment intersections* and *facet intersections*.

To bound the number of vertex containment intersections, we apply Proposition 5, with $\epsilon = \frac{1}{2}$. Specifically, we use Proposition 5(1) to bound the number of intersections in which a vertex of $a \in A$ is contained in a box $b \in B$. Furthermore, we use Proposition 5(2) *with the roles of n and m reversed* to bound the number of intersections in which a vertex of $b \in B$ is contained in a box $a \in A$. We get that the number of vertex containment intersections is at most

$$O_d\left(gt^2 n (\tfrac{\log n}{\log \log n})^{d-1} + tm (\tfrac{\log n}{\log \log n})^{d-1.5} + tm (\tfrac{\log m}{\log \log m})^{d-1} + gt^2 n (\tfrac{\log m}{\log \log m})^{d-1.5}\right).$$

By Proposition 6, the number of facet intersections is at most

$$O_d(gt^2n(\log n)^{d-2} + tm(\log n)^{d-2}).$$

Note that for a large n, we have $(\log n)^{d-2} = o((\frac{\log n}{\log\log n})^{d-1.5})$, and hence, the number of facet intersections is dominated by the number of vertex containment intersections.

Finally, since $m = \text{poly(n)}$, the terms $tm(\frac{\log n}{\log\log n})^{d-1.5}$ and $gt^2n(\frac{\log m}{\log\log m})^{d-1.5}$ are dominated by the two other terms. Therefore, we have

$$|E(G_{A,B})| \leq O_d\left(gt^2n(\tfrac{\log n}{\log\log n})^{d-1} + tm(\tfrac{\log m}{\log\log m})^{d-1}\right) = O_d(gtnf_{d,t}(n) + mf_{d,t}(m)),$$

as asserted. This completes the proof. ◀

2.2 Zarankiewicz problem for r-partite intersection hypergraph of boxes

Using Theorem 3, we can relatively easily obtain Proposition 8 below, which is a weaker version of Theorem 1. Since the proof of the weaker result may serve as a good introduction to the proof of Theorem 1, we present it in Section 2.2.1, and then we pass to the proof of Theorem 1 in Section 2.2.2.

2.2.1 Proof of a weaker variant of Theorem 1

We prove the following.

▶ **Proposition 8.** *Let $A_1, A_2, \ldots, A_r$ be families of n axis-parallel boxes in $\mathbb{R}^d$, and let H be their r-partite intersection hypergraph. If H is $K^r_{t,t,\ldots,t}$-free, then $|\mathcal{E}(H)| = O_r\left(t\left(nf_{d,t}(n)\right)^{r-1}\right)$.*

The proof of Proposition 8 is by induction on r. For the sake of simplicity, we present here the proof for $r = 3$, and complete the argument for a general r in the full version of the paper [4].

Proof of Proposition 8 for $r = 3$. Let $A_1 = A, A_2 = B, A_3 = C$. Let AB be the following multiset of axis-parallel boxes: $AB = \{a \cap b : a \in A, b \in B, a \cap b \neq \emptyset\}$. (Note that the intersection of two axis-parallel boxes is indeed an axis-parallel box). Let G be the bipartite intersection graph of the families C and AB. It is clear that $|E(G)| = |\mathcal{E}(H)|$, since there is a clear one-to-one correspondence between edges of G and hyperedges of H.

We claim that for a sufficiently large constant M, the graph G is $K_{t,Mtnf_{d,t}(n)}$-free. Indeed, assume to the contrary that G contains a copy of $K_{t,Mtnf_{d,t}(n)}$, for a "large" M. This means that there exist t boxes $c_1, c_2, \ldots, c_t \in C$ which all have a non-empty intersection with certain $Mtnf_{d,t}(n)$ axis-parallel boxes of the form $a_i \cap b_i$, with $a_i \in A$, $b_i \in B$. Denote by A' the set of all $a_i \in A$ that participate in such intersections, and by B' the set of all $b_i \in B$ that participate in such intersections. Let G' be the bipartite intersection graph of A', B'. We have $|E(G')| \geq Mtnf_{d,t}(n)$, and hence, by Theorem 3 (applied with $m = n$ and $g = 1$), it contains a $K_{t,t}$ (using the assumption that M is sufficiently large). This means that there exist $a_1, a_2, \ldots, a_t \in A$ and $b_1, b_2, \ldots, b_t \in B$ such that for all $1 \leq i, j \leq t$, we have $a_i \cap b_j \neq \emptyset$, and both a_i and b_j have a non-empty intersection with each of $c_1, \ldots, c_t$ (since they participate in pairs whose intersection with each of $c_1, \ldots, c_t$ is non-empty). As any three pairwise intersecting axis-parallel boxes have a non-empty intersection, this implies that $a_i \cap b_j \cap c_k \neq \emptyset$ for all $1 \leq i, j, k \leq t$, and consequently, H contains a $K_{t,t,t}$, a contradiction.

We have thus concluded that G is $K_{t,Mtnf_{d,t}(n)}$ free, for a sufficiently large constant M. By Theorem 3, applied with n, $m = n^2$, and $g = Mnf_{d,t}(n)$, this implies that

$$|\mathcal{E}(H)| = |E(G)| \leq O\left(t\left(nf_{d,t}(n)\right)^2\right).$$

This completes the proof. ◀

Note that in this proof, we used the fact that axis-parallel boxes in $\mathbb{R}^d$ have *Helly number 2*, namely, that any pairwise intersecting family of axis-parallel boxes has a non-empty intersection. Without this property, the existence of $K_{t,t}$ in G' does not necessarily imply the existence of $K_{t,t,t}$ in H.

The proof for a general r is an inductive process, in which at the ℓ'th step, one considers an auxiliary bipartite intersection graph of boxes, whose vertex sets are the multiset $A_1A_2 \dots A_{\ell+1} = \{(a_1 \cap \dots \cap a_{\ell+1}) : \forall i, a_i \in A_i, \cap_{i=1}^{\ell+1} a_i \neq \emptyset\}$, where the multiplicity of each element is the number of $(a_1, \dots, a_{\ell+1})$ tuples that lead to it, and $A_{\ell+2}$. At each step, Theorem 3 is applied once again, and the fact that axis-parallel boxes have Helly number 2 is deployed once again. As this part of the proof is a bit more cumbersome and is not needed for the proof of Theorem 1 presented below, we provide it in the full version of the paper [4].

2.2.2 Proof of Theorem 1

In Proposition 8, the polylogarithmic factor in the upper bound increases with r, since at each step of the inductive process we apply Theorem 3 and "pay" another $f_{d,t}(n)$-factor. In this subsection, we show how to avoid this dependency on r. Let us restate Theorem 1.

▶ **Theorem 9** (Theorem 1 restated). *Let $r, d, t \geq 2$ and let H be the intersection hypergraph of r families $A_1, \dots, A_r$ of n axis-parallel boxes in general position in $\mathbb{R}^d$. If H is $K^r_{t,\dots,t}$-free, then $|\mathcal{E}(H)| = O_{r,d}(tn^{r-1}f_{d,t}(n))$, where $f_{d,t}(n) = t(\frac{\log n}{\log\log n})^{d-1}$.*

In the proof, we use the following modification of the *biclique cover theorem* (see [7]) for axis-parallel boxes, that may be of independent interest. We note that the relation between Zarankiewicz's problem and biclique covers was already observed in [5, 9].

▶ **Definition 10.** *A biclique cover of a graph $G = (V, E)$ is a collection of pairs of vertex subsets $\{(A_1, B_1), \dots, (A_l, B_l)\}$ such that $E = \bigcup_{i=1}^{l}(A_i \times B_i)$.*

▶ **Lemma 11.** *Let $n, m, b \in \mathbb{N}$ be such that $b \leq \min(n, m)$, and let A, B be families of axis-parallel boxes in $\mathbb{R}^d$, with $|A| = n$ and $|B| = m$. Then the bipartite intersection graph of A, B can be partitioned into a union of $O(b\log^{d-1} b)$ bicliques (with no restriction on their size) and $O(b\log^{d-1} b)$ "partial bicliques" (namely, subgraphs of bicliques), each of size at most $\frac{n}{b} \cdot \frac{m}{b}$.*

Due to space constraints, the proof of the lemma is presented in the full version of the paper [4].

The motivation behind our proof of Theorem 1 is as follows: Assume that we add an additional assumption that the $(r-1)$-partite intersection graph of the families $A_1, \dots, A_{r-1}$ is complete. Namely, that for all $1 \leq i < j \leq r-1$, any box in A_i intersects any box in A_j. Furthermore, assume that by applying (only once!) Theorem 3, we show that the bipartite intersection graph of the multiset $A_1A_2 \dots A_{r-1} = \{a_1 \cap \dots \cap a_{r-1} : \forall i, a_i \in A_i, \cap_{i=1}^{r-1} a_i \neq \emptyset\}$ and the family A_r contains $K_{gt,t}$, for $g = n^{r-2}$. This means that some t boxes from A_r intersect $gt = tn^{r-2}$ boxes of the type $\{a_1 \cap \dots \cap a_{r-1} : a_i \in A_i\}$. These gt boxes involve at

least t boxes from each A_i ($1 \leq i \leq r-1$). Thus, by our additional assumption and by the fact that axis-parallel boxes admit Helly number 2, the existence of $K_{gt,t}$ in the bipartite intersection graph, implies the existence of $K^r_{t,\ldots,t}$ in the original r-partite intersection hypergraph. Therefore, the suggested additional assumption enables avoiding repeated applications of Theorem 3, and hence obtaining a bound with a logarithmic factor that does not increase with r.

In order to show that we can indeed make the additional assumption described above without affecting the final bound, we define a constraints graph G, not to be confused with the auxiliary graph G from Section 2.2.1.

▶ **Definition 12.** *For an r-tuple of set families $A_1, \ldots, A_r$, the* constraints graph $G = G_{A_1,\ldots,A_r}$ *is defined as follows. The vertex set of G is $V(G) = \{1, 2, \ldots, r\}$, and $(i, j) \in G$ if $\exists a_i \in A_i, a_j \in A_j$ such that $a_i \cap a_j = \emptyset$.*

This graph G represents the "distance" of the r-tuple of families $A_1, \ldots, A_r$ from the desired setting in which there exists i such that any box from A_j intersects any box from A_k, for all $j, k \neq i$. Our goal in the proof below is to remove all the edges from G, except for those emanating from a single vertex i. This will show that one can indeed make the additional assumption described above without affecting the final bound.

Proof of Theorem 9. Let $T_G(n_1, \ldots, n_r)$ be the maximum number of hyperedges in a $K^r_{t,\ldots,t}$-free r-partite intersection hypergraph H of r families $A_1, \ldots, A_r$ of axis-parallel boxes in $\mathbb{R}^d$, where $|A_i| = n_i$, and the constraints graph of r-tuple $A_1, \ldots, A_r$ is G. Denote $T_G(n) = T_G(n, \ldots, n)$. In order to prove the theorem, it is clearly sufficient to prove that for any constraint graph G, we have $T_G(n) \leq O(tn^{r-1} f_{d,t}(n))$. We prove this claim by induction on $|E(G)|$.

If $E(G) = \emptyset$, then for all $i \neq j$, any $a_i \in A_i$ intersects any $a_j \in A_j$. Since H is $K^r_{t,\ldots,t}$-free, this implies $n < t$ and we are done.

In the induction step, we assume that we have already proved the claim for any constraints graph with a smaller number of edges, and we now consider a constraints graph G. We consider three cases:

- Case A: G contains two non-adjacent edges;
- Case B: G is a triangle;
- Case C: G is a star.

Clearly, any graph G belongs to one of the three cases.

Case A: G contains two non-adjacent edges. Say these two non-adjacent edges are $(1, 2)$ and $(3, 4)$. Our goal now is to "remove" the edge $(1, 2)$ (and later, also the edge $(3, 4)$) from G, by partitioning the intersection graph of A_1 and A_2 into bicliques. To this end, we use Lemma 11 with a large constant b to partition the entire hypergraph into smaller hypergraphs, induced by a partition of the bipartite intersection of A_1 and A_2 to "full" bicliques and "partial" bicliques. We obtain the recursion:

$$T_G(n) \leq O(b \log^{d-1} b) T_G(\tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) + O(b \log^{d-1} b) O(tn^{r-1} f_{d,t}(n)), \tag{2}$$

where the right term comes from applying the induction hypothesis on the hypergraphs induced by full bicliques, whose constraints graph is $G \setminus \{(1, 2)\}$, and the left term comes from the partial bicliques.

Now, to bound $T_G(\frac{n}{b}, \frac{n}{b}, n, \ldots, n)$, we do the same with the sets A_3, A_4 and obtain

$$T_G(\tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) \leq O(b \log^{d-1} b) T_G(\tfrac{n}{b}, \tfrac{n}{b}, \tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) + O(b \log^{d-1} b) O(tn^{r-1} f_{d,t}(n)). \tag{3}$$

Combining (2) and (3) together, we get

$$T_G(n) \leq O(b^2 \log^{2d-2} b) \left(T_G(\tfrac{n}{b}, \tfrac{n}{b}, \tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) + O(tn^{r-1} f_{d,t}(n))\right). \tag{4}$$

To bound the left term in the right hand side, we partition the hypergraph into b^{r-4} hypergraphs which r sides of size $\frac{n}{b}$ by partitioning each of the sets $A_5, \ldots, A_r$ arbitrarily into b parts, each of size $\frac{n}{b}$. We obtain

$$T_G(n) \leq O(b^2 \log^{2d-2} b) \left(b^{r-4} T_G(\tfrac{n}{b}) + O(tn^{r-1} f_{d,t}(n))\right). \tag{5}$$

The recursion (5) solves to $T_G(n) = O(tn^{r-1} f_{d,t}(n))$, and we are done.

Case B: G is a triangle. Say $E(G) = \{(1,2), (2,3), (1,3)\}$. In this case, we first split A_1 and A_2 by Lemma 11, then we split A_2 and A_3 and then we split A_1 and A_3.

Using the induction hypothesis, by splitting A_1, A_2 we obtain

$$T_G(n) \leq O(b \log^{d-1} b) \left(T_G(\tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) + O(tn^{r-1} f_{d,t}(n))\right). \tag{6}$$

By splitting A_2, A_3, we get

$$T_G(\tfrac{n}{b}, \tfrac{n}{b}, n, \ldots, n) \leq O(b \log^{d-1} b) \left(T_G(\tfrac{n}{b}, \tfrac{n}{b^2}, \tfrac{n}{b}, n, \ldots, n) + O(tn^{r-1} f_{d,t}(n))\right), \tag{7}$$

and by splitting A_1, A_3 we have

$$T_G(\tfrac{n}{b}, \tfrac{n}{b^2}, \tfrac{n}{b}, n, \ldots, n) \leq O(b \log^{d-1} b) \left(T_G(\frac{n}{b^2}, \frac{n}{b^2}, \frac{n}{b^2}, n, \ldots, n) + O(tn^{r-1} f_{d,t}(n))\right). \tag{8}$$

Combining (6)-(8), we obtain

$$T_G(n) \leq O(b^3 \log^{3d-3} b) \left(T_G(\tfrac{n}{b^2}, \tfrac{n}{b^2}, \tfrac{n}{b^2}, n, \ldots, n) + O(tn^{r-1} f_{d,t}(n))\right), \tag{9}$$

and by partitioning each of $A_4, \ldots, A_r$ arbitrarily into b^2 equal parts, we get

$$T_G(n) \leq O(b^{2r-3} \log^{3d-3} b) T_G(\tfrac{n}{b^2}) + O(b^3 \log^{3d-3} b) O(tn^{r-1} f_{d,t}(n)). \tag{10}$$

As above, this recursion solves to $T_G(n) = O(tn^{r-1} f_{d,t}(n))$.

Case C: G is a star. Say G is a star centered at A_r. In this case we don't apply the induction hypothesis. Instead, we apply Theorem 3 (only once, in contrast to the argument in Section 2.2.1).

Assume on the contrary that $T_G(n) > Ctn^{r-1} f_{d,t}(n)$ for some large constant C. Consider the intersection graph between the multiset $A_1 A_2 \ldots A_{r-1} = \{a_1 \cap \ldots \cap a_{r-1} : \forall i, a_i \in A_i, \cap_{i=1}^{r-1} a_i \neq \emptyset\}$ and the family A_r. Since the number of edges in this graph is at least $\Omega(tn^{r-1} f_{d,t}(n))$, by Theorem 3 (with the parameters $n, m = n^{r-1}$ and $g = n^{r-2}$), this intersection graph contains $K_{gt,t}$.

This means that some t specific boxes from A_r intersect $gt = n^{r-2}t$ boxes of the type $a_1 \cap \ldots \cap a_{r-1}$ ($a_i \in A_i$). These gt boxes involve at least t boxes from each A_i ($1 \leq i \leq r-1$). Hence, there exist $A'_1 \subset A_1, \ldots, A'_r \subset A_r$, each of size t, such that any $a_1, \ldots, a_r$, where $a_1 \in A'_1, \ldots, a_r \in A'_r$, are pairwise intersecting. (Here we use the setting of case C in which for all $1 \leq i < j \leq r-1$, any two boxes $a_i \in A_i$ and $a_j \in A_j$ intersect.) Since axis-parallel boxes have Helly number 2, this implies that for all $a_1 \in A'_1, \ldots, a_r \in A'_r$ we have $a_1 \cap \ldots \cap a_r \neq \emptyset$, and hence, the restriction of H to $A'_1 \cup \ldots \cup A'_r$ is a copy of $K^r_{t,t,\ldots,t}$, a contradiction. This completes the proof of Theorem 9, and thus of Theorem 1. ◀

3 Intersection Hypergraphs of Pseudo-discs

In this section we prove Theorem 2. Let us recall the statement of the theorem.

▶ **Theorem 2.** Let $r, t \geq 2$ and let H be the r-partite intersection hypergraph of families $A_1, \ldots, A_r$ of n y-monotone pseudo-discs in general position in the plane. If H is $K^r_{t,\ldots,t}$-free, then $|\mathcal{E}(H)| = O(t^6 n^{r-1} (\log n)^{r-2})$.

A tool crucially used in the proof is a lopsided version of the following result from [5].

▶ **Theorem 13** ([5, Corollary 5.1])**.** *Let P be a set of n points in the plane, and let $\mathcal{F}$ be a family of m y-monotone pseudo-discs in general position in the plane. If the bipartite intersection graph $G(P, \mathcal{F})$ is $K_{t,t}$-free ($t \geq 2$), then*

$$|E(G(P, \mathcal{F}))| = O(tn + tm \log\log m + \log t).$$

The lopsided variant of Theorem 13 reads as follows.

▶ **Theorem 14** (Lopsided variant of Theorem 13)**.** *Let P be a multiset of n points in the plane, and let $\mathcal{F}$ be a multiset of m y-monotone pseudo-discs in general position in the plane. If the bipartite intersection graph $G(P, \mathcal{F})$ is $K_{gt,t}$-free ($t, g \geq 2$, gt on the side of the points), then*

$$|E(G(P, \mathcal{F}))| = O(tn + gtm \log m).$$

Due to space constraints, the proof of Theorem 14, which uses the geometric technique of *shallow cuttings* [16], is presented in the full version of the paper [4]. Note that for $g = 1$, the bound we obtain is weaker than the bound of Theorem 13. The reason for this is explained in the full version of the paper [4].

The proof of Theorem 2 uses induction on r, where the induction basis ($r = 2$) is the following theorem from [14]:

▶ **Theorem 15** ([14, Theorem 1.6])**.** *Let $t \geq 2$ and let G be the bipartite intersection graph of two families of pseudo-discs, each of size n. If G is $K_{t,t}$-free then $|E(G)| = O(t^6 n)$.*

Proof of Theorem 2. The proof is by induction on r, the base case being Theorem 15 above. In the induction step, we use the following observation: If r simple Jordan regions in $\mathbb{R}^2$ intersect, then either

- There is a pair of regions a_1, a_2 such that an intersection point of their boundaries is contained in all other $r - 2$ regions, or
- There is a region a_1 which is fully contained in all other $r - 1$ regions.

(To see why this holds, take a point in the intersection and look at all points on the boundaries of the regions which can be reached from it without crossing a boundary of one of the regions. If all these points belong to the same region, the second case holds. If not, by the assumptions one of these points is an intersection point of two regions and the first case holds).

We call an intersection of the first type a *type A* - intersection, and an intersection of the second type a *type B* - intersection.

For each intersecting pair $\{a_1, a_2\}$ of pseudo-discs we define a special point $p(a_1, a_2)$ as follows: If the intersection of a_1 and a_2 is of type A, then $p(a_1, a_2)$ is the left intersection point of the boundaries of a_1 and a_2. If this intersection is of type B, where $a_1 \subset a_2$, then $p(a_1, a_2)$ is the leftmost point of a_1.

Without loss of generality, both for Type A and for Type B, we count intersections of the type $a_1 \cap \ldots \cap a_r \neq \emptyset$ ($a_i \in A_i$), where $p(a_1, a_2)$ is contained in $a_3 \cap \ldots \cap a_r$. This affects the final bound by a multiplicative factor of $O_r(1)$.

Let P be the multiset $P = \{p(a_1, \ldots, a_{r-1}) : a_i \in A_i\}$, where $p(a_1, \ldots, a_{r-1})$ is $p(a_1, a_2)$ with multiplicity which is determined by the number of other tuples of a_i's whose intersection contains it. Clearly, $|P| \leq n^{r-1}$. Let G be the bipartite intersection graph of the multiset of points P and the family of pseudo-discs A_r. If $|\mathcal{E}(H)| > C_r t^6 n^{r-1} (\log n)^{r-2}$ (for a sufficiently large constant C_r), then by Theorem 14, G contains $K_{gt,t}$ for $g = C_{r-1} t^5 n^{r-2} (\log n)^{r-3}$ (where C_{r-1} will be specified below, and C_r is taken to be sufficiently large for the statement to hold by Theorem 14). This $K_{gt,t}$ comes from a set S of $gt = C_{r-1} t^6 n^{r-2} (\log n)^{r-3}$ distinct tuples of the form $(a_1, \ldots, a_{r-1})$ where $a_i \in A_i$, and a set T of t pseudo-discs from A_r. Let $Z = \bigcap\{a_r : a_r \in T\}$ be the intersection of all the pseudo-discs in T. Since all the elements of T contain a common point (e.g., each point in P), Z is non-empty, and it follows from properties of pseudo-discs families that Z is connected. (For a proof of this geometric fact, see [1, Theorem 4.4]).

Now, we clip each pseudo-disc $s \in A_1 \cup \ldots \cup A_{r-1}$ to $s' = s \cap Z$, and slightly perturb the boundaries such that the family $\{s' : s \in A_1 \cup \ldots \cup A_{r-1}\}$ is still a family of pseudo-discs. This perturbation can be performed as follows: Define a partial ordering on $A_1 \cup \ldots \cup A_{r-1}$, by considering the intersection of each pseudo-disc with the boundary of Z and ordering by inclusion. Extend the ordering into a linear ordering arbitrarily. Clip the pseudo-discs close to the boundary of Z from the inside, in such a way that a "smaller" pseudo-disc (according to the linear ordering) is clipped closer to the boundary. In this way, an intersection is added only to pairs of pseudo-discs whose boundaries have only one intersection point inside Z, and therefore, any two boundaries of $\{s' : s \in A_1 \cup \ldots \cup A_{r-1}\}$ intersect at most twice.

Consider the $(r-1)$-partite intersection hypergraph H' of the families $A'_1, \ldots, A'_{r-1}$, where $A'_i = \{s' : s \in A_i\}$. H' has at least $|S|$ hyperedges, since for each $(a_1, \ldots, a_{r-1}) \in S$, the point $p(a_1, \ldots, a_{r-1})$ is contained in Z, and hence, $a'_1 \cap a'_2 \cap \ldots \cap a'_{r-1} \neq \emptyset$. Since $|S| > C_{r-1} t^6 n^{r-2} (\log n)^{r-3}$, by the induction hypothesis H' contains a $K^{r-1}_{t,\ldots,t}$ all of whose elements are fully contained in $Z = \bigcap\{a : a \in T\}$. (Here, C_3 is chosen to be sufficiently large for applying the inductive hypothesis.) Together with the set $T \subset A_r$, we obtain $K^r_{t,\ldots,t}$ in H. This completes the proof. ◀

References

1 E. Ackerman, B. Keszegh, and D. Pálvölgyi. Coloring hypergraphs defined by stabbed pseudo-disks and abab-free hypergraphs. *SIAM J. Discret. Math.*, 34(4):2250–2269, 2020. doi:10.1137/19M1290231.

2 A. Basit, A. Chernikov, S. Starchenko, T. Tao, and C.-M. Tran. Zarankiewicz's problem for semilinear hypergraphs. *Forum Math. Sigma*, 9:Paper No. e59, 23, 2021. doi:10.1017/fms.2021.52.

3 W. G. Brown. On graphs that do not contain a Thomsen graph. *Canadian Math. Bulletin*, 9(3):281–285, 1966. doi:10.4153/CMB-1966-036-2.

4 T. Chan, C. Keller, and S. Smorodinsky. On Zarankiewicz's problem for intersection hypergraphs of geometric objects, 2024. arXiv:2412.06490.

5 T. M. Chan and S. Har-Peled. On the number of incidences when avoiding an induced biclique in geometric settings. In *Proceedings of SODA 2023*, pages 1398–1413. SIAM, 2023. doi:10.1137/1.9781611977554.ch50.

6 B. Chazelle. Lower bounds for orthogonal range searching: I. The reporting case. *J. ACM*, 37(2):200–212, 1990. doi:10.1145/77600.77614.

7 F. R. K. Chung, P. Erdős, and J. Spencer. *On the decomposition of graphs into complete bipartite subgraphs, in: Studies in Mathematics: To the Memory of Paul Turán (P. Erdős, L. Alpár, G. Haĺasz, and A. Sárközy, eds.)*, pages 95–101. Birkhäuser, Basel, 1983. doi:10.1007/978-3-0348-5438-2_10.

8 T. Do. Zarankiewicz's problem for semi-algebraic hypergraphs. *J. Combin. Th., Ser. A*, 158:621–642, 2018. doi:10.1016/J.JCTA.2018.04.007.

9 T. Do. Representation complexities of semi-algebraic graphs. *SIAM J. Discret. Math.*, 4(33):1864–1877, 2019. doi:10.1137/18M1221606.

10 P. Erdős. On extremal problems of graphs and generalized graphs. *Israel J. Math.*, 2:183–190, 1964. doi:10.1007/BF02759942.

11 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Euro. Math. Soc.*, 19(6):1785–1810, 2017. doi:10.4171/JEMS/705.

12 N. Frankl and A. Kupavskii. On the Erdős-Purdy problem and the Zarankiewitz problem for semialgebraic graphs, 2021. arXiv:2112.10245.

13 O. Janzer and C. Pohoata. On the Zarankiewicz problem for graphs with bounded VC-dimension. *Combinatorica*, 44(4):839–848, 2024. doi:10.1007/S00493-024-00095-2.

14 C. Keller and S. Smorodinsky. Zarankiewicz's problem via ϵ-t-nets. In *SoCG 2024*, volume 293 of *LIPIcs*, pages 66:1–66:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.66.

15 P. Kővári, V. Sós, and P. Turán. On a problem of Zarankiewicz. *Colloq. Math.*, 3:50–57, 1954.

16 J. Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2(3):169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

17 A. Milojević, B. Sudakov, and I. Tomon. Incidence bounds via extremal graph theory, 2024. arXiv:2401.06670.

18 B. Sudakov. Recent developments in extremal combinatorics: Ramsey and Turán type problems. In *Proceedings of the International Congress of Mathematicians. Volume IV*, pages 2579–2606, 2010. doi:10.1142/9789814324359_0159.

19 J. Tidor and H-H. H. Yu. Multilevel polynomial partitioning and semialgebraic hypergraphs: regularity, Turán, and Zarankiewicz results, 2024. arXiv:2407.20221.

20 I. Tomon. Coloring lines and Delaunay graphs with respect to boxes. *Random Struct. Algorithms*, 64(3):645–662, 2024. doi:10.1002/RSA.21193.

21 I. Tomon and D. Zakharov. Turán-type results for intersection graphs of boxes. *Comb. Probab. Comput.*, 30(6):982–987, 2021. doi:10.1017/S0963548321000171.

22 M. Tong. Zarankiewicz bounds from distal regularity lemma, 2024. arXiv:2410.13695.

Simplification of Trajectory Streams

Siu-Wing Cheng
HKUST, Hong Kong, China

Haoqiang Huang
HKUST, Hong Kong, China

Le Jiang
USTC, Hefei, China

Abstract

While there are software systems that simplify trajectory streams on the fly, few curve simplification algorithms with quality guarantees fit the streaming requirements. We present streaming algorithms for two such problems under the Fréchet distance d_F in $\mathbb{R}^d$ for some constant $d \geq 2$.

Consider a polygonal curve τ in $\mathbb{R}^d$ in a stream. We present a streaming algorithm that, for any $\varepsilon \in (0,1)$ and $\delta > 0$, produces a curve σ such that $d_F(\sigma, \tau[v_1, v_i]) \leq (1+\varepsilon)\delta$ and $|\sigma| \leq 2\,\mathrm{opt} - 2$, where $\tau[v_1, v_i]$ is the prefix in the stream so far, and $\mathrm{opt} = \min\{|\sigma'| : d_F(\sigma', \tau[v_1, v_i]) \leq \delta\}$. Let $\alpha = 2(d-1)\lfloor d/2 \rfloor^2 + d$. The working storage is $O(\varepsilon^{-\alpha})$. Each vertex is processed in $O(\varepsilon^{-\alpha} \log \frac{1}{\varepsilon})$ time for $d \in \{2,3\}$ and $O(\varepsilon^{-\alpha})$ time for $d \geq 4$. Thus, the whole τ can be simplified in $O(\varepsilon^{-\alpha}|\tau| \log \frac{1}{\varepsilon})$ time. Ignoring polynomial factors in $1/\varepsilon$, this running time is a factor $|\tau|$ faster than the best static algorithm that offers the same guarantees.

We present another streaming algorithm that, for any integer $k \geq 2$ and any $\varepsilon \in (0, \frac{1}{17})$, maintains a curve σ such that $|\sigma| \leq 2k-2$ and $d_F(\sigma, \tau[v_1, v_i]) \leq (1+\varepsilon) \cdot \min\{d_F(\sigma', \tau[v_1, v_i]) : |\sigma'| \leq k\}$, where $\tau[v_1, v_i]$ is the prefix in the stream so far. The working storage is $O((k\varepsilon^{-1} + \varepsilon^{-(\alpha+1)}) \log \frac{1}{\varepsilon})$. Each vertex is processed in $O(k\varepsilon^{-(\alpha+1)} \log^2 \frac{1}{\varepsilon})$ time for $d \in \{2,3\}$ and $O(k\varepsilon^{-(\alpha+1)} \log \frac{1}{\varepsilon})$ time for $d \geq 4$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases streaming algorithm, curve simplification, Fréchet distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.34

Related Version *Full Version*: `http://arxiv.org/abs/2503.23025`

Funding This research is supported by the Research Grants Council, Hong Kong, China (project no. 16208923).

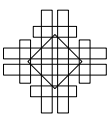
1 Introduction

The pervasive use of GPS sensors has enabled tracking of moving objects. For example, a car fleet can use real-time information about its vehicles to deploy them in response to dynamic demands. The sensor periodically samples the location of the moving object and sends it as a vertex to the remote cloud server for storage and processing. The remote server interprets the sequence of vertices received as a polygonal curve (trajectory).

For a massive stream, it has been reported [6, 7, 18, 19] that sending all vertices uses too much network bandwidth and storage at the server, and it may aggravate issues like out-of-order and duplicate data points. Software systems have been built for the sensor side to simplify trajectory streams on the fly (e.g. [16, 17, 18, 27, 28]). A local buffer is used. Every incoming vertex in the stream triggers a new round of computation that uses only the incoming vertex and the data in the local buffer. At the end of a round, these systems may determine the next vertex and send it to the cloud server, but they may also send nothing.

It is important to keep the local buffer small [16, 17, 18, 27, 28]. It means a small working storage and less computation in processing the next vertex in the stream, which enables real-time response and requires less computing power. In theoretical computer science,

© Siu-Wing Cheng, Haoqiang Huang, and Le Jiang;
licensed under Creative Commons License CC-BY 4.0



streaming algorithms have three goals [22]. Let n be the number of items in the stream so far. First, the working storage should be $o(n)$ if not polylog(n). Second, an item in the stream should be processed in polylog(n) time or at worst $o(n)$ time because items are arriving continuously. Third, although the best solution is unattainable as the entire input is not processed as a whole, the solution should be provably satisfactory. Let τ be the curve in the stream. The abovementioned software systems only deal with some local error measures between the simplified curve σ and τ. We want to provide guarantees on the size of σ and the global similarity between σ and τ.

A popular similarity measure is the *Fréchet distance* [14]. Let $\rho_\tau : [0,1] \to \mathbb{R}^d$ be a *parameterization* of τ such that, as t increases from 0 to 1, $\rho_\tau(t)$ moves from the beginning of τ to its end without backtracking. It is possible that $\rho_\tau(t) = \rho_\tau(t')$ for two distinct $t, t' \in [0,1]$. We can similarly define a parameterization ρ_σ for σ. These parameterizations induce a *matching* $\mathcal{M}$ between $\rho_\sigma(t)$ and $\rho_\tau(t)$ for all $t \in [0,1]$. A point may have multiple matching partners. The Fréchet distance is $d_F(\sigma,\tau) = \inf_{\mathcal{M}} \max_{t\in[0,1]} d(\rho_\sigma(t), \rho_\tau(t))$. We call a matching that realizes the Fréchet distance a *Fréchet matching*.

We study two problems. The *δ-simplification problem* is to compute, for a given $\delta > 0$, a curve σ of the minimum size (number of vertices) such that $d_F(\sigma,\tau) \le \delta$. The *$k$-simplification problem* is to compute, for a given integer $k \ge 2$, a curve σ of size k that minimizes $d_F(\sigma,\tau)$.

Given a polygonal curve $\tau = (v_1, v_2, \ldots)$, $|\tau|$ denotes its number of vertices, and for any $i < j$, $\tau[v_i, v_j]$ denotes the subcurve $(v_i, v_{i+1}, \ldots, v_j)$. For a subset $P \subset \mathbb{R}^d$, $d(x,P) = \inf_{p\in P} d(x,p)$. Given two points $x, y \in \mathbb{R}^d$, xy denotes the oriented line segment from x to y.

1.1 Related works

Static δ-simplification. Let $n = |\tau|$. In $\mathbb{R}$, if the vertices of σ are restricted to lie on τ (anywhere), then $|\sigma|$ can be minimized in $O(n)$ time [24]; if the vertices of σ must be a subset of those of τ, it takes $O(n)$ time to compute σ such that $|\sigma|$ is at most two more than the minimum [10]. In $\mathbb{R}$, there is a data structure [25] that, for any given δ, reports a curve σ with the minimum $|\sigma|$ such that the vertices of σ lie on τ (anywhere) and $d_F(\sigma,\tau) \le \delta$. The query time is $O(|\sigma|)$, and the preprocessing time is $O(n)$.

In $\mathbb{R}^2$, a curve σ with minimum $|\sigma|$ such that $d_F(\sigma,\tau) \le \delta$ can be computed in $O(n^2 \log^2 n)$ time [15]. In $\mathbb{R}^d$ for $d \ge 3$, if the vertices of σ must be a subset of those of τ, then $|\sigma|$ can be minimized in $O(n^3)$ time [5, 24, 26]. If there is no restriction on the vertices of σ, it has been recently shown that $|\sigma|$ can be minimized in $O\big((|\sigma|n)^{O(d|\sigma|)}\big)$ time [9].

Faster algorithms have been developed by allowing inexact output. Let $\kappa(\tau, r)$ be the minimum simplified curve size for an error r. In $\mathbb{R}^2$, it was shown [3] that for any $\delta > 0$, a curve σ can computed in $O(n \log n)$ time such that $d_F(\sigma,\tau) \le \delta$ and $|\sigma| \le \kappa(\tau, \delta/2)$. It is possible that $\kappa(\tau,\delta/2) \gg \kappa(\tau,\delta)$. A better control of the curve size has been obtained later. In $\mathbb{R}^d$, it was shown [24] that for any $\varepsilon \in (0,1)$ and $\delta > 0$, a curve σ can be constructed in $O(\varepsilon^{2-2d} n^2 \log n \log\log n)$ time such that $d_F(\sigma,\tau) \le (1+\varepsilon)\delta$ and $|\sigma| \le 2\kappa(\tau,\delta) - 2$.[1] Later, it was shown [8] that for any $\alpha, \varepsilon \in (0,1)$ and $\delta > 0$, a curve σ can be computed in $\tilde{O}\big(n^{O(1/\alpha)} \cdot (d/(\alpha\varepsilon))^{O(d/\alpha)}\big)$ time such that $d_F(\sigma,\tau) \le (1+\varepsilon)\delta$ and $|\sigma| \le (1+\alpha)\cdot\kappa(\tau,\delta)$.

The algorithms above for $d \ge 2$ do not fit the streaming setting [3, 5, 8, 9, 15, 24, 26]. In [3], the simplified curve σ is a subsequence of the vertices in τ, and the vertices of σ are picked from τ in a traversal of τ. Given the last pick v_i, for any $j > i$, whether v_j should

[1] In [24], the first and last vertices of σ are restricted to be the same as those of τ.

be included in σ is determined by examining the subcurve $\tau[v_i, v_j]$. This requires an $O(n)$ working storage which is too large. If the algorithms in [5, 8, 9, 15, 24, 26] are used in the streaming setting, the processing time per vertex would be $O(n)$ or more which is too high.

Static k-simplification. In $\mathbb{R}$, for any given k, the data structure for δ-simplification in [25] can be queried in $O(k)$ time to report a curve σ such that $|\sigma| = k$, the vertices of σ lie on τ (anywhere), and $d_F(\sigma, \tau)$ is minimized. In $\mathbb{R}^d$, if the vertices of σ must be a subset of those of τ, it was shown [14] that a solution σ can be computed in $O(n^4 \log n)$ time such that $d_F(\sigma, \tau) \leq 7 \cdot \min\{d_F(\sigma', \tau) : |\sigma'| \leq k$, no restriction on the vertices of $\sigma'\}$. The factor 7 can be reduced to 4 by a better analysis [3]. Nevertheless, if this algorithm is used in the streaming setting, the vertex processing time would be $O(n^3 \log n)$ or more, which is way too high.

Streaming line simplification. A *line simplification* σ of a stream τ is a subsequence of the vertices such that the first and last vertices of σ are the first and last vertices in the stream so far. In $\mathbb{R}^2$, for any integer $k \geq 2$ and any $\varepsilon \in (0, 1)$, it was shown [1, 2] how to maintain σ such that $|\sigma| \leq 2k$ and $d_F(\sigma, \tau) \leq (4\sqrt{2} + \varepsilon) \cdot \text{opt}_k$, where $\text{opt}_k = \min\{d_F(\sigma', \tau) :$ line simplification σ' of τ, $|\sigma'| \leq k\}$. The working storage is $O(k^2 + k\varepsilon^{-1/2})$. Each vertex is processed in $O(k \log \frac{1}{\varepsilon})$ amortized time. There are also streaming software systems for this problem that minimize some local error measures (e.g. [13, 20, 21, 23]); however, they do not provide any guarantee on the global similarity between τ and the output curve.

Discrete Fréchet distance. The *discrete* Fréchet distance $d_{\text{dF}}(\sigma, \tau)$ is obtained with the restriction that the parameterizations ρ_σ and ρ_τ must match the vertices of σ to those of τ and vice versa. Note that $d_{\text{dF}}(\sigma, \tau) \geq d_F(\sigma, \tau)$ and $d_{\text{dF}}(\sigma, \tau) \gg d_F(\sigma, \tau)$ in the worst case.

There are several results in $\mathbb{R}^3$ [4]. A curve σ with the minimum $|\sigma|$ such that $d_{\text{dF}}(\sigma, \tau) \leq \delta$ can be computed in $O(n \log n)$ time; if the vertices of σ must be a subset of those of τ, the computation time increases to $O(n^2)$. On the other hand, if k is given instead of δ, a curve σ such that $|\sigma| = k$ and $d_{\text{dF}}(\sigma, \tau)$ is minimized can be computed in $O(kn \log n \log(n/k))$ time; if the vertices of σ must a subset of those of τ, the computation time increases to $O(n^3)$.

In $\mathbb{R}^d$, a streaming k-simplification algorithm was proposed in [11] with guarantees on $d_{\text{dF}}(\sigma, \tau)$, where τ is the curve seen in the stream so far. It was shown that for any integer $k \geq 2$ and any $\varepsilon \in (0, 1)$, a curve σ can be computed such that $|\sigma| \leq k$ and $d_{\text{dF}}(\sigma, \tau) \leq 8 \cdot \min\{d_{\text{dF}}(\sigma', \tau) : |\sigma'| \leq k\}$. The working storage is $O(kd)$.

Improved results have been obtained subsequently [12]. For the streaming δ-simplification problem, there is a streaming algorithm in $\mathbb{R}^d$ that, for any $\gamma > 1$, computes a curve σ such that $d_{\text{dF}}(\sigma, \tau) \leq \delta$ and $|\sigma| \leq \kappa(\tau, \delta/\gamma)$. This algorithm relies on a streaming algorithm for maintaining a γ-approximate minimum closing ball (MEB) of points in $\mathbb{R}^d$; the processing time per vertex is asymptotically the same as the γ-approximate MEB streaming algorithm. There are two streaming k-simplification algorithms in [12]. The first one uses $O(\frac{1}{\varepsilon} kd \log \frac{1}{\varepsilon}) + O(\varepsilon)^{-(d+1)/2} \log^2 \frac{1}{\varepsilon}$ working storage and maintains a curve σ such that $|\sigma| \leq k$ and $d_{\text{dF}}(\sigma, \tau) \leq (1 + \varepsilon) \cdot \min\{d_{\text{dF}}(\sigma', \tau) : |\sigma'| \leq k\}$. The second algorithm uses $O(\frac{1}{\varepsilon} kd \log \frac{1}{\varepsilon})$ working storage and maintains a curve σ such that $|\sigma| \leq k$ and $d_{\text{dF}}(\sigma, \tau) \leq (1.22 + \varepsilon) \cdot \min\{d_{\text{dF}}(\sigma', \tau) : |\sigma'| \leq k\}$.

1.2 Our results

We present streaming algorithms for the δ-simplification and k-simplification problems in $\mathbb{R}^d$ for $d \geq 2$ under the Fréchet distance. Let $\tau = (v_1, v_2, \ldots)$ be the input stream.

Streaming δ-simplification. Our algorithm outputs vertices occasionally and maintains some vertices in the working storage. The simplified curve σ for the prefix $\tau[v_1, v_i]$ in the stream so far consists of vertices that have been output and vertices in the working storage. Vertices that have been output cannot be modified, so they form a prefix of all simplified curves to be produced in the future for the rest of the stream.

For any $\varepsilon \in (0, 1)$ and any $\delta > 0$, our algorithm produces a curve σ for the prefix $\tau[v_1, v_i]$ in the stream so far such that $d_F(\sigma, \tau[v_1, v_i]) \leq (1 + \varepsilon)\delta$ and $|\sigma| \leq 2\kappa(\tau[v_1, v_i], \delta) - 2$. Let $\alpha = 2(d-1)\lfloor d/2 \rfloor^2 + d$. The working storage is $O(\varepsilon^{-\alpha})$. Each vertex in the stream is processed in $O(\varepsilon^{-\alpha} \log \frac{1}{\varepsilon})$ time for $d \in \{2, 3\}$ and $O(\varepsilon^{-\alpha})$ time for $d \geq 4$. If our algorithm is used in the static case, the running time on a curve of size n is $O(\varepsilon^{-\alpha} n \log \frac{1}{\varepsilon})$. Ignoring polynomial factors in $1/\varepsilon$, our time bound is a factor n smaller than the $O(\varepsilon^{2-2d} n^2 \log n \log \log n)$ running time of the best static algorithm that achieves the same error and size bounds [24].

Intuitively, our streaming algorithm finds a line segment that stabs as many balls as possible that are centered at consecutive vertices of τ with radii $(1 + O(\varepsilon))\delta$. If such a line segment ceases to exist upon the arrival of a new vertex, we start a new line segment. Concatenating these segments forms the simplified curve. For efficiency, we approximate each ball by covering it with grid cells of $O(\varepsilon\delta)$ width. In [24], vertex balls and their covers by grid cells of $O(\varepsilon\delta)$ width are also used. Connections between all pairs of balls are computed to create a graph such that the simplified curve corresponds to a shortest path in the graph. In contrast, we maintain some geometric structures so that we can read off the next line segment from them. We prove an $O(\varepsilon^{-\alpha})$ bound on the total size of these structures, which yields the desired working storage and processing time per vertex.

Streaming k-simplification. A memory budget may cap the size of the simplified curve, motivating the streaming k-simplification problem. For example, in some wildlife tracking applications, the location-acquisition device is not readily accessible after deployment, and data is only offloaded from it after an extended period [19].

Our algorithm maintains the simplified curve σ in the working storage for the prefix $\tau[v_1, v_i]$ in the stream so far. For any $k \geq 2$ and any $\varepsilon \in (0, \frac{1}{17})$, our algorithm guarantees that $|\sigma| \leq 2k - 2$ and $d_F(\sigma, \tau[v_1, v_i]) \leq (1 + \varepsilon) \cdot \min\{d_F(\sigma', \tau[v_1, v_i]) : |\sigma'| \leq k\}$. Recall that $\alpha = 2(d-1)\lfloor d/2 \rfloor^2 + d$. The working storage is $O((k\varepsilon^{-1} + \varepsilon^{-(\alpha+1)}) \log \frac{1}{\varepsilon})$. Each vertex in the stream is processed in $O(k\varepsilon^{-(\alpha+1)} \log^2 \frac{1}{\varepsilon})$ time for $d \in \{2, 3\}$ and $O(k\varepsilon^{-(\alpha+1)} \log \frac{1}{\varepsilon})$ time for $d \geq 4$. We first simplify as in the case of δ-simplification. When the simplified curve reaches a size of $2k - 1$, the key idea is to run our δ-simplification algorithm with a suitable error tolerance to bring its size down to $2k - 2$.

There is only one prior streaming k-simplification algorithm [1, 2]. It works in $\mathbb{R}^2$. It restricts the vertices of the simplified curve σ to be a subset of those of τ, whereas our algorithm does not. It uses $O(k^2 + k\varepsilon^{-1/2})$ working storage, processes each vertex in $O(k \log \frac{1}{\varepsilon})$ amortized time, and guarantees that $|\sigma| \leq 2k$ and $d_F(\sigma, \tau) \leq (4\sqrt{2} + \varepsilon) \cdot \text{optimum}$ under the requirement that the vertices of σ must be a subset of those of τ. For $d = 2$, our algorithm uses $O((k\varepsilon^{-1} + \varepsilon^{-5}) \log \frac{1}{\varepsilon})$ working storage, processes each vertex in $O(k\varepsilon^{-5} \log^2 \frac{1}{\varepsilon})$ worst-case time, and guarantees that $|\sigma| \leq 2k - 2$ and $d_F(\sigma, \tau) \leq (1 + \varepsilon) \cdot \text{optimum}$. Ignoring the restriction on the vertices of σ, we offer a better approximation ratio for $d_F(\sigma, \tau)$. Although our vertex processing time bound is larger, it is worst-case instead of amortized. The comparison between working storage depends on the relative magnitudes of k and $1/\varepsilon$.

2 Streaming δ-simplification

Given a subset $X \subseteq \mathbb{R}^d$, we use $\text{conv}(X)$ to denote the convex hull of X. For any point $x \in \mathbb{R}^d$, let B_x denote the d-ball centered at x with radius δ. Consider the infinite d-dimenisonal grid with x as a grid vertex and side length $\varepsilon\delta/(2\sqrt{d})$. Let G_x be the subset of grid cells that intersect the ball centered at x with radius $(1+\varepsilon/2)\delta$. We say that an oriented line segment s *stabs the objects* $O_1, \ldots, O_m$ *in order* if there exist points $x_i \in s \cap O_i$ for $i \in [m]$ such that $x_1, \ldots, x_m$ appear in this order along s [15].

2.1 Algorithm

The high-level idea is to find the longest sequence $v_1, v_2, \ldots, v_i$ of vertices such that there is a segment s_1 that stabs B_{v_a} for $a \in [i]$ in order. Then, restart to find the longest sequence $v_{i+1}, \ldots, v_j$ such that there is a segment s_2 that stabs B_{v_a} for $a \in [i+1, j]$ in order. Repeating this way gives a sequence of segments $s_1, s_2, \ldots$. Concatenating these segments gives the simplified curve. The problem is the large working storage: a long vertex sequence may be needed to determine a line segment. We use several ideas to overcome this problem.

First, we approximate B_{v_a} by $\text{conv}(G_{v_a})$ and find a line segment that stabs the longest sequence $\text{conv}(G_{v_1}), \ldots, \text{conv}(G_{v_i})$ in order. We will show that it suffices for this line segment to start from the set P of grid points in G_{v_1}. So $|P| = O(\varepsilon^{-d})$.

Second, as the vertices arrive in the stream, for each point $p \in P$, we construct a structure $S_a[p]$ inductively as follows:

- Set $S_1[p] = \{p\}$ for all $p \in P$. We write $S_1[p] = p$ for convenience.
- For all $a \geq 1$, set $S_{a+1}[p] = \text{conv}(G_{v_{a+1}}) \cap F(S_a[p], p)$ for all $p \in P$, where $F(S_a[p], p) = \{y \in \mathbb{R}^d : py \cap S_a[p] \neq \emptyset\}$.

If $p \in S_a[p]$, then $F(S_a[p], p) = \mathbb{R}^d$ and hence $S_{a+1}[p] = \text{conv}(G_{v_{a+1}})$. If $S_a[p] = \emptyset$, then $F(S_a[p], p) = \emptyset$ and hence $S_{a+1}[p] = \emptyset$. Otherwise, by viewing p as a light source, $F(S_a[p], p)$ is the unbounded convex polytope consisting of $S_a[p]$ and the non-illuminated subset of $\mathbb{R}^d$.

We will show that for $a \geq 2$, $S_a[p]$ is equal to $\{x \in \text{conv}(G_{v_a}) : px$ stabs $\text{conv}(G_{v_1}), \ldots, \text{conv}(G_{v_a})$ in order$\}$. We will also show that for each $p \in P$, $S_a[p]$ and $F(S_a[p], p)$ have $\text{poly}(1/\varepsilon)$ complexities, and they can be constructed in $\text{poly}(1/\varepsilon)$ time. The array S_a is deleted after computing S_{a+1}, which keeps the working storage small.

We pause when we encounter a vertex v_{i+1} such that $S_{i+1}[p] = \emptyset$ for all $p \in P$. We choose any vertex q of any non-empty $S_i[p]$ (which must exist) and output the segment pq. Since pq stabs $\text{conv}(G_{v_1}), \ldots, \text{conv}(G_{v_i})$ in order, we can show that $d_F(pq, \tau[v_1, v_i]) \leq (1+\varepsilon)\delta$. All is well except that S_i has been deleted after computing S_{i+1}. A fix is that after processing a vertex v_a, we store a segment pq in a buffer $\tilde{\sigma}$ for an arbitrary vertex q of an arbitrary non-empty $S_a[p]$. We output the content of $\tilde{\sigma}$ after discovering that $S_{i+1}[p] = \emptyset$ for all $p \in P$.

Afterward, we reset P as the set of grid points in $G_{v_{i+1}}$ and restart the above processing from v_{i+1}. That is, reset $S_{i+1}[p] = p$ for all $p \in P$. One of the points in P will be the start of the next segment that will be output in the future. The start of this next segment may be far from $\text{conv}(G_{v_i})$, which contains the end of $\tilde{\sigma}$ that was just output. For $a = i+1, i+2, \ldots$, we set $S_{a+1}[p] = \text{conv}(G_{v_{a+1}}) \cap F(S_a[p], p)$ for all $p \in P$ until $S_a[p]$ becomes empty for all $p \in P$ again. In summary, the simplified curve σ is the concatenation of the output segments $s_1, s_2, \ldots$, i.e., the end of s_j is connected to the start of s_{j+1}. Figure 1 illustrates a few steps of this algorithm. The procedure Simplify(ε, δ) in Algorithm 1 gives the details.

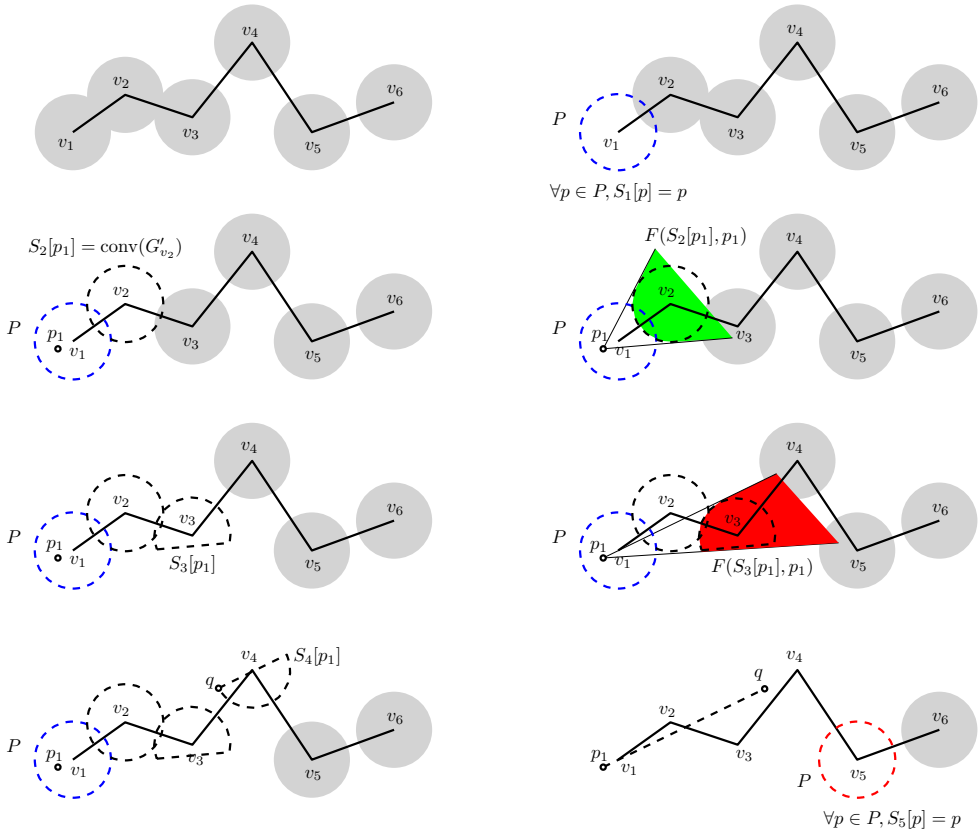


Figure 1 The shaded balls denote the $\mathrm{conv}(G_{v_a})$'s. The vertices arrive in order (from left to right and top to bottom). The first P consists of the grid points inside the blue dashed ball. Let p_1 be a grid point in P. $F(S_2[p_1], p_1)$ is the green unbounded convex region that is bounded by the tangents from p_1 to $S_2[p_1]$. $S_3[p_1] = \mathrm{conv}(G_{v_3}) \cap F(S_2[p_1], p_1)$ is denoted by the dashed clipped ball around v_3. Similarly, $F(S_3[p_1], p_1)$ is the red unbounded convex region that is bounded by the tangents from p_1 to $S_3[p_1]$. $S_4[p_1]$ is denoted by the dashed clipped ball around v_4. In this example, $S_5[p] = \emptyset$ for all $p \in P$, so we output $\tilde{\sigma}$ which may be equal to (p_1, q) for a vertex q of $S_4[p_1]$. We reset P as the set of grid points in G_{v_5} (inside the red dashed ball) and reset $S_5[p] = p$ for all $p \in P$.

2.2 Analysis

Given a polytope $Q \subset \mathbb{R}^d$, let $|Q|$ denote its complexity, i.e., the total number of its faces of all dimensions.

▶ **Lemma 1.** *Let P be the set of grid points in G_{v_i}. Assume that $S_i[p] = p$ for all $p \in P$. Take any $p \in P$ and any index $j \geq i+1$ such that $S_a[p] \neq \emptyset$ for all $a \in [i, j-1]$.*

(i) *$S_j[p] = \{x \in \mathrm{conv}(G_{v_j}) : px$ stabs $\mathrm{conv}(G_{v_i}), \ldots, \mathrm{conv}(G_{v_j})$ in order$\}$.*

(ii) *$S_j[p]$ is a convex polytope that can be computed in $O\big(|S_{j-1}[p]| + N \log N + N^{\lfloor d/2 \rfloor}\big)$ time, where N is any upper bound on the number of support hyperplanes of $F(S_{j-1}[p], p)$ such that $N = \Omega(\varepsilon^{(1-d)\lfloor d/2 \rfloor})$.*

▶ **Lemma 2.** *Let $\tau[v_1, v_i]$ be the prefix in the stream so far. The simplified curve σ satisfies the properties that $d_F(\sigma, \tau[v_1, v_i]) \leq (1+\varepsilon)\delta$ and $|\sigma| \leq 2\kappa(\tau[v_1, v_i], \delta) - 2$.*

▶ **Theorem 3.** *Let τ be a polygonal curve in $\mathbb{R}^d$ that arrives in a data stream. Let $\alpha = 2(d-1)\lfloor d/2 \rfloor^2 + d$. For every $\delta > 0$ and every $\varepsilon \in (0,1)$, the output curve σ of* Simplify(ε, δ) *satisfies $d_F(\sigma, \tau) \leq (1+\varepsilon)\delta$ and $|\sigma| \leq 2\kappa(\tau, \delta) - 2$. The working storage is $O(\varepsilon^{-\alpha})$. Each vertex is processed in $O(\varepsilon^{-\alpha} \log \frac{1}{\varepsilon})$ time for $d \in \{2, 3\}$ and $O(\varepsilon^{-\alpha})$ time for $d \geq 4$.*

Algorithm 1 SIMPLIFY.

Input: A stream $\tau = (v_1, v_2, \ldots)$, ε, and δ.

Output: The output curve σ consists of the sequence of vertices output so far and the vertices in the buffer $\tilde{\sigma}$. After processing the last vertex v_i at the end of the stream, SIMPIFY returns (P, S_i), which will be useful for the k-simplification problem.

function SIMPLIFY(ε, δ)
  read v_1 from the data stream; $i \leftarrow 1$; *init* $\leftarrow$ true; $\tilde{\sigma} \leftarrow \emptyset$
  while true do
    if *init* = true **then**
      init $\leftarrow$ false
      output the vertices in $\tilde{\sigma}$ ▷ output the line segment in $\tilde{\sigma}$
      $\tilde{\sigma} \leftarrow (v_i)$ ▷ view v_i as a degenerate line segment
      $P \leftarrow$ the grid points of G_{v_i}
      $S_i[p] \leftarrow p$ for all $p \in P$
    else
      choose a vertex q of some non-empty $S_i[p]$
      $\tilde{\sigma} \leftarrow (p, q)$ ▷ update the last edge of the simplified curve
    end if
    if end of data stream **then**
      output $\tilde{\sigma}$
      return (P, S_i) ▷ will be useful for k-simplification
    else
      read v_{i+1} from the data stream
    end if
    $S_{i+1}[p] \leftarrow \mathrm{conv}(G_{v_{i+1}}) \cap F(S_i[p], p)$ for all $p \in P$
    delete S_i
    init $\leftarrow$ true if $S_{i+1}[p] = \emptyset$ for all $p \in P$
    $i \leftarrow i + 1$
  end while
end function

Proof. The guarantees on σ follow from Lemma 2. We will bound the number of support hyperplanes and complexities of $S_j[p]$ and $F(S_{j-1}[p], p)$. The working storage and processing time then follow from Lemma 1(ii) and the fact that $|P| = O(\varepsilon^{-d})$.

A *bounding halfspace* of a convex polytope O is a halfspace that contains O and is bounded by the support hyperplane of a facet of O. We count the bounding halfspaces of $S_j[p]$. Some are bounding halfspaces of $F(S_{j-1}[p], p)$ whose boundaries pass through p and are tangent to $S_{j-1}[p]$. We call them the *pivot* bounding halfspaces. The remaining ones are bounding halfspaces of $\mathrm{conv}(G_{v_a})$ for some $a \le j$. We call these *non-pivot* bounding halfspaces.

Since $\mathrm{conv}(G_{v_a})$'s are translates of each other, their facets share a set V of unit outward normals. We have $|V| \le |\mathrm{conv}(G_{v_a})| = O(\varepsilon^{(1-d)\lfloor d/2 \rfloor})$. No two facets of $S_j[p]$ have the same vector in V because the two corresponding bounding halfspaces would be identical or nested – neither is possible. This feature helps us to prevent a cascading growth in the complexities of $S_j[p]$ in the inductive construction.

Take a pivot bounding halfspace h of $S_j[p]$. Suppose that h was introduced for the first time as a bounding halfspace of $S_i[p]$ for some $i \le j$. The boundary of h, denoted by ∂h, passes through p and is tangent to $S_{i-1}[p]$ at a $(d-2)$-dimensional face f of $S_{i-1}[p]$. Let

g_1 and g_2 be the bounding halfspaces of $S_{i-1}[p]$ whose boundaries contain the two facets of $S_{i-1}[p]$ incident to f. We make three observations. First, $F(S_{i-1}[p], p)$ lies inside the cone C_i of rays that shoot from p towards $S_{i-1}[p]$. Second, h is a bounding halfspace of C_i. Third, $S_a[p] \subseteq C_i$ for all $a \in [i, j]$.

We claim that neither ∂g_1 nor ∂g_2 passes through p. Neither g_1 nor g_2 is equal to h because h was not introduced before $S_i[p]$. If both ∂g_1 and ∂g_2 pass through p, then f spans a $(d-2)$-dimensional affine subspace ℓ that passes through p. But then ∂h meets C_i at ℓ only, which is a contradiction because ∂h should support a facet of $S_j[p]$. If either ∂g_1 or ∂g_2 passes through p, say ∂g_2, then the affine subspace spanned by f does not pass through p. But then p and f span a unique hyperplane, giving the contradiction that $g_2 = h$.

By our claim, g_1 and g_2 are non-pivot bounding halfspaces of $S_{i-1}[p]$. We charge h to (ϕ_1, ϕ_2) or (ϕ_2, ϕ_1), where ϕ_1 and ϕ_2 are the outward unit normals of g_1 and g_2, respectively. The pair (ϕ_1, ϕ_2) is charged if a ray that shoots from p to an interior point of $g_1 \cap g_2$ arbitrarily near $\partial g_1 \cap \partial g_2$ hits ∂g_1 before ∂g_2. Otherwise, the pair (ϕ_2, ϕ_1) is charged. Suppose that h is charged to (ϕ_1, ϕ_2). We argue that (ϕ_1, ϕ_2) is not charged again at some $(d-2)$-dimensional face of $S_a[p]$ for all $a \in [i, j-1]$ due to another pivot bounding halfspace of $S_j[p]$.

Assume to the contrary that (ϕ_1, ϕ_2) is charged again at some $(d-2)$-dimensional face of $S_b[p]$ for some $b \in [i, j-1]$ due to a pivot bounding halfspace h' of $S_j[p]$. Either the dimension of $\partial h' \cap C_i$ is zero or one, or the dimension of $\partial h' \cap C_i$ is two. In the first case, $\partial h'$ cannot support any facet of $S_j[p]$, a contradiction. In the second case, since h' is charged to the same ordered pair (ϕ_1, ϕ_2), $\partial h'$ must separate $S_b[p]$ from ∂h. Therefore, the cone of rays C_b that shoot from p towards $S_b[p]$ meets ∂h only at p. However, $S_j[p] \subseteq C_b$, which means that ∂h cannot support any facet of $S_j[p]$, a contradiction.

In all, $S_j[p]$ has $O(\varepsilon^{(1-d)\lfloor d/2 \rfloor})$ non-pivot bounding halfspaces and at most $|V|(|V|-1) = O(\varepsilon^{2(1-d)\lfloor d/2 \rfloor})$ pivot bounding halfspaces. The same bounds hold for $S_{j-1}[p]$. If a support hyperplane L of $F(S_{j-1}[p], p)$ is not a support hyperplane of $S_{j-1}[p]$, then L passes through p and is tangent to $S_{j-1}[p]$ at some $(d-2)$-dimensional face f. Since L is different from the support hyperplanes of the two facets of $S_{j-1}[p]$ incident to f, these two hyperplanes do not pass through p. So the two facets incident to f induce some non-pivot bounding halfspaces of $S_{j-1}[p]$, implying that $F(S_{j-1}[p], p)$ has $O(\varepsilon^{2(1-d)\lfloor d/2 \rfloor})$ bounding halfspaces. It follows that $\left|S_j[p]\right|$ and $\left|F(S_{j-1}[p], p)\right|$ are $O(\varepsilon^{2(1-d)\lfloor d/2 \rfloor^2})$. ◀

3 Streaming algorithm for the k-simplification problem

3.1 Algorithm

Let σ_i^* be the optimal solution for the k-simplification problem for $\tau[v_1, v_i]$. If we knew $\delta_i^* = d_F(\sigma_i^*, \tau[v_1, v_i])$, we could call Simplify$(\varepsilon, \delta_i^*)$ to obtain an approximate solution for the k-simplification problem. Therefore, we try to estimate δ_i^* on the fly.

We maintain an output simplified curve σ_i in the working storage after processing every vertex v_i in the stream. The first simplified curve σ_{2k-1} is obtained by calling Simplify, via an invocation of a new procedure Compress, on $\tau[v_1, v_{2k-1}]$ with δ equal to some value δ_{2k-1}. Given a curve of size $2k-1$, Compress simplifies it to a curve of size at most $2k-2$.

For $i \geq 2k$, when a vertex v_i arrives in the stream, we either update the last edge of σ_{i-1} to produce σ_i as in Simplify, or start a new segment (initially just the vertex v_i) as in Simplify. Suppose that we start new segment. If $|\sigma_{i-1}| < 2k-2$, we can append v_i to σ_{i-1} to form σ_i. Otherwise, σ_i is obtained by calling Simplify, via an invocation of Compress, on $\sigma_{i-1} \circ (v_i)$ with δ equal to some value δ_i to be specified later. We will prove in Lemmas 4–8 that $\delta_i \leq (1 + O(\varepsilon))\delta_i^*$. Afterward, we repeat the above to process the next vertex in the stream. The above processing is elaborated in Reduce in Algorithm 2.

We define a curve τ_i for every $i \geq 2k-1$ in the comments in lines 6, 11, and 22. These curves facilitate the analysis, but they are not maintained by the algorithm as variables. For each i, τ_i is the curve from which the simplification σ_i is computed. Therefore, $\tau_{2k-1} = \tau[v_1, v_{2k-1}]$ (line 6). Consider any $i \geq 2k$. If we call Compress to simplify $\sigma_{i-1} \circ (v_i)$ to a curve σ_i of size at most $2k-2$, we set $\tau_i = \sigma_{i-1} \circ (v_i)$ (line 22). Otherwise, σ_i is obtained by simplifying $\tau_{i-1} \circ (v_i)$ whether we start a new segment at v_i or not, so $\tau_i = \tau_{i-1} \circ (v_i)$ (line 11).

The input parameter r of Reduce needs an explanation. We will show how to compute a lower bound $\delta_{\min} > 0$ for $d_F(\sigma^*_{2k-1}, \tau[v_1, v_{2k-1}])$. For $i \geq 2k-1$, let $r_i \geq 1$ and $s_i \geq 0$ be integers such that

$$\frac{1}{\varepsilon^{s_i}}\delta_{\min} \leq \frac{(1+\varepsilon)^{r_i-1}}{\varepsilon^{s_i}}\delta_{\min} \leq \delta^*_i \leq \frac{(1+\varepsilon)^{r_i}}{\varepsilon^{s_i}}\delta_{\min} \leq \frac{1}{\varepsilon^{s_i+1}}\delta_{\min}.$$

We will show that if we can start the processing of the data stream with an initial error tolerance of $\delta_{2k-2} = \varepsilon(1+\varepsilon)^{r_i+1}(1-4\varepsilon)^{-1}\delta_{\min}$, then the output curve σ_i will be within a Fréchet distance of $\varepsilon^{-s_i}(1+\varepsilon)^{r_i+1}(1-4\varepsilon)^{-1}\delta_{\min}$ from $\tau[v_1, v_i]$. By the definitions of r_i and s_i, the error is at most $(1+O(\varepsilon))\delta^*_i$. The correct values of r_i and s_i are unknown to us. Nevertheless, no matter what s_i is, we have $1 \leq r_i \leq \lfloor \log_{1+\varepsilon} \frac{1}{\varepsilon} \rfloor$. We launch $\lfloor \log_{1+\varepsilon} \frac{1}{\varepsilon} \rfloor = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ independent runs of Reduce for each r between 1 and $\lfloor \log_{1+\varepsilon} \frac{1}{\varepsilon} \rfloor$. After processing the arriving vertex v_i, the run with the minimum δ_i gives the right answer. This idea of using multiple independent runs to capture the right δ was proposed in [12] for streaming k-simplification under the discrete Fréchet distance. We still have to obtain s_i. This is achieved by establishing a lower bound for δ^*_i (Lemma 5) and computing a particular upper bound for δ^*_i in line 2 of Compress.

We assume that no three consecutive vertices in the stream are collinear. We enforce it using $O(1)$ more working storage as follows. Remember the last two vertices (x, y) in the stream, but do not feed y to Reduce yet. Wait until the next vertex z arrives. If x, y and z are not colinear, feed y to Reduce and remember (y, z) as the last two vertices. If x, y and z are collinear, then drop y, make (x, z) the last two vertices, and wait for the next vertex.

Algorithm 3 shows the procedure Compress. If Compress is called when processing v_i, its task is to simplify $\tau_i = \sigma_{i-1} \circ (v_i)$ to a curve σ_i of size at most $2k-2$ with error estimate $\delta_i = \varepsilon^{-t}\delta_{i-1}$, where t is some judiciously chosen positive integer.

3.2 Analysis

Lemma 4 below gives a lower bound for the Fréchet distance between a curve of size m and another curve of size n such that $m \geq 2n-1$. An application of this result shows that $\delta_{\min}$ computed in line 4 of Reduce is a lower bound for $d_F(\sigma^*_{2k-1}, \tau[v_1, v_{2k-1}])$ as $|\sigma^*_{2k-1}| = k$.

▶ **Lemma 4.** *Let $\xi = (p_1, \ldots, p_m)$ and $\zeta = (q_1, \ldots, q_n)$ be any two curves such that $m \geq 2n-1$. Then, $d_F(\xi, \zeta) \geq \frac{1}{2}\min\{d(p_i, p_{i-1}p_{i+1}) : i \in [2, m-1]\}$.*

Lemma 5 below shows that a curve $(p_1, \ldots, p_{2k-1})$ can be simplified to a curve of size at most $2k-2$ using an error tolerance of $\frac{1}{2}\min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}$. By Lemma 5, Compress always returns a curve of size at most $2k-2$.

▶ **Lemma 5.** *Let $\xi = (p_1, \ldots, p_{2k-1})$. Assume no three consecutive vertices are collinear. Let $\hat{\delta}$ be the minimum value such that calling* Simplify$(\varepsilon, \hat{\delta})$ *on ξ produces a curve of size at most $2k-2$. Then,*

$$\hat{\delta} \leq \frac{1}{2}\min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\} \leq (1+\varepsilon)\hat{\delta}.$$

Algorithm 2 REDUCE.

Input: A data stream $\tau = (v_1, v_2, \ldots)$ and three parameters r, ε, and k.
Output: Let $\tau[v_1, v_i]$ be the prefix in the stream so far. A curve σ_i is maintained such that $|\sigma_i| \leq 2k-2$ and $d_F(\sigma_i, \tau[v_1, v_i]) \leq (1+\varepsilon)\delta_i$.

procedure REDUCE(r, ε, k)
  read the first $2k-1$ vertices $v_1, \ldots, v_{2k-1}$ of τ
  ▷ no three consecutive vertices are collinear
  $\delta_{\min} \leftarrow \frac{1}{2}\min\{d(v_i, v_{i-1}v_{i+1}) : i \in [2, 2k-2]\}$
  $\delta_{2k-2} \leftarrow \varepsilon(1+\varepsilon)^{r+1}(1-4\varepsilon)^{-1}\delta_{\min}$
  $\sigma_{2k-2} \leftarrow \tau[v_1, v_{2k-2}]$ ▷ $\tau_{2k-1} \leftarrow \tau[v_1, v_{2k-1}]$
  $(\sigma_{2k-1}, \delta_{2k-1}, P, S_{2k-1}) \leftarrow$ COMPRESS$(\tau[v_1, v_{2k-1}], \varepsilon, k, \delta_{2k-2})$
  $i \leftarrow 2k$
  while true do
    read v_i from the data stream
    $\delta_i \leftarrow \delta_{i-1}$ ▷ $\tau_i \leftarrow \tau_{i-1} \circ (v_i)$
    $S_i[p] \leftarrow \mathrm{conv}(G_{v_i}) \cap F(S_{i-1}[p], p)$ for all $p \in P$, where G_{v_i} is defined with $\delta = \delta_i$
    if $S_i[p] \neq$ null for some $p \in P$ **then**
      ▷ update the last possibly degenerate segment as in SIMPLIFY
      $q \leftarrow$ any vertex of $S_i[p]$
      $\sigma_i \leftarrow \sigma_{i-1}$ with the last segment replaced by pq
    else if $|\sigma_{i-1}| < 2k-2$ **then** ▷ start a new segment as in SIMPLIFY
      $P \leftarrow$ the grid points of G_{v_i} ▷ G_{v_i} is defined with $\delta = \delta_i$
      $S_i[p] \leftarrow p$ for all $p \in P$
      $\sigma_i \leftarrow \sigma_{i-1} \circ (v_i)$ ▷ treat v_i as a degenerate segment
    else ▷ simplify $\sigma_{i-1} \circ (v_i)$ to produce σ_i
      $(\sigma_i, \delta_i, P, S_i) \leftarrow$ COMPRESS$(\sigma_{i-1} \circ (v_i), \varepsilon, k, \delta_{i-1})$ ▷ $\tau_i = \sigma_{i-1} \circ (v_i)$
    end if
    delete σ_{i-1}, δ_{i-1}, and S_{i-1}
    $i \leftarrow i+1$
  end while
end procedure

Algorithm 3 COMPRESS.

Input: A polygonal curve $(x_1, \ldots, x_{2k-1})$ and three parameters ε, k, and δ.
Output: $(\zeta, \varepsilon^{-t}\delta, P, S)$, where ζ is the output curve of calling SIMPLIFY$(\varepsilon, \varepsilon^{-t}\delta)$ on $(x_1, \ldots, x_{2k-1})$ for some $t \in \mathbb{N}$, and (P, S) are the output returned by SIMPLIFY after processing x_{2k-1}.

function COMPRESS$((x_1, \ldots, x_{2k-1}), \varepsilon, k, \delta)$
  $t \leftarrow \min\left\{i \in \mathbb{N} : i \geq 1, \varepsilon^{-i}\delta \geq \frac{1}{2}\min\{d(x_j, x_{j-1}x_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}\right\}$
  $\zeta \leftarrow$ output curve produced by calling SIMPLIFY$(\varepsilon, \varepsilon^{-t}\delta)$ on $(x_1, \ldots, x_{2k-1})$
  $(P, S) \leftarrow$ output returned by the above call of SIMPLIFY after processing x_{2k-1}
  return $(\zeta, \varepsilon^{-t}\delta, P, S)$
end function

Proof. Consider the run of SIMPLIFY$(\varepsilon, \hat{\delta})$ on ξ. Let $p_{i_1}, p_{i_2} \ldots$ be the vertices in order along ξ at which SIMPLIFY starts a new segment. Note that $i_1 = 1$ and $i_{j+1} \geq i_j + 2$ for all j.

SIMPLIFY replaces each subcurve $\xi[p_{i_j}, p_{i_{j+1}-1}]$ by a segment. The simplified curve is a concatenation of these segments. Since $|\xi| = 2k-1$, there must be an index j such that $i_{j+1} > i_j + 2$ so that SIMPLIFY$(\varepsilon, \hat{\delta})$ simplifies ξ to a curve of size at most $2k-2$. Let $b = \min\{j : i_{j+1} > i_j + 2\}$. Because $i_1 = 1$ and $i_{j+1} = i_j + 2$ for all $j \in [b-1]$, we know that i_j is odd for every $j \in [b]$ and $i_b + 1$ is even. It follows from the choice of b that SIMPLIFY does not start a new segment at p_{i_b+2}. By the working of SIMPLIFY$(\varepsilon, \hat{\delta})$, there is a grid point p in $G_{p_{i_b}}$ (defined with $\delta = \hat{\delta}$) such that $S_{i_b+2}[p] \neq \emptyset$. By the reasoning in the proof of Lemma 2, for any point $x \in S_{i_b+2}[p]$, $d_F(px, \xi[p_{i_b}, p_{i_b+2}]) \leq (1+\varepsilon)\hat{\delta}$. On the other hand, by Lemma 4, $d_F(px, \xi[p_{i_b}, p_{i_b+2}]) \geq \frac{1}{2} d(p_{i_b+1}, p_{i_b} p_{i_b+2})$. Hence, $(1+\varepsilon)\hat{\delta} \geq \frac{1}{2} d(p_{i_b+1}, p_{i_b} p_{i_b+2}) \geq \frac{1}{2} \min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}$.

It remains to argue that $\hat{\delta} \leq \frac{1}{2} \min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}$. Let $\delta = \frac{1}{2} \min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2], j \text{ is even}\}$. It suffices to show that calling SIMPLIFY(ε, δ) on ξ returns a curve of size at most $2k-2$. Let $p_{a_1}, p_{a_2}, \ldots$ be the vertices in order along ξ at which SIMPLIFY(ε, δ) starts a new segment. Let c be the even number in $[2, 2k-2]$ such that $d(p_c, p_{c-1}p_{c+1}) = \min\{d(p_j, p_{j-1}p_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}$.

Suppose that there is an index j that satisfies $a_j \leq c-1$ and $a_{j+1} > a_j + 2$. Since $\xi[p_{a_j}, p_{a_{j+1}-1}]$ is simplified to a line segment, the prefix $\xi[p_1, p_{a_{j+1}-1}]$ is simplified to a curve of size at most $a_{j+1} - 2$, i.e., at least one vertex less. We are done because the size of the whole simplified curve must be at most $2k-2$.

The remaining possibility is that $a_{j+1} = a_j + 2$ for all $a_j \in [c-1]$. Since $a_1 = 1$ and c is even, we have $a_1 = 1, a_2 = 3, a_3 = 5, \ldots, a_j = c-1$. It follows that SIMPLIFY starts a new segment at p_{c-1}. Let $r = d(p_c, p_{c-1}p_{c+1})$.

We claim that there is a line segment xy such that $d_F(xy, \xi[p_{c-1}, p_{c+1}]) \leq r/2$. Let z be the nearest point in $p_{c-1}p_{c+1}$ to p_c. So zp_c has length r. Let z' be the midpoint of zp_c. Translate $p_{c-1}p_{c+1}$ by the vector $z' - z$ to obtain a segment xy. Observe that $z' \in xy$ and $d(x, p_{c-1}) = d(y, p_{c+1}) = d_F(z', p_c) = r/2$. By linear interpolations between xz' and $p_{c-1}p_c$ and between $z'y$ and $p_c p_{c+1}$, we have $d_F(xy, \xi[p_{c-1}, p_{c+1}]) \leq r/2$, establishing our claim.

By the choice of c, $\frac{1}{2}r = \frac{1}{2} d(p_c, p_{c-1}c_{c+1}) = \delta$. Then, our claim implies that xy stabs balls of radii δ centered at p_{c-1}, p_c, p_{c+1} in order. There is a grid point p' in $G_{p_{c-1}}$ (defined using δ) within a distance $\varepsilon\delta/2$ from x. By a linear interpolation between $p'y$ and xy, we know that $p'y$ stabs $\text{conv}(G_{p_{c-1}}), \text{conv}(G_{p_c}), \text{conv}(G_{p_{c+1}})$ in order. Therefore, $S_{c+1}[p'] \neq \emptyset$ by Lemma 1(i), implying that SIMPLIFY does not start a new segment at p_{c+1}. Hence, SIMPLIFY replaces a subcurve $\xi[p_{c-1}, p_e]$ for some $e \geq c+1$ by a segment and produces a curve of size at most $2k-2$. This proves that $\delta \geq \hat{\delta}$. ◀

Recall the curves τ_i for $i \geq 2k-1$ defined in the comments in lines 6, 11, and 22. The next result follows immediately from the working of REDUCE.

▶ **Lemma 6.** *For $i \geq 2k-1$, σ_i computed by* REDUCE *can also be produced by calling* SIMPLIFY(ε, δ_i) *on τ_i.*

▶ **Lemma 7.** *Assume that $\varepsilon \in (0, 1/3]$. For $i \geq 2k-1$, $d_F(\tau_i, \tau[v_1, v_i]) \leq 2\varepsilon\delta_i$.*

Recall that σ_i^* is the curve of size k at minimum Fréchet distance from $\tau[v_1, v_i]$ and that $\delta_i^* = d_F(\sigma_i^*, \tau[v_1, v_i])$. Also, recall that for $i \geq 2k-1$, $r_i \geq 1$ and $s_i \geq 0$ are integers that satisfy the following inequalities:

$$\frac{1}{\varepsilon^{s_i}}\delta_{\min} \leq \frac{(1+\varepsilon)^{r_i-1}}{\varepsilon^{s_i}}\delta_{\min} \leq \delta_i^* \leq \frac{(1+\varepsilon)^{r_i}}{\varepsilon^{s_i}}\delta_{\min} \leq \frac{1}{\varepsilon^{s_i+1}}\delta_{\min}. \tag{1}$$

By Lemma 4, $\delta_{\min} \le \delta^*_{2k-1}$. Also, $\delta^*_a \le \delta^*_b$ for all $a < b$ because a Fréchet matching between σ^*_b and $\tau[v_1, v_b]$ also matches $\tau[v_1, v_a]$ to a curve of size at most k. Therefore, $\delta_{\min} \le \delta^*_i$ for $i \ge 2k-1$, which implies that r_i and s_i are well defined for $i \ge 2k-1$. Note that r_i is an integer between 1 and $\lfloor \log_{1+\varepsilon}(1/\varepsilon) \rfloor$.

▶ **Lemma 8.** *For all $i \ge 2k-1$ and $\varepsilon \in \big(0, \frac{1}{17}\big)$, $\textsc{Reduce}(r_i, \varepsilon, k)$ computes $\delta_i \le (1+8\varepsilon)\delta^*_i$.*

Proof. We prove that $\textsc{Reduce}(r_i, \varepsilon, k)$ computes $\delta_i \le \varepsilon^{-s_i}1+\varepsilon)^{r_i+1}(1-4\varepsilon)^{-1}\delta_{\min}$. By (1), this is at most $(1+\varepsilon)^2(1-4\varepsilon)^{-1}\delta^*_i \le (1+8\varepsilon)\delta^*_i$ for $\varepsilon \le 1/17$.

Consider the case of $i = 2k-1$. We examine the call $\textsc{Reduce}(r_{2k-1}, \varepsilon, k)$. By (1), $\varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}-1}\delta_{\min} \le \delta^*_{2k-1} \le \varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}}\delta_{\min}$. Theorem 3 and Lemma 5 imply that $\varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}-1}\delta_{\min} \le \delta^*_{2k-1} \le \frac{1}{2}\min\big\{d(v_j, v_{j-1}v_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\big\} \le (1+\varepsilon)\delta^*_{2k-1} \le \varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}+1}\delta_{\min}$. Since the input parameter r of Reduce is equal to r_{2k-1}, line 5 of Reduce sets $\delta_{2k-2} = \varepsilon(1+\varepsilon)^{r_{2k-1}+1}(1-4\varepsilon)^{-1}\delta_{\min}$. For $\varepsilon \le 1/17$, we have $\varepsilon^{-1} > (1+\varepsilon)^2(1-4\varepsilon)^{-1}$, and so $\varepsilon^{1-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}+1}(1-4\varepsilon)^{-1}\delta_{\min} < \varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}-1}\delta_{\min} \le \frac{1}{2}\min\big\{d(v_j, v_{j-1}v_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\big\}$. As a result, line 2 of Compress ensures that $\delta_{2k-1} = \varepsilon^{-s_{2k-1}}(1+\varepsilon)^{r_{2k-1}+1}(1-4\varepsilon)^{-1}\delta_{\min}$.

Consider an index $i > 2k-1$. We examine the call $\textsc{Reduce}(r_i, \varepsilon, k)$. Define:

$$\Delta = \varepsilon^{-s_i}(1+\varepsilon)^{r_i+1}\delta_{\min}. \tag{2}$$

Our goal is to prove that $\delta_i \le (1-4\varepsilon)^{-1}\Delta$. We rewrite the middle inequalities in (1) as

$$(1+\varepsilon)^{-2}\Delta \le \delta^*_i \le (1+\varepsilon)^{-1}\Delta. \tag{3}$$

Define:

$$a = \max\big\{j \in [2k-2, i] : \delta_j \le \varepsilon(1-4\varepsilon)^{-1}\Delta\big\}. \tag{4}$$

The index a is well defined because one can verify that $\delta_{2k-2} = \varepsilon(1+\varepsilon)^{r_i+1}(1-4\varepsilon)^{-1}\delta_{\min} \le \varepsilon(1-4\varepsilon)^{-1}\Delta$.

By (4), (2), line 5 of Reduce, and line 2 of Compress, $\delta_a = \varepsilon^h(1-4\varepsilon)^{-1}\Delta$ for some integer $h \ge 1$. If $a = i$, then $\delta_i = \delta_a \le \varepsilon(1-4\varepsilon)^{-1}\Delta$, so we are done. Suppose that $a < i$. Note that $\delta_{a+1} \ne \delta_a$ by (4), which implies that $\delta_{a+1} > \delta_a$. So $\tau_{a+1} = \sigma_a \circ (v_{a+1})$, and $\textsc{Compress}(\tau_{a+1}, \varepsilon, k, \delta_a)$ is called to produce σ_{a+1} and δ_{a+1}. We have

$$\begin{aligned} d_F(\sigma^*_{a+1}, \tau_{a+1}) &\le d_F(\sigma^*_{a+1}, \tau[v_1, v_{a+1}]) + d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) \\ &= \delta^*_{a+1} + d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) \ \le\ \delta^*_i + d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]). \end{aligned} \tag{5}$$

If $a+1 = 2k-1$, then $\tau_{a+1} = \tau_{2k-1} = \tau[v_1, v_{2k-1}]$, so $d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) = 0$. Suppose that $a+1 > 2k-1$. By Lemma 6 and Theorem 3, $d_F(\sigma_a, \tau_a) \le (1+\varepsilon)\delta_a$. By Lemma 7, $d_F(\tau_a, \tau[v_1, v_a]) \le 2\varepsilon\delta_a$. So $d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) \le d_F(\sigma_a, \tau[v_1, v_a]) \le d_F(\sigma_a, \tau_a) + d_F(\tau_a, \tau[v_1, v_a]) \le (1+3\varepsilon)\delta_a$. Since $\delta_a \le \varepsilon(1-4\varepsilon)^{-1}\Delta$, we conclude that

$$d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) \le \varepsilon(1+3\varepsilon)(1-4\varepsilon)^{-1}\Delta. \tag{6}$$

Plugging (6) into (5) and using (3), we obtain the following inequality for $\varepsilon < 1/17$:

$$d_F(\sigma^*_{a+1}, \tau_{a+1}) \le \left(\frac{1}{1+\varepsilon} + \frac{\varepsilon+3\varepsilon^2}{1-4\varepsilon}\right)\Delta \le \frac{\Delta}{(1+\varepsilon)(1-4\varepsilon)}. \tag{7}$$

Consider the call of Simplify in $\textsc{Compress}(\tau_{a+1}, \varepsilon, k, \delta_a)$. By (7) and Theorem 3, calling Simplify on τ_{a+1} with an error tolerance of $\frac{1}{(1+\varepsilon)(1-4\varepsilon)}\Delta$ will produce a curve of size at most $2k-2$. Then, by Lemma 5, $\frac{1}{2}\min\big\{d(v_j, v_{j-1}v_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\big\} \le$

$(1+\varepsilon)\cdot\frac{1}{(1+\varepsilon)(1-4\varepsilon)}\Delta = (1-4\varepsilon)^{-1}\Delta$. Line 2 of Compress ensures that $\delta_{a+1} = \varepsilon^{-t}\delta_a = \varepsilon^{h-t}(1-4\varepsilon)^{-1}\Delta$ for the smallest integer $t \geq 1$ such that $\delta_{a+1} \geq \frac{1}{2}\min\{d(v_j, v_{j-1}v_{j+1}) : j \in [2, 2k-2],\ j \text{ is even}\}$. Hence, t cannot be greater than h. But t cannot be less than h because $\delta_{a+1} > \varepsilon(1-4\varepsilon)^{-1}\Delta$ by (4). So $\delta_{a+1} = (1-4\varepsilon)^{-1}\Delta$. For every $b \in [a+2, i]$,

$$\begin{aligned} d_F(\sigma_b^*, \tau_{a+1}\circ\tau[v_{a+2}, v_b]) &\leq d_F(\sigma_b^*, \tau[v_1, v_b]) + d_F(\tau_{a+1}\circ\tau[v_{a+2}, v_b], \tau[v_1, v_b]) \\ &\leq \delta_b^* + d_F(\tau_{a+1}, \tau[v_1, v_{a+1}]) \\ &\leq \delta_i^* + \varepsilon(1+3\varepsilon)(1-4\varepsilon)^{-1}\Delta \qquad (\because (6)) \\ &< (1-4\varepsilon)^{-1}\Delta = \delta_{a+1}. \qquad (\because (3)) \end{aligned}$$

Then, when processing v_{a+2}, Lemma 6 and Theorem 3 imply that Reduce executes lines 10–20 with $\delta_{a+2} = \delta_{a+1} = (1-4\varepsilon)^{-1}\Delta$ to produce σ_{a+2} of size at most $2k-2$. Applying this reasoning inductively gives $\delta_i = (1-4\varepsilon)^{-1}\Delta$. ◀

▶ **Theorem 9.** *Let τ be a polygonal curve in $\mathbb{R}^d$ that arrives in a data stream. Let $\alpha = 2(d-1)\lfloor d/2\rfloor^2 + d$. There is a streaming algorithm that, for any integer $k \geq 2$ and any $\varepsilon \in (0, \frac{1}{17})$, maintains a curve σ such that $|\sigma| \leq 2k-2$ and $d_F(\tau,\sigma) \leq (1+\varepsilon)\cdot\min\{d_F(\tau,\sigma') : |\sigma'| \leq k\}$. It uses $O\big((k\varepsilon^{-1} + \varepsilon^{-(\alpha+1)})\log\frac{1}{\varepsilon}\big)$ working storage and processes each vertex of τ in $O(k\varepsilon^{-(\alpha+1)}\log^2\frac{1}{\varepsilon})$ time for $d \in \{2,3\}$ and $O(k\varepsilon^{-(\alpha+1)}\log\frac{1}{\varepsilon})$ time for $d \geq 4$.*

Proof. Let $\delta^* = \min\{d_F(\tau,\sigma') : |\sigma'| \leq k\}$. By Lemma 8, if we launch $\lfloor\log_{1+\varepsilon/10}(10/\varepsilon)\rfloor = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ runs of Reduce using $\varepsilon/10$ instead of ε, there is a run that outputs a curve σ such that $d_F(\tau,\sigma) \leq (1+\varepsilon/10)(1+8\varepsilon/10)\delta^* \leq (1+\varepsilon)\delta^*$. Then, the working storage and processing time per vertex follow from Theorem 3. ◀

References

1. M.A. Abam, M. de Berg, P. Hachenberger, and A. Zarei. Streaming algorithms for line simplification. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 175–183, 2007.
2. M.A. Abam, M. de Berg, P. Hachenberger, and A. Zarei. Streaming algorithms for line simplification. *Discrete and Computational Geometry*, 43:497–515, 2010. doi:10.1007/S00454-008-9132-4.
3. P.K. Agarwal, S. Har-Peled, N.H. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3):203–219, 2005. doi:10.1007/S00453-005-1165-Y.
4. S. Bereg, M. Jiang, W. Wang, B. Yang, and B. Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Proceedings of the Latin American Symposium on Theoretical Informatics*, pages 630–641, 2008.
5. K. Bringmann and B.R. Chaudhury. Polyline simplification has cubic complexity. In *Proceedings of the International Symposium on Computational Geometry*, pages 18:1–18:16, 2019.
6. H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *The VLDB Journal*, 15(3):211–228, 2006. doi:10.1007/S00778-005-0163-7.
7. M. Chen, M. Xu, and P. Fränti. A fast $o(n)$ multiresolution polygonal approximation algorithm for gps trajectory simplification. *IEEE Transactions on Image Processing*, 21(5):2770–2785, 2012.
8. S.-W. Cheng and H. Huang. Curve simplification and clustering under Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1414–1432, 2023.
9. S.-W. Cheng and H. Huang. Solving Fréchet distance problems by algebraic geometric methods. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 4502–4513, 2024.
10. A. Driemel, A. Krivošija, and C. Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785, 2016.

11 A. Driemel, I. Psarros, and M. Schmidt. Sublinear data structures for short Fréchet queries, 2019. arXiv:1907.04420. arXiv:1907.04420.

12 A. Filtser and O. Filtser. Static and streaming data structures for Fréchet distance queries. *ACM Transactions on Algorithms*, 19(4):39:1–39:36, 2023. doi:10.1145/3610227.

13 Y. Gao, Z. Fang, J. Xu, S. Gong, C. Shen, and L. Chen. An efficient and distributed framework for real-time trajectory stream clustering. *IEEE Transactions on Knowledge and Data Engineering*, 36(5):1857–1873, 2024. doi:10.1109/TKDE.2023.3312319.

14 M. Godau. A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*, pages 127–136, 1991.

15 L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993. doi:10.1142/S0218195993000257.

16 X. Lin, J. Jiang, S. Ma, Y. Zuo, and C. Hu. One-pass trajectory simplification using the synchronous Euclidean distance. *The VLDB Journal*, 28:897–921, 2018.

17 X. Lin, S. Ma, J. Jiang, Y. Hou, and T. Wo. Error bounded line simplification algorithms for trajectory compression: an experimental evaluation. *ACM Transactions on Database Systems*, 46(3):11:1–11:44, 2021. doi:10.1145/3474373.

18 X. Lin, S. Ma, H. Zhang, T. Wo, and J. Huai. One-pass error bounded trajectory simplification. In *Proceedings of the VLDB Endowment*, volume 10, pages 841–852, 2017. doi:10.14778/3067421.3067432.

19 J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, and R. Jurdak. Bounded quadrant system: error-bounded trajectory compression on the go. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 987–998, 2015.

20 J. Muckell, J.-H. Hwang, V. Patil, C.T. Lawson, F. Ping, and S. Ravi. SQUISH: an online approach for GPS trajectory compression. In *Proceedings of the International Conference on Computing for Geospatial Research & Applications*, pages 13:1–13:8, 2011.

21 J. Muckell, P.W. Olsen, J.-H. Hwang, C.T. Lawson, and S. Ravi. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica*, 18(3):435–460, 2014. doi:10.1007/S10707-013-0184-0.

22 S. Muthukrishnan. *Data Streams: Algorithms and Applications*. now Publishers Inc., 2005.

23 M. Potamias, K. Patroumpas, and T. Sellis. Sampling trajectory streams with spatiotemporal criteria. In *Proceedings of the IEEE International Conference on Scientific and Statistical Database Management*, pages 275–284, 2006.

24 M. van de Kerkhof, I. Kostitsyna, M. Löffler, M. Mirzanezhad, and C. Wenk. Global curve simplification. In *Proceedings of the European Symposium on Algorithms*, pages 67:1–67:14, 2019.

25 T. van der Horst and T. Ophelders. Computing minimum complexity 1D curve simplifications under the Fréchet distance. In *The 39th European Workshop on Computational Geometry*, 2023.

26 M. van Kreveld, M. Löffler, and L. Wiratma. On optimal polyline simplification using the Hausdorff and Fréchet distance. In *Proceedings of the International Symposium on Computational Geometry*, pages 56:1–56:14, 2018.

27 Z. Wang, C. Long, G. Cong, and Q. Zhang. Error-bounded online trajectory simplification with multi-agent reinforcement learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1758–1768, 2021.

28 D. Zhang, M. Ding, D. Yang, Y. Liu, J. Fan, and H.T. Shen. Trajectory simplification: an experimental study and quality analysis. In *Proceedings of the VLDB Endowment*, volume 11, pages 934–946, 2018. doi:10.14778/3213880.3213885.

A Theory of Sub-Barcodes

Oliver A. Chubet
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Kirk P. Gardner
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Donald R. Sheehy
Department of Computer Science, North Carolina State University, Raleigh, NC, USA

Abstract

The primary tool in topological data analysis (TDA) is persistent homology, which involves computing a *barcode* – often from point-cloud or scalar field data – that serves as a topological signature for the underlying function. In this work, we introduce *sub-barcodes* and show how they arise naturally from *factorizations* of persistence module homomorphisms. We show that, as a partial order induced by factorizations, the relation of being a sub-barcode is strictly stronger than the rank invariant, and we apply sub-barcode theory to the problem of inferring information about the barcode of an unknown Lipschitz function from samples. The advantage of this approach is that it permits strong guarantees – with no noise – while requiring no sampling assumptions, and the resulting barcode is guaranteed to be a sub-barcode of every Lipschitz function that agrees with the data. We also present an algorithmic theory that allows for the efficient approximation of sub-barcodes using filtered Delaunay triangulations for Euclidean inputs.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Mathematics of computing → Geometric topology; Theory of computation → Computational geometry; Computing methodologies → Algebraic algorithms

Keywords and phrases Topology, Topological Data Analysis, Persistent Homology, Persistence Modules, Barcodes, Sub-barcodes, Factorizations, Lipschitz Extensions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.35

Related Version *Full Version*: `https://arxiv.org/abs/2206.10504` [10]

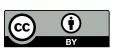
Funding This work was partially funded by the NSF under grant CCF-2017980.

1 Introduction

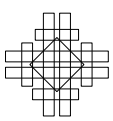
Topological data analysis (TDA) seeks to compute topological properties of unknown functions from finite samples. The most widely-used tool for TDA is persistent homology (PH), and the most widely-studied functions are distance functions. Thus, the algorithmic problems arising in this field tend to be either algebraic (PH computation), geometric (distance function representation), or both.

A PH pipeline starts with a real-valued function as input, and then considers the sublevel sets of this function. The homology of the sublevels at each threshold value are tracked and aggregated into a *barcode* (see Figure 1). The goal is to compute or approximate this barcode, which often requires substantial assumptions on the quality of the sample.

In this paper, we relax this goal to instead compute whatever *sub-barcode* is supported by the data we have. We form a sub-barcode by taking a sub-interval of each bar in the barcode. The formal definition of a sub-barcode and how it arises naturally in PH is covered in Section 3. The main idea is that, given *some* of the data, we compute *some* of the underlying structure.



© Oliver A. Chubet, Kirk P. Gardner, and Donald R. Sheehy;
licensed under Creative Commons License CC-BY 4.0



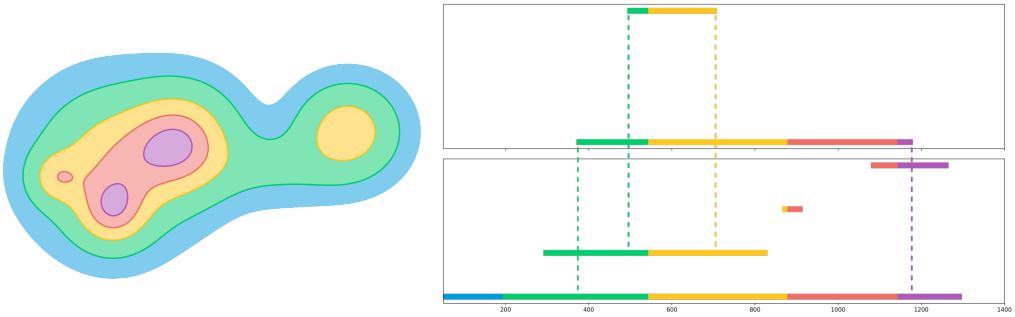


Figure 1 (Left) A contour plot of a Lipschitz function $f : X \to \mathbb{R}$. (Right) The barcode $\mathsf{Bar}(f)$ of f (bottom) and a sub-barcode (top). All colors correspond to the function values indicated on the axis (bottom-right).

In practice, analyzing the topology of a finite sample entails two main challenges:

- **Extension:** If S is a finite sample of the underlying space X, then we need to extrapolate the values for a function f from the known values at S to the unknown values at the rest of X. We won't get the exact answer, but we would like some guarantees on the extension.
- **Discretization:** Having a good approximation to the function is insufficient: we also need a reliable discrete representation of the underlying space. In practice, these representations usually arise in the form of a simplicial complex or a simplicial set.

In the classic setting of distance functions in Euclidean space, strong sampling assumptions guarantee that the distance to a sample approximates the distance to the underlying set. The Delaunay triangulation provides a perfect discretization, capturing the exact homology of the distance sublevels. This was the original setting for PH [13].

Lipschitz functions generalize distance functions and allow us to represent a much wider class of inputs. Several previous papers have shown that, given a sufficiently dense sample, a good approximation to the barcode of a Lipschitz function can be computed [6, 8, 11]. The guarantees from such work is that the output barcode is close to the true barcode in a standard metric called the bottleneck distance. Our approach eliminates all sampling assumptions, but gives guarantees in terms of a partial order on barcodes; that is, it is guaranteed to produce no noise or spurious bars, but finds only those topological features (bars) that are sufficiently supported by the sample.

Contributions and Outline

The background and definitions appear in sections 2 and 3. The *algebraic theory* of sub-barcodes is presented in Section 4. First, we show how sub-barcodes are related to the widely-studied rank invariant. Surprisingly, despite being a complete invariant for persistence, the natural extension of the rank invariant to a partial order is strictly less discriminating than sub-barcodes. This would be useless if it were not the case that sub-barcodes can be computed, a fact we show in the main sub-barcode theorem (Theorem 4), which states that sub-barcodes arise whenever we have one persistence module homomorphism factoring through another.

In Section 5, we lay out the *extension theory*. According to the algebraic theory of the previous section, the key to producing sub-barcodes is to find suitable upper and lower bounds on the unknown function. We show that maximum and minimum Lipschitz extensions serve

this purpose (Figure 2), as well as how to give good guarantees with only an approximation to the upper and lower bounds (Corollary 9). This section also contains a much more general result to encompass other classes of functions beyond Lipschitz functions (Theorem 11).

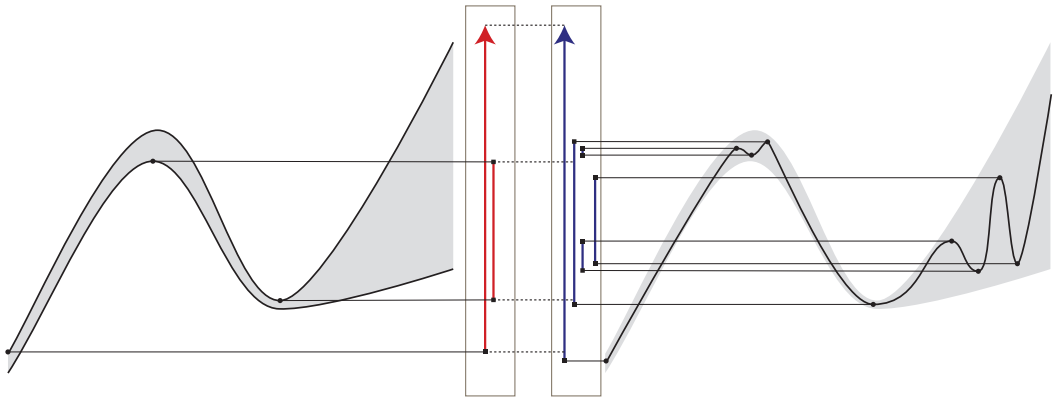


Figure 2 On the left, two functions are depicted, one is an upper bound and the other is a lower bound on an unknown function $f : \mathbb{R} \to \mathbb{R}$. There is a corresponding barcode associated with the pair that matches minima in the upper bound to maxima in the lower bound. On the right is a candidate function f that lies between the upper and lower bounds and its barcode $\mathsf{Bar}(f)$. The barcode of the inclusion of the upper and lower bounds is a sub-barcode of $\mathsf{Bar}(f)$.

The *discretization theory* is presented in Section 6. Here, we show how to approximate the Lipschitz extension sub-barcode of the previous section using Delaunay triangulations. This leads to a theory of semi-supervised TDA in which only a subset of function values are known (Theorem 13). We also give a tighter approximation using the barycentric decomposition of the Delaunay triangulation (Theorem 14), and show that this method also applies to a wider class of functions (Example 15).

The *categorical theory* of sub-barcodes is presented in Section 7. The TDA and PH pipeline depend on category theory to express the way relationships between inputs impact relationships between outputs, i.e., barcodes. We relate sub-barcodes to the classic approach of smoothing barcodes and show how this arises naturally from categorical considerations. We also give a new perspective on barcodes in general for which sub-barcodes are shown to be true subobjects in the categorical sense, and discuss how this construction is related to the theory of fuzzy sets [15, 1, 21].

2 Background

Functions and Filtrations

Let (X, d) be a metric space. When the metric is Euclidean, we denote the distance as a norm, i.e., $\mathrm{d}(x, y) := \|x - y\|$. The distance from a point x to a (finite) set U is denoted $\mathrm{d}(x, U) := \min_{u \in U} \mathrm{d}(x, u)$.

The real-valued functions $f : X \to \mathbb{R}$ form a vector space where $(f + \alpha\, g)(x) := f(x) + \alpha\, g(x)$, for all $\alpha \in \mathbb{R}$. We also have a poset of functions, where $f \geq g$ iff $f(x) \geq g(x)$ for all $x \in X$. A function $f : X \to \mathbb{R}$ is λ-*Lipschitz* if, for all $x, y \in X$, we have

$$f(y) - \lambda\, \mathrm{d}(x, y) \leq f(x) \leq f(y) + \lambda\, \mathrm{d}(x, y).$$

If λ is not specified, then it is assumed that $\lambda = 1$. This is perhaps not the most common way to express the Lipschitz condition, but it is closest to how it is used in this work.

The *t-sublevel set* $\mathrm{Sub}_f(t)$ of f is the set of points $x \in X$ with $f(x) \leq t$:

$$\mathrm{Sub}_f(t) := f^{-1}((-\infty, t]) = \{x \in X \mid f(x) \leq t\}.$$

This notation is designed to indicate that Sub_f is a *filtration*: for $s \leq t$ we have $\mathrm{Sub}_f(s) \subseteq \mathrm{Sub}_f(t)$. There is a partial order on sublevel filtrations where $\mathrm{Sub}_f \subseteq \mathrm{Sub}_g$ iff $\mathrm{Sub}_f(t) \subseteq \mathrm{Sub}_g(t)$ for all $t \in \mathbb{R}$. It is a simple exercise to check that, if $f \geq g$, then $\mathrm{Sub}_f \subseteq \mathrm{Sub}_g$; i.e., larger functions have smaller sublevel sets.

▶ Remark 1. The mapping from a function to its sublevel filtration is a contravariant functor from the poset of functions to the poset of filtrations. The filtrations themselves are functors $\mathbb{R} \to \mathbf{Top}$ from the totally ordered set $\mathbb{R}$ of real numbers to **Top**, the category of topological spaces. Throughout, we use remarks like this one to highlight the categorical interpretations. The reader who is unfamiliar with these terms may still follow all definitions and algorithms.

Persistence Modules

A *pointwise finite dimensional (p.f.d.) persistence module* $M : \mathbb{R} \to \mathbf{vec}$ assigns a (finite-dimensional) vector space $M(t)$ to all $t \in \mathbb{R}$ and a linear map $M_{s \leq t} : M(s) \to M(t)$ for all $s \leq t$. The map $M_{s \leq s}$ is the identity on $M(s)$, and for all $s \leq t \leq u$, $M_{s \leq u} = M_{t \leq u} \circ M_{s \leq t}$. For persistence modules M and N, a homomorphism $\phi : M \to N$ assigns linear maps $\phi_t : M(t) \to N(t)$ for all $t \in \mathbb{R}$ so that for all $s \leq t$, we have $\phi_t \circ M_{s \leq t} = N_{s \leq t} \circ \phi_s$.

▶ Remark 2. A persistence module M is a functor $M : \mathbb{R} \to \mathbf{vec}$, and a persistence module homomorphism $M \to N$ is a natural transformation $\phi : M \Rightarrow N$ between functors $\mathbb{R} \to \mathbf{vec}$, so that the category of persistence modules is precisely the category $\mathbf{vec}^{\mathbb{R}} = [\mathbb{R}, \mathbf{vec}]$ of functors $\mathbb{R} \to \mathbf{vec}$.

A *factorization* of homomorphisms $F : M \to N$ to $G : M' \to N'$ is given by a pair of homomorphisms $\varphi = \big(\varphi_1 : M' \to M,\ \varphi_2 : N \to N'\big)$ such that $G = \varphi_2 \circ F \circ \varphi_1$ (Diagram 1).

$$\begin{array}{ccc} M & \xleftarrow{\varphi_1} & M' \\ \downarrow{\scriptstyle F} & & \downarrow{\scriptstyle G} \\ N & \xrightarrow{\varphi_2} & N'. \end{array} \tag{1}$$

Let $\mathbf{Fac}(\mathbf{vec}^{\mathbb{R}})$ denote the category with homomorphisms of persistence modules as objects, and arrows $\varphi : F \to G$. Then $\varphi : F \to G$ is a factorization of G through F.

▶ **Example 3.** Letting $M(\epsilon)$ denote a persistence module M shifted by a constant $\epsilon \in \mathbb{R}$, a 2ϵ-interleaving is traditionally given by a pair of commuting diagrams

$$\begin{array}{ccc} M(-\epsilon) & \xrightarrow{M\varepsilon} & M(\epsilon) \\ & {\scriptstyle \Phi(-\epsilon)} \searrow \quad \nearrow {\scriptstyle \Psi} & \\ & N & \end{array} \tag{2}$$

$$\begin{array}{ccc} & M & \\ & {\scriptstyle \Psi(-\epsilon)} \nearrow \quad \searrow {\scriptstyle \Phi} & \\ N(-\epsilon) & \xrightarrow[N\varepsilon]{} & N(\epsilon) \end{array} \tag{3}$$

depicting factorizations of homomorphisms $M\varepsilon : M(-\epsilon) \to M(\epsilon)$ and $N\varepsilon : N(-\epsilon) \to N(\epsilon)$ by a pair $\big(\Phi : M \to N(\epsilon),\ \Psi : N \to M(\epsilon)\big)$. In this section, we show that the commutativity of Diagrams (2) and (3) imply sub-barcode relations $\mathsf{Bar}(M\varepsilon) \sqsubseteq \mathsf{Bar}(N)$ and $\mathsf{Bar}(M) \sqsupseteq \mathsf{Bar}(N\varepsilon)$.

Homology

Homology is a topological invariant that is particularly useful because it is both informative and efficiently computable for triangulable topological spaces. A homology functor $\mathrm{H} : \mathbf{Top} \to \mathbf{vec}$ associates a vector space $\mathrm{H}(X)$ to a topological space X, and takes continuous maps between topological spaces to linear maps between the corresponding vector spaces.

Persistence modules naturally arise in TDA by taking the homology of a sublevel filtration of a function. This assembles the functorial TDA pipeline from functions to filtrations to persistence modules. It is useful to see these as functors and not just functions because the additional structure of functoriality tells us that, if $f \geq g$, then there is a persistence module homomorphism $\mathrm{H}\,\mathrm{Sub}_f \to \mathrm{H}\,\mathrm{Sub}_g$. As we will see, these homomorphisms can be combined into factorizations (leading to sub-barcodes) and interleavings (leading to approximations).

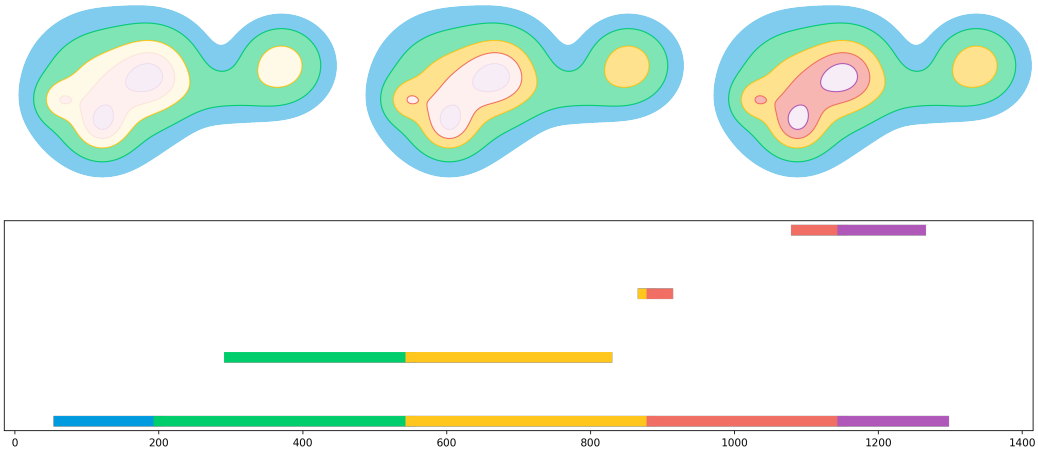


Figure 3 Sublevels of a 3-Lipschitz function $f : X \to \mathbb{R}$ on a subset of $\mathbb{R}^2$ (top) and its barcode $\mathsf{Bar}(f) = \mathsf{Bar}(\mathrm{H}_1\mathrm{Sub}_f)$ in dimension 1 (bottom). Colors correspond to the function values indicated on the bottom-axis.

3 Barcodes and Sub-Barcodes

Let $\mathcal{I}$ denote the poset of intervals on $\mathbb{R}$ ordered by inclusion. A *barcode* B is a function $B : \underline{B} \to \mathcal{I}$ where $\underline{B}$ is a set whose elements $\beta \in \underline{B}$ are called *bars*. The barcode assigns an interval to each bar. Note that B is not necessarily injective. When there is no confusion, we identify the bars in a barcode with their corresponding intervals. A *sub-barcode mapping* between barcodes A and B is a function $\phi : \underline{A} \to \underline{B}$ such that, for all $\alpha \in \underline{A}$, we have $A(\alpha) \subseteq B(\phi(\alpha))$. If ϕ is injective, then we call it a *sub-barcode matching* and say A is a *sub-barcode* of B, denoted $A \sqsubseteq B$. That is, A is a sub-barcode of B if A can be formed from B by taking a subinterval of each bar in a subset of $\underline{B}$ (Figure 4).

There is a barcode $\mathsf{Bar}(M)$ corresponding to any persistence module M. Note, the image of a persistence module homomorphism $\phi : M \to N$ is also persistence module. We overload notation and write $\mathsf{Bar}(\phi)$ to indicate $\mathsf{Bar}(\operatorname{im}\phi)$.

From the PH pipeline, we have persistence modules $\mathrm{H}\,\mathrm{Sub}_f$ associated with each continuous function $f : X \to \mathbb{R}$. We again simplify our notation and write $\mathsf{Bar}(f)$ to denote $\mathsf{Bar}(\mathrm{H}\,\mathrm{Sub}_f)$ and $\mathsf{Bar}(f \geq g)$ to denote $\mathsf{Bar}(\operatorname{im}\mathrm{H}(\mathrm{Sub}_f \subseteq \mathrm{Sub}_g))$. The object giving us a barcode is always clear from context.

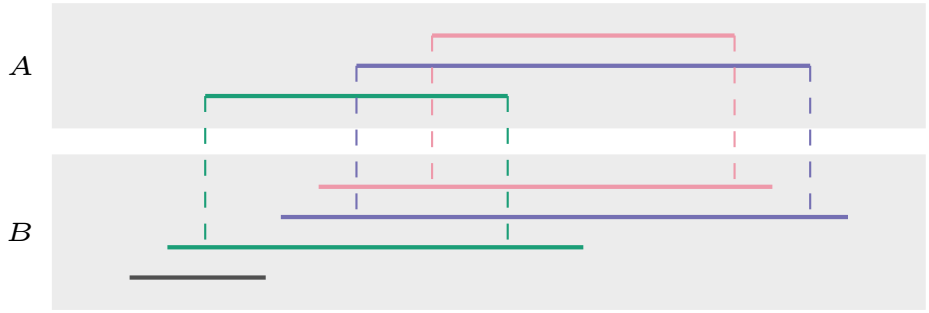


Figure 4 Two barcodes A and B and a sub-barcode matching $M : A \to B$.

We now have enough definitions to state the main theorem of sub-barcodes. The proof is postponed until Section 4 when we will have developed some necessary tools.

▶ **Theorem 4** (The Sub-Barcode Theorem)**.** *If there exists a factorization $\varphi : F \to G$ of persistence module homomorphisms (i.e., $G = \varphi_2 F \varphi_1$), then* $\mathsf{Bar}(G) \sqsubseteq \mathsf{Bar}(F)$.

4 The Algebraic Theory

4.1 Sub-barcodes vs. Ranks

Let $\mathrm{rk}\ m$ denote the *rank* of a linear map m defined as the dimension of its image, $\mathrm{im}\, m$. The *rank invariant* rank_M of a persistence module $M : \mathbb{R} \to \mathbf{vec}$ is a function from ordered pairs in $\mathbb{R}$ to the natural numbers defined as $\mathrm{rank}_M(s \leq t) := \mathrm{rk}\, M_{s \leq t}$. It is straightforward to extend this invariant to homomorphisms $F : M \to N$ by letting $\mathrm{rank}_F(s \leq t) := \mathrm{rk}\,(F_t \circ M_{s \leq t})$. In the following, we consider the rank invariant for a persistence module M to be the rank invariant of its identity homomorphism, i.e., $\mathrm{rank}_M := \mathrm{rank}_{1_M}$.

A basic fact from linear algebra is that, if a linear map g factors through a second linear map f, then $\mathrm{rk}\,(g) \leq \mathrm{rk}\,(f)$. It follows immediately that, if there is a factorization $F \to G$ of persistence module homomorphisms, then for all $s \leq t$,

$$\mathrm{rank}_G(s \leq t) \leq \mathrm{rank}_F(s \leq t).$$

This puts a partial order on rank invariants that we call the *sub-rank relation* and denote by $\mathrm{rank}_G \leq \mathrm{rank}_F$.

▶ Remark 5. rk is a contravariant functor $\mathbf{Fac}(\mathbf{vec}) \to \mathbb{N}$ and rank is a contravariant functor $\mathbf{Fac}(\mathbf{vec}^{\mathbb{R}}) \to \mathbb{N}$, i.e., from factorizations of vector spaces (resp. persistence modules) to the poset of natural numbers. An ordering of two rank invariants is a natural transformation between these functors.

In light of the sub-barcode theorem, the rank invariant is the same *type* of invariant as sub-barcodes. It puts a partial order on morphisms induced by factorizations. However, although the barcode can be constructed from the rank invariant, the natural ordering of rank invariants is a weaker invariant than sub-barcodes as expressed in the following.

▶ **Theorem 6.** *The sub-barcode order is a more discriminating invariant than the sub-rank order in the sense that, for all persistence module homomorphisms F and G,* $\mathsf{Bar}(G) \sqsubseteq \mathsf{Bar}(F)$ *implies that* $\mathrm{rank}_G \leq \mathrm{rank}_F$*, but not vice-versa.*

Proof. The rank invariant is easily extracted from a barcode: $\operatorname{rank}_F(s \leq t)$ is the number of bars in $\mathsf{Bar}(F)$ that contain the closed interval $s \leq t$. It follows immediately from the injectivity of the sub-barcode matching (and the pigeonhole principle) that $\mathsf{Bar}(G) \sqsubseteq \mathsf{Bar}(F)$ implies that $\operatorname{rank}_G \leq \operatorname{rank}_F$.

For the other side of the proof, it suffices to give an example of persistence modules M and N such that $\operatorname{rank}_M \leq \operatorname{rank}_N$ while $\mathsf{Bar}(M) \not\sqsubseteq \mathsf{Bar}(N)$.[1] Let M be a persistence module where $\mathsf{Bar}(M)$ has two bars corresponding to intervals $[0, 1)$ and $[2, 3)$, and let N be a persistence module such that $\mathsf{Bar}(N)$ has only one bar corresponding to the interval $[0, 3)$. Clearly, $\operatorname{rank}_M \leq \operatorname{rank}_N$, but it is not possible for a barcode with two bars to be a sub-barcode of one with only one bar. Thus $\mathsf{Bar}(M) \not\sqsubseteq \mathsf{Bar}(N)$. ◀

4.2 Induced Matching Theory Proof of the Sub-Barcode Theorem

Although not expressed in these terms, Bauer and Lesnick's theory of induced matchings [3] shows that sub-barcode matchings arise from certain homomorphisms. The induced matchings in that work are matchings between the bars that do not (necessarily) satisfy the inclusion requirement of a sub-barcode matching. They showed that for any homomorphism $F : M \to N$, there is a partial bijective function (a *matching*) between the bars with the property that every bar of $\mathsf{Bar}(F)$ is the intersection of the intervals of the matched bars from $\mathsf{Bar}(M)$ to $\mathsf{Bar}(N)$. These are constructed from the pair of canonical (functorial) matchings $\mathsf{Bar}(M) \to \mathsf{Bar}(F) \to \mathsf{Bar}(N)$ induced by the epi-mono factorization of F through its image. Translated into the vocabulary of sub-barcodes, this result implies that $\mathsf{Bar}(F) \sqsubseteq \mathsf{Bar}(M)$ and $\mathsf{Bar}(F) \sqsubseteq \mathsf{Bar}(N)$.

More generally, if F is a monomorphism (all maps F_t are injective), then $\mathsf{Bar}(M) \sqsubseteq \mathsf{Bar}(N)$. This is because the induced matching of a monomorphism matches every bar, in addition to the property that, if $[a, b]$ is matched to $[c, d]$, then $a \geq c$ and $b = d$; thus, a submodule relation between persistence modules implies a sub-barcode relation between the barcodes.

If F is an epimorphism (all maps F_t are surjective), then the induced matching is surjective, and if $[a, b]$ is matched to $[c, d]$, then $a = c$ and $d \leq b$. So, reversing this induced matching, we get $\mathsf{Bar}(N) \sqsubseteq \mathsf{Bar}(M)$. We use these matchings in the proof of the Sub-Barcode Theorem below.

Proof of The Sub-Barcode Theorem (Theorem 4). The factorization $\varphi = (\varphi_1, \varphi_2) : F \to G$ entails $G = \varphi_2 F \varphi_1$. Let $m : \operatorname{im} G \hookrightarrow \operatorname{im} \varphi_2 F$ be the unique monomorphism given by the universal property of images, and let $e : \operatorname{im} F \twoheadrightarrow \operatorname{im} \varphi_2 F$ be the epimorphism given by resticting φ_2 to $\operatorname{im} F$. Using the induced matching theory, m gives a sub-barcode matching $\mathsf{Bar}(\varphi_2 F) \sqsubseteq \mathsf{Bar}(F)$. Then, the reverse of the matching induced by e gives a sub-barcode matching $\mathsf{Bar}(G) \sqsubseteq \mathsf{Bar}(\varphi_2 F)$. Putting these together, we get

$$\mathsf{Bar}(G) \sqsubseteq \mathsf{Bar}(\varphi_2 F) \sqsubseteq \mathsf{Bar}(F).$$ ◀

The following corollary for ordered sequences of functions forms the basis for all of our applications of sub-barcodes in TDA.

▶ **Corollary 7.** *If $h \geq g \geq f \geq e$ are functions $X \to \mathbb{R}$, then $\mathsf{Bar}(h \geq e) \sqsubseteq \mathsf{Bar}(g \geq f)$.*

Proof. It suffices to observe that the ordering on the functions gives a factorization $(g \geq f) \to (h \geq e)$ of $h \geq e$ through $g \geq f$. By the functoriality of the sublevel functor and homology, we get the corresponding factorization of persistence modules. The claim then follows from Theorem 4. ◀

[1] As before, rank_M is shorthand for $\operatorname{rank} 1_M$.

5 The Extension Theory

Given an unknown Lipschitz function $f : X \to \mathbb{R}$ and a sample $S \subset X$, let f_S denote the function f restricted to the points of S. The pair (S, f_S) denotes input to a TDA problem where the goal is to infer as much as possible about f. Any function $g : X \to \mathbb{R}$ that agrees with f_S at the points of S is called an *extension* of f_S. If g is also Lipschitz, then it is called a *Lipschitz extension*. By the definition of Lipschitz continuity, each point $s \in S$ puts an upper bound and a lower bound on the unknown f so that

$$f(s) - \mathrm{d}(x, s) \leq f(x) \leq f(s) + \mathrm{d}(x, s).$$

The combination of these upper and lower bounds gives the *maximum and minimum Lipschitz extensions*, defined respectively for each $x \in X$ as

$$\check{f}_S(x) := \min_{s \in S} f(s) + \mathrm{d}(x, s) \quad \text{and} \quad \hat{f}_S(x) := \max_{s \in S} f(s) - \mathrm{d}(x, s).$$

Note that the max extension is the minimum of the upper bounds and vice versa (see also [17]).

The following theorem states that, for every Lipschitz function f that agrees with the data, the max and min Lipschitz extensions yield a barcode that is contained in $\mathsf{Bar}(f)$.

▶ **Theorem 8.** *Given $S \subseteq X$ and $f_S : S \to \mathbb{R}$, for all Lipschitz extensions f of f_S, we have $\mathsf{Bar}(\check{f}_S \geq \hat{f}_S) \sqsubseteq \mathsf{Bar}(f)$.*

Proof. It suffices to observe that for all Lipschitz extensions of f, we have $\check{f}_S \geq f \geq \hat{f}_S$. Because $\mathsf{Bar}(f) = \mathsf{Bar}(f \geq f)$, the result follows from Corollary 7. ◀

The Choice of Lipschitz Constants

It is natural to ask whether (and how) the choice of the Lipschitz constant affects this result. The previous statements have all assumed 1-Lipschitz functions. More generally, tuning the Lipschitz constant naturally leads to a filtration of the barcode itself as shown in the following.

▶ **Corollary 9.** *For any $t \in \mathbb{R}$, let $\check{f}_S^{(t)}$ and $\hat{f}_S^{(t)}$ denote the (resp.) max and min t-Lipschitz extensions. If f_S is t_0-Lipschitz and $t_0 \leq t_1 \leq t_2$, then $\mathsf{Bar}(\check{f}_S^{(t_1)} \geq \hat{f}_S^{(t_1)}) \sqsubseteq \mathsf{Bar}(\check{f}_S^{(t_2)} \geq \hat{f}_S^{(t_2)})$.*

A Sub-Barcode that is Close in Bottleneck Distance

A fundamental challenge in computing sub-barcodes from Lipschitz extensions is that it is not clear how to construct a simplicial complex that can represent the sublevel sets of both the max and min extension at the same time. So, in order to still derive guarantees, it will suffice to have approximations to these extensions. The following theorem shows that a bottleneck-close sub-barcode is possible if we have close approximations to the upper and lower bounds.

▶ **Theorem 10.** *Let $u' \geq u \geq f \geq \ell \geq \ell'$ be an ordered sequence of functions $X \to \mathbb{R}$ such that $\|\ell - \ell'\|_\infty \leq \varepsilon$ and $\|u - u'\| \leq \varepsilon$. Then, $\mathrm{d}_B\big(\mathsf{Bar}(u \geq \ell), \mathsf{Bar}(u' \geq \ell')\big) \leq \varepsilon$.*

Proof. The proof is a straightforward application of the general stability theorem for image persistence. The bounds $\|\ell - \ell'\|_\infty \leq \varepsilon$ and $\|u - u'\| \leq \varepsilon$ imply that $\ell' + \varepsilon \geq \ell \geq \ell'$ and $u' \geq u \geq u' - \varepsilon$. As all maps from ordered functions induce inclusions of sublevel sets, all required maps commute to form an interleaving in the arrow category of filtrations. This gives an interleaving in the arrow category of persistence modules which leads to an ε-interleaving of the image persistence modules. ◀

Beyond Lipschitz Functions

The careful reader may note that there is nothing particularly special about Lipschitz functions in the treatment above. One could consider other classes of functions as well. More generally, it suffices to have an upper and lower bound on the unknown function. In the Lipschitz case, the Lipschitz extensions form a lattice of hypotheses that we are considering. Recall that a poset is a lattice if every subset has a join (least upper bound) and a meet (greatest lower bound). The symbol $\vee$ is used for joins and $\wedge$ is used for meets. This is the origin of our notation $\check{f}_S$ and $\hat{f}_S$.

For a set $\mathcal{F}$ of functions, let $\bigvee \mathcal{F} := \bigvee_{f\in\mathcal{F}} f$ and $\bigwedge \mathcal{F} := \bigwedge_{f\in\mathcal{F}} f$. In particular, if $\mathcal{F}$ denotes the set of Lipschitz extensions of f_S, then $\check{f}_S = \bigvee \mathcal{F}$ and $\hat{f}_S = \bigwedge \mathcal{F}$. For other lattices of hypotheses, the same proof as for the Lipschitz case (Theorem 8) implies the following theorem.

▶ **Theorem 11.** *Given a lattice $\mathcal{F}$ of functions $X \to \mathbb{R}$, for all $f \in \mathcal{F}$, we have*

$$\mathsf{Bar}\big(\bigvee \mathcal{F} \geq \bigwedge \mathcal{F}\big) \sqsubseteq \mathsf{Bar}(f).$$

6 The Discretization Theory

The preceding sections laid out the theory of sub-barcodes for persistence modules, as well a the useful setting of Lipschitz functions. The following section shows how to approximate these barcodes.

The fundamental challenge is to provide a single discretization of space that approximates (or at least bounds) both Lipschitz extensions at the same time. We assume throughout this section that X is a convex polytope in $\mathbb{R}^d$.[2] For a finite set $S \subset X$, the *Voronoi diagram* assigns a convex polytope called the *Voronoi cell* to each point $s \in S$ defined as

$$\mathrm{Vor}_S(s) := \{x \in X \mid \mathrm{d}(x,s) = \mathrm{d}(x,S)\}.$$

The *radius* of $\mathrm{Vor}_S(s)$ is the maximum distance from s to any point in the cell: $r_s := \max_{x\in\mathrm{Vor}_S(s)} \mathrm{d}(x,s)$. More generally, for $\sigma \subseteq S$, we have a Voronoi cell

$$\mathrm{Vor}_S(\sigma) := \bigcap_{s\in\sigma} \mathrm{Vor}_S(s).$$

Dually, the *Delaunay triangulation* Del_S associated with S is the simplicial complex with vertex set S and simplices $\mathrm{Del}_S = \{\sigma \subseteq S \mid \mathrm{Vor}_S(\sigma) \neq \emptyset\}$. The Delaunay triangulation is widely used for persistent homology on Euclidean inputs.

6.1 Simple Voronoi Extensions and a Delaunay Filtration

The most direct way to use a Delaunay triangulation to estimate the persistence diagram of a set of points is to extend the function from the vertex set to the simplices, setting the value at the simplex be the maximum value at its vertices. This corresponds to the piecewise linear extension of the function to the geometric realization of the complex (see Section 2.5, Morozov [20]). Unfortunately, this can fail badly in the absence of strong sampling

[2] The convexity of X can be relaxed in many settings. Assuming X is convex simplifies our use of the Nerve theorem.

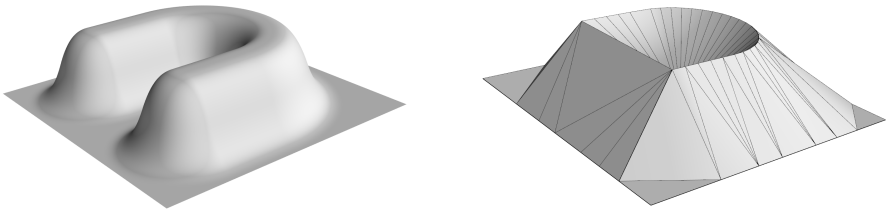


Figure 5 A function and its linear extension onto the Delaunay triangulation of a sample. The discretization introduces a spurious feature because it does not register the opening.

guarantees; Figure 5 shows an example where the piecewise linear extension introduces spurious features. Thus, the problem with some samples is not only that we can miss features, but that we may hallucinate them as well.

The key to using a discretization is to use an image filtration (see Cohen-Steiner et al. [12], Bauer [4]) capturing both an upper and lower bound. The following theorem states that we can compute a sub-barcode for any Lipschitz function by filtering the Delaunay triangulation by upper and lower bounds given on the input values and the radius of the Voronoi cells.

▶ **Theorem 12.** *Let S be a finite subset of a polytope $X \subset \mathbb{R}^d$. Let $f_S : S \to \mathbb{R}$ be a sample of an unknown Lipschitz function $f : X \to \mathbb{R}$. For each $s \in S$, let r_s denote the radius of the Voronoi cell $\mathrm{Vor}_S(s)$. Let u and ℓ be functions $\mathrm{Del}_S \to \mathbb{R}$ defined on Delaunay simplices $\sigma \in \mathrm{Del}_S$ as*

$$u(\sigma) := \max_{s\in\sigma} f_S(s) + r_s \quad \textit{and} \quad \ell(\sigma) := \max_{s\in\sigma} f_S(s) - r_s.$$

Then, $\mathsf{Bar}(u \geq l) \sqsubseteq \mathsf{Bar}(f)$.

Proof. To relate the barcodes, we will first construct functions on X that have the same barcode as the simplicial functions u and ℓ. Second, we will relate these functions to the Lipschitz extensions $\check{f}_S$ and $\hat{f}_S$.

We define piecewise constant functions on X using the Voronoi cells and f_S as

$$u^*(x) := \max_{s|x\in\mathrm{Vor}_S(s)} f_S(s) + r_s \quad \text{and} \quad \ell^*(x) := \max_{s|x\in\mathrm{Vor}_S(s)} f_S(s) - r_s.$$

By the Persistent Nerve Lemma (Chazal et al. [9], Lemma 3.4), there is a homotopy equivalence between the sublevels of u and u^* that commutes with inclusions (see also [2]). The same holds for ℓ and ℓ^*. It follows that $\mathsf{Bar}(u \geq \ell) \cong \mathsf{Bar}(u^* \geq \ell^*)$.

For any $x \in X$, there is an $s \in S$ such that $x \in \mathrm{Vor}_S(s)$ and $u^*(x) = f_S(s) + r_s$, so

$$u^*(x) = f_S(s) + r_s \geq f_S(s) + \mathrm{d}(x, s) \geq \min_{s_0\in S} f_S(s_0) + \mathrm{d}(x, s_0) = \check{f}_S(x).$$

Similarly, there is an $s \in S$ such that $x \in \mathrm{Vor}_S(s)$ and $\ell^*(x) = f_S(s) - r_s$, so

$$\ell^*(x) = f_S(s) - r_s \leq f_S(s) - \mathrm{d}(x, s) \leq \max_{s_0\in S} f_S(s_0) - \mathrm{d}(x, s_0) = \hat{f}_S(x),$$

thus $u^* \geq \check{f}_S \geq \hat{f}_S \geq \ell^*$, and Corollary 7 therefore implies

$$\mathsf{Bar}(u \geq \ell) \cong \mathsf{Bar}(u^* \geq \ell^*) \sqsubseteq \mathsf{Bar}(\check{f}_S \geq \hat{f}_S) \sqsubseteq \mathsf{Bar}(f).$$ ◀

6.2 Semi-Supervised TDA

We have assumed thus far that we have access to function values at all points in the sample. This resembles a supervised learning problem. If instead we only have function values at a subset $P \subset S$, we can still use the points of S to improve our approximation. We call this *semi-supervised TDA*.

The most common guarantees in PH are derived from interleavings, and thus yield bottleneck distance bounds on the resulting barcodes. In this section, we combine those results with the Lipschitz extension sub-barcode and the Delaunay filtration above to get a guaranteed approximation. Throughout, we assume that S is a sample of X, and that we only have function values $f_P : P \to \mathbb{R}$ for $P \subset S$. The algorithm extends $\check{f}_P$ and $\hat{f}_P$ to the points of S and then use the Delaunay filtration from the previous subsection.

In the following, we say that S is an *ε-sample* of X if every point of X is within ε of a point in S. Equivalently, the radius of every Voronoi cell $\mathrm{Vor}_S(s)$ is at most ε.

▶ **Theorem 13.** *Let $X \subset \mathbb{R}^d$ be a convex polytope, and suppose $S \subset X$ is an ε-sample. Let $f_P : P \to \mathbb{R}$ be a sample of an unknown Lipschitz function $f : X \to \mathbb{R}$ defined on a subset $P \subset S$, and let u and ℓ be functions $\mathrm{Del}_S \to \mathbb{R}$ defined on each Delaunay simplex $\sigma \subseteq S$ as*

$$u(\sigma) = \max_{s \in \sigma} \check{f}_P(s) + \varepsilon \quad \text{and} \quad \ell(\sigma) = \max_{s \in \sigma} \hat{f}_P(s) - \varepsilon.$$

Then, $\mathsf{Bar}(u \geq \ell) \sqsubseteq \mathsf{Bar}(f)$, and $\mathrm{d}_B\big(\mathsf{Bar}(u \geq \ell), \mathsf{Bar}(\check{f}_P \geq \hat{f}_P)\big) \leq \varepsilon$.

Proof. The sub-barcode relation follows from Theorem 12 and the observation that the radius $r_s \leq \varepsilon$ for all s in an ε-sample. The bottleneck bound follows from a more general theory of interleaving of image persitence modules. It depends only on the fact that the extensions u^* and ℓ^* of u and ℓ to piecewise constant functions on Voronoi cells (as in the proof of Theorem 12) satisfy $\|u^* - \check{f}_P\|_\infty \leq \varepsilon$ and $\|\ell^* - \hat{f}_P\|_\infty \leq \varepsilon$, respectively. These bounds yeild an ε-interleaving of the sublevels, and thus give an interleaving of the images.[3] ◀

6.3 Finer Approximation with Barycentric Subdivision

In general, we would like to have tight approximations to the sub-barcode of Lipschitz extensions without requiring any sampling assumptions. One way to get a better bound is to manually add points to reduce the radii of the cells. Another approach would be to consider the barycentric subdivision. This makes sense because it can be understood as assigning function values to Delaunay simplices individually rather than inducing the values from the function on the vertices. In this section, we show how to define such constructions in a way that applies to a much more general class of functions.

For each sample $s \in S$, knowing the value of $f(s)$ implies an upper bound and a lower bound on f. Thus far, these bounds came from the assumption that f is Lipschitz. Similar upper and lower bounds can be defined for other classes of functions. We only assume that these bounds are *distance monotone*. That is, the upper bound $\check{f}_s$ for a point s satisfies the condition that if $\mathrm{d}(x, s) \leq \mathrm{d}(y, s)$, then $\check{f}_s(x) \leq \check{f}_s(y)$. For the lower bound $\hat{f}_s$ at s values decrease with distance, i.e., if $\mathrm{d}(x, s) \leq \mathrm{d}(y, s)$, then $\hat{f}_s(x) \geq \hat{f}_s(y)$. We call $\{\check{f}_s\}$ and $\{\hat{f}_s\}$ a family of *pointwise distance monotone bounds*. As before, we let $\check{f}_S(x) := \min_{s \in S} \check{f}_s(x)$ and $\hat{f}_S(x) := \max_{s \in S} \hat{f}_s(x)$.

[3] A more general theorem for stabilty of image persistence can be found in the full version of this paper [10].

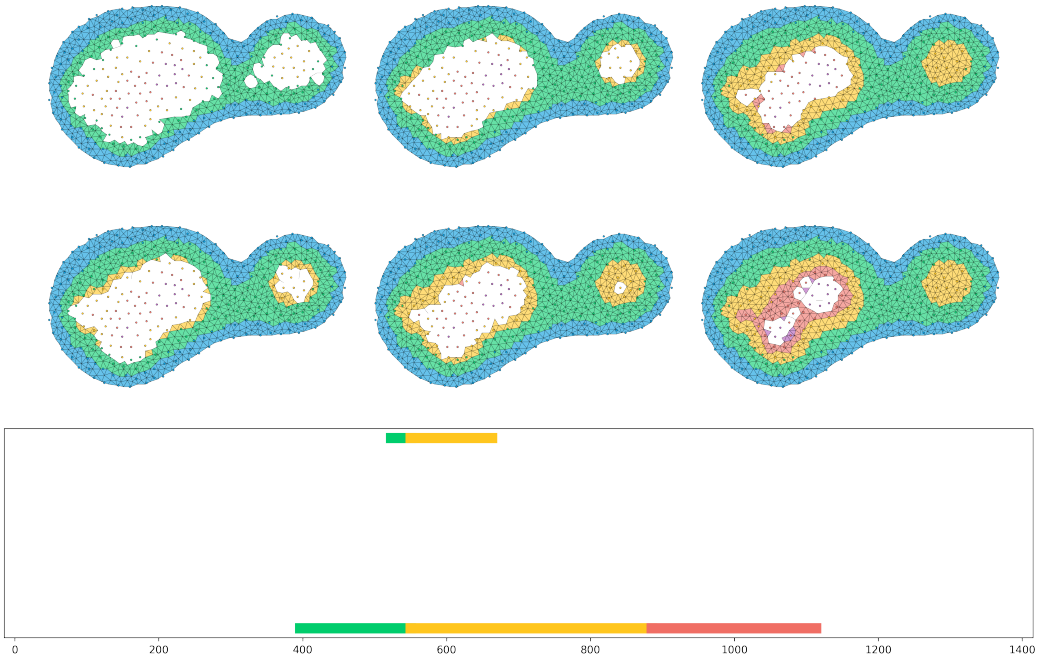


Figure 6 The top row shows the Delaunay filtration of the max Lipschitz extension. The next row shows the min extension. The barcode of the image suppresses the small spurious features and captures the two main features.

Although the term baryentric subdivision implies that subdivision happen at the barycenters of the cells, we define this operation instead as a combinatorial operation on simplicial complexes. For any simplicial complex K, we define $\mathsf{bary}K$ to be the simplicial complex with a vertex for each simplex in K and a simplex for each ordered subset of simplices (the ordering is by inclusion). For any function $g : K \to \mathbb{R}$, we can extend it to a filtration G on $\mathsf{bary}K$, as $\sigma_0 \hookrightarrow \cdots \hookrightarrow \sigma_k$ is in $G(t)$ if and only if $g(\sigma_i) \leq t$ for all $i \in \{0, \ldots, k\}$. Proof of the following theorem may be found in the full version [10].

▶ **Theorem 14.** *Let S be a finite sample of a convex polytope X, and let $\{\check{f}_s : \mathrm{Vor}_S(s) \to \mathbb{R} \mid s \in S\}$ and $\{\hat{f}_s : \mathrm{Vor}_S(s) \to \mathbb{R} \mid s \in S\}$ be families of pointwise distance-monotone bounds on some unknown function $f : X \to \mathbb{R}$. Aggregate these bounds as $\check{f}_S := \bigwedge_{s\in S} \check{f}_s$ and $\hat{f}_S := \bigvee_{s\in S} \hat{f}_s$. For $\sigma \in \mathrm{Del}_S$, let*

$$u(\sigma) := \min_{x \in \mathrm{Vor}_S(\sigma)} \check{f}_S(x) \quad \text{and} \quad \ell(\sigma) := \min_{x \in \mathrm{Vor}_S(\sigma)} \hat{f}_S(x).$$

Extend u and ℓ to $\mathsf{bary}\mathrm{Del}_S$. Then, $\mathsf{Bar}(u \geq \ell) = \mathsf{Bar}(\check{f}_S \geq \hat{f}_S) \sqsubseteq \mathsf{Bar}(f)$.

▶ **Example 15.** To apply this theorem to the special case of Lipschitz functions, the resulting filtration captures the usual Lipschitz upper and lower bounds restricted to the Voronoi cells. That is,

$$\check{f}_s : \mathrm{Vor}_S(s) \to \mathbb{R} : x \mapsto f_S(s) + \mathrm{d}(x, s) \quad \text{and} \quad \hat{f}_s : \mathrm{Vor}_S(s) \to \mathbb{R} : x \mapsto f_S(s) - \mathrm{d}(x, s).$$

Letting $\alpha(\sigma) := \min_{x\in \mathrm{Vor}_S(\sigma)} \mathrm{d}(x, \sigma)$ be the birth time of the simplex σ in the standard Delaunay filtration (also known as the α-complex filtration). Then, the corresponding filtration on $\mathsf{bary}\mathrm{Del}_S$ is induced by

$$u(\sigma) := \max_{s\in\sigma} f_S(s) + \alpha(\sigma), \quad \text{and} \quad \ell(\sigma) := \min_{s\in\sigma} f_S(s) - \alpha(\sigma).$$

7 Categorical Constructions

The following section describes how the concepts presented above fit neatly into a category-theoretic framework. It contains some technical concepts that may unfamiliar to computational geometers.

7.1 Smoothing Persistence Modules

Given a barcode B, an ε-smoothing is a new barcode defined by eliminating bars of length at most 2ε, and replacing all other bars $[b, d]$ with a $[b + \varepsilon, d - \varepsilon]$; that is, we trim ε from the beginning and the end of each bar. The idea of a smoothed barcode naturally arises in the literature on the stability of persistence modules [7, 5, 18]. It is clear from the definition that the smoothed barcode is a sub-barcode of the original. This relationship can be expressed clearly as arising from a factorization as follows.

Let δ_- and δ_+ be order-preserving maps $\mathbb{R} \to \mathbb{R}$ with the property that, for all $t \in \mathbb{R}$, we have $\delta_-(t) \leq t \leq \delta_+(t)$. Order preserving maps between posets are functors, so the pointwise order relations on δ_- and δ_+ correspond to natural transformations

$$\varepsilon_0 : \delta_- \Rightarrow 1_{\mathbb{R}} \quad \text{and} \quad \varepsilon_1 : 1_{\mathbb{R}} \Rightarrow \delta_+.$$

So, for any persistence module $M : \mathbb{R} \to \mathbf{vec}$, composition with M yields the following factorization.

$$\begin{array}{ccc} M\varepsilon_0 & \longrightarrow & M\varepsilon_1 \\ & \searrow \quad \nearrow & \\ & 1_M & \end{array} \tag{4}$$

Letting $\varepsilon = \varepsilon_1 \circ \varepsilon_0$, we can view $M\varepsilon$ (or equivalently, its image) as the smoothed persistence module. The factorization induces the sub-barcode matching in the obvious way:

$$\mathsf{Bar}(M\varepsilon) \sqsubseteq \mathsf{Bar}(M).$$

From this perspective, an interleaving of persistence modules may be viewed as a pair of compatible factorizations rather than a pair of compatible (shifted) homomorphisms. In fact, the pair (δ_-, δ_+) corresponds to an *adjunction* with counit ε_0 and unit ε_1 (see [14]).

7.2 Sub-Barcodes are Subobjects

Within the literature, there are several different categorifications of persistence barcodes (or equivalently persistence diagrams). Here, we identify a natural way to define a category of barcodes that plays nicely with our treatment of sub-barcodes.

First, recall that unlike much of the prior work, we do not regard barcodes as multisets. Instead, we define a barcode to be a function from a set to the intervals. In computer programming terms, we have an "is a" versus "has a" distinction; in the multiset view, each bar *is an* interval. In our view, each bar *has an* interval.

A standard categorical approach to construct a category $\int_{\mathbf{C}} F$ from a (pseudo)functor $\mathbf{C} \to \mathbf{Cat}$ is the Grothendieck construction. Let $F : \mathbf{C}^{\mathsf{op}} \to \mathbf{Cat}$ be a functor from a category $\mathbf{C}$ to the category $\mathbf{Cat}$ of (small) categories. For the purposes of this work, the *(covariant) Grothendieck construction* yields a category $\int_{\mathbf{C}} F$ with objects given by pairs (C, x) for each $C \in \mathbf{C}$ and $x \in F(C)$, and arrows $(f, h) : (C, x) \to (C', x')$ for each pair of arrows $f : C \to C'$ in $\mathbf{C}$ and $h : x \to F[f](x')$ in $F(C)$.

Let $[-,\mathcal{I}] : \mathbf{Set} \to \mathbf{Cat}$ be the (pseudo)functor that takes a set X and returns the category $\mathcal{I}^X = [X,\mathcal{I}]$ of functors $X \to \mathcal{I}$. Then the category $\mathbf{Bar}$ of barcodes described in Section 3 is a contravariant Grothendieck construction

$$\mathbf{Bar} = \int_{\mathbf{Set}} [-,\mathcal{I}].$$

The objects of $\mathbf{Bar}$ are pairs $(\underline{B}, B : \underline{B} \to \mathcal{I})$ – i.e. barcodes – and a morphism of $\mathbf{Bar}$ $(\underline{A}, A) \to (\underline{B}, B)$ is a function $\phi : \underline{A} \to \underline{B}$ such that for all $\alpha \in \underline{A}$, we have that $A(\alpha) \subseteq B(\phi(\alpha))$. To see this as a Grothendieck construction, it suffices to observe that the barwise inclusion ordering is a natural transformation $A \Rightarrow B\phi$ in $[\underline{A},\mathcal{I}]$.

The condition on morphisms ensures that a bar α can only map to a bar β when $A(\alpha) \subseteq B(\beta)$. Thus, a injective morphism is exactly a sub-barcode matching. These injective morphisms are the monomorphsisms in this category and thus sub-barcodes are subobjects in this category.

▶ Remark 16. The category $\mathbf{Bar}$ defined above is also known as the category of $\mathcal{I}$-fuzzy sets [15], and the construction of a category of fuzzy sets using the Grothendieck construction also noted by Jardine [16]. It is also possible to replace $\mathbf{Set}$ with $\mathbf{Mch}$, the category of sets with partial injective maps (i.e., matchings), to construct barcodes as $\int_{\mathbf{Mch}}[-,\mathcal{I}]$, or more generally, to replace $\mathbf{Set}$ with the category $\mathbf{Set}_+ \supset \mathbf{Mch}$ of sets and *partial functions*. In both cases, the monomorphisms are injective sub-barcode matchings that yield sub-barcode relations.

7.3 Ranks via Presheaves

The rank functor was previously constructed directly from a persistence module, or more generally, a factorization of persistence module homomorphisms. In this section, we show how the rank functor can also be constructed from a barcode. This is already well-known; the novelty here is that we show how the rank functor can be factored through the category of barcodes. This gives the more abstract proof that sub-barcodes are more discriminating than ranks.

For any category $\mathbf{C}$, there is a functor $L : \int_{\mathbf{Set}}[-,\mathbf{C}] \to [\mathbf{C}^{\mathsf{op}},\mathbf{Set}]$ given for $A : \underline{A} \to \mathbf{C}$ by

$$LA(c) = \{a \in \underline{A} \mid \exists c \to A(a)\}.$$

The morphisms $L(A)(c \to c')$ are inclusions $L(A)(c') \subseteq L(A)(c)$. For $(f,\varepsilon) : A \to B$ in $\int_{\mathbf{Set}}[-,\mathbf{C}]$, it is easy to check that f gives a natural transformation $LA \Rightarrow LB$.

For the special case where $\mathbf{C} = \mathcal{I}$, this is a functor from barcodes to presheaves of intervals. The intuitive meaning is that, for a barcode B, the presheaf LB maps an interval J to the set of bars in B that contain all of J. All morphisms in the image of L are inclusions, and so, for $J \supseteq I$, we have $LB(J) \subseteq LB(I)$ and thus $|LB(J)| \leq |LB(I)|$. Here the vertical bars indicate set cardinality which is a functor card from the poset of finite sets with inclusions to the poset $\mathbb{N}$. Thus, we have a functor $\mathsf{card} \circ L : \mathbf{Bar} \to \mathbf{Set}^{\mathcal{I}^{\mathsf{op}}}$.

It is now an easy exercise to show that $\mathrm{rank} = \mathsf{card} \circ L \circ \mathsf{Bar}$. All morphisms in the functor category $[\mathcal{I}^{op},\mathbb{N}] = \mathbb{N}^{\mathcal{I}^{\mathsf{op}}}$ are monomorphisms, so clearly a sub-barcode relation (a monomorphism of barcodes) yields a subrank relation (a monomorphism of rank invariants). The other direction doesn't hold. The most natural way to define a barcode from the rank invariant does not give a functor to barcodes. The example is given in the proof of Theorem 6.

▶ Remark 17. Barr [1] showed that, when the indexing poset L is a Heyting algebra, the category of L-fuzzy sets is equivalent to a category of sheaves of monomorphisms. Following this result, many of the recent applications of fuzzy sets [21, 19] have defined fuzzy sets

directly as functors $L \to \mathbf{Set}$. Unfortunately, because the poset of intervals $\mathcal{I}$ is not a Heyting algebra, this equivalence cannot be applied directly to the category $\mathbf{Bar}$. On the other hand, the category of intervals with the product or Egli-Milner ordering is a Heyting algebra. Extending Barr's equivalence to a category of barcodes and partial functions is the subject of future work.

References

1. Michael Barr. Fuzzy set theory and topos theory. *Canadian Mathematical Bulletin*, 29(4):501–508, 1986.
2. 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.
3. Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, page Vol. 6 No. 2 (2015): Special issue of Selected Papers from SoCG 2014, 2015. doi:10.20382/JOCG.V6I2A9.
4. Ulrich Bauer and Maximilian Schmahl. Efficient computation of image persistence. *arXiv preprint*, 2022. arXiv:2201.04170.
5. Peter Bubenik, Vin De Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015. doi:10.1007/S10208-014-9229-5.
6. 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.
7. 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*, SCG '09, pages 237–246, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542407.
8. 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.
9. Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in euclidean spaces. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, pages 232–241, New York, NY, USA, 2008. ACM. doi:10.1145/1377676.1377719.
10. Oliver A. Chubet, Kirk P. Gardner, and Donald R. Sheehy. A theory of sub-barcodes, 2022. doi:10.48550/arXiv.2206.10504.
11. David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l p-stable persistence. *Foundations of computational mathematics*, 10(2):127–139, 2010. doi:10.1007/S10208-010-9060-6.
12. David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 2009.
13. Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, November 2002. doi:10.1007/s00454-002-2885-2.
14. Kirk Patrick Gardner. *Verified Topological Data Analysis and a Theory of Sub-Barcodes*. PhD thesis, North Carolina State University, 2022.
15. Joseph A Goguen. L-fuzzy sets. *Journal of mathematical analysis and applications*, 18(1):145–174, 1967.
16. John F. Jardine. Fuzzy sets and presheaves. *Compositionality*, 1:3, December 2019. doi:10.32408/compositionality-1-3.

17 F William Lawvere. Metric spaces, generalized logic, and closed categories. *Rendiconti del seminario matématico e fisico di Milano*, 43:135–166, 1973.

18 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. doi:10.1007/S10208-015-9255-Y.

19 Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint*, 2018. arXiv:1802.03426.

20 Dmitriy Morozov. *Homological Illusions of Persistence and Stability*. PhD thesis, Duke University, 2008.

21 David I Spivak. Metric realization of fuzzy simplicial sets. Self-published Notes.

Strongly Sublinear Separators and Bounded Asymptotic Dimension for Sphere Intersection Graphs

James Davies
University of Cambridge, UK

Agelos Georgakopoulos
University of Warwick, UK

Meike Hatzel
Institute for Basic Science (IBS), Daejeon, South Korea

Rose McCarty
Georgia Institute of Technology, Atlanta, GA, USA

Abstract

In this paper, we consider the class $\mathcal{C}^d$ of sphere intersection graphs in $\mathbb{R}^d$ for $d \geq 2$. We show that for each integer t, the class of all graphs in $\mathcal{C}^d$ that exclude $K_{t,t}$ as a subgraph has strongly sublinear separators. We also prove that $\mathcal{C}^d$ has asymptotic dimension at most $2d + 2$.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Intersection graphs, strongly sublinear separators, asymptotic dimension

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.36

Related Version *Full Version*: https://arxiv.org/pdf/2504.00932

Funding *Agelos Georgakopoulos*: Supported by EPSRC grants EP/V048821/1 and EP/V009044/1.
Meike Hatzel: The research was supported by the Institute for Basic Science (IBS-R029-C1).
Rose McCarty: Supported by the National Science Foundation under Grant No. DMS-2202961.

Acknowledgements This research was initiated during the Banff workshop "New Perspectives in Colouring and Structure (24w5272)". We thank Alex Scott for the construction in Observation 8. We thank Louis Esperet for several helpful discussions.

1 Introduction

We write $\mathcal{C}^d$ for the class of intersection graphs of spheres in $\mathbb{R}^d$ for $d \geq 2$. The *sphere dimension* of a graph G is the smallest integer so that G is contained in $\mathcal{C}^d$. This notion of dimension is analogous to the *boxicity*, *cubicity*, or *sphericity*[1] of a graph. These "dimensionality" parameters were introduced by Roberts [47], further studied by Maehara [42], and have since become a topic of much research; see for instance [7, 23, 49, 52]. We formulate an equivalent definition of $\mathcal{C}^d$ utilising an appealing application of stereographic projection, allowing us to think of $\mathcal{C}^d$ as a higher-dimensional generalisation of the class of circle graphs. (See [5, 29, 45] for some classical work on circle graphs and [16, 48] for surveys.) It will, therefore, become natural to let $\mathcal{C}^1$ denote the class of circle graphs, which then have sphere dimension 1. Thus, $\mathcal{C}^d$ simultaneously generalises various well-studied graph classes.

Of particular note, graphs with low dimension often satisfy strong separator theorems along the lines of Lipton and Tarjan's [38] famous separator theorem for planar graphs. Formally, a *balanced separator* in an n-vertex graph G is a set $S \subseteq V(G)$ so that each

[1] We note that sphericity is more restrictive than sphere dimension as it only considers unit spheres.

© James Davies, Agelos Georgakopoulos, Meike Hatzel, and Rose McCarty;
licensed under Creative Commons License CC-BY 4.0

component of $G - S$ has at most $\frac{2}{3}n$ many vertices[2]. Lipton and Tarjan showed that every n-vertex planar graph has a balanced separator of size $\mathcal{O}(\sqrt{n})$. These separator theorems have many applications as they can be combined with divide-and-conquer techniques [9, 12, 39, 50].

Planar graphs are intersection graphs of internally disjoint circles by the Circle Packing Theorem of Andreev [1], Koebe [35], and Thurston [51]. So Miller, Teng, Thurston, and Vavasis [44] generalised the planar separator theorem by proving that the intersection graph of any sphere packing of size n in $\mathbb{R}^d$ has a balanced separator of size $\mathcal{O}(n^{1-\frac{1}{d}})$. In fact, their results hold for any collection of balls of bounded ply; the *ply* of a collection of geometric objects is the maximum number of objects with a non-empty common intersection. The ply of a collection of balls in $\mathbb{R}^d$ is closely related to the clique number of its intersection graph.

Many other geometric generalisations of Lipton and Tarjan's theorem have been proven, including some very recently [13, 14, 17, 30, 34, 50]. Some of these results, such as [13, 14, 34], consider relaxations of separators and allow for more general geometric objects that way. However, if we keep the original notion of balanced separators, then two main improvements have been made. First, instead of considering balls, we can consider compact convex subsets of $\mathbb{R}^d$ of bounded aspect ratio (the ratio of their diameter to their height) by [30, Lemma 2.21]. Second, it is enough that for any pair of objects under consideration, we can "move one of them around inside the other to roughly trace its boundary"; see [17, Theorem 2].

All the results discussed above are for convex subsets of $\mathbb{R}^d$. We prove a new separator theorem for the intersection graphs of spheres in $\mathbb{R}^d$. Instead of bounding the ply, there is a different natural obstruction which we must bound the size of: the class of all complete bipartite graphs has bounded sphere dimension, but does not admit small balanced separators. Our first result is that this is the only obstruction. For any integers $t, d \geq 2$, we write $\mathcal{C}_t^d$ for the class of all graphs that have sphere dimension at most d and do not contain the complete bipartite graph $K_{t,t}$ as a subgraph. (This subgraph need not be induced.)

▶ **Theorem 1.** *For any integers $t, d \geq 2$, the class $\mathcal{C}_t^d$ has strongly sublinear separators. More precisely, every n-vertex graph $G \in \mathcal{C}_t^d$ has a balanced separator of size $\widetilde{\mathcal{O}}_d(t^{10}n^{1-\frac{1}{2d+8}})$.*

To the best of our knowledge, this is the first high-dimensional separator theorem that applies to classes that contain large induced complete bipartite graphs. In two dimensions, Lee [37] proved that every string graph with m edges has a balanced separator of size $\mathcal{O}(\sqrt{m})$. This implies a separator theorem for string graphs with a forbidden $K_{t,t}$-subgraph by the Kővári-Sós-Turán Theorem [36]. This theorem of Lee was conjectured by Fox and Pach [24], and a slightly weaker version with an extra $\log(m)$ factor was proven by Matoušek [43]. As far as we know, the same bound could hold for graphs with sphere dimension at most d; maybe every graph with m edges and sphere dimension d has a balanced separator of size $\mathcal{O}_d(\sqrt{m})$. If so, a theorem of Fox, Pach, Sheffer, Suk, and Zahl [25] about semi-algebraic graphs could then be used to obtain a strong separator theorem for $\mathcal{C}_t^d$.

Next, we turn our attention to the asymptotic dimension of our graphs. Gromov [28] introduced the notion of asymptotic dimension of a metric space as a coarse version of the classical topological dimension. This was part of his coarse-geometric approach, which has tremendously impacted geometric group theory. For instance, Yu [53] showed that (any Cayley graph of) every finitely generated group G with finite asymptotic dimension embeds coarsely into Hilbert space, and therefore G satisfies a seminal conjecture of Novikov. Roughly speaking, asymptotic dimension corresponds to the number of colours needed to colour a graph, or family of graphs, with certain restrictions on the diameters and spacing of monochromatic components; the precise definition is given in Section 2.

[2] The exact constant does not matter much; we could replace $\frac{2}{3}$ by any constant $c \in (0, 1)$.

Although asymptotic dimension was originally introduced to study spaces of infinite diameter, the notion can be applied to classes of finite graphs, and it is gaining popularity in this setting [4, 15, 17, 26, 27, 33, 40]. This popularity is partly motivated by a connection to sparse partition schemes in algorithm development, as we discuss in Section 6. It is also motivated by connections to graph colouring, as discussed in [20].

Thus, understanding how the asymptotic dimension of a graph class interacts with other geometric properties is important. In this spirit, Bonamy et al. [4] proved that planar graphs have asymptotic dimension at most 2, and asked if every class with strongly sublinear separators has bounded asymptotic dimension [4]. (We note that this question was phrased in terms of classes of polynomial expansion; however, a class has polynomial expansion if and only if it has strongly sublinear separators [18, 46]. Formally, a class has *strongly sublinear separators* if there exists $\delta > 0$ so that every n-vertex graph that is an induced subgraph of some graph in the class admits a balanced separator of size $\mathcal{O}(n^{1-\delta})$.) We prove that the answer is positive for graphs of bounded sphere dimension.

▶ **Theorem 2.** *For any integer $d \geq 1$, the class of graphs of sphere dimension at most d has asymptotic dimension at most $2d + 2$.*

This theorem generalises and also uses a theorem of Dvořák and Norin [19], at the cost of increasing the dimension by 1. In fact, we prove a more general result by considering intersection graphs of shapes other than spheres (Theorem 18).

Paper structure. We introduce some preliminaries in Section 2. In Section 3, we discuss basic properties of sphere dimension, particularly the connection to circle graphs. In Section 4, we prove our main result that $\mathcal{C}_t^d$ has strongly sublinear separators. We prove that $\mathcal{C}^d$ has bounded asymptotic dimension in Section 5. We conclude with some open problems in Section 6.

2 Preliminaries

All graphs in this paper can be assumed to be finite and simple, although some of our results extend verbatim to the infinite case. We use $\widetilde{\mathcal{O}}_d$ for the version of big-$\mathcal{O}$ notation where we view d as being fixed and suppress polylogarithmic factors. (In fact it will be clear from our proof of Theorem 1 that we only suppress a single $\log n$ factor and a constant factor of the form c^d for c a universal constant.)

Geometric intersection graphs. Let $\mathcal{G}$ be a class of geometric objects in a given space, for example, unit squares in the plane. Then an *intersection graph of* $\mathcal{G}$ is a graph G whose vertices can be identified with objects in $\mathcal{G}$ such that there is an edge between two vertices in G if and only if the two corresponding objects intersect. Similarly, we write $G(\mathcal{G})$ for the graph with vertex-set $\mathcal{G}$ where two vertices are adjacent if they have a non-empty intersection. The *ply* of $\mathcal{G}$ is the maximum size of a set of objects in $\mathcal{G}$ with a non-empty common intersection.

Shallow minors. Our proof of Theorem 1 relies on a theorem of Plotkin, Rao, and Smith [46] and Dvořák and Norin [18] which characterises classes that admit strongly sublinear separators in terms of forbidden shallow minors. We give some relevant definitions.

Given a positive integer r, we say that a graph H is an *r-shallow minor* of a graph G if there is a function $\varphi : V(H) \to 2^{V(G)}$ such that:

- for every $x \in V(H)$, the subgraph of G induced by $\varphi(x)$ contains a rooted spanning tree of height at most r,
- for all distinct $x, y \in V(H)$, the sets $\varphi(x)$ and $\varphi(y)$ are disjoint, and
- for every $xy \in E(H)$, there is an edge in G with one end in $\varphi(x)$ and the other in $\varphi(y)$.

We call φ an *r-shallow minor model* of H in G. For $v \in V(H)$, we call $\varphi(v)$ the *bag of v*.

Strongly sublinear separators. Recall that a *balanced separator* in an n-vertex graph G is a set $S \subseteq V(G)$ so that each component of $G - S$ has at most $\frac{2}{3}n$ vertices. A graph class $\mathcal{C}$ has *strongly sublinear separators* if there exist $\varepsilon > 0$ so that for any n-vertex graph H which is an induced subgraph of a graph in $\mathcal{C}$, the graph H has a balanced separator of size $\mathcal{O}(n^{1-\varepsilon})$. (The big-$\mathcal{O}$ notation can hide a constant factor that depends on $\mathcal{C}$.) Note that to obtain strongly sublinear separators, it suffices to show that there exists $\delta > 0$ so that there are balanced separators of size $\widetilde{\mathcal{O}}(n^{1-\delta})$. In order to provide this, we show that there exists $\delta > 0$ so that there are separators of size $\widetilde{\mathcal{O}}(n^{1-\delta})$. Plotkin, Rao, and Smith [46] proved that any graph with a forbidden shallow minor has a small balanced separator.

▶ **Theorem 3** (Plotkin, Rao, and Smith [46]). *For any $r, h \in \mathbb{N}$, any n-vertex graph without an r-shallow K_h minor has a balanced separator of size $\mathcal{O}(rh^2 \log n + \frac{n}{r})$.*

We apply this theorem with r being a small power of n and h being a polynomial in r. With this choice of parameters, Theorem 3 yields a sufficiently small balanced separator.

Strong colouring number. Let G be a graph, $r \in \mathbb{N}$, and $\leq$ be an ordering of the vertices of G. For a vertex $v \in V(G)$, we call a vertex $u \in V(G)$ *strongly r-reachable from v* if $u \leq v$ and there exists a v-u-path P of length at most r in G such that u is the only vertex in P that is smaller than v with respect to $\leq$. The *r-width* of $\leq$ is the maximum over all $v \in V(G)$ of the number of vertices which are strongly r-reachable from v. Finally, the *r-strong colouring number* of a graph G, denoted $\mathsf{scol}_r(G)$, is the minimum r-width over all possible vertex orderings of G. We require the observation that intersection graphs of balls of bounded ply have bounded strong colouring numbers.

▶ **Lemma 4** ([21, Lemma 1]). *Let $\mathcal{B}$ be a finite collection of balls in $\mathbb{R}^d$ of ply at most t. Then the intersection graph $G(\mathcal{B})$ has r-strong colouring number at most $t(2r+2)^d$.*

We also use the following well-known observation that graphs with large shallow clique minors have large strong colouring numbers.

▶ **Observation 5.** *If a graph G contains an r-shallow K_h-minor, then $\mathsf{scol}_{4r+1}(G) \geq h - 1$.*

Asymptotic and Assouad–Nagata dimension. Let G be a graph, and let $\mathsf{dist}_G(\cdot,\cdot)$ denote its usual graph metric. If the graph is clear from the context, we drop the index. Let $\mathcal{U}$ be a family of subsets of $V(G)$. We say that $\mathcal{U}$ is *D-bounded* if each set $U \in \mathcal{U}$ has diameter at most D. We say that $\mathcal{U}$ is *r-disjoint* if for any a, b belonging to different elements of $\mathcal{U}$ we have $\mathsf{dist}(a, b) > r$.

We say that $D : \mathbb{R}_+ \to \mathbb{R}_+$ is an *n-dimensional control function* for a graph class $\mathcal{C}$, if for any $G \in \mathcal{C}$ and $r > 0$, $V(G)$ has a cover $\mathcal{U} = \bigcup_{i=1}^{n+1} \mathcal{U}_i$ such that each $\mathcal{U}_i$ is r-disjoint and each element of $\mathcal{U}$ is $D(r)$-bounded. The *asymptotic dimension* of $\mathcal{C}$ denoted by $\mathsf{asdim}(\mathcal{C})$, is the least integer n such that $\mathcal{C}$ has an n-dimensional control function. If no such integer n exists, then $\mathsf{asdim}(\mathcal{C})$ is infinite. The *Assouad–Nagata dimension* of $\mathcal{C}$, denoted by $\mathsf{ANdim}(\mathcal{C})$, is defined similarly, except that we insist that $D(r) \in \mathcal{O}(r)$.

3 Sphere graphs and sphere dimension

In this section, we show that the graph classes $\mathcal{C}^d$ can be thought of as a higher-dimensional analogue of the well-studied class of circle graphs. Recall that a *circle graph* is an intersection graph of a family of chords of $\mathbb{S}^1$. We generalise this definition as follows. A *d-sphere graph* is an intersection graph of a subfamily of

$$\mathcal{H}_d := \{\mathbb{D}^d \cap H \mid H \text{ is a hyperplane of } \mathbb{R}^{d+1}\},$$

where the d-disc $\mathbb{D}^d$ is the set $\{x \in \mathbb{R}^{d+1} \mid |x,0| \leq 1\}$, and $\mathbb{S}^d$ is its boundary. In particular, $\mathcal{H}_1$ consists of all chords of the circle $\mathbb{S}^1$, and $\mathcal{H}_2$ consists of the flat discs with boundary on $\mathbb{S}^2$. Thus, 1-sphere graphs are exactly the circle graphs, which by definition are the graphs in $\mathcal{C}^1$. We observe that the class of d-sphere graphs coincides with $\mathcal{C}^d$:

▶ **Observation 6.** *For every $d \geq 1$, a countable graph G is a d-sphere graph if and only if it belongs to $\mathcal{C}^d$.*

Proof. For $d = 1$ this is so by definition. For $d \geq 2$, suppose $\Theta = \{H_v \mid v \in V(G)\}$ is a d-sphere representation of G, i.e. a family of elements of $\mathcal{H}_d$ witnessing that G is a d-sphere graph. Assume that no H_v contains the *north pole* $\{0,\dots,0,1\}$ of $\mathbb{S}^d$, which we can since G is countable (which is a much stronger assumption than we actually need). Notice that $H_v \cap \mathbb{S}^d$ is homeomorphic to $\mathbb{S}^{d-1}$ for every v. Letting f be the stereographic projection of $\mathbb{S}^d$ onto $\mathbb{R}^d$, we observe that $f(H_v \cap \mathbb{S}^d) =: H'_v$ is also homeomorphic to $\mathbb{S}^{d-1}$, where we use the well-known property of stereographic projection that it preserves spheres not containing the north pole [44, Lemma 4.2]. Then G is isomorphic to the intersection graph of the family $\Theta' = \{H'_v \mid v \in V(G)\}$ by construction.

Conversely, we use f^{-1} to stereographically project a representation of G as an intersection graph of spheres in $\mathbb{R}^d$ (avoiding the origin) to a d-sphere graph representation. ◀

We now give some other observations about sphere graphs and sphere dimension. As a warm-up, we observe that if G is planar, then it has sphere dimension 2. This is because the Circle Packing Theorem [35] asserts that every finite[3] planar graph admits a sphere packing in $\mathbb{R}^2$. (A *sphere packing* of a graph G is an arrangement of closed euclidean spheres $\{S_v : v \in V(G)\}$ in $\mathbb{R}^d$ or $\mathbb{S}^d$ such that $S_v \cap S_w$ is empty whenever $vw \notin E(G)$ and $S_v \cap S_w$ is a single point whenever $vw \in E(G)$. In other words, G is the tangency graph of the family $\{S_v : v \in V(G)\}$.) It is known that every finite graph G admits a sphere packing in $\mathbb{R}^d$ for $d \leq |V(G)| - 1$ [22], and so $G \in \mathcal{C}^{|V(G)|-1}$. This proves the following observation.

▶ **Observation 7.** *Every graph G with $|V(G)| \geq 3$ has sphere dimension at most $|V(G)| - 1$.*

Conversely, we prove the following.

▶ **Observation 8.** *For every $d \in \mathbb{N}$, there is a finite graph which is not a d-sphere graph.*

Proof. We may assume that $d \geq 2$ since it is well known that not every graph is a circle graph; see [5]. Let n be an integer; we think of n as being large in comparison to d.

Let G_n be a graph of minimum degree n that does not contain a triangle as a subgraph, and such that G_n remains connected after removing the closed neighbourhood $N[y] := N(y) \cup \{y\}$ of any $y \in V(G_n)$; for example, G_n could be a portion of the n-dimensional cubic lattice.

[3] More generally, this applies to any connected, locally finite, planar graph; see [31] and references therein.

Suppose G_n can be realised as a d-sphere graph, and therefore as an intersection graph of spheres $\{S_v \mid v \in V(G_n)\}$ in $\mathbb{R}^d$ by Observation 6. Pick a sphere of minimum radius among the S_v. Then, the spheres representing the neighbours of v intersect S_v and are mutually disjoint since G_n is triangle-free. Moreover, there is no triple S_x, S_y, S_z of such spheres such that S_y lies in the interior of S_x and S_z lies in the interior of S_y because $N[y]$ would separate x from z in this case. Thus, after deleting any such spheres lying in the interior of others, we are left with at least $n/2$ spheres with disjoint interiors intersecting S_v, each of radius at least that of S_v. By a volume argument, the number of such spheres is bounded by a function of d. Thus G_n is not a d-sphere graph when n is large enough. ◀

It is possible to strengthen Observation 8 by providing cubic graphs of arbitrarily high sphere dimension. Indeed, this is a corollary of Theorem 1, as we now discuss. Let $\{G_n^d : n \in \mathbb{N}\}$ be a family of cubic expander graphs with $\lim_{n\to\infty} |V(G_n^d)| = \infty$. Then, since expanders do not have small separators, at most finitely many members of this family can have sphere dimension at most d by Theorem 1 and by Lee's [37] separator theorem for circle graphs. Alternatively, we can let $\{G_n^d : n \in \mathbb{N}\}$ be a family of finite cubic graphs of asymptotic dimension more than $2d+1$; for instance, such graphs can be obtained from a portion of the $(2d+2)$–dimensional hypercubic lattice by replacing each vertex with a subcubic tree. Then at most finitely many members of this family can have sphere dimension at most d by the result of [19] that our Theorem 18 generalises.

4 Strongly sublinear separators

In this section, we prove Theorem 1, the separator theorem for the class $\mathcal{C}_t^d$ of all graphs which have sphere dimension at most d and do not have $K_{t,t}$ as a subgraph.

We need some additional notation. Given a sphere S in $\mathbb{R}^d$, we write $\mathsf{ball}(S)$ for the (closed) ball with boundary S. We say that a sphere S' *contains* a different sphere S if $\mathsf{ball}(S')$ contains $\mathsf{ball}(S)$. Given a collection $\mathcal{S}$ of spheres in $\mathbb{R}^d$, we say that a sphere $S \in \mathcal{S}$ is *maximal* if there is no $S' \in \mathcal{S} \setminus \{S\}$ such that S' contains S. Likewise, a sphere $S \in \mathcal{S}$ is *minimal* if there is no $S' \in \mathcal{S} \setminus \{S\}$ such that S contains S'. We call $\mathcal{S}$ *nested* if for any two spheres in $\mathcal{S}$, one contains the other. In this section, every collection of spheres is finite.

We start by noting that excluding $K_{t,t}$ provides a bound on the number of short paths between the "inside" and the "outside" of a large set of nested spheres.

▶ **Lemma 9.** *Let $r, t \in \mathbb{N}$, let $\mathcal{S}$ be a collection of spheres in $\mathbb{R}^d$, and let $\mathcal{N} \subseteq \mathcal{S}$ be a set of $t(r+1)$-many nested spheres. Suppose that the intersection graph $G(\mathcal{S} \setminus \mathcal{N})$ contains a collection $\mathcal{P}$ of $t^2(r+1)$-many pairwise vertex-disjoint paths such that for each path $P \in \mathcal{P}$,*

- *one end of P intersects the minimal sphere in $\mathcal{N}$,*
- *the other end of P intersects the maximal sphere in $\mathcal{N}$, and*
- *P has length at most r.*

Then, $G(\mathcal{S})$ contains a $K_{t,t}$ as a subgraph.

Proof. We sort the vertices in $\mathcal{N}$ by containment and consider $\leq$ to be this ordering. For every path $P \in \mathcal{P}$ and every vertex $x \in V(P)$, the neighbourhood of x in $\mathcal{N}$ is an interval with respect to $\leq$. We denote this interval by I_x. The first two conditions of the lemma guarantee that every sphere in $\mathcal{N}$ is a neighbour of at least one vertex in P. Thus, since $|\mathcal{N}| = t(r+1)$ and P has length at most r, every $P \in \mathcal{P}$ contains a vertex x_P with $|I_{x_P}| \geq t$. Also, there are at most $t(r+1)$ choices for the first vertex (according to $\leq$) inside I_{x_P}. Therefore, there are t vertices of the form x_P (on different paths) such that their intervals all start in the same vertex. These t vertices, plus the first t vertices in their intervals, yield the desired $K_{t,t}$-subgraph. ◀

Using this insight, we next prove that "minimal" graphs that contain a certain shallow clique minor cannot be represented by spheres with too much nesting.

▶ **Lemma 10.** *Let $t, d, r, h \in \mathbb{N}$ with $t, d \geq 2$, and suppose that $G \in \mathcal{C}_t^d$ has as few vertices as possible subject to containing an r-shallow K_h-minor. Let $\mathcal{S}$ be a collection of spheres in $\mathbb{R}^d$ such that $G = G(\mathcal{S})$. Then every nested subset of $\mathcal{S}$ has size at most $36t^4(3r+1)^3$.*

Proof. Suppose towards a contradiction that there is a nested set $\mathcal{N} \subseteq \mathcal{S}$ of size at least $36t^4(3r+1)^3$. Let φ be an r-shallow minor model of K_h in G. For each vertex $x \in V(K_h)$, we fix some rooted spanning tree T_x of the subgraph of G induced on $\varphi(x)$ of height at most r. Note that, as G is minimal with respect to containing an r-shallow K_h-minor, every vertex of G is in a bag of φ. Similarly, for every vertex $x \in V(K_h)$, every leaf of T_x has an edge to some other bag which has no other neighbours in T_x. We now break into cases based on whether many vertices in $\mathcal{N}$ lie in the same bag or in different bags.

First, consider the case that at least $6t^2(3r+1)^2$ vertices of $\mathcal{N}$ lie in a single bag, say the bag of $x \in V(K_h)$. As T_x has height at most r, at least a $\frac{1}{(r+1)}$ proportion of these elements all have the same distance to the root of T_x. Thus, there exists a set $\mathcal{X} \subseteq \mathcal{N}$ of size at least $6t^2(3r+1)$ such that all vertices in $\mathcal{X}$ are not only in T_x but also have the same distance to the root of T_x. So, no vertex in $\mathcal{X}$ lies on the path between the root of T_x and another vertex in $\mathcal{X}$. For each $X \in \mathcal{X}$, we fix an arbitrary leaf vertex $\ell(X)$ of T_x that is a descendent of X in T_x. (It is possible that $\ell(X) = X$.) Let us denote the path of T_x joining X and $\ell(X)$ by $P(X)$. Since all of the vertices in $\mathcal{X}$ have the same distance to the root, the paths $\{P(X) \mid X \in \mathcal{X}\}$ are pairwise vertex-disjoint. Additionally, for each of these leaves $\ell(X)$, there is a vertex $y(X) \in V(K_h)$ such the bag of $y(X)$ has an edge to $\ell(X)$ and does not have an edge to any other vertex in $\varphi(x)$. Thus, since the leaves $\{\ell(X) \mid X \in \mathcal{X}\}$ are all distinct, the vertices $\{y(X) \mid X \in \mathcal{X}\}$ are also all distinct.

Now we sort the elements of $\mathcal{X}$ by containment as $X_1, \ldots, X_{|\mathcal{X}|}$. We pair up the first $t^2(6r+2)$ elements of $\mathcal{X}$ with the last $t^2(6r+2)$ elements of $\mathcal{X}$. So we consider the pairs $(X_1, X_{|\mathcal{X}|})$, $(X_2, X_{|\mathcal{X}|-1})$, and so on. As K_h is the complete graph, for every pair $(X_i, X_{|\mathcal{X}|-i+1})$ there is a path between X_i and $X_{|\mathcal{X}|-i+1}$ whose vertex-set is contained in $P(X_i) \cup \varphi(y(X_i)) \cup \varphi(y(X_{|\mathcal{X}|-i+1})) \cup P(X_{|\mathcal{X}|-i+1})$. The two paths $P(X_i)$ and $P(X_{|\mathcal{X}|-i+1})$ have length at most $r-1$ since no vertex in $\mathcal{X}$ is the root of T_x. In each of the two bags $\varphi(y(X_i))$ and $\varphi(y(X_{|\mathcal{X}|-i+1}))$, we can choose paths of length at most $2r$. Finally, we need 3 edges to join the parts together. So overall, this path between X_i and $X_{|\mathcal{X}|-i+1}$ has length at most $2(r-1)+2(2r)+3 = 6r+1$ (see Figure 1). Note that $|\mathcal{X}| \geq 6t^2(3r+1) = 3t^2(6r+2) \geq 2t^2(6r+2) + t(6r+2)$. So at least $t(6r+2)$ spheres lie between any two paired spheres in $\mathcal{X}$. Thus, by Lemma 9, G contains $K_{t,t}$ as a subgraph, a contradiction.

Second, consider the case that there exists a set $\mathcal{X} \subseteq \mathcal{N}$ of size at least $6t^2(3r+1)$ so that every pair of vertices in $\mathcal{X}$ lie in different bags of φ. So for every $X \in \mathcal{X}$, there exists a vertex $y(X) \in V(K_h)$ such that X lies in $\varphi(y(X))$. Again, we sort the elements of $\mathcal{X}$ by containment as $X_1, \ldots, X_{|\mathcal{X}|}$ and pair the first and last elements. This time, we pair the first $t(4r+2)$ elements of $\mathcal{X}$ with the last $t(4r+2)$ elements of $\mathcal{X}$. So for $1 \leq i \leq t(4r+2)$, the sphere X_i is paired to the sphere $X_{|\mathcal{X}|-i+1}$. As K_h is the complete graph, there exists a path between X_i and $X_{|\mathcal{X}|-i+1}$ of length at most $4r+1$ in the subgraph of G induced by $\varphi(y(X_i)) \cup \varphi(y(X_{|\mathcal{X}|-i+1}))$ (see Figure 2). Additionally, $|\mathcal{X}| \geq 6t^2(3r+1) = 3t^2(6r+2) \geq 2t(4r+2) + t^2(4r+2)$. Thus, at least $t^2(4r+2)$ spheres lie between any two paired spheres in $\mathcal{X}$. Thus, again by Lemma 9, G contains $K_{t,t}$ as a subgraph, a contradiction.

One of the two cases must occur, and so this completes the proof of Lemma 10. ◀

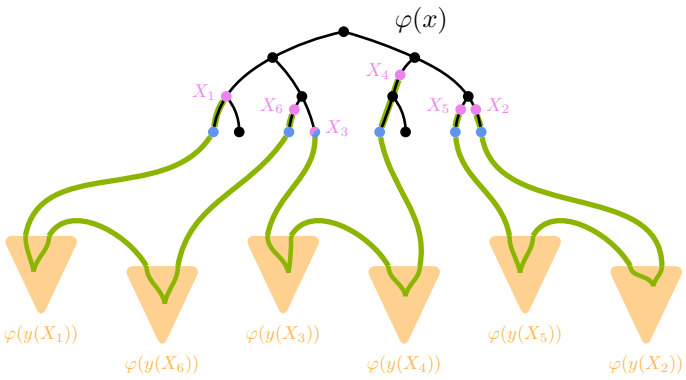


Figure 1 This illustrates the case that many vertices of $\mathcal{N}$ lie in a single bag. The set $\mathcal{X} = \{X_1, \ldots, X_6\}$ is depicted together with the corresponding leaves (note that X_3 is equal to its leaf). Using two paired bags, the green paths are of length at most $6r + 1$ and pairwise disjoint.

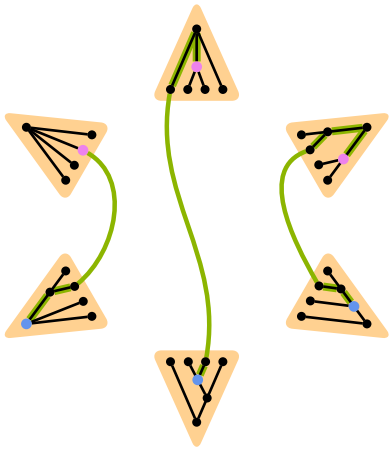


Figure 2 This illustrates the case that many vertices of $\mathcal{N}$ lie in different bags. The outermost spheres are mapped to the innermost spheres such that the green paths (all of length at most $4r + 2$) have to cross the spheres lying in between.

It is known, as seen in Lemma 4, that intersection graphs of balls of small ply have bounded strong colouring number. We make use of this to show that we can obtain a similar result for sphere intersection graphs excluding a $K_{t,t}$-subgraph, assuming they can be represented without a large set of nested spheres.

▶ **Lemma 11.** *Let $\mathcal{S}$ be a collection of spheres in $\mathbb{R}^d$ whose intersection graph $G = G(\mathcal{S})$ does not contain $K_{t,t}$ as a subgraph. Suppose that every nested subset of $\mathcal{S}$ has size at most k. Then for every $r \in \mathbb{N}$, the r-strong colouring number of G is at most $2kt(2r + 2)^d$.*

Proof. We claim that the collection $\{\mathsf{ball}(S) \mid S \in \mathcal{S}\}$ of balls in $\mathbb{R}^d$ has ply at most $2kt$. This yields the desired statement by Lemma 4.

So, let $\mathcal{X} \subseteq \mathcal{S}$ be a collection of spheres such that their common intersection as balls is non-empty, that is, such that $\bigcap_{X \in \mathcal{X}} \mathsf{ball}(X) \neq \emptyset$. Let us consider the poset $\preceq$ on $\mathcal{X}$ where for any spheres $X, X' \in \mathcal{X}$, we have $X \preceq X'$ if X is contained in X'. Every chain in this poset has size at most k by the assumption about nested sets. Next, note that a single point in

$\mathbb{R}^d$ lies in at most $2t$ pairwise incomparable (according to $\preceq$) spheres in $\mathcal{X}$, as otherwise, G would contain a clique of size $2t$ and therefore have $K_{t,t}$ as a subgraph. Thus, every antichain in this poset has size at most $2t$, and the lemma follows. ◀

The penultimate step is to combine lemmata 10 and 11 with Observation 5 in order to exclude shallow minors from the class $\mathcal{C}_t^d$.

▶ **Lemma 12.** *For any integers $t, d \geq 2$ and any $r \in \mathbb{N}$, no graph $G \in \mathcal{C}_t^d$ has a clique of size $72t^5(3r+2)^{d+3}$ as an r-shallow minor.*

Proof. For convenience, we set $h = 72t^5(3r+2)^{d+3}$. Suppose for a contradiction that there is a graph $G \in \mathcal{C}_t^d$ which contains K_h as an r-shallow minor. Choose such a graph G with as few vertices as possible. Let $\mathcal{S}$ be a collection of spheres in $\mathbb{R}^d$ such that $G = G(\mathcal{S})$. Then by Lemma 10, every nested subset of $\mathcal{S}$ has size at most $36t^4(3r+1)^3$. So by Lemma 11, the $(4r+1)$-strong colouring number of G is at most $2(36t^4(3r+1)^3)t(2r+2)^d$. Note that

$$2(36t^4(3r+1)^3)t(2r+2)^d = 72t^5(3r+1)^3(2r+2)^d \leq 72t^5(3r+2)^{d+3} - 2 = h - 2.$$

Finally, by Observation 5, the $(4r+1)$-strong colouring number of G is at least $h-1$. So $h - 1 \leq \mathsf{scol}_{4r+1}(G) \leq h - 2$, a contradiction. ◀

Now we can combine our results with the theorem of Plotkin, Rao, and Smith [46] (stated as Theorem 3) to obtain strongly sublinear separators in graphs of sphere dimension at most d which exclude a $K_{t,t}$-subgraph. We restate the theorem below for convenience.

▶ **Theorem 1.** *For any integers $t, d \geq 2$, the class $\mathcal{C}_t^d$ has strongly sublinear separators. More precisely, every n-vertex graph $G \in \mathcal{C}_t^d$ has a balanced separator of size $\widetilde{\mathcal{O}}_d(t^{10}n^{1-\frac{1}{2d+8}})$.*

Proof. Let $G \in \mathcal{C}_t^d$. For convenience, set $r = n^{\frac{1}{2d+8}}$. (We omit floors and ceilings for the sake of readability when they do not affect the proof.) By Lemma 12, G does not have a clique of size $72t^5(3r+2)^{d+3}$ as an r-shallow minor. Since $72t^5(3r+2)^{d+3} \in \mathcal{O}_d(t^5r^{d+3})$, by Theorem 3 we have that G contains a balanced separator of size

$$\mathcal{O}_d\left(rt^{10}r^{2d+6}\log n + \frac{n}{r}\right) = \widetilde{\mathcal{O}}_d\left(t^{10}r^{2d+7} + n^{1-\frac{1}{2d+8}}\right) = \widetilde{\mathcal{O}}_d\left(t^{10}n^{1-\frac{1}{2d+8}}\right).$$ ◀

5 Asymptotic dimension

In this section, we prove that $\mathcal{C}^d$ has bounded asymptotic dimension. In fact, we prove that the following more general class of graphs has bounded asymptotic dimension. Let $\mathcal{C}_\alpha^d$ be the class of connected intersection graphs of systems of path-connected sets in $\mathbb{R}^d$, each of which is obtained from a compact convex set of aspect ratio at most α by removing some subset of its interior. Clearly, the class $\mathcal{C}^d$ of intersection graphs of spheres in $\mathbb{R}^d$ is a subclass of $\mathcal{C}_\alpha^d$. If the sets are restricted so that none of their interiors is removed, then this subclass was proved to have bounded asymptotic dimension by Dvořák and Norin [19]. This theorem shall be one of our main tools.

▶ **Theorem 13** ([19])**.** *For every positive integer d and every $\alpha \geq 1$, the class of intersection graphs of systems of compact convex sets of aspect ratio at most α in $\mathbb{R}^d$ has asymptotic dimension at most $2d+1$.*

We introduce some notations similar to those used for spheres as in Section 4. For a graph $G \in \mathcal{C}^d_\alpha$ and a vertex $x \in V(G)$ we refer by $\mathsf{ccl}(x)$ to the convex closure of x. We say that a vertex y *contains* a different vertex x if $\mathsf{ccl}(y)$ contains $\mathsf{ccl}(x)$. We say the vertex x is *maximal* if there is no $y \in V(G)\backslash\{x\}$ such that y contains x.

The *height* of a bounded convex set $X \subset \mathbb{R}^d$ is the infimum $h \geq 0$ such that X is contained between two parallel hyperplanes at distance h. The *aspect ratio* of X is the ratio of the diameter of X over its *height*. Let $\mathcal{B}^d_\alpha$ denote the class of intersection graphs of systems of compact convex sets of aspect ratio at most α in $\mathbb{R}^d$. If $G \in \mathcal{C}^d_\alpha$ contains no pair of distinct vertices x, y with y containing x, then G belongs to $\mathcal{B}^d_\alpha$, i.e. Theorem 13 applies.

Our next main tool is a theorem of Bonamy, Bousquet, Esperet, Groenland, Liu, Pirot, and Scott [4] that shall allow us to apply Theorem 13. Before introducing this theorem, we require some further definitions.

A *weighted graph* (G, λ) consists of a graph G and a function $\lambda : E(G) \to \mathbb{R}^+$. We call $\lambda(e)$ the *weight* of e for each $e \in E(G)$. The length in (G, λ) of a path P in G is the sum of the weights of the edges of P. Given two vertices $x, y \in V(G)$, we define $\mathsf{dist}_{(G,\lambda)}(x, y)$ to be the infimum of the length in (G, λ) of a path between x and y; we define $\mathsf{dist}_{(G,\lambda)}(x, y) = \infty$ if there exists no path between x and y. We remark that all the weighted graphs we consider are actually graphs whose edges have weights 1 or 2. Given $A \subseteq V(G)$, we let $G[A]$ denote the *induced subgraph* of G on vertex set A. In other words, $G[A]$ is the subgraph of G obtained by restricting to the vertex set A. For weighted graphs, we define $(G, \lambda)[A]$ similarly.

A *real projection* of a (weighted) graph (G, λ) is a function $p : V(G) \to \mathbb{R}$ such that $|p(x) - p(y)| \leq \mathsf{dist}(x, y)$ for every pair of vertices $x, y \in V(G)$. Given $S > 0$ and a real projection p of a weighted graph G, we say that a vertex set $A \subseteq V(G)$ is (p, S)-*bounded* if $\max_{x,y\in A} |p(x) - p(y)| \leq S$. Given a class $\mathcal{C}$ of weighted graphs and a sequence $\mathcal{L} = (\mathcal{L}^1, \mathcal{L}^2 \ldots)$ of classes of weighted graphs with $\mathcal{L}^1 \subseteq \mathcal{L}^2 \subseteq \cdots$, we say that $\mathcal{C}$ is $\mathcal{L}$-*layerable* if there is a function $f : \mathbb{R}^+ \to \mathbb{N}$ such that any weighted graph $(G, \lambda) \in \mathcal{C}$ has a real projection $p : V(G) \to \mathbb{R}$ such that for any $S > 0$, any maximal (p, S)-bounded set in (G, λ) induces a (weighted) subgraph that belongs to $\mathcal{L}^{f(S)}$.

▶ **Theorem 14** ([4, Theorem 4.3]). *Let $\mathcal{L} = (\mathcal{L}_1, \mathcal{L}_2, \ldots)$ be a sequence of classes of weighted graphs of asymptotic dimension at most n. Let $\mathcal{C}$ be an $\mathcal{L}$-layerable class of weighted graphs. Then the asymptotic dimension of $\mathcal{C}$ is at most $n + 1$.*

We use the well-known fact that quasi-isometric classes of graphs have equal asymptotic dimension. A (Π, Σ)-*quasi-isometry* between (weighted) graphs (G, λ) and (H, ψ) is a map $f : V(G) \to V(H)$ such that the following hold for fixed constants $\Pi \geq 1$, $\Sigma \geq 0$:

1. $\Pi^{-1}\mathsf{dist}(x, y) - \Sigma \leq \mathsf{dist}(f(x), f(y)) \leq \Pi\mathsf{dist}(x, y) + \Sigma$ for every $x, y \in V(G)$, and
2. for every $z \in V(H)$, there is a $x \in V(G)$ such that $\mathsf{dist}(z, f(x)) \leq \Sigma$.

We say that classes of weighted graphs $\mathcal{G}$ and $\mathcal{H}$ are *quasi-isometric*, if there exist fixed constants $\Pi \geq 1$, $\Sigma \geq 0$ such that for every $G \in \mathcal{G}$ there exists a $H \in \mathcal{H}$ and a (Π, Σ)-*quasi-isometry* between G and H. It is well known that quasi-isometric classes of (weighted) graphs have equal asymptotic dimension (see for instance [4]).

▶ **Lemma 15.** *If classes of weighted graphs $\mathcal{G}_1, \mathcal{G}_2$ are quasi-isometric, then they have equal asymptotic dimension.*

Instead of working with $\mathcal{C}^d_\alpha$, we shall define an essentially equivalent (we only add edges of weight 2 between some vertices at distance 2) class of weighted graphs $\widehat{\mathcal{C}}^d_\alpha$ that is layerable by a class of weighted graphs that are quasi-isometric to graphs in $\mathcal{B}^d_\alpha$.

For $G \in \mathcal{C}^d_\alpha$, choose some maximal $v \in V(G)$. We call v the *pivot* vertex of G. Now, for each non-negative integer i, let D_i be the vertices in G at distance i from v. Let $(\widehat{G}, \lambda)$ be the weighted graph obtained from G by adding, for each pair $x, y \in V(G)$ with $x, y \in D_i$ for some i, an edge of weight 2 between x and y if x is contained in y (all the edges $(E(G))$ have weight 1 in $(\widehat{G}, \lambda)$). The following observation is key.

▶ **Lemma 16.** *Let $G \in \mathcal{C}^d_\alpha$. Then for every pair $x, y \in V(G)$ holds $\mathsf{dist}_G(x, y) = \mathsf{dist}_{(\widehat{G},\lambda)}(x, y)$.*

Proof. It is enough to show that if there is an edge of weight 2 between vertices x, y in $(\widehat{G}, \lambda)$, then in G, there is a vertex adjacent to both x and y. Since there is an edge of weight 2 between vertices x, y in $(\widehat{G}, \lambda)$, there exists some positive integer i such that $x, y \in D_i$, and without loss of generality, we may assume that x is contained in y. Let P be a path of length i between x and v, and let z be the unique vertex of $D_{i-1} \cap V(P)$. Then z intersects x and $\bigcup_{u \in V(P)} u$ contains a curve C from x to a point in v not contained in w for any $w \in V(G)$. Since the curve C starts in the interior of y and ends in its exterior, C must intersect y. Therefore, y is adjacent to a vertex of P. However, y is non-adjacent to x in G and cannot be adjacent to any vertex of $D_0 \cup \cdots \cup D_{i-2}$ since $y \in D_i$. Therefore, y must be adjacent to z in G. Hence, the distance between x and y in G is equal to 2, as desired. ◀

Let $\mathcal{L}^i_{d,\alpha}$ be the class of weighted graphs that are $(i, 1)$-quasi-isometric to a graph in $\mathcal{B}^d_\alpha$. Note that $\mathcal{L}^1_{d,\alpha} \subseteq \mathcal{L}^2_{d,\alpha} \subseteq \cdots$. Let $\mathcal{L}_{d,\alpha} = (\mathcal{L}^1_{d,\alpha}, \mathcal{L}^2_{d,\alpha}, \ldots)$. We show that $\widehat{\mathcal{C}}^d_\alpha$ is $\mathcal{L}_{d,\alpha}$-layerable.

▶ **Lemma 17.** *The class of weighted graphs $\widehat{\mathcal{C}}^d_\alpha$ is $\mathcal{L}_{d,\alpha}$-layerable.*

Proof. Let $(\widehat{G}, \lambda) \in \widehat{\mathcal{C}}^d_\alpha$, let $G \in \mathcal{C}^d_\alpha$ be the graph obtained from $(\widehat{G}, \lambda)$ by removing its edges of weight 2, and let v be the pivot-vertex of $(\widehat{G}, \lambda)$. For $x \in V(\widehat{G})$, let $p(x) \coloneqq d_{\widehat{G}}(v, x)$. Then, p is a real projection of $(\widehat{G}, \lambda)$. Let $S > 0$, and let $A \subseteq V(\widehat{G})$ be some maximal (p, S)-bounded set in $(\widehat{G}, \lambda)$. Then, there exists some non-negative integer t such that A is the set of vertices in $(\widehat{G}, \lambda)$ (or equivalently G) whose distance from v is between t and $t + \lceil S \rceil$; indeed, we can let $t \coloneqq \min\{p(x) \mid x \in A\}$.

Let M be the set of maximal vertices of $G[A]$. Observe that $(\widehat{G}, \lambda)[M] = G[M]$ contains no edge of weight 2 and furthermore that it is a graph in $\mathcal{B}^d_\alpha$. Let $f : A \to M$ be such that for every $u \in A$, $f(u)$ is some maximal vertex containing u. We shall show that f is a $(2\lceil S \rceil + 4, 1)$-quasi-isometry from $(\widehat{G}, \lambda)[A]$ to $G[M]$, which implies that $(\widehat{G}, \lambda)[A] \in \mathcal{L}^{2\lceil S \rceil + 4}_{d,\alpha}$ and therefore that $\widehat{\mathcal{C}}^d_\alpha$ is $\mathcal{L}_{d,\alpha}$-layerable as we desire.

If there is an edge of weight 1 or 2 between vertices x and y of $(\widehat{G}, \lambda)[A]$, then $f(x)$ and $f(y)$ intersect or coincide, and therefore $\mathsf{dist}_{G[M]}(f(x), f(y)) \leq 1$. It follows that $\mathsf{dist}_{G[M]}(f(x), f(y)) \leq \mathsf{dist}_{(\widehat{G},\lambda)[A]}(x, y)$ for every pair of vertices $x, y \in A$.

Consider some $m \in M$. Note that $f^{-1}(m)$ is non-empty since $f(m) = m$, and so f is surjective. We aim to show that each vertex x of $f^{-1}(m)$ has distance at most $S + 2$ from m in $(\widehat{G}, \lambda)[A]$. Let $x_0 x_1 \ldots x_\ell$ be a v–x geodesic in $(\widehat{G}, \lambda)$ (or G), i.e. a path with $x_0 = v$, $x_\ell = x$ and $p(x_i) = \mathsf{dist}_{(\widehat{G},\lambda)}(v, x_i) = i$ for each i. Then $\bigcup_{i=0}^{\ell} x_i$ contains a curve C from x to a point in v not contained in the interior of z for any $z \in V(G) \backslash v$. Since the curve C starts in the interior of $\mathsf{ccl}(m)$ and ends outside the interior of $\mathsf{ccl}(m)$, it follows that m is adjacent in G to some vertex of $\{x_0, \ldots, x_\ell\}$. We further have that m must be adjacent to one of $x_{p(m)-1}, x_{p(m)}, x_{p(m)+1}$. Note that $t \leq p(m) \leq p(x) = \ell \leq t + \lceil S \rceil$ since $m, x \in A$. If either $p(m) > t$ or m is adjacent to one of $x_{p(m)}$ or $x_{p(m)+1}$, then there is a path in $G[A]$ (and thus in $(\widehat{G}, \lambda)[A]$) between m and x of length at most $\lceil S \rceil + 1$. So, we may assume now that $p(m) = t$ and that m is non-adjacent to both $x_{p(m)}$ and $x_{p(m)+1}$. It follows that $x_{p(m)}$

is contained in $\mathsf{ccl}(m)$ and that m is adjacent to $x_{p(m)-1}$ in G. Therefore, there is an edge of weight 2 in $(\widehat{G},\lambda)$ between m and $x_{p(m)}$. Hence, there is a path of length at most $\lceil S\rceil+2$ in $(\widehat{G},\lambda)[A]$ between m and x. Thus, x has distance at most $\lceil S\rceil+2$ from m in $(\widehat{G},\lambda)[A]$.

So $f^{-1}(m)$ has radius at most $\lceil S\rceil+2$ in $(\widehat{G},\lambda)[A]$ and therefore diameter at most $2\lceil S\rceil+4$. Thus, as $G[M]$ is a graph, for every $x,y\in A$, we have that $(2\lceil S\rceil+4)^{-1}\mathsf{dist}_{(\widehat{G},\lambda)[A]}(x,y)-1\leq \mathsf{dist}_{G[M]}(f(x),f(y))$. Hence f is a $(2\lceil S\rceil+4)$-quasi-isometry between $(\widehat{G},\lambda)[A]$ and $G[M]$. ◀

We have assembled all the ingredients to prove the main result of this section:

▶ **Theorem 18.** *For every positive integer d and every $\alpha\geq 1$, the class $\mathcal{C}^d_\alpha$ of intersection graphs of systems of path-connected sets in $\mathbb{R}^d$, each of which is obtained from a compact convex set of aspect ratio at most α by removing some subset of its interior, has asymptotic dimension at most $2d+2$.*

Proof. By Lemma 16, the asymptotic dimension of $\mathcal{C}^d_\alpha$ and $\widehat{\mathcal{C}}^d_\alpha$ is equal. By Lemma 17, $\widehat{\mathcal{C}}^d_\alpha$ is $\mathcal{L}_{d,\alpha}$-layerable. The graph classes $\mathcal{L}^1_{d,\alpha},\mathcal{L}^2_{d,\alpha},\ldots$ all have asymptotic dimension at most $2d+1$ by Lemma 15 and Theorem 13. Therefore, by Theorem 14, $\widehat{\mathcal{C}}^d_\alpha$ has asymptotic dimension at most $2d+2$. Hence $\mathcal{C}^d_\alpha$ has asymptotic dimension at most $2d+2$. ◀

6 Final remarks and open problems

As we have seen, every graph G with at least three vertices has sphere dimension at most $|V(G)|-1$ by Observation 7. Can this be improved? Providing a bound with respect to the number of edges is also interesting.

▶ **Problem 19.** *What is the maximum sphere dimension of a graph on n vertices?*

In light of other known separator theorems for ball intersection graphs [44], it seems likely that the exponent in Theorem 1 is not optimal. We conjecture the following.

▶ **Conjecture 20.** *For any integers $t,d\geq 1$, every n-vertex graph $G\in\mathcal{C}^d_t$ has a balanced separator of size $\mathcal{O}_{t,d}(n^{1-\frac{1}{d+1}})$.*

This conjecture is true for $d=1$ since every n-vertex circle graph with a forbidden $K_{t,t}$ subgraph has a separator of size $\mathcal{O}_t(\sqrt{n})$ due to [37] and [24, Theorem 3].

It is also likely possible to improve the asymptotic dimension bound in Theorem 18 for graphs of bounded sphere dimension as follows.

▶ **Conjecture 21.** *For any integer $d\geq 1$, the class of graphs with sphere dimension at most d has asymptotic dimension at most d.*

This conjecture would be tight since the hypercubic lattice $\mathbb{Z}^d$ has asymptotic dimension and sphere dimension equal to d. For the latter claim, the sphere dimension is at most d because of the standard sphere packing centred at the integers. The fact that the sphere dimension is at least d follows from a deep result by Benjamini and Schramm [2, 3] stating that $\mathbb{Z}^d$ cannot be quasi-sphere packed in $\mathbb{R}^{d-1}$, with a little additional work. The case $d=2$ of the conjecture (furthermore for Assouad–Nagata dimension and in the more general class of string graphs) will be proved in separate work [10]. It is also easy to deduce the $d=1$ case for circle graphs from known results [8, 26]. Recently Liu and Norin [41] independently, using different methods, proved an improved asymptotic dimension bound of $d+1$. It might be possible to reduce Conjecture 21 via quasi-isometry to the class of intersection graphs of balls in $\mathbb{R}^d$ (this is the case for $d=2$ [10]).

▶ **Conjecture 22.** *For any integer $d \geq 2$, the class of graphs with sphere dimension at most d is quasi-isometric to the class of intersection graphs of balls in $\mathbb{R}^d$.*

Recall that an *induced minor* of a graph is obtained by deleting vertices and contracting edges. Let $\mathrm{Forb}_{\mathrm{ind}}(H)$ denote the class of graphs that do not contain a graph H as an induced minor.

▶ **Problem 23.** *Do the graphs in $\mathrm{Forb}_{\mathrm{ind}}(H)$ have bounded sphere dimension for every finite graph H? If not, is each graph in $\mathrm{Forb}_{\mathrm{ind}}(H)$ uniformly quasi-isometric to a graph with bounded sphere dimension?*

The same questions can be asked for minors instead of induced minors. These two versions of Problem 23 are related via the following problem, which is a special case of [27, Conjecture 1.1] (which has been disproven [11], but special cases remain open):

▶ **Problem 24.** *For every finite graph H, each graph in $\mathrm{Forb}_{\mathrm{ind}}(H)$ is uniformly quasi-isometric to a graph with no H minor.*

For $d \geq 3$, it is not possible to replace asymptotic dimension by Assouad–Nagata dimension in Theorem 18, even for the subclass of ball intersection graphs. This was observed by Dvořák and Norin [17]. Assouad–Nagata dimension is closely related to a concept in computer science called sparse partition schemes. Roughly speaking, if a graph class has bounded Assouad–Nagata dimension, and the colourings witnessing this can be computed, then certain universal versions of classical optimisation problems such as TSP, Steiner Tree and Set Cover allow approximations of bounded ratio. These approximate solutions are computed using sparse partition schemes; see, for instance, [4, 32, 6]. As mentioned above, graphs of bounded sphere dimension have unbounded Assouad–Nagata dimension, and so we cannot hope for sparse partition schemes. Still, it would be interesting to obtain bounds on these parameters as a function of the size of the graph. Our proof of Theorem 18 utilises Theorem 14, which has a strengthening preserving boundedness of Assouad–Nagata dimension. Thus, the bottleneck lies in the proof of Theorem 13 in [17].

References

1 E. M. Andreev. Convex polyhedra of finite volume in Lobačevskiĭ space. *Mat. Sb. (N.S.)*, 83(125):256–260, 1970.

2 Itai Benjamini and Oded Schramm. Harmonic functions on planar and almost planar graphs and manifolds, via circle packings. *Invent. Math.*, 126(3):565–587, 1996. doi:10.1007/s002220050109.

3 Itai Benjamini and Oded Schramm. Lack of sphere packing of graphs via nonlinear potential theory. *J. Topol. Anal.*, 5(1):1–11, 2013. doi:10.1142/S1793525313500039.

4 Marthe Bonamy, Nicolas Bousquet, Louis Esperet, Carla Groenland, Chun-Hung Liu, François Pirot, and Alexander Scott. Asymptotic dimension of minor-closed families and assouad–nagata dimension of surfaces. *Journal of the European Mathematical Society*, 26(10):3739–3791, 2023.

5 André Bouchet. Circle graph obstructions. *J. Comb. Theory, Ser. B*, 60(1):107–144, 1994. doi:10.1006/jctb.1994.1008.

6 Costas Busch, Ryan Lafortune, and Srikanta Tirthapura. Sparse covers for planar graphs and graphs that exclude a fixed minor. *Algorithmica*, 69(3):658–684, 2014. doi:10.1007/s00453-013-9757-4.

7 L. Sunil Chandran, Mathew C. Francis, and Naveen Sivadasan. Geometric representation of graphs in low dimension using axis parallel boxes. *Algorithmica*, 56(2):129–140, 2010. doi:10.1007/s00453-008-9163-5.

8 Victor Chepoi, Feodor F Dragan, Ilan Newman, Yuri Rabinovich, and Yann Vaxes. Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs. *Discrete & Computational Geometry*, 47:187–214, 2012. doi:10.1007/S00454-011-9386-0.

9 Fan R. K. Chung. Separator theorems and their applications. In *Paths, flows, and VLSI-layout (Bonn, 1988)*, volume 9 of *Algorithms Combin.*, pages 17–34. Springer, Berlin, 1990.

10 James Davies. Riemannian planes and string graphs are quasi-isometric to planar graphs. (In Preparation).

11 James Davies, Robert Hickingbotham, Freddie Illingworth, and Rose McCarty. Fat minors cannot be thinned (by quasi-isometries). arXiv:2405.09383.

12 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, and Sudeshna Kolay. An ETH-tight exact algorithm for Euclidean TSP. *SIAM J. Comput.*, 52(3):740–760, 2023. doi:10.1137/22M1469122.

13 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.

14 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.

15 Marc Distel. Proper minor-closed classes of graphs have Assouad-Nagata dimension 2, 2023. arXiv:2308.10377.

16 Guillermo Durán, Luciano N. Grippo, and Martín D. Safe. Structural results on circular-arc graphs and circle graphs: a survey and the main open problems. *Discrete Appl. Math.*, 164:427–443, 2014. doi:10.1016/j.dam.2012.12.021.

17 Zdeněk Dvořák, Rose McCarty, and Sergey Norin. Sublinear separators in intersection graphs of convex shapes. *SIAM J. Discrete Math.*, 35(2):1149–1164, 2021. doi:10.1137/20M1311156.

18 Zdeněk Dvořák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM J. Discrete Math.*, 30(2):1095–1101, 2016. doi:10.1137/15M1017569.

19 Zdeněk Dvořák and Sergey Norin. Asymptotic dimension of intersection graphs. *European Journal of Combinatorics*, page 103631, 2022.

20 Zdeněk Dvořák and Sergey Norin. Weak diameter coloring of graphs on surfaces. *Eur. J. Comb.*, 121:13, 2024. Id/No 103845. doi:10.1016/j.ejc.2023.103845.

21 Zdeněk Dvořák, Jakub Pekárek, Torsten Ueckerdt, and Yelena Yuditsky. Weak coloring numbers of intersection graphs. In *38th international symposium on computational geometry, SoCG 2022, Berlin, Germany, June 7–10, 2022*, page 15. Wadern: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. Id/No 39. doi:10.4230/LIPIcs.SoCG.2022.39.

22 David Eppstein. Minimum dimension for sphere packing a graph in euclidean space. MathOverflow. URL: https://mathoverflow.net/q/86274 (version: 2012-01-21), Author: https://mathoverflow.net/users/440/david-eppstein.

23 Louis Esperet. Boxicity and topological invariants. *European J. Combin.*, 51:495–499, 2016. doi:10.1016/j.ejc.2015.07.020.

24 Jacob Fox and János Pach. A separator theorem for string graphs and its applications. *Comb. Probab. Comput.*, 19(3):371–390, 2010. doi:10.1017/S0963548309990459.

25 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Eur. Math. Soc. (JEMS)*, 19(6):1785–1810, 2017. doi:10.4171/JEMS/705.

26 Koji Fujiwara and Panos Papasoglu. Asymptotic dimension of planes and planar graphs. *Trans. Am. Math. Soc.*, 374(12):8887–8901, 2021. doi:10.1090/tran/8487.

27 Agelos Georgakopoulos and Panos Papasoglu. Graph minors and metric spaces, 2023. arXiv:2305.07456.

28 Mikhael Gromov. *Geometric group theory. Volume 2: Asymptotic invariants of infinite groups. Proceedings of the symposium held at the Sussex University, Brighton, July 14-19, 1991*, volume 182 of *Lond. Math. Soc. Lect. Note Ser.* Cambridge: Cambridge University Press, 1993.

29 A. Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Math.*, 55:161–166, 1985. doi:10.1016/0012-365X(85)90044-5.

30 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.

31 Zheng-Xu He and Oded Schramm. Fixed points, Koebe uniformization and circle packings. *Ann. Math. (2)*, 137(2):369–406, 1993. doi:10.2307/2946541.

32 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proceedings of the 37th annual ACM symposium on theory of computing, STOC'05. Baltimore, MD, USA, May 22–24, 2005*, pages 386–395. New York, NY: Association for Computing Machinery (ACM), 2005. doi:10.1145/1060590.1060649.

33 Martina Jørgensen and Urs Lang. Geodesic spaces of low Nagata dimension. *Ann. Fenn. Math.*, 47(1):83–88, 2022. doi:10.54330/afm.112472.

34 Sándor Kisfaludi-Bak, Jana Masaříková, Erik Jan van Leeuwen, Bartosz Walczak, and Karol Węgrzycki. Separator theorem and algorithms for planar hyperbolic graphs. In *40th International Symposium on Computational Geometry*, volume 293 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 67, 17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/lipics.socg.2024.67.

35 Paul Koebe. *Kontaktprobleme der konformen Abbildung.* Ber. Verh. Sächs. Akad. Leipzig 88, 1936.

36 T. Kövári, Vera T. Sós, and Pál Turán. On a problem of K. Zarankiewicz. *Colloq. Math.*, 3:50–57, 1954. doi:10.4064/cm-3-1-50-57.

37 James R. Lee. Separators in region intersection graphs. In *8th innovations in theoretical computer science conference, ITCS 2017, Berkeley, CA, USA, January 9–11, 2017*, page 8. Wadern: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. Id/No 1. doi:10.4230/LIPIcs.ITCS.2017.1.

38 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.

39 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.

40 Chun-Hung Liu. Assouad-nagata dimension of minor-closed metrics, 2023. doi:10.48550/arXiv.2308.12273.

41 Chun-Hung Liu and Sergey Norin. personal comunication.

42 Hiroshi Maehara. Space graphs and sphericity. *Discrete Appl. Math.*, 7(1):55–64, 1984. doi:10.1016/0166-218X(84)90113-6.

43 Jiří Matoušek. Near-optimal separators in string graphs. *Comb. Probab. Comput.*, 23(1):135–139, 2014. doi:10.1017/S0963548313000400.

44 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.

45 Walid Naji. Reconnaissance des graphes de cordes. *Discrete Math.*, 54:329–337, 1985. doi:10.1016/0012-365X(85)90117-7.

46 Serge Plotkin, Satish Rao, and Warren D. Smith. Shallow excluded minors and improved graph decompositions. In *Proceedings of the 5th annual ACM-SIAM symposium on discrete algorithms, SODA '94, Arlington, VA, USA, January 23–25, 1994*, pages 462–470. New York, NY: ACM; Philadelphia, PA: SIAM, 1994. URL: http://dl.acm.org/citation.cfm?id=314464.314625.

47 Fred S. Roberts. On the boxicity and cubicity of a graph. Recent Prog. Comb., Proc. 3rd Waterloo Conf. 1968, 301-310 (1969)., 1969.

48 Rose McCarty. *Local Structure for Vertex-Minors.* PhD Thesis, UWSpace, 2021.

49 Alex Scott and David R. Wood. Better bounds for poset dimension and boxicity. *Trans. Amer. Math. Soc.*, 373(3):2157–2172, 2020. doi:10.1090/tran/7962.

50 Warren D. Smith and Nicolas 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.

51 William P. Thurston. *Collected works of William P. Thurston with commentary: IV. The geometry and topology of three-manifolds: with a preface by Steven P. Kerckhoff. Edited by Benson Farb, David Gabai and Steven P. Kerckhoff.* Providence, RI: American Mathematical Society (AMS), 2022.

52 W. T. jun. Trotter. Interval graphs, interval orders, and their generalizations. Applications of discrete mathematics, Proc. 3rd SIAM Conf., Clemson/South Carolina 1986, 45-58, 1988.

53 Guoliang Yu. The Novikov conjecture for groups with finite asymptotic dimension. *Ann. Math. (2)*, 147(2):325–355, 1998. doi:10.2307/121011.

Nearest Neighbor Searching in a Dynamic Simple Polygon

Sarita de Berg
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

In the nearest neighbor problem, we are given a set S of point sites that we want to store such that we can find the nearest neighbor of a (new) query point efficiently. In the dynamic version of the problem, the goal is to design a data structure that supports both efficient queries and updates, i.e. insertions and deletions in S. This problem has been widely studied in various settings, ranging from points in the plane to more general distance measures and even points within simple polygons. When the sites do not live in the plane but in some domain, another dynamic problem arises: what happens if not the sites, but the domain itself is subject to updates?

Updating sites often results in local changes to the solution or data structure, while updating the domain may incur many global changes. For example, in the closest pair problem, inserting a point only requires us to check if this point is in the new closest pair, while updating the domain might change the distances between most pairs of points in our set. Presumably, this is the reason that this form of dynamization has received much less attention. Only some basic problems, such as shortest paths and ray shooting, have been studied in this setting.

Here, we tackle the nearest neighbor problem in a dynamic simple polygon. We allow insertions into both the set of sites and the polygon. An insertion in the polygon is the addition of a line segment starting at the boundary of the polygon. We present a near-linear size –in both the number of sites and the complexity of the polygon– data structure with sublinear update and query time. This is the first nearest neighbor data structure that allows for updates to the domain.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases dynamic data structure, simple polygon, geodesic distance, nearest neighbor

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.37

Related Version *Full Version*: `https://arxiv.org/abs/2503.03435` [12]

1 Introduction

The nearest neighbor problem is a fundamental problem in computer science: given a set of n point sites S in some domain P, the goal is to store S so that given a query point $q \in P$ we can efficiently find the point in S closest to q [5, 19]. In case P is the two-dimensional Euclidean plane, such queries can be answered in $O(\log n)$ time by computing the Voronoi diagram of S, and storing it for efficient point location queries. In many applications, the domain however may contain obstacles (e.g. buildings, cars, rivers, lakes etc.) that are impassible. Hence, we may want to model P as a simple polygon, or even a polygonal domain (i.e. a polygon with holes), and measure the distance between two points by the length of their shortest obstacle avoiding path. In this setting, it is also possible to build the Voronoi diagram. Aronov [3] showed that the so called *geodesic Voronoi diagram* in a simple polygon with m vertices has complexity $O(n + m)$, and gave an $O((n + m) \log(n + m) \log m)$ time algorithm to construct it. This then allows for $O(\log(n + m))$ time nearest neighbor queries. Finding an optimal algorithm to construct the geodesic Voronoi diagram remained elusive for

© Sarita de Berg and Frank Staals;
licensed under Creative Commons License CC-BY 4.0

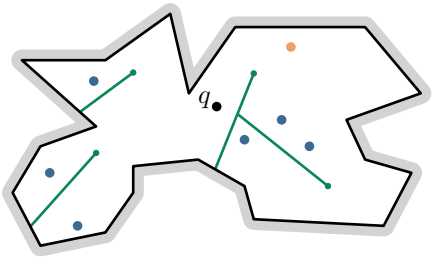


Figure 1 After inserting the green segments, the orange site is the nearest neighbor of q.

quite a while. Only a few years ago, after a series of improvements [21, 24, 26], Oh showed an optimal $O(m + n \log n)$ time algorithm [22]. When P is a polygon with holes, the Voronoi diagram has the same complexity, and can be computed in $O((n + m) \log(n + m))$ time [16].

Nearest neighbor searching is a *decomposable search problem*, meaning that to find the nearest neighbor in the set $A \cup B$, we can compute the nearest neighbor in set A, separately compute the nearest neighbor in set B, and return the closest of these two candidates [4]. Therefore, supporting efficient insertions of new point sites in $\mathbb{R}^2$ (while still answering queries in polylogarithmic time) is relatively simple using, for example, the logarithmic method [25]. However, supporting both insertions and deletions in polylogarithmic time, turned out to be quite challenging. Chan [5] presented the first such results, achieving $O(\log^2 n)$ query time while supporting insertions in $O(\log^3 n)$ and deletions in $O(\log^6 n)$ amortized expected time. Kaplan et al. [19] and later Chan [7] further improved the update times to $O(\log^2 n)$ for insertions, and $O(\log^4 n)$ for deletions. Kaplan et al. also generalized these results to other (constant complexity) distance functions [19]. Agarwal, Arge, and Staals [1] showed that these techniques can be extended to efficiently answer nearest neighbor queries in a simple polygon P. De Berg and Staals even presented a data structure that can report efficiently report the k-nearest neighbors, for some query parameter $k \in [n]$, in this setting [11].

The above results thus support applications in which the set of input points may change over time. However, in such applications, the obstacles are also likely to change. Unfortunately, none of the approaches above support updates to the domain P, e.g. by inserting or deleting an obstacle. Even more generally, there is relatively little work in which the input domain P may change while maintaining some non-trivial structure on P or S. Results of this type are mostly restricted to general purpose range or intersection searching [2, 17] or point location [14]. The most prominent work that we are aware of that specifically maintains some data structure on top of a dynamic polygonal domain is the work by Goodrich and Tamassia [13]. They support ray shooting and shortest path queries in a dynamic connected planar subdivision $\hat{P}$ in $O(\log^2 n)$ time. Ishaque and Tóth show how to maintain the (geodesic) convex hull of S in $\hat{P}$ while supporting the insertion of barriers (non-crossing segments such that all faces are simply connected) and site deletions in $O(\text{polylog}(n + m))$ time [18]. Oh and Ahn generalize this result to supporting both insertions and deletions (of barriers and sites) [23]. They achieve update times of $\tilde{O}((n + m)^{2/3})$, where the $\tilde{O}(\cdot)$ notation hides logarithmic factors in n and m. A recent result of Choudhury and Inkulu [10] shows how to maintain the visibility graph inside a dynamic simple polygon P using $\tilde{O}(\Delta)$ time per update, where Δ is the number of changes.

Challenges. A possible explanation for the lack of work in this dynamic domain setting is that the problems seem extremely challenging: even just adding a single line segment to the boundary of the polygon may significantly change the distances between the sites in the

polygon. This severely limits what information (i.e. which distances) the data structure can store. An insertion may also structurally change the triangulation of the polygon (which is the foundation upon which, e.g., the data structure of Agarwal, Arge, and Staals [1] is built).

Our results. We overcome these challenges, and present the first dynamic data structure for the geodesic nearest neighbor problem that allows updates to both the sites S and the simple polygon P containing S. Our data structure achieves sublinear query and update time and allows the following updates on P and S (Fig. 1): **1.** inserting a segment (barrier) between points u and v, where u on the boundary ∂P of P and $v \in P$ an arbitrary point, provided that the interior of this segment does not intersect ∂P, and **2.** inserting a site $s \in P$ into S.

This type of barrier insertion is similar to the insertion operation considered by Ishaque and Tóth [18] and Oh and Ahn [23]. Note that in case v also lies on the boundary of P, this actually splits the polygon into two subpolygons, which can no longer interact. Hence, we can essentially treat these subpolygons independently. In the remainder of the paper, we thus focus on the more interesting case in which v lies strictly in the interior of P. This also slightly simplifies our presentation, since P then remains an actual simple polygon.

Our data structure uses $O(n(\log\log n + \log m) + m)$ space, supports the above updates in amortized $\tilde{O}(n^{3/4} + m^{3/4})$ time, and answers queries in expected $\tilde{O}(n^{3/4} + n^{1/4}m^{1/2})$ time. While these query and update times are still far off from the polylogarithmic query and update times for a static domain, they are comparable to, for example, the result of Oh and Ahn for maintaining the geodesic hull in a dynamic polygon [23].

We believe it should be possible to extend our data structure to support deleting sites as well. In particular, by further incorporating the results of Agarwal, Arge, and Staals [1]. However, for ease of exposition we defer site deletions for now. Supporting deleting vertices of the polygon seems more interesting, however also much more complicated. Further ideas are likely required to support deletions of vertices as well.

Organization. Our data structure essentially consists of two independent parts, both of which critically build upon balanced geodesic triangulations [9]. So, we first review some of their useful properties in Section 2. We then give an overview of the data structure in Section 3. Section 4 describes the first part of our data structure, which can be regarded as an extension of the dynamic two-point shortest path data structure of Goodrich and Tamassia [13]. Our main technical contribution is in Section 5, which describes the second main part: a (static) data structure that can answer nearest neighbor queries in a subpolygon Q of P that can be specified at query time. This result may be of independent interest. Omitted proofs as well as the generalization of some of the results of Agarwal, Arge, and Staals [1] that we need for our data structure, can be found in the full version [12].

2 Preliminaries

Let P be a simple polygon with m vertices, let ∂P be the boundary of P, and let S be a set of n point sites inside P. We denote the shortest path that is fully contained within P between two point $p, q \in P$ by $\pi_P(p, q)$, or simply $\pi(p, q)$ if the polygon is clear from the context. Similar to earlier papers about geodesic Voronoi diagrams, we will use the general position assumptions that no point is (ever) equidistant to four sites in S, and no vertex of P is equidistant to three sites.

A *geodesic triangulation* of P is a triangulation-like decomposition of P into geodesic triangles. A *geodesic triangle* with corners $u, v, w \in P$ is formed by the three shortest paths $\pi(u, v)$, $\pi(v, w)$, $\pi(w, u)$. In general, such a geodesic triangle consists of a simple polygon

bounded by three concave chains, and three polygonal chains emanating from the vertices where the the concave chains join. The interior of the simple polygon region is called the *deltoid* region, and the polygonal chains are called the *tails* of the geodesic triangle. Note that the deltoid region may be empty, for example if w lies on $\pi(u, v)$.

Similar to a triangulation, a geodesic triangulation decomposes P into $O(m)$ geodesic triangles with corners that are vertices of P whose boundaries do not cross. This triangulation induces a dual tree $\mathcal{T}$, where two nodes are connected if their respective geodesic triangles share two corners. We root this tree at a node whose geodesic triangle shares a side with an edge of P. A geodesic triangulation is called *balanced* if the height of this tree is $O(\log m)$. A balanced geodesic triangulation can be constructed in $O(m)$ time [9].

A dynamic shortest path data structure. Here we shortly discuss the data structure of Goodrich and Tamassia for dynamic planar subdivisions [13]. This data structure supports both shortest-path and ray-shooting queries in $O(\log^2 m)$ time. Updates to the subdivision, such as inserting or deleting a vertex or edge, also take $O(\log^2 m)$ time. The main component is a balanced geodesic triangulation with dual tree $\mathcal{T}$. A secondary dynamic point location data structure [14] is build for this geodesic triangulation. A tertiary data structure stores, for each deltoid Δ, the three concave chains bounding Δ in balanced tree structures. To support shortest path queries, they store for each node in $\mathcal{T}$ the tail of its geodesic triangle that is not shared by its parent in a balanced tree structure. The size of the data structure is $O(m)$. Using rotations, insertions and deletions can be performed in $O(\log^2 m)$ time.

3 Overview of the data structure

Our dynamic nearest neighbor data structure for a dynamic simple polygon consists of two independent data structures: a Dynamic Shortest Path data structure, which remains up-to-date, and a Cone Query data structure, which is static and is rebuild after a number of updates. We denote by P the current polygon with m vertices and by S the current set of n sites in P. Furthermore, we denote by $\tilde{P}$ and $\tilde{m}$ the polygon and number of vertices and by $\tilde{S}$ and $\tilde{n}$ the set of sites and its size at the last rebuild of the Cone Query data structure. Let k be the number of updates (insertions) since the last rebuild, i.e. $k = m + n - \tilde{m} - \tilde{n}$.

The main purpose of the two data structures is the following. When given a query point q, we first use the Dynamic Shortest Path data structure to compute the distance from q to each of the newly inserted sites, i.e. in $S \setminus \tilde{S}$. This gives us the nearest neighbor of q among these sites. To find the nearest neighbor of q among $\tilde{S}$, we first use the Dynamic Shortest Path data structure to find $O(k)$ regions in P, where each region R is paired with an apex point v_R on ∂R. Each regions has the property that the shortest path $\pi_P(p, q)$ from a point $p \in R$ to q is given by $\pi_{\tilde{P}}(p, v_R) \cup \pi_P(v_R, q)$. This means that the nearest neighbor of q in P of the sites $\tilde{S} \cap R$ is the same as the nearest neighbor of v_R in $\tilde{P}$ of the same sites $\tilde{S} \cap R$.

This is where the Cone Query query data structure comes into play. This data structure allows us the answer such a nearest neighbor query for some region R with apex v_R in $\tilde{P}$. After querying the Cone Query data structure for each of the $O(k)$ regions generated by the Dynamic Shortest Path data structure, we return the nearest neighbor of q among $\tilde{S}$, which is the closest of these $O(k)$ sites. Next, we define the queries for the Cone Query data structure more precisely. We refer to these queries as *bounded nearest neighbor queries*.

▶ **Definition 1** (Figure 2). *A* bounded nearest neighbor query (R, v_R) *in* $\tilde{P}$ *consists of a region* R *and an point* $v_R \in \partial R$ *and asks for the nearest neighbor of* v_R *in* $\tilde{S} \cap R$*, where* R *is of one of the following two types:*

- *a* geodesic triangle*: a region bounded by three shortest paths in* $\tilde{P}$*, or*
- *a* geodesic cone*: a region bounded by two shortest paths in* $\tilde{P}$ *and part of* $\partial\tilde{P}$*.*

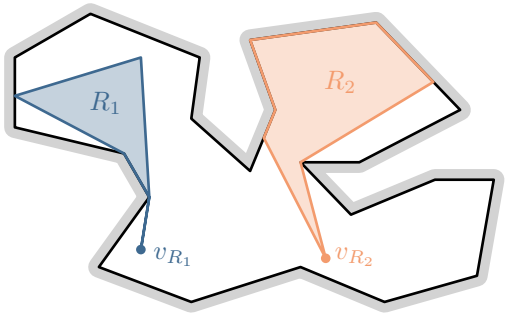


Figure 2 Two bounded nearest neighbor queries: a geodesic triangle (R_1) and a geodesic cone (R_2).

The Dynamic Shortest Path data structure that finds these geodesic triangle and geodesic cone regions is based on the data structure of Goodrich and Tamassia [13] for ray shooting and shortest path queries in a dynamic planar subdivision. We augment this data structure to distinguish between "old" and "new" vertices, i.e. the vertices in $\tilde{P}$ and in $P \setminus \tilde{P}$. In Section 4, we discuss this data structure in more detail, and obtain the following result.

▶ **Theorem 2.** *A Dynamic Shortest Path data structure maintains a dynamic simple polygon with $O(\log^2 m)$ update time using $O(m)$ space such that for a query point q a set of $O(k)$ bounded nearest neighbor queries that together find the nearest neighbor of q in P among $\tilde{S}$ in P can be computed in $O(k \log^2 m)$ time.*

The Cone Query data structure is the main data structure we present. We give a short overview of the data structure here, illustrated in Figure 3, and discuss the details and proofs in Section 5. The data structure is based on a balanced geodesic triangulation $\mathcal{T}$, augmented with three auxiliary data structures. We call the geodesic triangulation the first level data structure, and the three auxiliary data structures the second level data structures. The second level data structures are the following. First, for each node in $\mathcal{T}$, we store a partition tree [6] on the sites that lie in the corresponding deltoid. Furthermore, we store several third level data structures for each node of the partition tree. This allows us to find the nearest neighbor of v_R for (a subset of) the sites contained in the deltoid. Second, for each node in $\mathcal{T}$, we store the sites that lie on the boundary of the geodesic triangle in a binary search tree. This allows us the find the nearest neighbor of v_R for a subset of the sites that lie on the boundary of the geodesic triangle. Third, for each edge in $\mathcal{T}$, which corresponds to a shortest path between two points on $\partial\tilde{P}$, we store a nearest neighbor data structure that allows us to compute the nearest neighbor of v_R among the sites in one of the subpolygons defined by this shortest path, provided that v_R lies in the other subpolygon. We show that we can answer a bounded nearest neighbor query by querying $O(\log \tilde{m})$ of these level two data structures. This results in the following theorem.

▶ **Theorem 3.** *For a static polygon $\tilde{P}$ with $\tilde{m}$ vertices and a set $\tilde{S}$ of $\tilde{n}$ sites in $\tilde{P}$, the Cone Query data structure can be constructed in $O(\tilde{m} \log\log \tilde{n} + \tilde{n} \log \tilde{n} \log\log \tilde{n} + \tilde{n} \log \tilde{n} \log^2 \tilde{m}))$ time and $O(\tilde{n} \log\log \tilde{n} + \tilde{n} \log \tilde{m} + \tilde{m})$ space, such that a bounded nearest neighbor query can be answered in $O(\sqrt{\tilde{n}} \log^{3/2} \tilde{m} + \log \tilde{n} \log^2 \tilde{m})$ expected time.*

To obtain our final result, we rebuild the Cone Query data structure once k becomes larger than $\tilde{n}^{1/4} + \tilde{m}^{1/4}$. This results in update and query times that are $\tilde{O}(n^{3/4} + m^{3/4})$. Note that by rebuilding the Cone Query data structure earlier (or later), we can increase (or decrease) the update time and decrease (or increase) the query time, respectively.

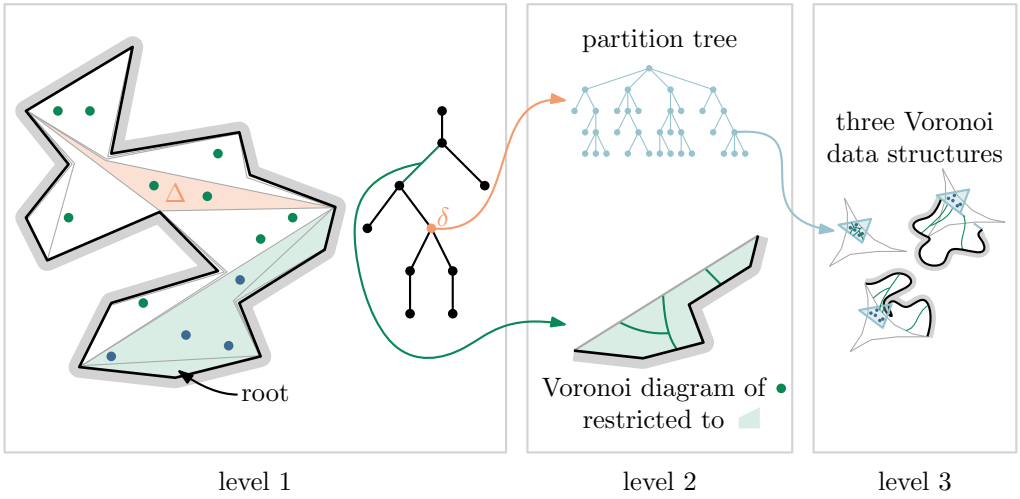


Figure 3 Overview of the Cone Query data structure. For each node in the geodesic triangulation (level 1) we store a partition tree and for each edge we store some restricted Voronoi diagram (level 2). For each node in the partition tree we store three Voronoi based data structures (level 3). Nodes of the geodesic triangulation whos deltoid is empty are omitted.

▶ **Theorem 4.** *We can build an insertion-only data structure for a dynamic simple polygon P with m vertices and a dynamic set S of n sites in P using $O(n(\log\log n + \log m) + m)$ space with $O(n^{3/4}\log n(\log\log n + \log^2 m) + m^{3/4}\log\log n)$ amortized update time that can answer nearest neighbor queries in $O(n^{3/4}\log^{3/2} m + m^{1/4}(n^{1/2}\log^{3/2} m + \log n \log^2 m))$ expected time.*

4 Dynamic shortest path data structure

The Dynamic Shortest Path data structure is an extension of the dynamic data structure of Goodrich and Tamassia [13], discussed in Section 2. We first explain how we adapt it to support efficient access to our newly inserted vertices. Then, we show how to use the Dynamic Shortest Path data structure to generate $O(k)$ bounded nearest neighbor queries that can together answer a nearest neighbor query, thus proving Theorem 2.

Extending the dynamic shortest path data structure. We extend the dynamic shortest path data structure of Goodrich and Tamassia [13] so that we have efficient access to the vertices and edges that were inserted since the last rebuild. We call these *marked* vertices and edges. In the data structure of [13], each polygonal chain is stored in a *chain tree*: a balanced binary tree where leaves correspond to the edges in the chain and internal nodes to the vertices of the chain. Each internal node also corresponds to a subchain and stores its length. We additionally store a boolean in each node that that is true whenever its subchain contains a marked edge, and false otherwise. Next to this dynamic data structure, we keep a list of all marked vertices. For this list, we consider the endpoint of an inserted edge to be marked, even if the vertex already existed in $\tilde{P}$.

▶ **Lemma 5.** *The Dynamic Shortest Path data structure maintains a dynamic simple polygon with $O(\log^2 m)$ update time using $O(m)$ space.*

▶ **Lemma 6.** *Given two points $p, q \in P$, the Dynamic Shortest Path data structure can find the first (or last) marked vertex on $\pi(p, q)$ in $O(\log^2 m)$ time.*

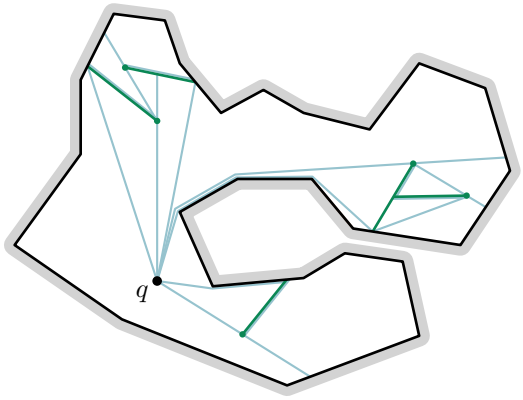


Figure 4 The bounded nearest neighbor query regions that are obtained for the query point q. The green segments are the segments inserted after the last rebuild.

Computing the bounded nearest neighbor queries. Next, we explain how to use the Dynamic Shortest Path data structure to find for a query point $q \in P$ a set of bounded nearest neighbor queries such that the nearest neighbor of q is the result of one of these queries. For each bounded nearest neighbor query we return the apex v_R, the two other endpoints s, t of the shortest paths bounding R, and a boolean indicating whether R is a geodesic triangle or geodesic cone. In case R is a geodesic cone, we order s, t such that the part of ∂P in R is given by the clockwise traversal from s to t along ∂P. We will choose our region R in such way that the paths bounding R are shortest paths in both P and $\tilde{P}$. In other words, there are no marked vertices (that are reflex vertices) on these shortest paths.

▶ **Lemma 7.** *We can compute a set of $O(k)$ bounded nearest neighbor queries in $\tilde{P}$ that allows us to find the nearest neighbor of a point $q \in P$ among the sites in $\tilde{S}$ in $O(k \log^2 m)$ time.*

Proof. We first describe the partition of P that will result in the $O(k)$ bounded nearest neighbor queries, and then discuss how to compute them. We partition P into $O(k)$ maximal regions such that for every point in a region R, the path to q uses the same *marked* vertices, see Figure 4. This can be viewed as a shortest path map for q, where we only consider marked vertices. Each region in this map will be a bounded nearest neighbor query together with its apex, i.e. the first marked vertex on the path from any point in the region to q.

To find these regions, we keep track of all points on ∂P that define the boundary between two such regions in a balanced binary tree T based on the order along ∂P. Observe that all marked vertices define such a boundary point. We thus insert all $O(k)$ marked vertices into T. Then, for each marked vertex v, we:

1. Compute the shortest path $\pi(v, q)$.
2. Perform a ray shooting query in the direction of the last segment of $\pi(v, q)$ to find the point x where the extension of this segment hits ∂P.
3. Insert x into T.

The Dynamic Shortest Path data structure supports shortest path queries in $O(\log^2 m)$ time. We can also obtain the last segment on the shortest path using the chain trees representing the path. The ray shooting query that follows again takes $O(\log^2 m)$ time. Finally, inserting x into T takes just $O(\log k)$ time. As there are $O(k) = O(m)$ marked vertices, this takes $O(k \log^2 m)$ time in total.

What remains is to find the bounded nearest neighbor queries using T. Each pair s, t of consecutive points in T defines such a query. If s and t are endpoints of the same marked edge, then their query region R is a geodesic triangle, otherwise R is a geodesic cone. The apex v_R of the bounded nearest neighbor is the last marked vertex on $\pi(q, s) \cap \pi(q, t)$, or q if there is no such vertex. We distinguish the following cases:

- If s (or t) is a marked, then v_R is the first (other) marked vertex on $\pi(s, q)$ (or $\pi(t, q)$).
- If both s and t are not marked, and thus included in T in the ray shooting step, then v_R is the second marked vertex on $\pi(s, q)$, i.e. the first vertex after the vertex used in the ray shooting step.

Note that in both cases v_R is also on $\pi(t, q)$, as otherwise the ray shooting step on v_R would add some point x between s and t on ∂P to T. Lemma 6 implies that we can find v_R in $O(\log^2 m)$ time (to find the second marked vertex we can search the corresponding chain tree from the first marked vertex in $O(\log m)$ time). ◀

5 Static cone query nearest neighbor data structure

In this section, we present the static Cone Query data structure. Throughout this section, let P be a *static* simple polygon with m vertices and S a *static* set of n sites in P. We want our data structure to answer bounded nearest neighbor queries for a region R and a query point q on ∂R, which in our case will be the apex point v_R, as in Definition 1. We assume that the region R is given as endpoints of the (at most three) shortest paths bounding it, and in case R is a geodesic cone, an indication which part of ∂P is in R.

The base of our data structure is a geodesic triangulation $\mathcal{T}$. We construct several data structures for the nodes and edges of $\mathcal{T}$. We assign each site $s \in S$ to the highest node in $\mathcal{T}$ whose geodesic triangle contains s. Note that there is only one node whose geodesic triangle contains s if s lies in some deltoid region, but there may be more if s is on the boundary of a geodesic triangle. For a geodesic triangle $\delta \in \mathcal{T}$, we denote the set of sites assigned to δ by S_δ. For the deltoid Δ of δ, we define $S_\Delta := S_\delta \cap \Delta$.

As a preprocessing step, we triangulate P [8] and construct the shortest path data structure of Guibas and Hershberger [15]. We also construct a point location data structure [20] on the geodesic triangulation. Each face in this subdivision is associated with its node in $\mathcal{T}$, and each edge is associated with the highest node in $\mathcal{T}$ that has this edge on its boundary. The geodesic triangulation can also be used to answer ray shooting queries in $O(\log m)$ time [9].

Next, we define two fundamental queries for a node in $\mathcal{T}$.

▶ **Definition 8.** *Let δ be a node in $\mathcal{T}$. Let (R, q) be a bounded nearest neighbor query, then we define the following subqueries.*

- Type (a)*: return the nearest neighbor of q in $S_\delta \cap R$.*
- Type (b)*: let $\mathcal{T}_x$ be the subtree of $\mathcal{T}$ rooted at a child x of δ and P_x the corresponding subpolygon, i.e. $P_x := \bigcup_{y \in \mathcal{T}_x} y$. For $q \notin P_x$, return the nearest neighbor of q in $S \cap P_x$.*

For each node $\delta \in \mathcal{T}$, we store two data structures that together can answer type (a) queries. One that finds the nearest neighbor of q among the sites on the boundary of the geodesic triangle, and one that finds the nearest neighbor of q among the sites in the deltoid region. For each edge $(\delta, x) \in \mathcal{T}$ we store a data structure that can answer type (b) queries.

Next, we show that we can answer bounded nearest neighbor queries using $O(\log m)$ type (a) and (b) queries. We first state two lemmas that prove useful in finding these queries.

▶ **Lemma 9.** *Let p, r be two points on ∂P, and let $s \in P$. In $O(\log m)$ time we can test if s lies right of the shortest path $\pi(p, r)$.*

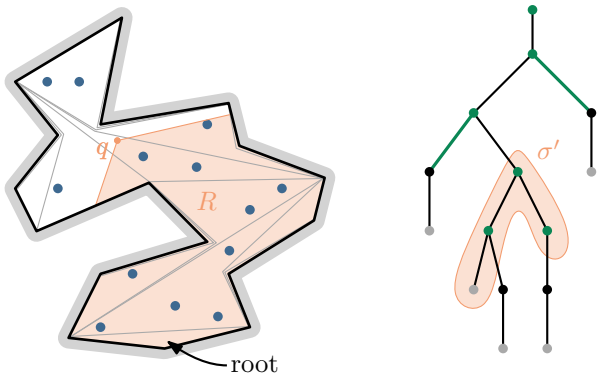


Figure 5 A geodesic triangulation and its dual tree. Nodes with empty deltoid regions are gray. We perform a type (a) query for all green nodes, and a type (b) query for all green edges.

▶ **Lemma 10.** *Let p, r be two points on ∂P, and let $s, t \in P$. We can determine in $O(\log^2 m)$ time the first and last point of $\pi(s,t) \cap \pi(p,r)$, or conclude the paths are disjoint.*

▶ **Lemma 11.** *A bounded nearest neighbor query can be answered using $O(\log m)$ type (a) and (b) queries. We can determine these $O(\log m)$ queries in $O(\log^2 m)$ time.*

Proof. We first prove the first part of the lemma, and then show how to find these type (a) and (b) queries efficiently. Figure 5 illustrates which nodes and edges we query. Let (R, q) be a bounded nearest neighbor query. The region R is either bounded by three shortest paths, or by two shortest path and part of ∂P. We first consider one of these shortest path that bounds R. Let $\pi(s,t)$ be this shortest path, and let $v(s)$ and $v(t)$ be geodesic triangles in $\mathcal{T}$ that contain s and t respectively. Let σ be the subset of nodes of $\mathcal{T}$ on the paths from $v(s)$ and $v(t)$ to the root of $\mathcal{T}$. Furthermore, let P_σ be the union of the geodesic triangles in σ, i.e. $P_\sigma = \bigcup_{\delta \in \sigma} \delta$. We claim that $\pi(s,t)$ is contained P_σ. Indeed, consider a deltoid region Δ that is entered by $\pi(s,t)$ of a geodesic triangle $\delta \notin \sigma$. Let $\pi(a,b)$ be the shortest path that is the boundary of δ through which $\pi(s,t)$ enters Δ. Note that by definition a and b are on ∂P. So, because $\pi(s,t)$ starts and ends in P_σ, it should at some point reenter P_σ by crossing $\pi(a,b)$ again. This contradicts $\pi(s,t)$ being a shortest path.

Let σ' be the the union of the σ sets for the shortest paths bounding R, excluding any geodesic triangles that intersect none of the shortest paths bounding R. Observe that for any point $p \in P$ the subgraph of nodes in $\mathcal{T}$ whose geodesic triangle contains p is connected. As each pair of shortest paths bounding R has a common endpoint, the subgraph of σ' is also connected. For each $\delta \in \sigma'$, we perform a type (a) query. Because $\mathcal{T}$ has height $O(\log m)$, we perform $O(\log m)$ of these queries. Together, these queries find the nearest neighbor of q in R for all geodesic triangles whose intersection with one of the shortest paths bounding R is non-empty. Clearly, any node whose deltoid region is intersected by such a shortest path is in σ'. Any site that is on the boundary of a geodesic triangle intersected by such a shortest path is stored in the highest node in $\mathcal{T}$ that contains the site. As this subset of nodes is connected, some node in σ' must contain this site.

What remains is to query the geodesic triangles that are fully contained in R. If R is a geodesic triangle, no node in $\mathcal{T}$ can be contained in the interior of R, as there are no vertices of P in its interior, and the corners of the geodesic triangle are vertices of P. If R is a geodesic cone, there might be $O(m)$ nodes in $\mathcal{T}$ contained inside R that do not intersect the two bounding paths. Because R is connected, there is an edge (δ, x) in $\mathcal{T}$, with δ the parent of x, whose removal splits $\mathcal{T}$ into two subtrees, one of which contains exactly the

geodesic triangles that are fully contained in R. If this subtree is $\mathcal{T}_x$ (recall that $\mathcal{T}_x$ is the subtree rooted at x), then we perform a type (b) query on (δ, x). If the subtree is $\mathcal{T} \setminus \mathcal{T}_x$, then for each node on the path from δ to the root, we perform a type (a) query and a type (b) query for the incident edge not on this path.

Finally, we prove the second part of the lemma. We first compute the set σ'. Consider one of the shortest paths $\pi(s,t)$ that bounds R. We first perform a point location query for both s and t. This gives us two nodes $v(s), v(t)$ in $\mathcal{T}$ that contain s and t. We then find the lowest common ancestor w of $v(s)$ and $v(t)$ in $O(\log m)$ time. All nodes on the paths from $v(s)$ and $v(t)$ to w are in σ'. Note that the previous argument that $\pi(s,t)$ cannot cross a geodesic triangle boundary and return even implies that $\pi(s,t)$ is contained in the subpolygon correspoding to these nodes. However, there may be more nodes on the path from w to the root of $\mathcal{T}$ whose boundary intersects $\pi(s,t)$. Let $\pi(p,r)$ be the shortest path that is the boundary between w and its parent geodesic triangle. Lemma 10 implies that we can find the first point a and the last point b of $\pi(s,t) \cap \pi(p,r)$, or conclude the paths are disjoint, in $O(\log^2 m)$ time. Any geodesic triangle in $\mathcal{T}$ above w that intersects $\pi(s,t)$ intersects either a or b (or both). It follows we can find the highest node of $\mathcal{T}$ that intersects $\pi(s,t)$ by a point location query on a and b, and choosing the highest node of the returned nodes. All nodes on the path in $\mathcal{T}$ from w and to this node are in σ'. By following the same procedure for all shortest path bounding R, we can compute σ' in $O(\log^2 m)$ time.

If R is a geodesic triangle, we are finished. If R is instead a geodesic cone, we need to find the edge whose removal splits off the fully contained geodesic triangles as described earlier. Note that this edge must be incident to σ'. Let s and t be the endpoints other than q of the shortest paths that bound R. We first find the subsequence of polygon vertices contained in R by a point location query for s and t. By storing the boundary of P in a balanced tree, we can then find the subsequence of vertices between the edges containing s and t in $O(\log m)$ time. For each node in σ', we check whether the incident edge that is not in σ' is contained in R by checking whether its endpoints are contained in the subsequence in $O(\log m)$ time. We thus find the desired edge of $\mathcal{T}$ in $O(\log^2 m)$ time. The required type (a) and (b) queries can then be found by traversing the tree as described before in $O(\log m)$ time. ◀

5.1 Second level data structures

For each node $\delta \in \mathcal{T}$, we store two data structures that together can answer type (a) queries, and for each edge (δ, x) we store a data structure that can answer type (b) queries.

Data structures for type (a) queries. Fix a node $\delta \in \mathcal{T}$ and let Δ be its deltoid. For each shortest path bounding δ, we store the sites of S_δ that are on the path in a binary search tree based on the order along the path. Together, these trees contain the sites in $S_\delta \setminus S_\Delta$.

On the sites in S_Δ we construct Chan's partition tree [6]. Each leaf node stores a constant number of sites, and each site is stored in exactly one leaf. For each node μ in the partition tree, let $S(\mu)$ denote the sites stored in the leaves below μ and let $\blacktriangle(\mu)$ denote the triangular cell of μ that contains $S(\mu)$. We store the following data structures for μ, see Figure 6:

1. The geodesic Voronoi diagram of the sites $S(\mu)$ with respect to the polygon $\blacktriangle(\mu) \cap \Delta$.
2. For each segment bounding $\blacktriangle(\mu)$: the forest representing the Voronoi diagram of $S(\mu)$ restricted to the subpolygon bounded by ∂P and the chord that contains the segment that does not contain $S(\mu)$.
3. For each shortest path π bounding δ: the forest representing the Voronoi diagram of $S(\mu)$ restricted to the subpolygon bounded by ∂P and π that does not contain $S(\mu)$.

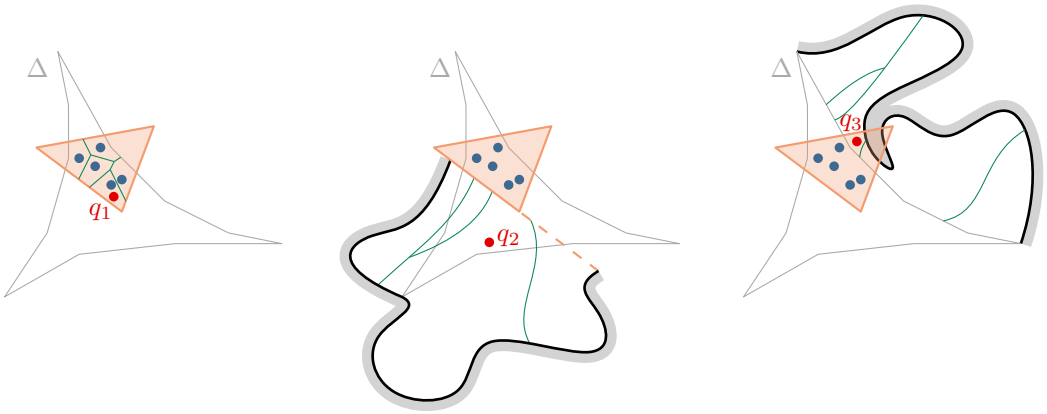


Figure 6 The three data structures stored for the orange cell $\blacktriangle(\mu)$ for answering type (a) queries. The first is used for a query point inside $\blacktriangle(\mu) \cap \Delta$, the second for a query point outside $\blacktriangle(\mu)$, and the third for a query point that is in $\blacktriangle(\mu)$ but outside of Δ.

The first data structure is simply the standard geodesic Voronoi diagram [22]. For the second and third data structure, we adapt a data structure that Agarwal, Arge, and Staals [1] use in their dynamic nearest neighbor data structure. They essentially prove that when a polygon is partitioned into two subpolygons P_ℓ, P_r by a line segment, the forest representing the Voronoi diagram of the sites in P_ℓ restricted to P_r can be stored and queried efficiently. In the full version [12], we generalize this result to the case where the polygon is partitioned using an arbitrary shortest path instead of a line segment, and obtain the following result.

▶ **Theorem 12.** *Let P be a polygon with m vertices appropriately preprocessed for two-point shortest path and ray-shooting queries. Let P_ℓ, P_r be a partition of P into two subpolygons by a shortest path between two points on ∂P. Given a set of n sites S in P_ℓ, there is an $O(n)$ size data structure storing the combinatorial representation of the Voronoi diagram of S in P_r so that given a query point $q \in P_r$, we can find the site in S closest to q in $O(\log n \log m)$ time. Building the data structure takes $O(n \log^2 m + n \log n)$ time.*

Data structure for type (b) queries. Let δ be a node in $\mathcal{T}$ and x be a child of δ. To be able to answer type (b) queries we store Voronoi diagram of the sites in $S \cap P_x$ restricted to $P \setminus P_x$, for P_x as in Definition 8, in the forest data structure of Theorem 12, as the boundary between the two geodesic triangles is a shortest path.

5.2 Analysis

Next, we prove that the Cone Query data structure achieves the bounds in Theorem 3. We use the following lemma about multilevel data structures using Chan's partition tree.

▶ **Lemma 13** (Corollary 6.2 of [6]). *Given n points in $\mathbb{R}^2$, for any fixed $\gamma < 1/2$, we can form $O(n)$ canonical subsets of total size $O(n \log\log n)$ in $O(n \log n)$ time, such that the subset of all points inside any query triangle can be reported as a union of disjoint canonical subsets C_i with $\sum_i |C_i|^\gamma \leq O(\sqrt{n})$ in time $O(\sqrt{n})$ w.h.p.(n).*

Query time. Let (R, q) be a bounded nearest neighbor query. Lemma 11 states that we can answer the query using $O(\log m)$ type (a) and (b) queries and that we can compute these queries in $O(\log^2 m)$ time. What remain is to analyze the time of a type (a) and (b) query.

First, consider a type (a) query. Recall that this asks for the nearest neighbor of q in $S_\delta \cap R$ for a geodesic triangle $\delta \in \mathcal{T}$ with deltoid Δ. To answer this query we have to query the three binary trees storing the sites in $S_\delta \setminus S_\Delta$ and the partition tree on S_Δ. For a shortest path of $\partial\delta$, the sites on this path that are contained in R form a contiguous subsequence. We can thus obtain this subset of sites in a binary tree in $O(\log n)$ time. Because the distance from q to a shortest path is a convex function, we can then binary search in this subtree to find the site closest to q in $O(\log n \log m)$ time.

To find the closest site of q in S_Δ, we query the partition tree. However, we cannot apply Lemma 13 directly, as this requires the query region to be a triangle instead of a geodesic triangle or cone. Oh and Ahn [23] essentially prove that seven triangles can cover a geodesic triangle without including any sites of S_Δ outside of the geodesic triangle in their interior. Here, we need a slightly more general statement that also includes geodesic cones.

▶ **Lemma 14.** *We can construct in $O(\log^2 m)$ time a constant number of interior disjoint triangles whose union contains $\Delta \cap R$ but does not contain any sites in $S_\Delta \setminus R$.*

For each of the $O(1)$ triangles given by Lemma 14 we query the partition tree. Lemma 13 states that the partition tree gives us a set of disjoint canonical subsets C_i. For each canonical subset, which corresponds to some node μ in the partition tree, we perform a nearest neighbor query using one of the data structures stored in the node. Which data structure we query depends on the location of q in relation to Δ and $\blacktriangle(\mu)$, see Figure 6.

- If q is inside both the deltoid Δ and the cell $\blacktriangle(\mu)$, i.e. $q \in \Delta \cap \blacktriangle(\mu)$, we query the geodesic Voronoi diagram of $\Delta \cap \blacktriangle(\mu)$.
- If q is outside the cell $\blacktriangle(\mu)$, i.e. $q \notin \blacktriangle(\mu)$, we query the forest constructed for the segment of $\blacktriangle(\mu)$ whose extension separates q from $\blacktriangle(\mu)$.
- If q is inside the cell $\blacktriangle(\mu)$ but outside the deltoid Δ, i.e. $q \in \blacktriangle(\mu)$ and $q \notin \Delta$, we query the forest constructed for side of δ that separates q from Δ.

Consider a canonical subset C_i for which the complexity of Δ restricted to the cell of C_i is m_i. The first type of query, which is in the (regular) geodesic Voronoi diagram, takes $O(\log(|C_i| + m_i))$ time [22]. The second and third type of query take $O(\log |C_i| \log m)$ time by Theorem 12. The query time thus becomes

$$\begin{aligned}\sum_i (\log |C_i| \log m + \log(|C_i| + m_i)) &= \log m \sum_i \log |C_i| + \sum_i \log(|C_i| + m_i) \\ &\leq \log m \sum_i \log |C_i| + \sum_i (\log |C_i| + \log m_i) \\ &= (\log m + 1) \sum_i \log |C_i| + \sum_i \log m.\end{aligned}$$

Here we used that for $|C_i|, m_i \geq 2$ it holds that $\log(|C_i| + m_i) \leq \log(|C_i| m_i)$. Choosing any $\gamma \in (0, 1/2)$ in Lemma 13 implies that $\sum_i \log |C_i| = O(\sqrt{|S_\Delta|})$. The number of terms in the sum $\sum_i \log m$ is bounded by $O(\sqrt{|S_\Delta|})$ as well. A type (a) query can thus be answered in $O(\sqrt{|S_\Delta|} \log m)$ time.

Finally, we note that the $O(\log m)$ type (a) queries we perform are on distinct deltoids. Thus, if we sum over all deltoids Δ_i that we query, we have $\sum_i |S_{\Delta_i}| = O(n)$. To bound the sum of square roots over these set sizes, we use the HM-GM-AM-QM inequalities, which states that for positive reals $x_1, \ldots, x_a$, we have $x_1 + \cdots + x_a \leq \sqrt{a(x_1^2 + \cdots + x_a^2)}$. It then follows that the total query time of all type (a) queries is $\sum_{i=0}^{O(\log m)} \sqrt{|S_{\Delta_i}|} \log m = O(\sqrt{n} \log^{3/2} m)$.

A type (b) query requires only a single query of the data structure of Theorem 12, and thus takes $O(\log n \log m)$ time. The total query time of the type (b) queries is $O(\log n \log^2 m)$.

Construction time. As preprocessing, we build the shortest path data structure of Guibas and Hershberger in $O(m)$ time [8, 15]. We then build the level one data structure: a balanced geodesic triangulation together with a point location data structure, in $O(m)$ time [9].

There are three level two data structures: one for each edge of $\mathcal{T}$ (for the type (b) queries), and two for each node of $\mathcal{T}$ (for the type (a) queries). Let δ be a node in $\mathcal{T}$ and Δ the corresponding deltoid.

First, consider the data structure on each edge of $\mathcal{T}$. This is the forest data structure of Theorem 12. For a child x of δ this data structure can be build in $O(|S \cap P_x| \log^2 m + |S \cap P_x| \log |S \cap P_x|)$ time, with P_x as in Definition 8. Note that each site in S can occur in at most $O(\log m)$ polygons P_x, once for each level of $\mathcal{T}$. It follows that constructing the data structure for each edge of $\mathcal{T}$ takes $O(n \log^3 m + n \log n \log m)$ time in total.

Second, consider the two data structures on the nodes of $\mathcal{T}$. For each shortest path bounding δ, we store the sites on the path in a binary search tree. These trees can be constructed in $O(n_\delta(\log n_\delta + \log m))$ time. Furthermore, we store the sites in the deltoid Δ in a partition tree. The partition tree (excluding auxiliary data structures) can be build in $O(n_\Delta \log n_\Delta)$ time (Lemma 13).

For each node μ of the partition tree, we store three third level Voronoi data structures for the sites $S(\mu)$. Let $m(\mu)$ be the number of vertices of $\blacktriangle(\mu) \cap \Delta$. The first level three data structure is a (regular) geodesic Voronoi diagram restricted to μ, including a point location data structure, and can be constructed in $O(m(\mu) + |S(\mu)| \log |S(\mu)|)$ time [22]. The second and third data structures are the forest data structure of Theorem 12 on each segment bounding the cell of μ and each shortest path bounding δ. These can be constructed in $O(|S(\mu)| \log^2 m + |S(\mu)| \log |S(\mu)|)$ time (Theorem 12). The total construction time for a node μ of the partition tree is then $O(m(\mu) + |S(\mu)| \log |S(\mu)| + |S(\mu)| \log^2 m)$. To bound the total construction time for all nodes of the partition tree, observe that the cells in each of the $O(\log \log n_\Delta)$ levels of the partition tree are disjoint. So, $\sum_\mu |m(\mu)| = O(m_\Delta \log \log n_\Delta)$. Furthermore, $\sum_\mu |S(\mu)| = O(n_\Delta \log \log n_\Delta)$. The total construction time over all nodes is then $O(m_\Delta \log \log n_\Delta + n_\Delta \log \log n_\Delta (\log n_\Delta + \log^2 m))$.

Together, the level two and three data structures on a node $\delta \in \mathcal{T}$ are constructed in $O(m_\delta \log \log n_\delta + n_\delta \log \log n_\delta (\log n_\delta + \log^2 m))$ time. This implies that the total construction time for all nodes in $\mathcal{T}$ is $O(m \log \log n + n(\log n \log \log n + \log^2 m \log \log n))$.

The total construction time of all first, second, and third level data structures then becomes $O(m \log \log n + n \log n \log \log n + n \log^3 m + n \log n \log^2 m + n \log n \log m) = O(m \log \log n + n \log n \log \log n + n \log n \log^2 m)$.

Space usage. The space of the shortest path data structure on P is $O(m)$. For a node $\delta \in \mathcal{T}$ and Δ the corresponding deltoid, the trees for $\partial\delta$ have size $O(n_\delta)$. As for the partition tree, Lemma 13 states that the total size of the canonical subsets is $O(n_\Delta \log \log n_\Delta)$. The three data structures that are built for each node of the partition tree use just linear space in the number of sites, thus the total size of the partition tree including its auxiliary data structures remains $O(n_\Delta \log \log n_\Delta)$. The total space over all nodes in $\mathcal{T}$ is thus $O(n \log \log n)$. What remains is to analyze the space of the data structures for type (b) queries. Theorem 12 states that this forest data structure for a child x of δ uses $O(|S \cap P_x|)$ space. As before, each site in S can occur in at most $O(\log m)$ polygons P_x, once for each level of $\mathcal{T}$. Thus the total space of these data structures is $O(n \log m)$.

References

1 Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polygon. In *34th International Symposium on Computational Geometry, SoCG*, 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 Micha Sharir. Applications of a new space-partitioning technique. *Discret. Comput. Geom.*, 9:11–38, 1993. doi:10.1007/BF02189304.

3 Boris Aronov. On the Geodesic Voronoi Diagram of Point Sites in a Simple Polygon. *Algorithmica*, 4(1):109–140, 1989. doi:10.1007/BF01553882.

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 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.

6 Timothy M. Chan. Optimal partition trees. *Discret. Comput. Geom.*, 47(4):661–690, 2012. doi:10.1007/S00454-012-9410-Z.

7 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.

8 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discret. Comput. Geom.*, 6:485–524, 1991. doi:10.1007/BF02574703.

9 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas J. Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994. doi:10.1007/BF01377183.

10 Tameem Choudhury and R. Inkulu. Maintaining the visibility graph of a dynamic simple polygon. In *5th International Conference on Algorithms and Discrete Applied Mathematics, CALDAM*, volume 11394 of *Lecture Notes in Computer Science*, pages 42–52. Springer, 2019. doi:10.1007/978-3-030-11509-8_4.

11 Sarita de Berg and Frank Staals. Dynamic data structures for k-nearest neighbor queries. *Comput. Geom.*, 111:101976, 2023. doi:10.1016/J.COMGEO.2022.101976.

12 Sarita de Berg and Frank Staals. Nearest neighbor searching in a dynamic simple polygon, 2025. doi:10.48550/arXiv.2503.03435.

13 Michael T. Goodrich and Roberto Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *J. Algorithms*, 23(1):51–73, 1997. doi:10.1006/JAGM.1995.0797.

14 Michael T. Goodrich and Roberto Tamassia. Dynamic trees and dynamic point location. *SIAM J. Comput.*, 28(2):612–636, 1998. doi:10.1137/S0097539793254376.

15 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.

16 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.

17 Mashhood Ishaque, Bettina Speckmann, and Csaba D. Tóth. Shooting permanent rays among disjoint polygons in the plane. *SIAM J. Comput.*, 41(4):1005–1027, 2012. doi:10.1137/100804310.

18 Mashhood Ishaque and Csaba D. Tóth. Relative convex hulls in semi-dynamic arrangements. *Algorithmica*, 68(2):448–482, 2014. doi:10.1007/S00453-012-9679-6.

19 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.

20 David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983. doi:10.1137/0212002.

21 Chih-Hung Liu. A nearly optimal algorithm for the geodesic voronoi diagram of points in a simple polygon. *Algorithmica*, 82(4):915–937, 2020. doi:10.1007/S00453-019-00624-2.

22 Eunjin Oh. Optimal algorithm for geodesic nearest-point voronoi diagrams in simple polygons. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 391–409. SIAM, 2019. doi:10.1137/1.9781611975482.25.

23 Eunjin Oh and Hee-Kap Ahn. Dynamic geodesic convex hulls in dynamic simple polygons. In *33rd International Symposium on Computational Geometry, SoCG*, volume 77 of *LIPIcs*, pages 51:1–51:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.51.

24 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.

25 Mark H. Overmars. *The Design of Dynamic Data Structures*, volume 156 of *Lecture Notes in Computer Science*. Springer, 1983. doi:10.1007/BFB0014927.

26 Evanthia Papadopoulou and Der-Tsai Lee. A New Approach for the Geodesic Voronoi Diagram of Points in a Simple Polygon and Other Restricted Polygonal Domains. *Algorithmica*, 20(4):319–352, 1998. doi:10.1007/PL00009199.

An Algorithm for Tambara-Yamagami Quantum Invariants of 3-Manifolds, Parameterized by the First Betti Number

Colleen Delaney
Department of Mathematics, Department of Physics and Astronomy, Purdue University, West Lafayette, IN, USA

Clément Maria
INRIA Université Côte d'Azur, Nice, France
FGV/EMAp, Rio de Janeiro, Brazil

Eric Samperton
Department of Mathematics, Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

Quantum topology provides various frameworks for defining and computing invariants of manifolds inspired by quantum theory. One such framework of substantial interest in both mathematics and physics is the Turaev-Viro-Barrett-Westbury state sum construction, which uses the data of a spherical fusion category to define topological invariants of triangulated 3-manifolds via tensor network contractions. In this work we analyze the computational complexity of state sum invariants of 3-manifolds derived from Tambara-Yamagami categories. While these categories are the simplest source of state sum invariants beyond finite abelian groups (whose invariants can be computed in polynomial time) their computational complexities are yet to be fully understood. We first establish that the invariants arising from even the smallest Tambara-Yamagami categories are #P-hard to compute, so that one expects the same to be true of the whole family. Our main result is then the existence of a fixed parameter tractable algorithm to compute these 3-manifold invariants, where the parameter is the first Betti number of the 3-manifold with $\mathbb{Z}/2\mathbb{Z}$ coefficients.

Contrary to other domains of computational topology, such as graphs on surfaces, very few hard problems in 3-manifold topology are known to admit FPT algorithms with a topological parameter. However, such algorithms are of particular interest as their complexity depends only polynomially on the combinatorial representation of the input, regardless of size or combinatorial width. Additionally, in the case of Betti numbers, the parameter itself is computable in polynomial time. Thus while one generally expects quantum invariants to be hard to compute classically, our results suggest that the hardness of computing state sum invariants from Tambara-Yamagami categories arises from classical 3-manifold topology rather than the quantum nature of the algebraic input.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Theory of computation → Computational geometry; Theory of computation → Fixed parameter tractability

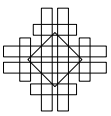
Keywords and phrases 3-manifold, quantum invariant, fixed parameter tractable algorithm, topological parameter, Gauss sums, topological quantum field theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.38

Related Version *Full Version*: `https://arxiv.org/pdf/2311.08514` [8]

Funding *Colleen Delaney*: Partially supported through the Simons Collaboration on Global Categorical Symmetries (Simons Foundation Award 888988).
Clément Maria: Partially supported by the ANR project ANR-20-CE48-0007 (AlgoKnot).
Eric Samperton: Partially supported by NSF award DMS-2038020.

© Colleen Delaney, Clément Maria, and Eric Samperton;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

Quantum physics – especially quantum field theory – is a profuse source of invariants in low-dimensional topology, and these invariants often provide information about manifolds beyond the reach of classical topology. For example, on one hand, the Alexander polynomial of a knot is determined "classically" (from the first homology of the knot's universal abelian cover) and infinitely many different knots have trivial Alexander polynomial; on the other hand, the Jones polynomial (the prototypical example of a quantum invariant) has never been shown to be trivial on any non-trivial knot. However, the native paradigm for (approximately) computing quantum invariants is manifestly quantum computational, so that in general one should expect them to be hard to compute on a classical computer [19]. Naturally then we seek to understand the full landscape of the computational complexities of quantum invariants. From the point of view of computational topology we are particularly interested in a middle ground where there may be interesting invariants that are not too hard to compute, and to develop algorithms to compute them as efficiently as possible.

In this work, we investigate the complexity landscape of a specific class of quantum invariants of 3-manifolds, namely, the *Turaev-Viro-Barrett-Westbury (TVBW) state sum construction* [30, 2]. Unlike other quantum invariants (e.g. Donaldson invariants), the TVBW construction is *a priori* algorithmically computable, combining a finite amount of algebraic data in the form of a *(skeletalized) spherical fusion category* $\mathcal{C}$ with a triangulation $\mathfrak{T}$ of a closed, oriented 3-manifold M to generate a complex algebraic number $|M|_{\mathcal{C}} \in \mathbb{C}$ that is an orientation-preserving homeomorphism invariant of M. The invariant $|M|_{\mathcal{C}}$ is defined as the contraction of a certain tensor network built by decorating $\mathfrak{T}$ with the data of $\mathcal{C}$, and can be formally described via a combinatorial *state sum formula*. We refer the reader to the full version of this article [8] for a brief review of the TVBW construction (in the case of multiplicity-free, pseudo-unitary spherical categories) and [32] for more background.

As we review at the end of this introduction, it is desirable to understand the computational complexity of evaluating the TVBW invariant $|\cdot|_{\mathcal{C}}$ on a given triangulated 3-manifold

$$|\cdot|_{\mathcal{C}} : \{\text{triangulations of closed, oriented 3-manifolds}\} \to \mathbb{C}$$
$$\text{triangulation } \mathfrak{T} \text{ of } M \mapsto |M|_{\mathcal{C}}.$$

More precisely, we would like to understand how the computation of these invariants depends on the choice of spherical fusion category $\mathcal{C}$, since there are countably infinitely many equivalence classes of such categories, and their classification is at least as hard as the classification of finite groups. As usual, we aim to understand the complexity of computing $|M|_{\mathcal{C}}$ as a function of the size of a triangulation $\mathfrak{T}$ used to encode M.

We approach this complexity-theoretic classification problem from the perspective that fusion categories are "quantum" generalizations of (the representation theory of) finite groups. While this perspective has not been made explicitly overt in the computational topology literature, it is common in quantum algebra, where the extent to which a fusion category deviates from a finite group G can be made precise in various ways. For the sake of space, we will not review these notions here, but will note that anyway one cuts it, the simplest spherical fusion categories are of the type $\mathrm{Vec}(A)$ – where A is an abelian group – consisting of A-graded finite-dimensional vector spaces, with a tensor product that comes from the group operation in A. In fact, for any finite group G, the category of G-graded vector spaces $\mathrm{Vec}(G)$ forms a spherical fusion category, but if G is non-abelian, we consider this category to be more "complicated" than $\mathrm{Vec}(A)$. One reason for this is that $|\cdot|_{\mathrm{Vec}(A)}$ is always polynomial-time computable, whereas $|\cdot|_{\mathrm{Vec}(G)}$ is often $\#P$-hard for non-abelian groups [21].

Arguably, after $\mathrm{Vec}(A)$, the next simplest family of fusion categories consists of the Tambara-Yamagami categories $\mathrm{TY}(A,\chi,\nu)$. Such a category is determined by a finite abelian group A, together with a small amount of additional data: a *bicharacter* (*i.e.*, a non-degenerate symmetric bilinear pairing) $\chi : A \times A \to U(1)$, and a choice of square root $\nu = \pm 1/\sqrt{|A|} = \pm |A|^{-1/2}$ [28]. (See [8] for a complete description of these categories using this data.) And yet, in the full version of this article [8], we show that already in the simplest non-trivial case with $A = \mathbb{Z}/2\mathbb{Z}$, the invariants $|\cdot|_{\mathrm{TY}(\mathbb{Z}/2\mathbb{Z},\chi,-1/\sqrt{2})}$ are $\#P$-hard to compute. This is more-or-less a restatement of known results about the Turaev-Viro invariant TV_4 [14, 23], since $|\cdot|_{\mathrm{TY}(\mathbb{Z}/2\mathbb{Z},\chi,1/\sqrt{2})} = \mathrm{TV}_4(\cdot)$. However, as we shall explain, our approach offers a conceptual clarification by placing TV_4 within the broader taxonomy of spherical fusion categories. In particular, we conjecture that the TVBW state sum invariant of *every* non-trivial Tambara-Yamagami category $\mathrm{TY}(A,\chi,\nu)$ is #P-hard.

Our main result is a parameterized algorithm to compute the TVBW state sum invariants $|M|_{\mathrm{TY}(A,\chi,\nu)}$ associated to Tambara-Yamagami categories, where our parameter is the first $\mathbb{Z}/2\mathbb{Z}$ Betti number β_1.

▶ **Main Result** (Informal; see Theorem 4). *Fix an integer $B \geq 0$ and a Tambara-Yamagami category $\mathrm{TY}(A,\chi,\nu)$. Then there exists a polynomial time algorithm to evaluate $|M|_{\mathrm{TY}(A,\chi,\nu)}$ for triangulated 3-manifolds with $\beta_1(M) \leq B$.*

While the hardness of general Tambara-Yamagami invariants is still conjectural, we interpret our main result as showing that any such hardness is explainable by a simple classical topological fact: there exist triangulations $\mathfrak{T}$ of 3-manifolds M with n simplices where $\beta_1(M)$ can be as large as $\Theta(n)$.

Our algorithm succeeds by converting the natively quantum topological computational problem of computing a TVBW state sum invariant of a triangulated 3-manifold – which *a priori* requires an expensive tensor network contraction – into a sequence of other much more efficient classical computational problems in algebraic number theory and combinatorial-algebraic topology. Prior to our work, to the best of our knowledge, the only published algorithm in 3-manifold topology directly parameterized by an efficiently computable *topological* quantity (whose value is independent of the presentation of the input) is the algorithm of Maria and Spreer to compute the Turaev-Viro quantum invariant TV_4 of 3-manifolds [30] at a 4th root of unity in $O(2^{\beta_1} n^3)$ operations [23].

The technical underpinnings of our algorithm are explained in Section 3; the algorithm itself is precisely described and analyzed in Section 4. At the heart of our algorithm is a (non-obvious!) observation: $|M|_{\mathrm{TY}(A,\chi,\nu)}$ can be identified with a (normalized) sum of $|H^1(M,\mathbb{Z}/2\mathbb{Z})| = 2^{\beta_1}$ many *Gauss sums of $\mathbb{Q}/\mathbb{Z}$-valued quadratic forms* on certain finite abelian groups derived from A and the given triangulation of M.

$$
\begin{aligned}
|M|_{\mathrm{TY}(A,\chi,\nu)} &= \sum_{\theta \in \mathrm{Adm}(\mathfrak{T})} |\mathfrak{T}|_\theta && \text{(def. of TVBW state sum invariant; Sec. 2.3)} \\
&= \sum_{\alpha \in Z^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})} \underbrace{\sum_{\theta \in \mathrm{Adm}(\mathfrak{T},\alpha)} |\mathfrak{T}|_\theta}_{\substack{\textit{partial state sum at} \\ \textit{cocycle } \alpha\text{; Sec. 3.2}}} && \text{($\mathbb{Z}/2\mathbb{Z}$-grading of $\mathrm{TY}(A,\chi,\nu)$; Sec. 3.1)} \\
&= \#B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z}) \cdot \sum_{[\alpha] \in H^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})} \underbrace{\sum_{\theta \in \mathrm{Adm}(\mathfrak{T},\alpha)} |\mathfrak{T}|_\theta}_{\substack{(\star)\ \text{poly. time computable as a} \\ \text{quadratic Gauss sum}}} && \text{(follows from [31]; Sec. 3.5)}
\end{aligned}
$$

spherical category $\mathcal{C}$	complexity of $\lvert M\rvert_{\mathcal{C}}$	FPT in β_1?	Reference
$\mathrm{Vec}(A)$ for A abelian	in fP	Yes	well-known, e.g. [20]
$\mathrm{TY}(\mathbb{Z}/2\mathbb{Z}, \exp(\pi iab), -1/\sqrt{2})$	$\#\mathsf{P}$-hard	Yes	[14, 23], long version [8]
$\mathrm{TY}(A, \chi, \nu)$	unknown in general	Yes	this work
$\mathrm{Vec}(G)$ for G non-abelian solvable	unknown	unknown	
$\mathrm{Vec}(G)$ for G non-abelian simple	$\#\mathsf{P}$-complete	No (unless $\mathsf{fP} = \#\mathsf{P}$)	[20]
$U_q\mathfrak{sl}_2$-mod (q a non-lattice root of 1)	$\#\mathsf{P}$-hard	No (unless $\mathsf{fP} = \#\mathsf{P}$)	[13, 19]

Figure 1 State of the art for computing TVBW invariants based on "simple" spherical fusion categories.

The second and third equalities above follow for a rather general reason: $\mathrm{TY}(A, \chi, \nu)$ possesses the structure of a $\mathbb{Z}/2\mathbb{Z}$-*grading* and so by general facts related to *homotopy quantum field theory* [31], we are able to reduce $\lvert M\rvert_{\mathrm{TY}(A,\chi,\nu)}$ to a sum of 2^{β_1} terms in a manner similar in spirit to [23]. Notably, however, the concept of *graded fusion categories* is absent from [23], and this structure enables us to analyze the state sum invariant of M directly from *any* simplicial triangulation, without having to pass to a one-vertex triangulation via the crushing algorithm of Jaco and Rubinstein [12] (as in [23]).

The bulk of Section 3 is then concerned with explaining and justifying ($\star$), and our methods here diverge substantially from [23]. The efficient algorithmic evaluation of quadratic Gauss sums appears to be well-known to experts in algebraic number theory, although we were unable to find specific references for this fact, the closest apparently being [3]; see Section 2.1 and [8] for further discussion.

The general nature of our main result allows us to establish new conceptual insight into the complexity-theoretic landscape of TVBW invariants, a portion of which we summarize in Figure 1. It is known that TVBW 3-manifold invariants are often #P-hard to compute exactly, and sometimes hard even just to approximate [13, 14, 18, 19, 20, 21]; moreover, in many of these cases, it is known that there can be no tractable algorithm in β_1. Thus the existence of a parametrized algorithm in a topological and efficiently computable parameter for the Tambara-Yamagami invariants is an interesting counterpoint to their expected hardness. This suggests that the hardness of computing the invariants is due to some intrinsic hardness in 3-manifold topology, rather than the Tambara-Yamagami category alone, whose construction is only a small modification of $\mathrm{Vec}(A)$.

After scrutinizing Figure 1, it seems sensible to expect that there should be a dichotomy theorem for TVBW invariants. A naïve conjecture would be that $\lvert\cdot\rvert_{\mathcal{C}}$ is in fP if and only if $\mathcal{C} \simeq \mathrm{Vec}(A)$, and otherwise $\lvert\cdot\rvert_{\mathcal{C}}$ is $\#\mathsf{P}$-hard. However, this cannot be correct, as there are "twisted" versions of A-graded vector spaces $\mathrm{Vec}(A, \eta)$, $\eta \in H^3(A, U(1))$, for which $\lvert\cdot\rvert_{\mathrm{Vec}(A,\eta)}$ is also known to be polynomial-time computable. More subtly, if G is any 2-step nilpotent group, then $\lvert M\rvert_{\mathrm{Vec}(G)} = \#\{\pi_1(M) \to G\}/\#G$ should also be in fP.

A systematic understanding of the computational complexity of state sum invariants is significant because they fit into the larger framework of a fully-extended 3D TQFT. Indeed, this connection endows the invariants with important applications to *fault-tolerant quantum computation* and the characterization of *topologically ordered phases of matter*, e.g. via *topological quantum computing* [16, 17, 26, 34], as well as to the developing paradigm of "non-invertible" or "categorical" symmetries and dualities present in *quantum field theories* and *lattice models* [1, 6, 9, 10, 29]. Of course, there is plenty of motivation from low-dimensional topology as well, where these invariants can be used in practice to distinguish non-homeomorphic 3-manifolds. TVBW invariants are heuristically strong, with a typical category $\mathcal{C}$ capable of distinguishing "most" pairs of non-homeomorphic manifolds.

2 Background

In this section we introduce the TVBW invariants of 3-manifolds derived from Tambara-Yamagami categories, as well as some of the basic components necessary for our algorithm.

2.1 Elements of algorithmic number theory

Finite abelian groups. Let $(A,+)$ be a *finite abelian group* with $|A|$ elements, denoted additively. The fundamental theorem of finitely generated abelian groups shows that A admits a unique *primary decomposition* of the form:

$$A \cong \bigoplus_{p \text{ prime}} \bigoplus_{k_i,n_i \,:\, i\in I_p} \left(\mathbb{Z}/p^{k_i}\mathbb{Z}\right)^{n_i}$$

for finitely many primes p, and positive integers $k_i, n_i > 0$. For a finite abelian group A, we denote by $d(A)$ the number of non-trivial summands in its primary decomposition. We use $A_{(p)}$ to denote the subgroup elements whose order is a power of p. Hence,

$$A = \bigoplus_{p \text{ prime}} A_{(p)}, \quad \text{where } A_{(p)} = \bigoplus_{k_i,n_i \,:\, i\in I_p} \left(\mathbb{Z}/p^{k_i}\mathbb{Z}\right)^{n_i}.$$

The following is a standard algorithmic result for finite abelian groups, formulated according to our needs in this article; see for example [25, Chap. 1.11].

▶ **Theorem 1.** *Let H be a finite abelian group presented via a primary decomposition with generators $h_1,\dots,h_{d(H)}$ for each of its prime summands. If $G \subseteq H$ is a subgroup of H specified via a set of generators $g_1,\dots,g_{d(G)}$ (each of which is encoded by a list of integers n_{ij} such that $g_i = \sum_j n_{ij}h_j$), then there exists a polynomial time algorithm using $O(d(|H|)^3)$ operations in H that finds a presentation of the primary decomposition of G – that is, a new* minimal *set of generators of G denoted $g'_1,\dots,g'_{d(G)}$ (each of which is, as before, represented by a list of integers n'_{ij} such that $g'_i = \sum_j n'_{ij}h_j$) together with relations of the form $(g'_i)^{p_i^{k_i}}$ for primes p_i and positive integers k_i.*

Pairings and forms on finite abelian groups. We now quickly review the basics of quadratic forms on finite abelian groups. We refer the reader to [27] for more background.

Let $(A,+)$ be a finite abelian group, denoted additively. A *symmetric bilinear pairing* $b\colon A\times A\to G$ into the (not necessarily finite) abelian group $(G,+)$ is a map satisfying

$$b(x,y) = b(y,x), b(x+y,z) = b(x,z)+b(y,z), \text{ and } b(x,y+z) = b(x,y)+b(x,z) \text{ for all } x,y,z.$$

A pairing is *non-degenerate* if for every $x\in A$, $x\neq 0$, there exists some $y\in A$ such that $b(x,y)$ is not trivial in G. When the symmetric bilinear pairing takes value in $(G,+) = (\mathbb{Q}/\mathbb{Z},+)$, where $(\mathbb{Q}/\mathbb{Z},+)$ is the additive group of rational numbers modulo the integers, we call the pair (A,b) a *discriminant pairing*. For any real number $x\in\mathbb{R}$, denote by $\mathbf{e}(x)$ the quantity $\exp(2\pi i x)$. Let $(U(1),\times)$ be the multiplicative group of complex numbers $\{\mathbf{e}(x) : x\in\mathbb{R}/\mathbb{Z}\}$ on the unit circle. There is a natural group isomorphism $\mathbb{Q}/\mathbb{Z}\to U(1), x\mapsto \mathbf{e}(x)$. When a symmetric bilinear pairing b is non-degenerate and takes value in $(U(1),\times)$, we call it a *bicharacter*. Note that (A,b) is a non-degenerate discriminant pairing if and only if $(A,\mathbf{e}(b(\cdot,\cdot)))$ is a bicharacter. If (A,b) is a discriminant pairing and $e_1,\dots,e_k\in A$, then the *Gram matrix* of $(e_1,\dots,e_k)$ is the $k\times k$ matrix $(b(e_i,e_j))_{i,j}$.

For a map $q\colon A\to G$, define $\partial q\colon A\times A\to G$ by $\partial q(x,y) := q(x+y)-q(x)-q(y)$ for any $x,y\in A$. The map q is a *(homogeneous) quadratic form* if, for any $x\in A$ and $u\in\mathbb{Z}$, $q(ux) = u^2q(x)$ and the map ∂q is bilinear. For a quadratic form q, we call ∂q the *bilinear*

pairing associated to q. A quadratic form q is *non-degenerate* if the bilinear pairing ∂q is non-degenerate. If q takes value in $\mathbb{Q}/\mathbb{Z}$, the pair (A, q) is called a *pre-metric group*, and if moreover q is non-degenerate, (A, q) is a *metric group*.

Gauss sums on pre-metric groups. If $\gamma \in \mathbb{Q}/\mathbb{Z}$, let $\mathbf{e}(\gamma) := \exp(2\pi i\gamma)$. To every pre-metric group (G, q) we associate a complex number $\Theta(G, q)$ called the *Gauss sum*:

$$\Theta(G, q) := \frac{1}{|\sqrt{|G|}|} \sum_{x \in G} \mathbf{e}(q(x)).$$

Our next theorem is one of the central tools in our algorithm for computing the TVBW invariants of Tambara-Yamagami categories.

▶ **Theorem 2.** *Let G be a finite abelian group presented via a primary decomposition with $d(G)$ summands. If oracle access to a quadratic form $q : G \to \mathbb{Q}/\mathbb{Z}$ is provided, then the Gauss sum $\Theta(G, q)$ can be evaluated in polynomial time using $O(d(G)^3 \operatorname{poly}(\log |G|))$ operations, $O(d(G)^2)$ oracle accesses, and $O(d(G)^2 \operatorname{poly}(\log |G|))$ memory words.*

Theorem 2 is seemingly folklore in the algebraic number theory literature, but we were unable to find any references proving it. We understand it as a consequence of a scattered collection of works on pre-metric groups and algorithms. Wall [33] abstractly classifies metric groups by showing how to decompose them into direct sums of elementary pieces; Basak and Johnson [3] later provide an algorithmic formulation of the decomposition theorem and explain that the Gauss sum of a metric group can be computed readily from the decomposition. We have however not found a complexity analysis of this construction anywhere in the literature. The closest we are aware of is [5, Thm. 1.1], which shows that Gauss sums of (not-necessarily-homogeneous) quadratic forms on cyclic groups $\mathbb{Z}/N\mathbb{Z}$ are computable in polynomial time; however, even if A itself is cyclic, the Gauss sums we will need to compute in our algorithm for $|M|_{\mathrm{TY}(A,\chi,\nu)}$ are on non-cyclic pre-metric groups of the form (A^n, q). In order to fill this gap, we prove Theorem 2 in the full version of the paper [8] by supplementing the algorithm of [3] with a detailed complexity analysis.

2.2 Combinatorial and algebraic topology

(Co)homology. Throughout this article we employ the simplicial cohomology of simplicial complexes with $\mathbb{Z}/2\mathbb{Z}$ coefficients. We assume the reader is familiar with the basic definitions (see [11] for an introduction to the topic). The following well-known fact can be understood as an application of Theorem 1.

▶ **Theorem 3.** *Let $\mathfrak{T}$ be a simplicial complex with n simplices. For any dimension $d = 0, 1, 2, \ldots$, the following can be computed in polynomial time using $O(n^3)$ operations: a basis of $Z^d(\mathfrak{T}, \mathbb{Z}/2\mathbb{Z})$; a basis of $B^d(\mathfrak{T}; \mathbb{Z}/2\mathbb{Z})$; a set of elements in $Z^d(\mathfrak{T}; \mathbb{Z}/2\mathbb{Z})$ that represent a basis of $H^d(\mathfrak{T}; \mathbb{Z}/2\mathbb{Z})$. In particular, the dimension of each of these spaces can be computed in polynomial time.*

Triangulated 3-manifolds. The most common data type in computational 3-manifold topology is that of a generalized triangulation; e.g., see [24]. However, strictly speaking, the TVBW invariants are only defined for *simplicial (or PL)* triangulations. It is possible to extend the definition to "polytope decompositions" (including generalized triangulations) [15], but we will not need to work at this level of generality.

Thus, throughout this work, we make the following conventions: a 3-manifold M is always closed and oriented, and presented by a simplicial triangulation, denoted $\mathfrak{T}$, together with an orientation of $\mathfrak{T}$. The orientability of M is reflected in the fact that the simplicial homology group $H_3(\mathfrak{T};\mathbb{Z})$ is isomorphic to $\mathbb{Z}$ (this can be checked quickly from $\mathfrak{T}$), and then an *orientation* of $\mathfrak{T}$ is encoded by labeling each tetrahedron in $\mathfrak{T}$ with a sign $+1$ or -1 in a way that gives rise to a $\{+1,-1\}$-valued cellular 0-cocycle on the cellulation dual to $\mathfrak{T}$.

Later, it will also be convenient to decorate each *edge* of an oriented triangulation with an arbitrarily chosen orientation. We stress that these edge orientations are arbitrary, and have nothing to do with the orientation of the manifold's tetrahedra. Going forward, by a small abuse of language, if we say *triangulation*, then we will usually mean an oriented triangulation equipped with such an edge orientation. We denote the set of vertices of $\mathfrak{T}$ by V, the set of edges by E, the set of triangular faces by F, and the set of tetrahedra by T.

2.3 Tambara-Yamagami quantum invariants

The *quantum invariants* we consider in this article are $\mathbb{C}$-valued topological invariants of closed 3-manifolds derived from algebraic data (a *spherical fusion category*) attached to a combinatorial presentation of the 3-manifold (a *simplicial triangulation*), together with evaluation rules described in the works of Turaev and Viro [30] and Barrett and Westbury [2]. We now give an extremely brief introduction to *Tambara-Yamagami quantum invariants*, which are the TVBW invariants produced by a particular infinite family of spherical fusion categories called *Tambara-Yamagami categories* [28]. For the sake of space, here we provide only the details necessary to follow the proofs of our main results. We refer the reader to [8] for the full details of the combinatorial description of spherical fusion categories in general and for Tambara-Yamagami categories in particular, as well as the general construction of TVBW invariants (for unitary and multiplicity-free categories).

Let $(A,+)$ be a finite abelian group with $|A|$ elements, $\chi : A \times A \to U(1)$ a bicharacter on A, and ν a choice of square root $\nu = \pm 1/\sqrt{|A|} = \pm |A|^{-1/2}$. The *Tambara-Yamagami category* $\mathrm{TY}(A,\chi,\nu)$ is an algebraic object which gives rise to the following data:

- a set of *colors* $L \stackrel{\text{def}}{=} A \sqcup \{m\}$ (where m is some symbol not in A), together with a *duality* operation * defined by $a^* \stackrel{\text{def}}{=} -a$ and $m^* \stackrel{\text{def}}{=} m$.
- scalars d_a and d_m associated to every color in L called *quantum dimensions*, as well as the *(global) quantum dimension* scalar D. These scalars are defined as follows:

$$d_a \stackrel{\text{def}}{=} 1 \text{ for all } a \in A, \qquad d_m \stackrel{\text{def}}{=} +|A|^{1/2}, \qquad D \stackrel{\text{def}}{=} \sum_{x \in L} d_x^2 = 2|A|.$$

- a set of *admissible triples* of colors

$$\{(a,b,-a-b),(a,m,m),(m,a,m),(m,m,a) : a,b \in A\} \subset L \times L \times L,$$

- and a system of $\mathbb{C}$-valued *tetrahedron weights* defined below.

A *coloring* $\theta\colon E \to L$ of a triangulation $\mathfrak{T}$ is an assignment of a color to each edge of $\mathfrak{T}$. Given an edge e_0 on the boundary of a face f, the orientation of e_0 determines a direction along which to traverse the three boundary edges of f; call these edges e_0, e_1 and e_2, according to the order in which we first see them while doing this traversal. Now associate a triple of colors $(\theta(e_0), \theta(e_1)^{(*)}, \theta(e_2)^{(*)})$, where $\theta(e_i)^{(*)}$ means we apply the duality operation * to the label $\theta(e_i)$ if the direction of traversal around f disagrees with the orientation of e_i. The coloring θ is called *admissible* if every such triple associated to an edge-face pair is admissible. The set of *admissible colorings* is denoted by $\mathrm{Adm}_L(\mathfrak{T})$.

We now describe the tetrahedral weights involved in the definition of the Tambara-Yamagami invariants. Let θ be a coloring of $\mathfrak{T}$ and let $\theta_e := \theta(e) \in L$ denote the label of an edge $e \in E$. For each tetrahedron $t \in T$ with edges $e_1, e_2, \dots e_6$ and orientation $O(t) \in \{+, -\}$, we assign a *tetrahedral weight* $|t|_\theta^{O(t)} \in \mathbb{C}$. If θ is inadmissible, then $|t|_\theta^{O(t)} \stackrel{\text{def}}{=} 0$. Otherwise, θ is admissible and so, as described in [8], takes on one of four possible types; taking $O(t)$ into account we then assign the tetrahedral weights as follows:

m-empty

$$\begin{vmatrix} a & b & a+b \\ b-c & a+c & c \end{vmatrix}^{+} = 1 \qquad \begin{vmatrix} a & b & a+b \\ c-b & a+c & c \end{vmatrix}^{-} = 1$$

m-triangle

$$\begin{vmatrix} m & m & a \\ a & a+b & m \end{vmatrix}^{+} = d_m \qquad \begin{vmatrix} a & b & a+b \\ m & m & m \end{vmatrix}^{-} = d_m$$

flat m-quad

$$\begin{vmatrix} a & m & m \\ b & m & m \end{vmatrix}^{+} = d_m \chi(a,b) \qquad \begin{vmatrix} a & m & m \\ b & m & m \end{vmatrix}^{-} = d_m \overline{\chi(a,b)}$$

crooked m-quad

$$\begin{vmatrix} m & m & a \\ m & m & b \end{vmatrix}^{+} = \nu d_m^2 \overline{\chi(a,b)} \qquad \begin{vmatrix} m & m & a \\ m & m & b \end{vmatrix}^{-} = \nu^{-1} d_m^2 \chi(a,b),$$

where a, b, c are arbitrary elements in A.

Finally, the Tambara-Yamagami quantum invariant $|M|_{\mathrm{TY}(A,\chi,\nu)}$ of a 3-manifold M with triangulation $\mathfrak{T}$ is defined via the following "state sum formula:"

$$|M|_{\mathrm{TY}(A,\chi,\nu)} \stackrel{\text{def}}{=} \sum_{\theta \in \mathrm{Adm}_L(\mathfrak{T})} \frac{\prod_{t \in T} |t|_\theta^{O(t)} \prod_{e \in E} d_{\theta e}}{\prod_{f \in F} \sqrt{d_{\theta e_1} d_{\theta e_2} d_{\theta e_3}} \prod_{v \in V} D}.$$

We note that for a fixed coloring θ and tetrahedron t, testing its admissibility on t and determining its type can be carried out in constant time. Thus, there is a naïve algorithm to compute $|M|_{\mathrm{TY}(A,\chi,\nu)}$ directly from this definition with running time exponential in the size of $\mathfrak{T}$. In fact, this is true of $|M|_{\mathcal{C}}$ for any spherical fusion category $\mathcal{C}$. We now turn to showing that we can do much better when $\mathcal{C} = \mathrm{TY}(A, \chi, \nu)$ if we bound the $\mathbb{Z}/2\mathbb{Z}$ Betti number $\beta_1(M)$.

3 The role of the Betti number in Tambara-Yamagami invariants

We are interested in the following algorithmic problem:

$\mathrm{TY}(A, \chi, \nu)$-INVARIANT COMPUTATION
Input: $\mathfrak{T}$ a triangulation of an oriented closed 3-manifold M
Output: $\lvert M\rvert_{\mathrm{TY}(A,\chi,\nu)}$

Since we are not working uniformly in the data A, χ, ν, there is no harm in assuming we have full "explicit" access to it. Our main result is

▶ **Theorem 4.** *For a fixed Tambara-Yamagami category $\mathrm{TY}(A, \chi, \nu)$, there is a deterministic algorithm that, given as input a triangulation of an orientable closed 3-manifold M with n tetrahedra, computes the quantum invariant $|M|_{\mathrm{TY}(A,\chi,\nu)}$ in $O(2^{\beta_1} n^3)$ operations and using $O(n^2)$ memory words.*

In this section we expound the ideas introduced in Section 1 that lead to this algorithm; its proof is deferred until Section 4. Hereafter A, χ, and ν are fixed.

3.1 Admissible colorings induce simplicial 1-cocycles

We begin by noting that each admissible coloring determines a $\mathbb{Z}/2\mathbb{Z}$-valued simplicial 1-cocycle. More precisely, consider the function

$$\phi\colon \mathrm{Adm}_L(\mathfrak{T}) \to C^1(\mathfrak{T}, \mathbb{Z}/2\mathbb{Z}), \qquad \phi(\theta)(e) = \begin{cases} 1 & \text{if } \theta(e) = m \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

that assigns to an admissible coloring of $\mathfrak{T}$ a $\mathbb{Z}/2\mathbb{Z}$-valued simplicial 1-cochain of the triangulation (that is, a function $E \to \mathbb{Z}/2\mathbb{Z}$).

▶ **Lemma 5.** *For any admissible coloring θ of a triangulation $\mathfrak{T}$, $\phi(\theta)$ is a 1-cocycle.*

In the case of $\mathrm{TY}(\mathbb{Z}/2\mathbb{Z}, \exp(i\pi ab), -1/\sqrt{2}) = \mathrm{TV}_4$, this lemma was proved in [4, 22] and used in the algorithm of [23]. It can be proved easily in the general case by examining the definition of admissible triples in $\mathrm{TY}(A, \chi, \nu)$ – the calculation boils down to the simple observation that in any admissible coloring, the number of edges around any face of $\mathfrak{T}$ colored by m is either 0 or 2. More conceptually, this lemma follows from the fact that $\mathrm{TY}(A, \chi, \nu)$ is a spherical $\mathbb{Z}/2\mathbb{Z}$-*graded* fusion category, and a similar result holds for any spherical G-graded fusion category, cf. [31].

3.2 The partial state sum at a 1-cocycle

Fix a 1-cocycle $\alpha \in Z^1(\mathfrak{T}, \mathbb{Z}/2\mathbb{Z})$ and consider the *partial state sum at* α, defined by only summing over the admissible labelings associated to α:

$$|\mathfrak{T}, \alpha|_{\mathrm{TY}(A,\chi,\nu)} \overset{\text{def}}{=} \sum_{\theta \in \phi^{-1}(\alpha)} \frac{\displaystyle\prod_{t \in T} |t|_\theta^{O(t)} \prod_{e \in E} d_{\theta e}}{\displaystyle\prod_{f \in F} \sqrt{d_{\theta e_1} d_{\theta e_2} d_{\theta e_3}} \prod_{v \in V} D}.$$

By definition, all admissible colorings of $\phi^{-1}(\alpha)$ have the same set of edges colored by the object m. We partition E, F, and T, such that for every coloring in $\phi^{-1}(\alpha)$:

1. $E = E_\emptyset \sqcup E_\bullet$ where $E_\emptyset$ is the set of edges colored by an element of A, and $E_\bullet$ is the set of edges colored by m,
2. $F = F_\emptyset \sqcup F_{\bullet\bullet}$ where $F_\emptyset$ is the set of faces whose edges are colored exclusively by elements of A, and $F_{\bullet\bullet}$ is the set of faces with exactly two edges colored by m,
3. $T = T_\emptyset \sqcup T_\triangle \sqcup T_\boxplus \sqcup T_\boxminus \sqcup T_\diamondsuit \sqcup T_\diamond$, where $T_\emptyset$ is the set of m-empty faces, $T_\triangle$ is the set of m-triangles, $T_\boxplus$ and $T_\boxminus$ are respectively the set of positively and negatively oriented flat m-quads, and $T_\diamondsuit$ and $T_\diamond$ are respectively the set of positively and negatively oriented crooked m-quads (see end of Section 2.3 for conventions).

Replacing $|t|_\theta^{O(t)}$ by the appropriate values and factorizing, we now rewrite the partial state sum at the 1-cocycle α.

For an admissible coloring θ and an m-quad tetrahedron t in θ (either flat or crooked), define $\chi_{\theta(t)}$ to be the value $\chi(a, b)$ where a, b are the colors in θ assigned to the two opposite edges not colored by m in the (flat or crooked) m-quad tetrahedron t. One can then check:

$$|\mathfrak{T}, \alpha|_{\mathrm{TY}(A,\chi,\nu)} = D^{-|V|} \cdot d_m^{C_\alpha} \cdot \nu^{|T_\diamondsuit| - |T_\diamond|} \sum_{\theta \in \phi^{-1}(\alpha)} \left(\prod_{t \in T_\boxplus} \chi_{\theta(t)} \cdot \prod_{t \in T_\boxminus} \overline{\chi_{\theta(t)}} \cdot \prod_{t \in T_\diamondsuit} \overline{\chi_{\theta(t)}} \cdot \prod_{t \in T_\diamond} \chi_{\theta(t)} \right) \tag{2}$$

where $C_\alpha = |E_\bullet| - |F_{\bullet\bullet}| + |T_\boxplus| + |T_\boxminus| + 2|T_\diamondsuit| + 2|T_\diamond|$.

In Section 3.4 we prove:

▶ **Theorem 6.** *Fix a category* $\mathrm{TY}(A,\chi,\nu)$*. Let* $e_1,\dots,e_{|E|}$ *be an arbitrary ordering of the edges of* $\mathfrak{T}$*. For any collections* $\mathcal{P},\mathcal{Q}$ *of pairs of indices* (i,j)*,* $1\le i,j\le|E|$*, with potentially repeated pairs, such that* $|\mathcal{P}|,|\mathcal{Q}|\in O(n)$*, there is an algorithm to evaluate the sum:*

$$\sum_{\theta\in\phi^{-1}(\alpha)}\prod_{(i,j)\in\mathcal{P}}\chi(\theta_{e_i},\theta_{e_j})\prod_{(r,s)\in\mathcal{Q}}\overline{\chi(\theta_{e_r},\theta_{e_s})} \tag{3}$$

in $O(n^3)$ *operations, using* $O(n^2)$ *memory words.*

Using the theorem, we arrive at a key step towards our main result.

▶ **Corollary 7.** *Fix a category* $\mathrm{TY}(A,\chi,\nu)$*. Given as input a 3-manifold triangulation* $\mathfrak{T}$ *with* n *tetrahedra, and a 1-cocycle* α *in* $\mathfrak{T}$*, the sum (2) can be evaluated in* $O(n^3)$ *operations, using* $O(n^2)$ *memory words.*

Proof. By definition of the projection ϕ (Equation (1)), the 1-cocycle α gives a description of the edges colored by the object m in all admissible colorings of $\phi^{-1}(\alpha)$. Additionally, the multiplicative constant $D^{-|V|}\cdot d_m^{C_\alpha}\cdot\nu^{|T_{\diamondsuit}|-|T_{\diamondsuit}|}$ in front of the sum (Equation (2)) depends exclusively on the set of edges colored by m. In consequence, the multiplicative constant can be evaluated in linear time by checking the set of m-colored edges of each tetrahedron. Using the notations of Theorem 6, we consider the pairs $\mathcal{P}$ (resp. $\mathcal{Q}$) of indices (i,j) of opposite edges (e_i,e_j) not colored by m in tetrahedra of type $T_{\boxplus}\sqcup T_{\diamondsuit}$ (resp. $T_{\boxminus}\sqcup T_{\diamondsuit}$), in order to evaluate the sum $\sum_{\theta\in\phi^{-1}(\alpha)}\left(\prod_{t\in T_{\boxplus}}\chi_{\theta(t)}\cdot\prod_{t\in T_{\boxminus}}\overline{\chi_{\theta(t)}}\cdot\prod_{t\in T_{\diamondsuit}}\overline{\chi_{\theta(t)}}\cdot\prod_{t\in T_{\diamondsuit}}\chi_{\theta(t)}\right)$ in cubic time. ◀

3.3 Efficiently characterizing colorings over a 1-cocycle

For a given 1-cocycle $\alpha\in Z^1(\mathfrak{T};\mathbb{Z}/2\mathbb{Z})$, we now characterize the set $\phi^{-1}(\alpha)\subseteq\mathrm{Adm}(\mathfrak{T})$. Recall that $E_\emptyset$ is the set of edges not colored by m and $F_\emptyset$ is the set of faces with no edge colored by m.

▶ **Lemma 8.** *If the finite abelian group* A *is given by its primary decomposition with* $d(A)$ *summands, and the triangulation* $\mathfrak{T}$ *has* $n=|\mathfrak{T}|$ *tetrahedra, then the set* $\phi^{-1}(\alpha)$ *is in bijection with the elements of a finite abelian group* G *with* $O(d(A)n)$ *generators.*

Proof. Fix an arbitrary orientation of the edges of the triangulation. The admissibility constraints for colorings in a Tambara-Yamagami category imply that the set $\phi^{-1}(\alpha)$ is in bijection with the set of assignments $\theta\colon E_\emptyset\to A$ satisfying the following linear equations: for any m-empty face with edges e_1,e_2,e_3,

$$\varepsilon_1\theta(e_1)+\varepsilon_2\theta(e_2)+\varepsilon_3\theta(e_3)=0,$$

where $\varepsilon_i\in\{\pm1\}$ depending whether the orientations of edges e_1,e_2,e_3 agree or not with an arbitrarily chosen orientation of the triangular face. Consider the group homomorphism $\lambda\colon A^{|E_\emptyset|}\to A^{|F_\emptyset|}$ where the $|F_\emptyset|$-many values are the relations $\varepsilon_1\theta(e_1)+\varepsilon_2\theta(e_2)+\varepsilon_3\theta(e_3)$ above. The set $\phi^{-1}(\alpha)$ is in bijection with the kernel of this application. Because $A^{|E_\emptyset|}$ is a finite abelian group with $d(A)|E_\emptyset|$ summands in its primary decomposition, with $|E_\emptyset|\le|E|\in O(n)$, then $\ker\lambda$ is a finite abelian group with $O(d(A)n)$ generators. ◀

Additionally, a minimal family of generators $(y_1,\dots,y_\ell), y_i\in A^{|E_\emptyset|}$, for $\ker\lambda$ can be computed in polynomial time.

▶ **Lemma 9.** *For a fixed abelian group A, with the notations above, a minimal family of generators for* $\ker\lambda$ *can be computed in $O(n^3)$ operations, using $O(n^2)$ memory words.*

Proof. The proof is standard linear algebra over $\mathbb{Z}$ using Smith normal form, see the proof of [8, Lemma 3.0] for details. ◀

In conclusion, any generator of the kernel $\ker\lambda \subseteq A^{|E_\emptyset|}$ can be interpreted as an assignment of colors to the empty edges of the triangulation $E_\emptyset \to A$. For a fixed 1-cocycle α of the triangulation, any admissible coloring in $\phi^{-1}(\alpha)$ can be expressed as a sum (with integer coefficients) of elements of the kernel (where the sum of colors is edge-wise) following the rule of addition in A.

3.4 Partial state sums are (normalized) Gauss sums

The goal of this subsection is to prove Theorem 6. Consider a finite abelian group $(A,+)$ with $d := d(A)$ generators $x_1,\dots,x_d$ for the summands of its primary decomposition, its product A^k with generators $(x_{ij})_{i=1\dots d, j=1\dots k}$, and a subgroup $(G,+)$ with $g := d(G)$.

For a group element $y \in G$, we express $y = \sum_{i,j} n_{ij}(y)x_{ij}$, for integer coefficients $n_{ij}(y)$, in the set of generators for A^k. We also write $y^{(j)} := \sum_{i=1\dots d} n_{ij}(y)x_i \in A$ for the projection of y into the j-th component of the product A^k.

Let $\mathcal{P},\mathcal{Q}$ be arbitrary sets of pairs of indices (i,j), with $1 \le i,j \le k$, and possible multiplicities $p_{ij} \ge 1$ for the pair $(i,j) \in \mathcal{P}$, and q_{ij} for the pair $(i,j) \in \mathcal{Q}$. Consider the product:

$$\psi\colon G \to U(1), \qquad y \mapsto \prod_{(i,j)\in\mathcal{P}} \chi(y^{(i)},y^{(j)})^{p_{ij}} \prod_{(r,s)\in\mathcal{Q}} \chi(y^{(r)},y^{(s)})^{-q_{rs}}. \tag{4}$$

and define the complex valued sum:

$$\Omega(\mathcal{P},\mathcal{Q}) := \frac{1}{|\sqrt{|G|}|} \sum_{y\in G} \psi(y). \tag{5}$$

▶ **Lemma 10.** *The sum $\Omega(\mathcal{P},\mathcal{Q})$ in Equation (5) is a* quadratic Gauss sum *over G,* i.e.*, there exists a quadratic form $q\colon G \to \mathbb{Q}/\mathbb{Z}$ s.t.,*

$$\Omega(\mathcal{P},\mathcal{Q}) = \Theta(G,q) = \frac{1}{|\sqrt{|G|}|} \sum_{x\in G} \mathbf{e}(q(x)).$$

Proof. Define the pairing $b\colon A\times A \to \mathbb{Q}/\mathbb{Z}$ satisfying, for any $x,y \in A$, the equality $\chi(x,y) = \mathbf{e}(b(x,y))$. The map b is a bilinear pairing. This follows from the definition of a bicharacter. For any $x,y,z \in A$,

$$\chi(x+y,z) = \mathbf{e}(b(x+y,z)) = \chi(x,z)\chi(y,z) = \mathbf{e}(b(x,z)+b(y,z)),$$

and in consequence $b(x+y,z) = b(x,z)+b(y,z) \mod \mathbb{Z}$. By symmetry of χ, bilinearity follows. Define $q\colon G \to \mathbb{Q}/\mathbb{Z}$ by:

$$q(y) := \sum_{(i,j)\in\mathcal{P}} p_{ij}\cdot b(y^{(i)},y^{(j)}) - \sum_{(r,s)\in\mathcal{Q}} q_{rs}\cdot b(y^{(r)},y^{(s)}),$$

such that $\left|\sqrt{|G|}\right| \Omega(\mathcal{P},\mathcal{Q}) = \sum_{x\in G} \mathbf{e}(q(x))$. The map q is a quadratic form. Indeed, first notice that, for any integer u, $q(uy) = u^2 q(y)$ follows from the fact that $b(uy^{(i)},uy^{(j)}) = u^2 b(y^{(i)},y^{(j)})$. Second, the form $\partial q : (y,y') \mapsto q(y+y') - q(y) - q(y')$ is bilinear, as it can be expressed as a sum of evaluations of the bilinear pairing b on $(y^{(i)},y'^{(j)})$ for some i,j. ◀

Proof of Theorem 6. Using the notation of Theorem 6, we identify the following sum on the triangulation:

$$\sum_{\theta\in\phi^{-1}(\alpha)} \prod_{(i,j)\in\mathcal{P}} \chi(\theta_{e_i},\theta_{e_j}) \prod_{(r,s)\in\mathcal{Q}} \overline{\chi(\theta_{e_r},\theta_{e_s})} \tag{6}$$

with the quantity $\left|\sqrt{|G|}\right| \Omega(\mathcal{P},\mathcal{Q})$ in Equation (5), where the group G is the subgroup $\ker\lambda$ of the product $A^{|E_\emptyset|}$, whose presentation we identify in polynomial time via Lemma 9. The pairs of indices $\mathcal{P}$ and $\mathcal{Q}$ represent indices of edges in Equation (6), and particular copies of A in the product $A^{|E_\emptyset|}$ in Equation (4), with a natural correspondence induced by the fact that copies of A in $A^{|E_\emptyset|}$ are in bijection with edges not colored by m (once the 1-cocycle α is fixed). The order of the finite group $G=\ker\lambda$ can be evaluated by finding the primary decomposition of G, which can be done in polynomial time (Theorem 1). Let q be the quadratic form defined in the proof of Theorem 10. We can compute the Gram matrix of a minimal family of generators of $\ker\lambda$ (computed in Lemma 9) for the discriminant pairing $(G,\partial q)$ by evaluating ∂q, and compute the Gauss sum $\Omega(\mathcal{P},\mathcal{Q})$ in polynomial time (by virtue of Theorem 2), with the expected complexity. ◀

3.5 Cohomologous 1-cocycles have equal partial state sums

The tools introduced so far in this section would only be enough to construct a parameterized algorithm in terms of the number of 1-cocycles $Z^1(\mathfrak{T};\mathbb{Z}/2\mathbb{Z})$. The final ingredient necessary to further reduce the parameter to $\beta_1(M)$ follows from a general statement about TVBW invariants of spherical G-graded fusion categories (for any finite group G) first described by Turaev and Virelizier [31, Thm. 7.1], specialized to our situation: $\mathrm{TY}(A,\chi,\nu)$ is a spherical $\mathbb{Z}/2\mathbb{Z}$-graded fusion category.

▶ **Theorem 11** (special case of Thm. 7.1 in [31]). *If α and β are cohomologous elements of the group of cocycles $Z^1(\mathfrak{T};\mathbb{Z}/2\mathbb{Z})$, then $|\mathfrak{T},\alpha|_{\mathrm{TY}(A,\chi,\nu)} = |\mathfrak{T},\beta|_{\mathrm{TY}(A,\chi,\nu)}$.*

4 Fixed parameter tractable algorithm in the first Betti number

Fix the data of $\mathrm{TY}(A,\chi,\nu)$. Given as input an arbitrary triangulation $\mathfrak{T}$ of a closed oriented 3-manifold M, our algorithm to compute the TVBW invariant $|M|_{\mathrm{TY}(A,\chi,\nu)}$ is as follows:

1. Compute a set of 1-cocycles $\{\alpha_1,\dots,\alpha_{\beta_1}\}\subset Z^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$ such that $\{[\alpha_1],\dots,[\alpha_{\beta_1}]\}$ forms a basis of the 1-cohomology vector space $H^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$.

   Compute a basis for $B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$; the cardinality $\#B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$ of the set of 1-coboundaries is equal to $2^{\dim B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})}$. [Theorem 3]

2. For each of the 2^{β_1} distinct 1-cohomology classes $[\alpha]$ of $H^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$, construct a representative $\alpha\in Z^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$ by enumerating all 2^{β_1} $\mathbb{Z}/2\mathbb{Z}$-combinations of elements in $\{\alpha_1,\dots,\alpha_{\beta_1}\}$, *i.e.*, all $\alpha=\varepsilon_1\alpha_1+\ldots+\varepsilon_{\beta_1}\alpha_{\beta_1}$, for $\varepsilon_i\in\mathbb{Z}/2\mathbb{Z}$.

   Now, for each such α, apply Corollary 7 to compute the partial state sum $|\mathfrak{T},\alpha|_{\mathrm{TY}(A,\chi,\nu)}$, with the following steps:

   a. Compute a minimal family of generators for $\ker\lambda=\langle x_1,\dots,x_g\rangle$, [Lemma 9]

   b. Compute the primary decomposition of $\ker\lambda$ with generators $\ker\lambda=\langle x'_1,\dots,x'_g\rangle$ for the summands, [Theorem 1]

   c. Compute the Gram matrix of $x'_1,\dots,x'_g$ for the pre-metric form $(\ker\lambda,q)$, by evaluating ∂q on all pairs (x'_i,x'_j). [defined in Lemma 10]

d. Evaluate the Gauss sum of $(\ker\lambda, q)$, [Theorem 2]
and multiply by $\left|\sqrt{|G|}\right|$. [proof of Theorem 6]
Normalize to get the state sum $|\mathfrak{T},\alpha|_{\mathrm{TY}(A,\chi,\nu)}$, with multiplicative factor as in Equation (2). [Corollary 7]

3. Sum over all α:

$$|M|_{\mathrm{TY}(A,\chi,\nu)} = \sum_{\substack{\alpha=\varepsilon_1\alpha_1+\ldots\varepsilon_{\beta_1}\alpha_{\beta_1}\\ \text{with } \varepsilon_i\in\mathbb{Z}/2\mathbb{Z}, \text{ and s.t.,}\\ \langle[\alpha_1],\ldots,[\alpha_{\beta_1}]\rangle = H^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})}} \#B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})\cdot|\mathfrak{T},\alpha|_{\mathrm{TY}(A,\chi,\nu)}.$$

Our main Theorem 4 follows readily from the following:

▶ **Theorem 12.** *Fix a Tambara-Yamagami category* $\mathrm{TY}(A,\chi,\nu)$*. Given a triangulation* $\mathfrak{T}$ *of a closed 3-manifold* M *with* n *tetrahedra, the above algorithm computes* $|M|_{\mathrm{TY}(A,\chi,\nu)}$ *in* $O(2^{\beta_1}n^3)$ *operations and using* $O(n^2)$ *memory words.*

Proof. Computing a basis for $B^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$, as well as a set of generators $\{\alpha_1,\ldots,\alpha_{\beta_1}\}$ whose cohomology classes form a basis of $H^1(\mathfrak{T},\mathbb{Z}/2\mathbb{Z})$, in step (1) is a standard procedure in computer algebra (Theorem 3), done by normalizing a $O(n)\times O(n)$ matrix with $\mathbb{Z}/2\mathbb{Z}$ coefficients. This can be done in $O(n^3)$ operations in $\mathbb{Z}/2\mathbb{Z}$ with Gaussian eliminations, where an arithmetic operation in $\mathbb{Z}/2\mathbb{Z}$ has constant complexity.

Given the 1-cocycles $\{\alpha_1,\ldots,\alpha_{\beta_1}\}$, we can enumerate every combination $\alpha=\varepsilon_1\alpha_1+\ldots+\varepsilon_{\beta_1}\alpha_{\beta_1}$ in (amortized) $O(n)$ operations per α. Indeed, enumerating all $\{\varepsilon_1,\ldots,\varepsilon_{\beta_1}\}$ is akin to incrementing a β_1-bits binary counter from $0\ldots0$ to $1\ldots1$, which induces an (amortized) change of $O(1)$ bits per increment [7, Chapter 16]. The 1-cocycles are represented by formal sums of $O(n)$ edges, and computing the sum of two 1-cocycles takes $O(n)$ operations.

For each α as above, we compute the partial state sum using Corollary 7 in polynomial time. Specifically, step (2a) is solved by computing a Smith normal form, together with transformation matrices, of a particular $O(d(A)n)\times O(d(A)n)$-integer matrix, which has complexity $O(n^3)$ (Lemma 9) with (modular) Gaussian eliminations, once A is fixed.

We can read-off the generators $x_1,\ldots,x_g$ of $\ker\lambda$ from the transformation matrices, and compute the generators $x'_1,\ldots,x'_g$ of step (2b) by computing the primary decomposition of the subgroup $\ker\lambda$ (Theorem 1). Computing the Gram matrix in step (2c) is done by evaluating the bilinear pairing ∂q, defined in Lemma 10, on all pairs (x'_i,x'_j). The quadratic form q is defined as a sum of $O(n)$ terms of the form $b(x'_i,x'_j)$, and the overall Gram matrix can be computed in $O(g^2n)$ operations. Recall that we have an oracle access to the function $b(\cdot,\cdot)$ (see beginning of Section 3). By Theorem 2, the Gauss sum of $(\ker\lambda,q)$ can be evaluated from the Gram matrix of $x'_1,\ldots,x'_g$ in $O(g^3)$ operations in $\mathbb{Q}/\mathbb{Z}$, where the arithmetic complexity of operations in $\mathbb{Q}/\mathbb{Z}$ depend solely on the maximal order of an element of A (independently of the size of the Gram matrix). See the full version [8] for details.

Finally, the invariant $|M|_{\mathrm{TY}(A,\chi,\nu)}$ computed in step (3) is the sum of 2^{β_1} complex numbers represented with a constant number of memory words each for a fixed group A (with $\beta_1\leq n$). In consequence, the overall complexity of the algorithm, knowing $g\in O(d(A)n)$ and considering $d(A),|A|$ constant for a fixed group A, is $O(2^{\beta_1}n^3)$. The memory complexity is bounded by the size of the matrices, which is $O(n^2)$, since we have a fixed A (coefficients are represented with a constant number of memory words, and $d(A)$ is a constant). ◀

References

1 David Aasen, Roger S. K. Mong, and Paul Fendley. Topological defects on the lattice: I. The Ising model. *J. Phys. A*, 49(35):354001, 46, 2016. doi:10.1088/1751-8113/49/35/354001.

2 John W. Barrett and Bruce W. Westbury. Invariants of piecewise-linear 3-manifolds. *Trans. Amer. Math. Soc.*, 348(10):3997–4022, 1996. doi:10.1090/S0002-9947-96-01660-1.

3 Tathagata Basak and Ryan Johnson. Indicators of Tambara-Yamagami categories and Gauss sums. *Algebra & Number theory*, 9(8):1793–1823, 2015.

4 Benjamin A. Burton, Clément Maria, and Jonathan 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.

5 Jin-Yi Cai, Xi Chen, Richard Lipton, and Pinyan Lu. On tractable exponential sums. In *Frontiers in Algorithmics: 4th International Workshop, FAW 2010, Wuhan, China, August 11-13, 2010. Proceedings 4*, pages 148–159. Springer, 2010. doi:10.1007/978-3-642-14553-7_16.

6 Yichul Choi, Ho Tat Lam, and Shu-Heng Shao. Noninvertible Global Symmetries in the Standard Model. *Phys. Rev. Lett.*, 129:161601, October 2022. doi:10.1103/PhysRevLett.129.161601.

7 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 4th edition, 2001.

8 Colleen Delaney, Clément Maria, and Eric Samperton. An algorithm for Tambara-Yamagami quantum invariants of 3-manifolds, parameterized by the first Betti number. *CoRR*, abs/2311.08514, 2023. doi:10.48550/arXiv.2311.08514.

9 Daniel S. Freed, Gregory W. Moore, and Constantin Teleman. Topological symmetry in quantum field theory, 2023. arXiv:2209.07471.

10 Daniel S. Freed and Constantin Teleman. Topological dualities in the Ising model. *Geom. Topol.*, 26(5):1907–1984, 2022. doi:10.2140/gt.2022.26.1907.

11 Allan Hatcher. *Algebraic Topology*. Cambridge University Press, 1 edition, 2002.

12 William Jaco and J. Hyam Rubinstein. 0-efficient triangulations of 3-manifolds. *J. Differential Geom.*, 65(1):61–168, 2003.

13 F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Cambridge Philos. Soc.*, 108(1):35–53, 1990. doi:10.1017/S0305004100068936.

14 Robion Kirby and Paul Melvin. Local surgery formulas for quantum invariants and the Arf invariant. *Geometry & Topology Monographs*, 7:213–233, 2004.

15 Alexander Kirillov Jr and Benjamin Balsam. Turaev-Viro invariants as an extended TQFT. *arXiv preprint*, 2010. arXiv:1004.1533.

16 A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Physics*, 303(1):2–30, 2003. doi:10.1016/S0003-4916(02)00018-0.

17 Robert Koenig, Greg Kuperberg, and Ben W. Reichardt. Quantum computation with Turaev-Viro codes. *Ann. Physics*, 325(12):2707–2749, 2010. doi:10.1016/j.aop.2010.08.001.

18 Hari Krovi and Alexander Russell. Quantum Fourier transforms and the complexity of link invariants for quantum doubles of finite groups. *Comm. Math. Phys.*, 334(2):743–777, 2015. doi:10.1007/s00220-014-2285-5.

19 Greg Kuperberg. How hard is it to approximate the Jones polynomial? *Theory Comput.*, 11:183–219, 2015. doi:10.4086/toc.2015.v011a006.

20 Greg Kuperberg and Eric Samperton. Computational complexity and 3-manifolds and zombies. *Geom. Topol.*, 22(6):3623–3670, 2018. doi:10.2140/gt.2018.22.3623.

21 Greg Kuperberg and Eric Samperton. Coloring invariants of knots and links are often intractable. *Algebr. Geom. Topol.*, 21(3):1479–1510, 2021. doi:10.2140/agt.2021.21.1479.

22 Clément Maria and Jonathan Spreer. Admissible colourings of 3-manifold triangulations for Turaev-Viro type invariants. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ESA.2016.64.

23 Clément Maria and Jonathan Spreer. A polynomial-time algorithm to compute Turaev-Viro invariants TV4,q of 3-manifolds with bounded first Betti number. *Found. Comput. Math.*, 20(5):1013–1034, 2020. doi:10.1007/s10208-019-09438-8.

24 Sergei Matveev. *Algorithmic Topology and Classification of 3-Manifolds*. Springer, 9 edition, 2003.

25 James R. Munkres. *Elements of algebraic topology*. CRC Press, 1 edition, 1984.

26 Eric C. Rowell and Zhenghan Wang. Mathematics of topological quantum computing. *Bull. Amer. Math. Soc. (N.S.)*, 55(2):183–238, 2018. doi:10.1090/bull/1605.

27 Winfried Scharlau. *Quadratic and Hermitian Forms*. Grundlehren der mathematischen Wissenschaften. Springer, 1985.

28 Daisuke Tambara and Shigeru Yamagami. Tensor Categories with Fusion Rules of Self-Duality for Finite Abelian Groups. *Journal of Algebra*, 209:692–707, 1998.

29 Ryan Thorngren and Yifan Wang. Fusion category symmetry I: anomaly in-flow and gapped phases. *arXiv preprint*, 2019. arXiv:1912.02817.

30 V. G. Turaev and O. Ya. Viro. State sum invariants of 3-manifolds and quantum 6j-symbols. *Topology*, 31(4):865–902, 1992. doi:10.1016/0040-9383(92)90015-A.

31 Vladimir Turaev and Alexis Virelizier. On 3-dimensional homotopy quantum field theory, i. *International Journal of Mathematics*, 23(09):1250094, 2012.

32 Vladimir G. Turaev. *Quantum invariants of knots and 3-manifolds*, volume 18 of *De Gruyter Studies in Mathematics*. De Gruyter, Berlin, third edition, 2016. doi:10.1515/9783110435221.

33 Charles T C Wall. Quadratic forms on finite groups and related topics. *Topology*, 2:281–298, 1963.

34 Zhenghan Wang. *Topological quantum computation*, volume 112 of *CBMS Regional Conference Series in Mathematics*. Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 2010. doi:10.1090/cbms/112.

Tiling with Three Polygons Is Undecidable

Erik D. Demaine
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA

Stefan Langerman
Computer Science Department, Université libre de Bruxelles, Belgium

Abstract

We prove that the following problem is co-RE-complete and thus undecidable: given three simple polygons, is there a tiling of the plane where every tile is an isometry of one of the three polygons (either allowing or forbidding reflections)? This result improves on the best previous construction which requires five polygons.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Problems, reductions and completeness

Keywords and phrases plane tilings, polygons, undecidability, co-RE

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.39

Related Version *Full Version*: `https://arxiv.org/pdf/2409.11582`

Supplementary Material
Software (Implementation): `https://github.com/edemaine/three-tiles` [4]
archived at `swh:1:dir:aef5f642bf8a40c0cecf56bfb62b9e089effb265`

Acknowledgements This research was initiated at the Second Adriatic Workshop on Graphs and Probability, and continued at the Eleventh Annual Workshop on Geometry and Graphs. The authors thank all participants of both workshops for providing a stimulating research environment. They also thank Joshua Blömer for spotting a bug in the implementation [4] and Figure 6.

> "Three Rings for the Elven-kings under the sky, . . . "
> – J. R. R. Tolkien, *The Lord of the Rings*, epigraph

1 Introduction

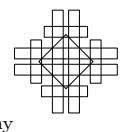
A ***tiling*** of the plane [9] is a covering of the plane by nonoverlapping polygons called ***tiles***, isometric copies of one or more geometric shapes called ***prototiles***, without gaps or overlaps. In this paper, we study the most fundamental computational problem about tilings:

▶ **Problem 1** (Tiling). *Given one or more prototiles, can they tile the plane?*

The tiling problem is ***undecidable*** – solved by no finite algorithm. Golomb [6] was first to prove this result, by building n polyominoes that simulate n ***Wang tiles*** [15] – unit squares with edge colors that must match – by adding color-specific bumps and dents to each edge. Four years earlier, Berger [1] proved that tiling with Wang tiles is undecidable (disproving Wang's original conjecture [15]) by showing how they can simulate a Turing machine. Robinson [13] later simplified Berger's proof. The worst-case number n of tiles (Wang or polyomino) is $\Theta(|Q| \cdot |\Sigma|)$, where $|Q|$ and $|\Sigma|$ are the number of states and symbols in the simulated Turing machine, respectively.

Constant Number of Prototiles. The first constant and previously best upper bound on the number of prototiles required to make the tiling problem undecidable is 5, as proved by Ollinger fifteen years ago [11]. Our main result, proved in Section 3, is an improvement of this upper bound to 3:

© Erik D. Demaine and Stefan Langerman;
licensed under Creative Commons License CC-BY 4.0



▶ **Theorem 1.1.** *Given three simple-polygon prototiles, determining whether they tile the plane is undecidable.*

It remains open whether tiling with one or two given prototiles is decidable. ***Periodic*** tilings (tilings with two translational symmetries) can be found algorithmically by enumerating fundamental domains, as we show in the full paper. (Surprisingly, this intuitive fact does not seem to have been explicitly proved before, except in special settings like Wang tiles [15].) Thus a necessary condition for undecidability is the existence of prototile(s) with only aperiodic tilings. Recently, Smith, Myers, Kaplan, and Goodman-Strauss [14] found a single prototile with this property, so there are no obvious obstacles to undecidability.

Tiling by Translation. Our construction relies on rotation of the prototiles (but works independent of whether we allow reflections). If we restrict to tiling by translation only, then Ollinger's construction can be modified to use 11 prototiles, by adding some rotations of the five polyominoes [11]. This upper bound was improved to 10 by Yang [17] and to 8 by Yang and Zhang [18]. All of these constructions use polyominoes. In higher dimensions, Yang and Zhang [19] improved the upper bound to five polycube prototiles in 3D, and four polyhypercube prototiles in 4D.

The tiling-by-translation problem also has a lower bound of 2 for undecidability: any single polygon that tiles the plane by translation can do so by periodic (even isohedral) tiling [5]. This result also holds for disconnected polyominoes [2]. If we generalize to tiling a specified periodic subset of d-dimensional space, where d is part of the input, then Greenfeld and Tao [8] recently proved tiling to be undecidable with a single disconnected polyhypercube.

Periodic Target. We show that Greenfeld and Tao's generalization to tiling a specified periodic subset [8] changes the best known results also for undecidability of tiling the plane. Our 3-polygon construction and Ollinger's 5-polyomino construction [11], and Yang and Zhang's 8-polyomino translation-only construction [18] all have one prototile (our shurikens, and their jaws) that appear periodically in any tiling of the plane. Thus, if we remove that pattern from the target, we obtain a periodic subset of the plane which can be tiled using a reduced number of prototiles of 2, 4, and 7, respectively. In particular, we prove

▶ **Corollary 1.2.** *Given two simple-polygon prototiles, and given a periodic subset of the plane, determining whether the two prototiles tile the periodic subset is undecidable.*

Logical Undecidability. Algorithmic undecidability implies ***logical undecidability*** (as explained in [7] in the context of tilings). In particular, our result implies that there are three polygon prototiles that cannot be proved or disproved to tile the plane, for any fixed set of axioms (e.g., ZFC). Otherwise, we would obtain a finite algorithm to decide tileability, by enumerating all proofs.

▶ **Corollary 1.3.** *For any fixed set of axioms, there are three fixed simple-polygon prototiles such that both "these prototiles tile the plane" and "these prototiles do not tile the plane" have no proof.*

Tiling Completion. Undecidability of tiling requires the set of prototiles to depend on the Turing machine simulation. To obtain undecidability with a fixed set of prototiles, we can generalize the tiling problem as follows [13]:

▶ **Problem 2** (Tiling Completion)**.** *Given one or more prototiles, and given some already placed tiles, can this placement be extended to a tiling of the plane?*

Robinson [13] gave the first result on this problem: a set of 36 prototiles (Wang tiles or polygons) for which tiling completion is undecidable. This result applies the general Turing machine simulation to Minsky's 4-symbol 7-state universal Turing machine, so only a finite number of tiles need to be preplaced to represent the Turing machine to simulate. Likely this result could be improved using newer smaller universal Turing machines [16]. If we allow for (countably) infinitely many tiles to be preplaced, we can use semi-universal Turing machines and simulate Rule 110, enabling undecidability with just six supertiles (Wang tile or polygons) [20]. Our main result reduces this upper bound to 3, in the stronger model of finitely many preplaced tiles:

▶ **Corollary 1.4.** *There are three fixed simple-polygon prototiles such that, given a finite set of already placed tiles, determining whether this placement can be extended to a tiling of the plane is undecidable.*

Co-RE-completeness. While past results on tiling and tiling completion have focused on undecidability, all such proofs actually show ***co-RE-hardness***: the simulated Turing machine halts if and only if the prototiles fail to tile. Co-RE-hardness is a more precise statement than undecidability, so we use that phrasing here. But it raises the question: are tiling and tiling completion in co-RE? Surprisingly, this question does not seem to have been solved (or even asked) in the literature before. In Section 4, we prove the answer is "yes":

▶ **Theorem 1.5.** *Given a finite set of polygon prototiles, and given a (possibly empty) connected set of already placed tiles, determining whether this placement can be extended to a tiling of the plane is in co-RE.*

This result holds in a very general model for polygons: the angles and edge lengths can be represented as ***computable*** numbers (meaning that a Turing machine can output the first n bits, given n). Our three-polygon construction uses a more restricted model, where the angles are rational multiples of π and the edge lengths are constant-size radical expressions, showing the problem to be co-RE-complete for every model in between.

▶ **Corollary 1.6** (Stronger form of Theorem 1.1)**.** *Given three simple-polygon prototiles, where the edge lengths and angles in degrees are specified by computable numbers or by constant-size radical expressions, determining whether they tile the plane is co-RE-complete.*

2 Wang Tiling: Signed and Unsigned

We reduce from Wang tiling, which is known to be undecidable. A ***Wang tile*** is a square with a ***glue*** on each edge. Classically, Wang tiles are ***unsigned***, meaning that glues match if they are equal, and ***translation-only***, meaning they have a specified orientation of which edge is north, east, south, and west. In 1966, Berger proved unsigned Wang tiling undecidable:

▶ **Theorem 2.1** ([1])**.** *Given a set of Wang tiles, it is co-RE-hard to determine whether they tile the plane by translation only, matching glues of equal value.*

Our first reduction converts unsigned Wang tiling to a variant that is ***signed***, meaning every glue has a sign ($+$ or $-$) and value, and glues match if they have opposite sign and equal value, and ***free***, meaning the tile can be rotated and/or reflected. This result was proved by Robinson [13, p. 179]. Figure 1 sketches the reduction; the full paper gives the details for completeness.

▶ **Lemma 2.2** ([13]). *Given a set of unsigned translation-only Wang tiles T, we can construct a set of signed free Wang tiles T' that has the same tilings as T up to global isometry, allowing or forbidding reflection.*

(a) 11 unsigned translation-only Wang tiles T that tile only aperiodically, the minimum possible [10].

(b) Equivalent 11 signed free Wang tiles T'. Bumps/dents denote signs.

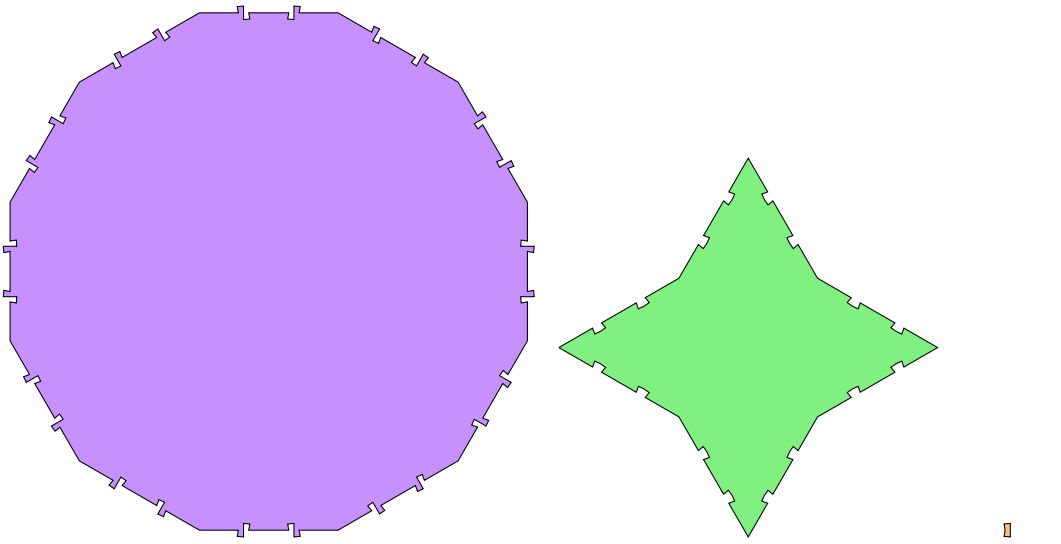
Figure 1 Example of converting unsigned translation-only Wang tiles to signed free Wang tiles.

Henceforth when we say "Wang tiles" we mean signed free Wang tiles.

3 Three Tiles That Simulate n Signed Wang Tiles

We implement any set of n Wang tiles with three tiles, illustrated in Figure 2:

1. the *wheel* which encodes all of the Wang tiles,
2. the *staple* which covers the unused Wang tiles of each wheel, and
3. the *shuriken* which fills in the remaining gaps.

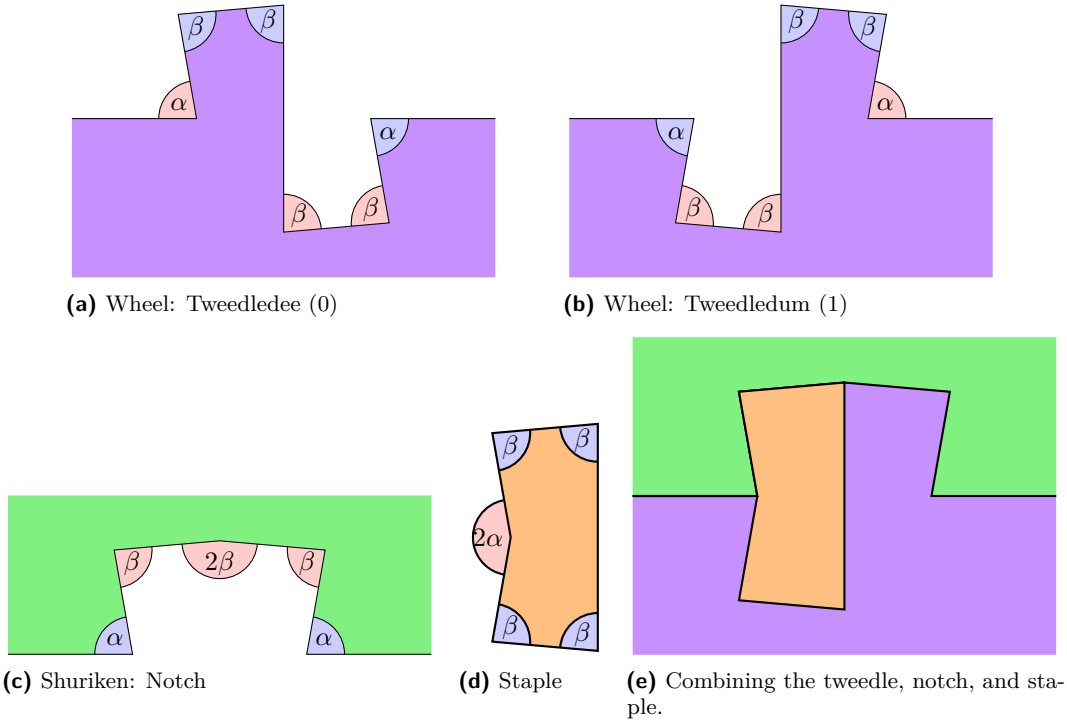


(a) Wheel **(b)** Shuriken **(c)** Staple

Figure 2 The three tiles in our construction, to scale; Figure 3 shows zoomed details of the construction. The wheel is just an example; it depends on the n Wang tiles being simulated. The shuriken depends (only) on n.

3.1 Construction and Intended Tiling

Suppose we are given a set of n Wang tiles, where the ith tile ($1 \le i \le n$) has signed glues n_i, e_i, s_i, w_i on its north, east, south, and west edges respectively. Assume n is an odd integer ≥ 5 by possibly adding duplicate tiles.



(a) Wheel: Tweedledee (0)

(b) Wheel: Tweedledum (1)

(c) Shuriken: Notch

(d) Staple

(e) Combining the tweedle, notch, and staple.

Figure 3 Zoomed views of portions of the three tiles in our construction (15× scale vs. Figure 2).

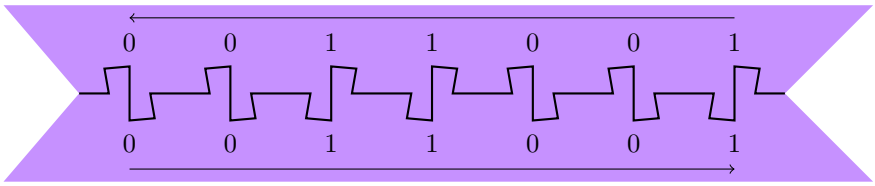


Figure 4 Matching a glue (top) and its negative (bottom) between two wheels.

The ***wheel*** is a regular $4n$-gon with each edge adorned by bumps and notches representing the $4n$ glues. For tile i, the glues n_i, e_i, s_i, w_i adorn sides $i, n+i, 2n+i, 3n+i$ of the $4n$-gon, respectively. To encode a glue, we encode its value in binary using $b = O(\log n)$ bits, prepend a 00 at the beginning, and append 01 at the end. For negative glues, we reverse the order of the bits, which puts a 10 at the beginning and a 00 at the end. Then we represent each bit with a ***tweedledee*** (0) or ***tweedledum*** (1) gadget, which are rotationally symmetric zig-zags shown in Figures 3a and 3b. Both follow the sequence of angles $\alpha, \beta, \beta, \beta, \beta, \alpha$ (where α and β are defined below); for tweedledee, this sequence measures defect, angle, angle, defect, defect, angle, respectively; while for tweedledum, this measures the opposite (angle, defect, defect, angle, angle, defect).[1] As shown in Figure 4, two adjacent glues match exactly if and only if they have the same value and opposite sign (where the opposite sign is enforced by the 00 and 01 at either end). This representation also ensures that reflecting a wheel will produce reflected glues that do not match unreflected glues: a reflection causes the bits of a

[1] It is also possible to use two different convex angles β_1, β_2 in place of each repetition β, β, but the notation is messier, so we opt for this simpler construction.

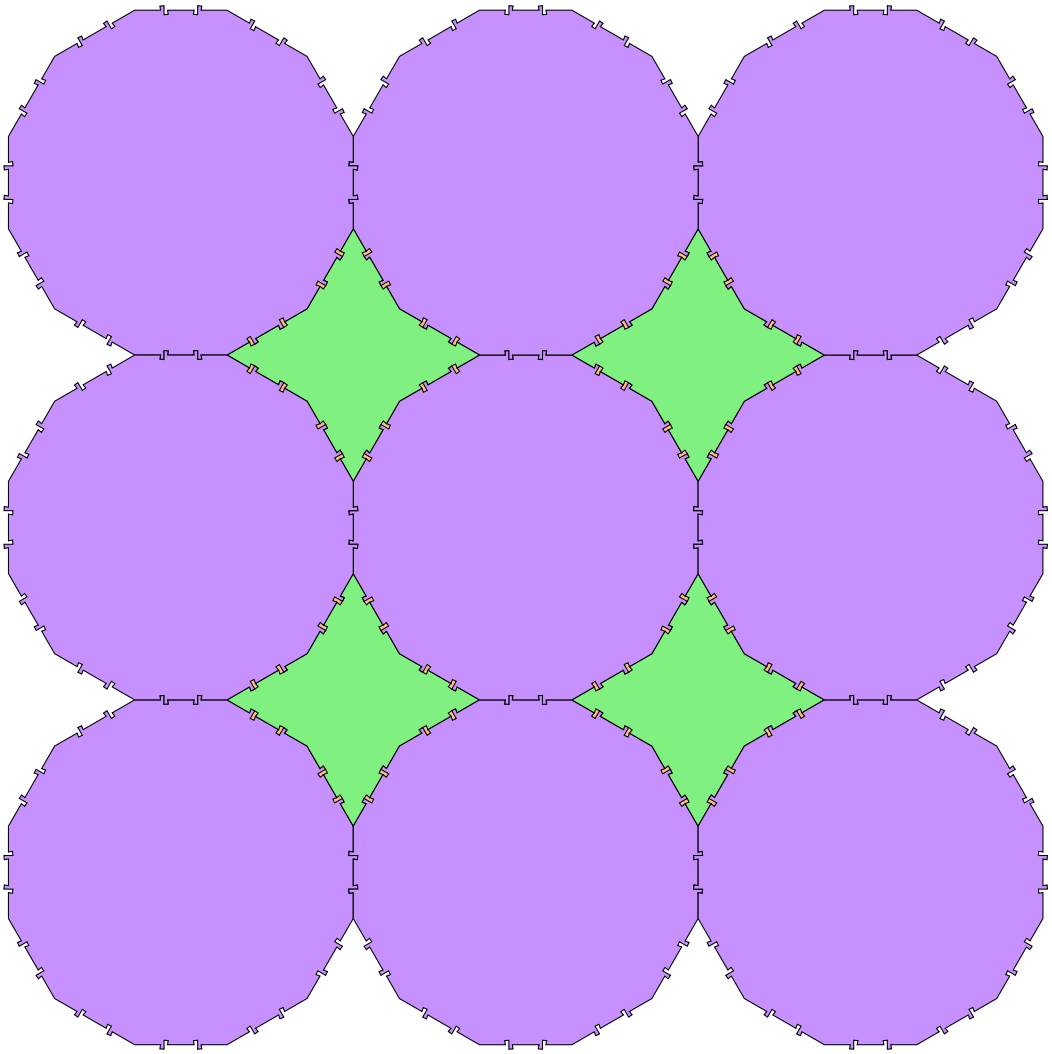


Figure 5 Example tiling with the wheel, shuriken, and staple.

glue to be reversed and negated, so the reflection of a positive glue starts with 01 and ends with 11, and the reflection of a negative glue starts with 11 and ends with 10, both of which are incompatible with unreflected glues.

By this construction, rotating the wheel so that its ith side is horizontal and at the top will have its north, east, south, and west sides represent the glues n_i, e_i, s_i, w_i of tile i. Given a tiling of the plane using this set of Wang tiles, we can place copies of the rotated wheel exactly as in the Wang tiling, and the glues will match exactly. Some space remains between the wheels, which we fill with "staples" and "shurikens". See Figure 5.

The ***shuriken*** is composed of four regular concave chains of $n-1$ sides, matching the lengths and complementary to the angles of the regular $4n$-gon. Each side is adorned with b reflectionally symmetric ***notches***, shown in Figure 3c, each consisting of convex angle α; reflex deficits $\beta, 2\beta, \beta$; and convex angle α. As shown in Figure 3e, each notch can fit a tweedle of either kind, leaving a space that is filled exactly by a ***staple*** (shown in Figure 3d, and consisting of convex angles $\beta, \beta, \beta, \beta$ and reflex deficit 2α). Thus each side of the shuriken can exactly match any glue, effectively hiding the unused tiles of each wheel (the glues that are not on the north, east, south, or west sides). Thus we have shown:

▶ **Lemma 3.1.** *Given a set of n Wang tiles and a tiling of the plane with them, we can construct a tiling of the plane with the wheel, the shuriken, and the staple constructed above.*

It remains to show that this is the only way our three tiles can tile the plane.

3.2 Angle Structure

We start with a few definitions and observations on the angles of the tiles.

Call an angle ***clean*** (and color it purple) if it is an integer multiple of $\frac{\pi}{2n}$. The sum of clean angles is clean, and the sum of clean angles and one nonclean angle is not clean.

The vertices of the convex $4n$-gon, which we call ***corners***, have clean convex angle $\pi(1-\frac{1}{2n})$. The matching shuriken reflex ***anticorners*** have a matching defect $\pi(1-\frac{1}{2n})$, and each of the four concave chains are connected at the convex ***tip*** vertices, which have a clean convex angle of $\frac{\pi}{n}$.

Define angles $\alpha = \frac{\pi}{2} - 2\varepsilon$ and $\beta = \frac{\pi}{2} - \varepsilon$, and pick $\varepsilon = \frac{\pi}{16}$.[2] These angles and their combinations are not clean:

▶ **Property 1.** *For any $\theta_1, \theta_2 \in \{\alpha, \beta\}$, neither θ_1 nor $\theta_1 + \theta_2$ is clean.*

Proof. The relevant angles are $\alpha = \frac{\pi}{2} - 2\varepsilon = \frac{3\pi}{8}$, $\beta = \frac{\pi}{2} - \varepsilon = \frac{7\pi}{16}$, $2\alpha = \frac{3\pi}{4}$, $2\beta = \frac{7\pi}{8}$, and $\alpha + \beta = \frac{13\pi}{16}$, which all have a doubly even denominator (divisible by 4). Because n is odd, these numbers cannot be equal to $\frac{i\pi}{2n}$ for any integer i, so none of these angles are clean. ◀

Table 1 Angles used in the wheel, shuriken, and staple. For convex vertices, we give the interior angle, while for reflex vertices, we give the defect (2π minus the interior angle).

Shape	angle of convex vertices	defect of reflex vertices
staple	β	2α
shuriken	$\alpha, \frac{\pi}{n}$	$\beta, 2\beta, \pi(1-\frac{1}{2n})$
wheel	$\alpha, \beta, \pi(1-\frac{1}{2n})$	α, β

Table 1 lists the angles used by each polygon, colored to indicate which are clean. Angles α, β are all a bit less than 90°, so a sum of two of them is a bit less than 180°, and a sum of three of them is a bit less than 270°. More precisely:

▶ **Lemma 3.2.** *For any $\theta_1, \theta_2, \theta_3 \in \{\alpha, \beta\}$, $\theta_1 \in (\frac{3}{8}\pi, \frac{7}{16}\pi)$, $\theta_1 + \theta_2 \in (\frac{3}{4}\pi, \frac{7}{8}\pi)$ which is $< \pi$, and $\theta_1 + \theta_2 + \theta_3 \in (\frac{9}{8}\pi, \frac{21}{16}\pi)$ which is $> \pi$. Note that these intervals are disjoint, so the value of a sum $\sum_i \theta_i$ with at most three terms determines the number of terms in the sum.*

3.3 Edge Lengths

We design the edge lengths of the tweedledee and tweedledum in Figures 3a and 3b so that the near-vertical and near-horizontal edges all have the same length, which we call 1, and the total horizontal traversal is exactly 4. Equivalently, we choose the sequence of edge lengths for either tweedle to be

$$\begin{aligned}&\langle 2 - \sin\beta + \cos\alpha,\ 1,\ 1,\ 2(\sin\alpha + \cos\beta),\ 1,\ 1,\ 2 - \sin\beta + \cos\alpha\rangle \\ &= \langle 2 - \cos\varepsilon + \sin 2\varepsilon,\ 1,\ 1,\ 2(\cos 2\varepsilon + \sin\varepsilon),\ 1,\ 1,\ 2 - \cos\varepsilon + \sin 2\varepsilon\rangle.\end{aligned}$$

[2] Other choices of ε also work; the choice here is so that all edge lengths can be expressed by radical expressions, and all angles are rational multiples of π (or equivalently, rational numbers of degrees).

Given our choice of $\varepsilon = \frac{\pi}{16}$,

$$\cos\varepsilon = \frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2}}}, \qquad \sin\varepsilon = \frac{1}{2}\sqrt{2-\sqrt{2+\sqrt{2}}},$$
$$\cos 2\varepsilon = \frac{1}{2}\sqrt{2+\sqrt{2}}, \qquad \sin 2\varepsilon = \frac{1}{2}\sqrt{2-\sqrt{2}}$$

are all radical expressions, so all our edge lengths can be expressed by radical expressions.

To prevent the notches from intersecting each other at the tip of the shuriken (with the sharp angle of π/n), we place the notches of the shuriken and the tweedles of the wheel at a distance of n from the anticorners of the shuriken and the corners of the wheel respectively. That is, each side of the shuriken is composed of an edge of length n, followed by the b notches, followed by an edge of length n; and each side of the wheel is composed of an edge of length n, followed by the b tweedles composing the glue, followed by an edge of length n.

3.4 Forced Tiling Structure

▶ **Lemma 3.3.** *The staple does not tile the plane (even allowing reflections).*

Proof. The staple has convex vertices just of angle β and one reflex vertex of defect 2α. In any tiling of staples, every reflex vertex must have its defect 2α filled by some convex angles. But $2\alpha = \frac{3}{4}\pi$, so by Lemma 3.2, it must be filled by exactly two convex angles. But $2\beta = \frac{7}{8}\pi > 2\alpha$, so two of the convex angles do not fit. Thus it is impossible to exactly fill the deficit. ◀

▶ **Lemma 3.4.** *The staple and shuriken do not tile the plane (even allowing reflections).*

Proof. By Lemma 3.3, any tiling with staples and shurikens has a shuriken. Any shuriken has a reflex anticorner of defect $\pi(1-\frac{1}{2n})$, which is clean and less than π. This defect must be filled by some convex angles, of which we have three: $\alpha, \beta, \frac{\pi}{n}$. By Lemma 3.2, this defect can be filled by at most two angles $\in \{\alpha, \beta\}$. But by Property 1, summing one or two of these angles is not clean, while the remaining convex angle $\frac{\pi}{n}$ and the target sum $\pi(1-\frac{1}{2n})$ are. Thus we cannot use any angles $\in \{\alpha, \beta\}$ to fill the deficit, leaving only the convex angle $\frac{\pi}{n}$. But $\frac{\pi}{n}$ is an even multiple of $\frac{\pi}{2n}$, while the target sum is an odd multiple of $\frac{\pi}{2n}$. Thus it is impossible to exactly fill the deficit. ◀

▶ **Lemma 3.5.** *Any tiling of the plane with staples, shurikens, and wheels (even allowing reflections) must consist of an infinite square grid of wheels corresponding to a Wang tiling.*

Proof. By Lemma 3.4, any tiling with staples, shurikens, and wheels has a wheel. Any wheel has a convex corner of angle $\pi(1-\frac{1}{2n})$, which has a deficit of $\pi(1+\frac{1}{2n})$, which is clean. We claim that this deficit can be filled in exactly two ways: one reflex anticorner of a shuriken of deficit $\pi(1-\frac{1}{2n})$, or one convex tip vertex of a shuriken of angle $\frac{\pi}{n}$ and one convex corner of a wheel of angle $\pi(1-\frac{1}{2n})$.

Because the target deficit is $> \pi$, we need to consider both convex and reflex angles as well as flat edges (of angle π) for possible fillings. First consider the unclean angles starting with reflex angles of deficit $\alpha, \beta, 2\alpha, 2\beta$. But $\alpha < \beta < 2\alpha < 2\beta = \frac{7}{8}\pi = \pi(1-\frac{1}{8}) < \pi(1-\frac{1}{2n})$ for any $n \geq 5$. Thus none of the unclean reflex angles can be used to fill the deficit. The only remaining reflex angle that can fill the deficit is a shuriken anticorner of deficit $\pi(1-\frac{1}{2n})$, and if we use that angle, we are done.

Next consider the unclean convex angles α and β. Because $\pi(1+\frac{1}{2n}) < \frac{9}{8}\pi$ for any $n \geq 5$, by Lemma 3.2 this deficit can be filled by at most two angles $\in \{\alpha, \beta\}$. But by Lemma 1, summing one or two of these angles is not clean, while the remaining convex angles $\frac{\pi}{n}, \pi(1-\frac{1}{2n})$, flat edge of angle π, and the target sum $\pi(1+\frac{1}{2n})$ are all clean. Thus we cannot use any angles $\in \{\alpha, \beta\}$ to fill the deficit.

This leaves only the flat edge of angle π and two convex angles: the shuriken tip of angle $\frac{\pi}{n}$ and the wheel corner of angle $\pi(1-\frac{1}{2n})$. If we used only copies of the tip $\frac{\pi}{n}$, we would get an even multiple of $\frac{\pi}{2n}$, but the target sum $\pi(1+\frac{1}{2n})$ is an odd multiple of $\frac{\pi}{2n}$. Using a flat edge π would leave a gap of angle $\frac{\pi}{2n}$, which is too small to fill with any of the available angles. Finally, using the wheel corner $\pi(1-\frac{1}{2n})$ (gluing a second wheel to the first) will leave a gap of angle $\frac{\pi}{n}$, which can only be filled by the tip angle $\frac{\pi}{n}$.

Thus we have shown that, in all cases, the wheel's convex angle $\pi(1-\frac{1}{2n})$ has to be matched with an anticorner or a tip of the shuriken. Furthermore, an edge of the wheel adjacent to a corner must be glued to an edge of the concave chain adjacent to a tip or anticorner of the shuriken. We can follow the path of both the wheel and the concave chain of the shuriken and observe that the $n-2$ anticorners and the two tips of that concave chain will be glued to consecutive corners of the wheel.

At the end of the concave chain of the shuriken, we find a tip glued to the corner of a wheel, leaving a deficit of $\pi(1-\frac{1}{2n})$ to fill. As shown in Lemma 3.4, this can only be filled by another wheel corner. Therefore, the surround of a wheel must be filled by an alternating sequence of wheels and shurikens (omitting the small gaps left between the tweedles and the notches, which are filled by staples).

Pick one wheel T in the tiling, translate the tiling so that its center[3] is at the origin $(0,0)$, and rotate the plane so that edge i of its $4n$-gon is glued to another wheel at a horizontal edge of the $4n$-gon, with T below that edge. Also rescale so that the width of a wheel's $4n$-gon (the distance between parallel edges, without adornments) is 1. Then the wheel adjacent to edge i of T will have its center at coordinate $(0,1)$. Following the boundary of T clockwise, we find a shuriken glued to the edges $i+1, \ldots, i+n-1$ of T, and a wheel glued to the edge $i+n$ of T. The wheel glued to edge $i+n$ has its center at coordinates $(1,0)$. Continuing this reasoning, we find that the tiling is a grid of wheels with centers on all integer lattice points. The shurikens and staples ensure that all spaces are filled, and the tweedles ensure that the tiles are compatible as in a Wang tiling. Therefore, for any tiling with the three tiles, we can produce a tiling of the plane with the original Wang tiles. ◀

3.5 Undecidability

The angles of the polygons, as listed in Table 1, are all rational multiples of π. The edge lengths of the polygons, as listed in Section 3.3, can all be expressed as radical expressions. We call polygons with such angles and edge lengths ***nice***.

▶ **Theorem 3.6.** *Given n Wang tiles, we can construct three nice polygons that can tile the plane (allowing or forbidding reflections) if and only if the Wang tiles can tile the plane.*

Proof. Combine Lemma 3.1 (if) and Lemma 3.5 (only if). ◀

Combining this with membership of the tiling problem in co-RE (to be proved in the next section), we obtain:

[3] Define the ***center*** of a wheel to be the center of gravity of its $4n$-gon, without adornments.

▶ **Corollary 3.7** (Nice form of Corollary 1.6). *Given three nice polygons in the plane, deciding whether they tile the plane is co-RE-complete and thus undecidable.*

In a recent paper, Greenfeld and Tao [8] consider a generalized version of the tiling problem, where only a periodic subset of space needs to be covered by the tiles. In our reduction, the union of the shurikens form a periodic subset of $\mathbb{R}^2$, and so does its complement. Thus, tiling the complements of the shurikens with the two remaining tiles is undecidable:

▶ **Corollary 3.8** (Stronger form of Corollary 1.2). *Given two nice polygons, deciding whether they tile a periodic subset of the plane is co-RE-complete and thus undecidable.*

By plugging in Wang tiles that simulate a universal Turing machine, such as Robinson's 36 Wang tiles [13], we also obtain undecidability of tiling completion with three specific tiles:

▶ **Corollary 3.9** (Stronger form of Corollary 1.4). *There exist three fixed tiles for which completing a given finite partial tiling is co-RE-complete and thus undecidable.*

We implemented our construction in an open-source web application [4], where the user can input any set of Wang tiles and see the resulting polygons. Figure 6 shows the output for the 11 Wang tiles from Figure 1.

4 Membership in Co-RE

The previous sections show the co-RE-hardness of the tiling problem for three given tiles, and the tiling completion problem for three fixed tiles. We counterbalance these intractability results by showing that the tiling problem and tiling completion problem are in co-RE, for any finite set of prototiles. Refer to the full paper for omitted proofs in this section.

4.1 Model: Computable Polygons

To make these positive results as strong as possible, we use the weakest possible model of computation. We use a standard Turing machine, and represent polygons by their sequences of angles and edge lengths (equivalently, instructions in the Logo/Turtle graphics language), which need not be given explicitly but can be computed to any desired precision. More precisely, we assume the input polygons are "computable" in the following sense:

▶ **Definition 4.1.** *A real number a is **computable** [3] if there is a Turing machine T_a that, given a natural number n, outputs an integer $T_a(n)$ such that $\frac{T_a(n)-1}{n} \le a \le \frac{T_a(n)+1}{n}$. A polygon is **computable** if it is promised to be simple and closed, and its n angles and edge lengths are computable.*

Computability is likely the most general representation of real numbers that is still usable for our problem. Computable numbers include all rational numbers, algebraic numbers, and transcendental numbers that can be computed to any desired precision. In particular they are closed under addition, subtraction, multiplication, division, integer roots and even trigonometric functions:

▶ **Theorem 4.2** ([3, Theorem 4.14]). *If x, y, and z are computable real numbers with $z > 0$, then $x + y$, $x - y$, xy, x/z, $|x|$, $\min(x, y), \max(x, y)$, $\exp(x), \sin(x), \cos(x)$, $\log(z)$, and $\sqrt{z}$ are computable as well.*

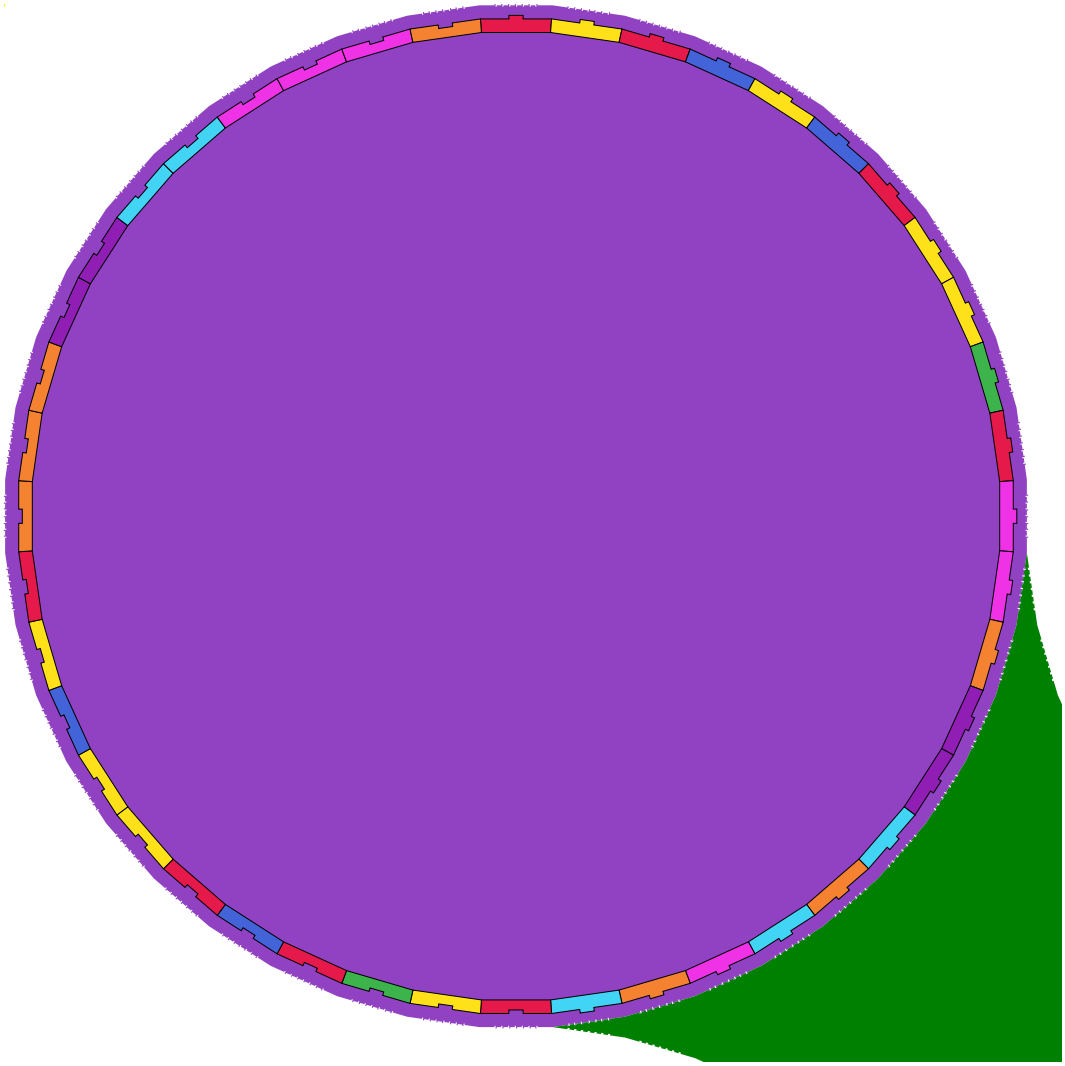


Figure 6 Output of our implemented reduction [4] for the 11 Wang tiles from Figure 1, focusing on the wheel and how the shuriken fits around the bottom-right corner. The wheel is annotated with edge colors from Figure 1b.

Note that some basic operations can be intractable for computable numbers. For instance, determining whether a computable number is zero, or the equality between two computable numbers, is undecidable (in fact, co-RE-complete). To see this, given a Turing machine T, define a number a_T to have its nth bit after the binary point be 1 if T halts after n steps, and 0 otherwise. The number a_T is computable, and is zero if and only if T does not halt. For our problem, we can show a similar undecidability:

▶ **Theorem 4.3.** *Given a single computable pentagon, determining whether it tiles the plane is co-RE-hard and thus undecidable.*

Proof. Pick a generic quadrilateral, and glue a very flat isosceles triangle to one of its edges, where the apex angle is $180° - r/100$ for a given constructible number $r \in [0, 1]$. The other angles and edge lengths of the triangle are constructible via trigonometric functions.

The resulting pentagon tiles the plane only in the degenerate case where the triangle is degenerate, i.e., a line segment, which happens exactly when the obtuse angle of the triangle is exactly 180°. Thus the tiling problem is equivalent to testing whether $r = 0$ for a given constructible number $r \in [0, 1]$, which is co-RE-hard as shown above. ◀

Of course, this result is quite unsatisfying, as the reason for undecidability stems from the extreme weakness of the model and generality of the representation of the polygons, and the inability to even check locally that a tiling is valid. Yet surprisingly, in this same model, we are able to show membership in co-RE.

4.2 Co-RE Algorithm

The high-level idea of our algorithm is to try to build partial tilings that cover a larger and larger disk. If we ever fail to cover a disk, then we know that the plane cannot be tiled; and if we never fail, the well-known Extension Theorem (Theorem 4.11 below) guarantees that the plane can be tiled. To determine whether we can tile enough to cover a disk, we bound the number of tiles that could possibly intersect the disk, then enumerate all possible combinatorial ways for these tiles to fit together, and for each, check whether the tiles fit together properly. Checking fit is limited to tiles that share vertices, however, so we need to take care to handle the case that there are seam lines in the tiling where no tiles on opposite sides of the line share a vertex (as in, e.g., the classic brick tiling). We also avoid checking for global intersection between tiles (because doing so is tricky in our model), opting instead to check just locally that angles add up correctly at vertices and that edge lengths add up correctly along edges. Our notion of "neat carpet" handles both of these issues by forbidding only local self-overlap, and guaranteeing that every boundary vertex is either outside the specified disk or has total angle 180° so potentially forms a seam boundary. We are then able to show that arbitrarily large neat carpets imply the existence of a plane tiling.

Our co-RE algorithm will in particular need to repeatedly test for equality among constructible numbers, a co-RE-complete problem. Thus we need a way to compose co-RE decisions. We use the following standard result (mentioned, e.g., in [12]):

▶ **Lemma 4.4.** *Finite disjunctions and recursively enumerable conjunctions of co-RE decision problems are in co-RE.*

Let $\mathcal{T}$ be a set of prototiles, where each tile is a (simple closed) computable polygon. Define a ***carpet*** to be a topological disk produced by gluing together a finite collection of tiles from $\mathcal{T}$, where every interior vertex has 360° total angle from incident tiles. We assume that the carpet is laid out in the Euclidean plane, that is, every point in a tile has real coordinates, but we allow the surface to be self-overlapping, that is, a point of the plane might be covered by more than one tile. A ***patch*** is a carpet whose embedding in the plane is not self-overlapping.

A carpet can be described by its ***combinatorial gluing***, which specifies (1) the set of tiles, each of which is an instance of a prototile; (2) a partition of the tile vertices into coincident (glued together) points; and (3) for each tile edge, the sequence of other tile edges and/or boundary that the edge has positive-length overlap with, in order along the edge. Call a carpet or a patch ***seamless*** if the position of all tiles in the carpet is fully determined by the combinatorial gluing and the position of its first tile. This notion forbids carpets whose tiles can be separated by a line along which the two sides could slide (causing an uncountable infinity of solutions). To verify seamlessness, build the ***incidence graph*** on the tiles of the carpet, where two tiles are connected by an edge if they share a vertex, and

check that the graph is connected. For the case of tiling completion, we can ensure that the given patch is seamless by adding a vertex along each seam, common to both adjacent tiles; as this modification to the preplaced tiles does not change their shape, it does not change the outcome of the decision.

First we show how to verify that a carpet is valid:

▶ **Lemma 4.5.** *Given a combinatorial gluing of a possible seamless carpet, deciding whether it corresponds to a seamless carpet is co-RE-complete.*

Define the ***distance*** between two points in a carpet to be the Euclidean distance between those two points when the carpet is laid out in the Euclidean plane (note that this embedding may be self-overlapping, and this distance is no larger than the intrinsic distance within the carpet). Call a vertex of a carpet ***neat*** if it is either interior to the carpet and surrounded by tiles summing up to an angle of 2π, or is on the boundary of the carpet and is surrounded by contiguous tiles summing up to an angle of π. A carpet is ***neat within radius $< r$*** if every vertex at distance $< r$ from the origin is neat. In an ***anchored*** carpet, we assume one ***anchored*** tile in the combinatorial gluing has been chosen to be placed with its center of gravity at the origin, and with a canonical rotation (say, matching its prototile).

▶ **Lemma 4.6.** *Given an anchored carpet and a computable positive number r, deciding whether it is neat within radius $< r$ is in co-RE.*

Let ρ be the maximum "radius" of tiles in $\mathcal{T}$, meaning that a disk of radius ρ centered at the center of gravity of each tile in $\mathcal{T}$ covers that tile. Let $A_{\min}$ be the minimum area of a tile in $\mathcal{T}$. Let D_r denote the disk of radius r centered at the origin.

▶ **Lemma 4.7.** *If $\mathcal{T}$ can tile the plane, then for any $r > 0$, (a) there is an anchored patch with a finite number $N(r)$ of tiles that covers the disk D_r, and (b) there is an anchored seamless patch with $\leq N(r)$ tiles that is neat within radius $< r$.*

If a seamless patch P can be extended to tile the plane, then for any $r > 0$, (a) there is an anchored patch containing P with a finite number $|P| + N(r)$ of tiles that covers the disk D_r, and (b) there is a seamless patch containing P with $\leq |P| + N(r)$ tiles that is neat within radius $< r$.

Proof. For the (a) statements, translate (and rotate) the tiling so that one of its tiles is anchored; and if we are given a seamless patch P, choose to anchor one of its tiles. Consider the disk $D_{r+2\rho}$ of radius $r + 2\rho$ centered at the origin (the anchored tile's center of gravity). Take all the tiles in the plane tiling that are fully inside $D_{r+2\rho}$, and take all the tiles of P (if given). Because the tiles do not overlap and are each of area $\geq A_{\min}$, there are at most $\pi(r + 2\rho)^2/A_{\min}$ tiles inside $D_{r+2\rho}$. Take the connected component that contains the origin (and thus the tiles of P, if given), which only decreases the number of tiles. This is an anchored patch that covers the smaller disk D_r.

For the (b) statements, build the incidence graph on the tiles of the carpet, where two tiles are connected by an edge if they share a vertex. If this graph is connected, then the patch is seamless. If this graph is disconnected, it is because of seams. Seam lines cannot intersect: otherwise, their intersection point is a vertex common to both sides of the seams. Thus, cutting the patch along all seams, or equivalently taking one connected component of the incidence graph, will create neat vertices on the new boundary, where the seams were. Take the component that contains a tile covering the origin. This is a seamless patch that is neat within radius $< r$. ◀

▶ **Lemma 4.8.** *Given prototiles $\mathcal{T}$ and a computable radius r, deciding whether there is an anchored carpet that is neat within radius $< r$ is in co-RE. If there is no such carpet, then $\mathcal{T}$ cannot tile the plane.*

Given prototiles $\mathcal{T}$, a seamless patch P, and a computable radius r, deciding whether P can be extended to a carpet that is neat within radius $< r$ is in co-RE. If there is no such carpet, then P cannot be extended to tile the plane.

▶ **Lemma 4.9.** *If an anchored carpet is neat within radius $< r + 2\rho$, then the carpet contains an anchored patch that is neat within radius $< r$.*

Proof. Consider the intersection between the carpet C and the disk $D_{r+2\rho}$, which might have multiple connected components and/or be self-overlapping. Pick the component S that contains the anchored tile. The boundary of S is composed of straight lines (the edges of the tiles connected by neat vertices) and circular arcs (portions of the boundary of $D_{r+2\rho}$), connected together at convex angles. Thus S is convex, so non-self-overlapping.

Back in the carpet, remove all tiles that are not entirely contained in $D_{r+2\rho}$. Again, this possibly results in several connected components. Retain the component C' that contains the anchored tile, where connectivity is defined by interior paths, so that C' does not have pinch points. By construction, this component is contained in S and thus is not self-overlapping. Also C' is a topological disk: a hole in C' could only come from a removed tile, which must touch the outside of $D_{r+2\rho}$, contradicting that it is surrounded by C' (by planarity). Therefore, C' is a patch.

Finally, because all tiles removed have diameter $\leq 2\rho$, none of the deleted tiles intersect the disk D_r, and therefore all vertices within radius $< r$ remain untouched. Therefore the patch C' is neat within radius $< r$. ◀

▶ **Lemma 4.10.** *If there exists an anchored carpet C that is neat within radius $< r + 2\rho$, then there exists a patch that covers the disk $D_{r/2}$. Furthermore, that patch contains all tiles of C that intersect $D_{r/2}$.*

Proof. Suppose we have a carpet that is neat within radius $< r + 2\rho$. By Lemma 4.9, we have a patch that is neat within radius $< r$. The intersection of the patch and the disk D_r is a disk cut by noncrossing chords; refer to Figure 7. Note that any chord of D_r that intersects $D_{r/2}$ cuts off an arc of angle $\frac{2}{3}\pi$ from D_r. Because the chords defined by the patch do not intersect in D_r, at most two of them intersect $D_{r/2}$.

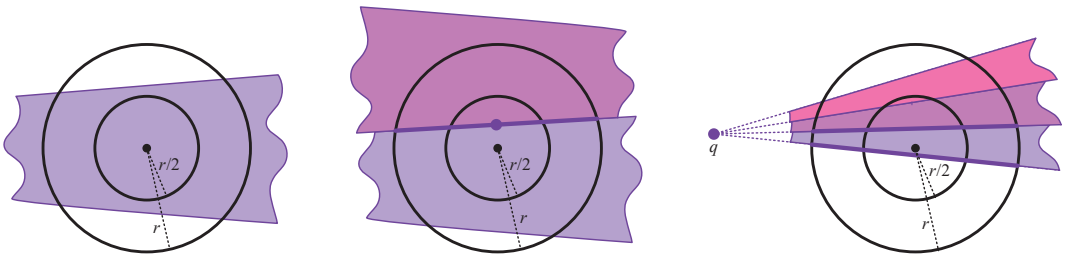


Figure 7 A neat patch within $< r$, and how it can interact with the smaller disk $D_{r/2}$. From left to right: no chords, one chord, and two chords.

If no chord intersects $D_{r/2}$, then the patch covers $D_{r/2}$, and we are done.

If exactly one chord intersects $D_{r/2}$, then the patch is contained in a half-plane that contains the origin and is bounded by the extension of the cord. Rotate a copy of the patch by 180° about the center of the cord, and glue the two pieces together. This produces a patch that covers $D_{r/2}$.

If two chords intersect $D_{r/2}$, then the patch is contained in a wedge defined by the extensions of the two chords. Let q be the apex of the wedge, and assume q is on the x axis, with one of the chords above the x axis and the other chord below. Assume without loss of generality that the chord below is the longest one. Repeatedly stitch copies of the patch by rotating it about q, upward, gluing the longer chord of the previous copy to the shorter chord of the next copy, until the upper half of $D_{r/2}$ is covered. Now repeat the procedure for a single chord. The resulting patch covers the disk of radius $r/2$. ◀

▶ **Theorem 4.11** (Extension Theorem [9, p. 151])**.** *Given a finite collection $\mathcal{T}$ of prototiles, if they tile arbitrarily large disks, then they admit a tiling of the plane.*

Given a finite collection $\mathcal{T}$ of prototiles and patch P using tiles of $\mathcal{T}$, if P can be extended to cover arbitrarily large disks centered in P, then P can be completed to tile the plane.[4]

We translate the above theorem to seamless anchored patches.

▶ **Lemma 4.12.** *Given a collection $\mathcal{T}$ of prototiles, if there exist anchored carpets that are neat within radius $< r$ for arbitrarily large r, then $\mathcal{T}$ admits a tiling of the plane.*

Given a collection $\mathcal{T}$ of prototiles, and given an anchored patch P using tiles of $\mathcal{T}$, if P can be extended to a carpet that is neat within radius $< r$ for arbitrarily large r, then P can be completed to tile the plane.

▶ **Theorem 4.13** (Precise form of Theorem 1.5)**.** *Given a set $\mathcal{T}$ of k polygons in our model, deciding whether they tile the plane is in co-RE. Also given a patch P of tiles from $\mathcal{T}$, deciding whether P can be completed to tile the plane is in co-RE.*

Proof. For every positive integer k, set $r = k\rho$ and use Lemma 4.8 to determine whether there exists an anchored carpet that is neat within radius $< r$, or whether P can be completed to produce such a carpet. This is a recursively enumerable disjunction of co-RE problems, so by Lemma 4.4 is in co-RE. By Lemma 4.12, if all of these problems output true, then the polygons tile the plane or complete P to tile the plane. By Lemma 4.8, if any of these problems outputs false, then the polygons do not tile or complete P to tile the plane. ◀

References

1. Robert Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66:1–72, 1966. doi:10.1090/memo/0066.
2. Siddhartha Bhattacharya. Periodicity and decidability of tilings of $\mathbb{Z}^2$. *American Journal of Mathematics*, 142(1):255–266, 2020. doi:10.1353/ajm.2020.0006.
3. Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms*, pages 425–491. Springer, New York, NY, 2008. doi:10.1007/978-0-387-68546-5_18.
4. Erik D. Demaine and Stefan Langerman. edemaine/three-tiles, 2024. Software, swhId: swh:1:dir:aef5f642bf8a40c0cecf56bfb62b9e089effb265 (visited on 2025-06-05). URL: https://github.com/edemaine/three-tiles, doi:10.4230/artifacts.23339.
5. D. Girault-Beauquier and M. Nivat. Tiling the plane with one tile. In *Topology and Category Theory in Computer Science*. Oxford University Press, August 1991. doi:10.1093/oso/9780198537601.003.0012.
6. Solomon W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9(1):60–71, 1970. doi:10.1016/S0021-9800(70)80055-2.

[4] Although [9, p. 151] states only the first version of the theorem, the same proof establishes both.

7 Rachel Greenfeld and Terence Tao. Undecidable translational tilings with only two tiles, or one nonabelian tile. *Discrete & Computational Geometry*, 70(4):1652–1706, 2023. doi:10.1007/s00454-022-00426-4.

8 Rachel Greenfeld and Terence Tao. Undecidability of translational monotilings. *Journal of the European Mathematical Society*, 2024. To appear. arXiv:2309.09504.

9 Branko Grünbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman, 1987.

10 Emmanuel Jeandel and Michaël Rao. An aperiodic set of 11 Wang tiles. *Advances in Combinatorics*, January 2021. doi:10.19086/aic.18614.

11 Nicolas Ollinger. Tiling the plane with a fixed number of polyominoes. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA 2009)*, volume 5457 of *Lecture Notes in Computer Science*, pages 638–647, Tarragona, Spain, April 2009. Springer. doi:10.1007/978-3-642-00982-2_54.

12 H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953. doi:10.2307/1990888.

13 Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12(3):177–209, September 1971. doi:10.1007/BF01418780.

14 David Smith, Joseph Samuel Myers, Craig S. Kaplan, and Chaim Goodman-Strauss. An aperiodic monotile, 2023. doi:10.48550/arXiv.2303.10798.

15 Hao Wang. Proving theorems by pattern recognition – II. *Bell Systems Technical Journal*, 40(1):1–41, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.

16 Damien Woods and Turlough Neary. The complexity of small universal turing machines: A survey. *Theoretical Computer Science*, 410(4):443–450, 2009. doi:10.1016/j.tcs.2008.09.051.

17 Chao Yang. Tiling the plane with a set of ten polyominoes. *International Journal of Computational Geometry & Applications*, 33(03n04):55–64, 2023. doi:10.1142/S0218195923500012.

18 Chao Yang and Zhujun Zhang. A proof of Ollinger's conjecture: undecidability of tiling the plane with a set of 8 polyominoes, 2024. doi:10.48550/arXiv.2403.13472.

19 Chao Yang and Zhujun Zhang. Undecidability of translational tiling of the 4-dimensional space with a set of 4 polyhypercubes, 2024. doi:10.48550/arXiv.2409.00846.

20 Jed Chang-Chun Yang. *Computational Complexity and Decidability of Tileability*. PhD thesis, University of California, Los Angeles, 2013.

Apex Representatives

Tamal K. Dey
Purdue University, West Lafayette, IN, USA

Tao Hou
University of Oregon, Eugene, OR, USA

Dmitriy Morozov
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

Given a zigzag filtration, we want to find its barcode representatives, i.e., a compatible choice of bases for the homology groups that diagonalize the linear maps in the zigzag. To achieve this, we convert the input zigzag to a levelset zigzag of a real-valued function. This function generates a Mayer–Vietoris pyramid of spaces, which generates an infinite strip of homology groups. We call the origins of indecomposable (diamond) summands of this strip their apexes and give an algorithm to find representative cycles in these apexes from ordinary persistence computation. The resulting representatives map back to the levelset zigzag and thus yield barcode representatives for the input zigzag. Our algorithm for lifting a p-dimensional cycle from ordinary persistence to an apex representative takes $O(p \cdot m \log m)$ time. From this we can recover zigzag representatives in time $O(\log m + C)$, where C is the size of the output.

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

Keywords and phrases zigzag persistent homology, Mayer–Vietoris pyramid, cycle representatives

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.40

Related Version *Full Version*: `https://arxiv.org/abs/2502.17704` [8]

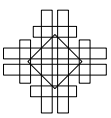
Funding *Tamal K. Dey*: partially supported by NSF grants CCF-2437030 and DMS-2301360.
Tao Hou: supported by NSF fund CCF 2439255.
Dmitriy Morozov: supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

1 Introduction

In topological data analysis, one often wants to not only describe the distribution of topological features in data, but also to identify individual features directly in the input. For persistent homology, a natural starting point are the cycle representatives of homology classes. Specifically, one wants to find *barcode representatives*, a consistent choice of bases for the input sequence of homology groups. For ordinary persistence, this comes directly from the computation: a representative at the start of the bar includes into every consecutive space and therefore remains a valid choice of the basis element.

For zigzag persistence, the situation is more complicated. Because simplices can enter and leave the complex, a representative cycle at one step in the zigzag filtration may not exist at a later step. So even though zigzag persistence algorithms keep track of representative cycles, the specific choices that they make do not always produce compatible barcode representatives. To cope with this problem, Dey et al. [9] introduced an algorithm that keeps track of additional information that allows it to report consistent barcode representatives. For a zigzag filtration with m insertions and deletions, the algorithm requires $O(m^3)$ preprocessing and $O(m^2)$ time to report one or all representatives for any single bar, and $O(m^3)$ time to report all representatives.

© Tamal K. Dey, Tao Hou, and Dmitriy Morozov;
licensed under Creative Commons License CC-BY 4.0



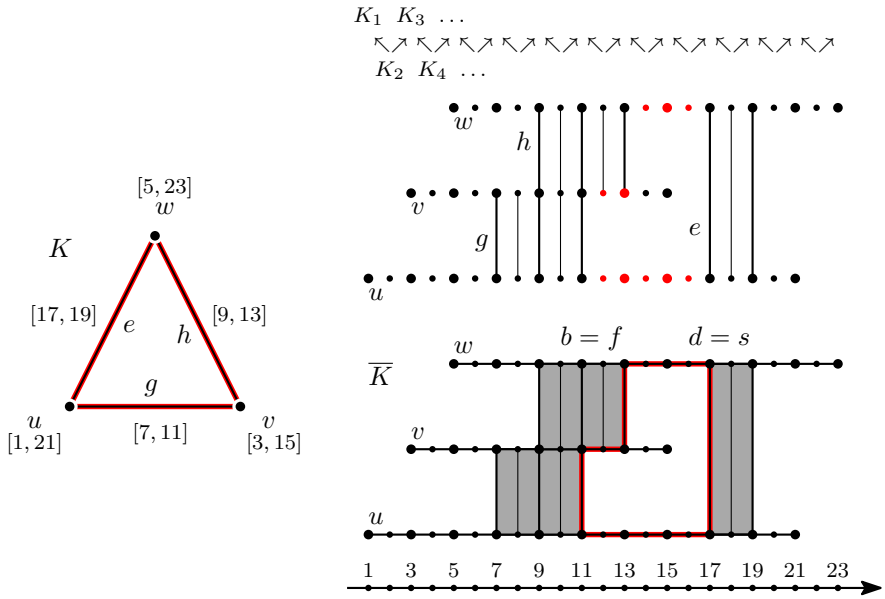


Figure 1 A zigzag on the top. A total complex K on the left. The support of each simplex in a zigzag, $T(\sigma)$, shown as an interval. The corresponding prism $\overline{K}$ on the bottom. A 1-cycle in $K[11, 17]$ and its lift in $\overline{K}[11, 17]$ are highlighted in red. The lifted cycle is an apex representative; its slices (pairs of vertices in red) are representatives of a bar in the original zigzag.

On the surface, this is as well as one can hope to do: there are $O(m)$ bars, over $O(m)$ spaces, with each representative consisting of $O(m)$ simplices. But we can do better. Every zigzag filtration is isomorphic to a levelset zigzag filtration of some function. The function generates the Mayer–Vietoris pyramid [5], which decomposes into diamond summands [2]. Each diamond has a single space as its origin, its apex: this space maps into every other space in the diamond. By finding the diamond's representative in the apex – an apex representative – we can quickly recover its representative in every other space, including those that fall in the levelset zigzag, and therefore in the original zigzag; see Figure 1. All the information can be recovered from an ordinary persistence computation. It takes matrix multiplication time, $O(m^\omega)$, to compute persistence [16], and an extra $O(p \cdot m \log m)$ time to recover a p-dimensional apex representative of size $O(p \cdot m)$. After the preprocessing, zigzag representatives can be recovered in $O(\log m + C)$ time, where C is the size of the output.

2 Background

We assume familiarity with simplicial and cell complexes, homology, persistent homology, including zigzag persistence. We only recap select topics to establish notation. Any gaps can be filled with the standard literature [14, 12, 13].

Throughout the paper, we take care to work with arbitrary field coefficients, but abuse notation as follows. Cell $\sigma \in z$ means the coefficient of σ in z is non-zero, $\langle \sigma, z \rangle \neq 0$. If α is a chain in some cell complex K, then $\alpha \subseteq L$ means that α is supported on the subcomplex $L \subseteq K$, i.e., $\forall \sigma \in (K - L), \langle \sigma, \alpha \rangle = 0$. Similarly, $z \cap L$ refers to the restriction of chain z to L, i.e., $z \cap L = \sum_{\sigma \in L} \langle \sigma, z \rangle \cdot \sigma$.

Persistence. Given a filtered cell complex K, let D be a matrix that represents its boundary operator, with rows and columns ordered according to the filtration. To find ordinary persistence pairing, one can compute a decomposition, $R = DV$, where matrix R is reduced, meaning its pivots – lowest non-zero elements in its columns – appear in unique rows, and matrix V is full-rank upper-triangular. We index the columns and rows of the matrices by the (totally ordered) cells of the input complex. The persistence pairing is given by the pivots of matrix R: a p-dimensional homology class created by the addition of simplex σ is destroyed by the addition of simplex τ iff $\operatorname{low} R[\tau] = \sigma$, where $\operatorname{low} R[\tau]$ returns the pivot in the column $R[\tau]$. By definition, the corresponding column $V[\tau]$ stores a $(p+1)$-dimensional chain whose boundary is the cycle $R[\tau]$.

The decomposition $R = DV$ is not unique: multiple matrices R and V satisfy it. However, the original persistence algorithm [10] performs operations in a lazy fashion, subtracting columns from left to right only if their pivots collide. We call this a *lazy reduction*. It has the following special properties that simplify our algorithms.

▶ **Lemma 1** (Lemma 1 in [17]). *Assume decomposition $R = DV$ is obtained via the lazy reduction, and $\sigma_i = \operatorname{low} R[\tau_i]$ and $\sigma_j = \operatorname{low} R[\tau_j]$ are such that $\tau_i < \tau_j$. If $\sigma_i < \sigma_j$, then entry $V[\tau_i, \tau_j] = 0$.*

Proof. Induction on V; see [17]. ◀

▶ Remark 2. The authors of [17] simplify the statement of the preceding lemma by abusing the notation. They assume that if $R[\tau] = 0$, then $\operatorname{low} R[\tau] = \bar{\sigma}$, a special imaginary cell that precedes every cell in the complex. In particular, it follows that in a lazy reduction, entry $V[\tau_i, \tau_j] = 0$ if $R[\tau_i] = 0$.

The contrapositive of the lemma is the following corollary.

▶ **Corollary 3.** *Assume decomposition $R = DV$ is obtained via the lazy reduction, and $\sigma_i = \operatorname{low} R[\tau_i]$ and $\sigma_j = \operatorname{low} R[\tau_j]$ are such that $\tau_i < \tau_j$. If $V[\tau_i, \tau_j] \neq 0$, then $\sigma_j < \sigma_i < \tau_i < \tau_j$.*

Zigzag persistence. We start with a zigzag of simplicial complexes,

$$K_0 \to K_1 \leftarrow K_2 \to K_3 \leftarrow K_4 \to \ldots \leftarrow K_n, \tag{1}$$

where every consecutive pair of complexes vary by at most one simplex, i.e., K_i can be equal to K_{i+1} in the sequence. It is convenient to assume that every simplex σ appears and disappears exactly once in this sequence, and therefore is present during some interval $T(\sigma) = [i, j]$, i.e., $\sigma \in K_k$ iff $k \in T(\sigma)$. This assumption can be made without loss of generality by assuming that the union K of all simplices across all times, $K = \bigcup_i K_i$ forms a Δ-complex. The full definition of this object is too verbose – we refer the reader to [7], which adapts the classical concept of Δ-complexes in [14, p. 103] to our case of zigzag filtration – but informally it relaxes the requirement that in a simplicial complex two simplices intersect in a common face. All the familiar notions of simplicial homology work the same, but now instead of having a simplex supported over multiple intervals in the zigzag, we can have multiple simplices, with the same boundary, each supported on a single interval. We use $m = |K|$ to denote the size of the input complex.

▶ Remark. To simplify the notation we let $\min(\sigma) = \min T(\sigma)$ and $\max(\sigma) = \max T(\sigma)$.

▶ Remark 4. We assume that $\max(\sigma) > \min(\sigma)$, i.e., $T(\sigma)$ is not a single point. We can assume this without loss of generality because we can always pad the zigzag with extra copies of a space.

Applying homology, we get a zigzag of homology groups, where we suppress dimension for simplicity,

$$\mathsf{H}(K_0) \to \mathsf{H}(K_1) \leftarrow \mathsf{H}(K_2) \to \mathsf{H}(K_3) \leftarrow \mathsf{H}(K_4) \to \ldots \leftarrow \mathsf{H}(K_n). \tag{2}$$

Just like with ordinary persistence, this zigzag decomposes into k bars or persistence intervals [3]. Let $\{\alpha_i^1, \ldots, \alpha_i^k\}$ be a choice of elements in $\mathsf{H}(K_i)$ such that the non-zero elements form a basis for $\mathsf{H}(K_i)$. We say that such bases are *compatible* across the zigzag if the maps in Equation (2) diagonalize, i.e., $\alpha_{2i-1}^j \leftarrow\!\shortmid\, \alpha_{2i}^j \mapsto \alpha_{2i+1}^j$. For every j, α_*^j are non-zero over a single persistence interval $[b^j, d^j]$. Cycles z_i^j that give a set of compatible bases $\alpha_i^j = [z_i^j]$ are called *zigzag representatives*, with z_*^j being the representatives of the j-th bar in the zigzag barcode.

Real-valued function. A convenient setting for persistence is that of a Morse-like [4] real-valued function $f : \mathbb{X} \to \mathbb{R}$. We denote with $\mathbb{X}_a^b = f^{-1}[a, b]$ the preimage of an interval and allow the endpoints to be infinite, $a, b = \pm\infty$, in which case the interval is understood to be open at the infinite ends. We denote the following pairs of spaces:

$$\mathbb{X}[b, d] = (\mathbb{X}_b^d, \emptyset), \qquad \mathbb{X}(b, d] = (\mathbb{X}_{-\infty}^d, \mathbb{X}_{-\infty}^b),$$
$$\mathbb{X}[b, d) = (\mathbb{X}_b^\infty, \mathbb{X}_d^\infty), \qquad \mathbb{X}(b, d) = (\mathbb{X}, \mathbb{X}_{-\infty}^b \cup \mathbb{X}_d^\infty).$$

Let $a_1 < \ldots < a_n$ be the critical values of the function f. Let s_i be regular values interleaved with the critical values: $s_0 < a_1 < s_1 < a_2 < \ldots < s_{n-1} < a_n < s_n$. The following constructions play an important role in the theory of persistent homology:

- Extended persistence (EP):

$$\begin{aligned} 0 \to \mathsf{H}(\mathbb{X}[-\infty, s_0]) \to \ldots \to \mathsf{H}(\mathbb{X}[-\infty, s_n]) = \mathsf{H}(\mathbb{X}) \\ \to \mathsf{H}(\mathbb{X}[-\infty, s_n)) \to \ldots \to \mathsf{H}(\mathbb{X}[-\infty, s_1)) \to \mathsf{H}(\mathbb{X}[-\infty, s_0)) = 0. \end{aligned}$$

- Levelset zigzag (LZZ):

$$0 \leftarrow \mathsf{H}(\mathbb{X}[s_0, s_0]) \to \mathsf{H}(\mathbb{X}[s_0, s_1]) \leftarrow \mathsf{H}(\mathbb{X}[s_1, s_1]) \to \ldots \to \mathsf{H}(\mathbb{X}[s_{n-1}, s_n]) \leftarrow \mathsf{H}(\mathbb{X}[s_n, s_n]) \to 0.$$

Carlsson et al. [5] showed that these two sequences contain the same information by arranging the four types of spaces into a *Mayer–Vietoris Pyramid*, see Figure 2(left). Once homology is applied, the pyramid unrolls into an infinite Mayer–Vietoris strip of homology groups, where every diamond belongs to the Mayer–Vietoris long exact sequence, making it *exact* in the terminology of [3], see Figure 2(right). This allows one to translate the decomposition between any two paths that differ by a single diamond (and therefore between any pair of paths via composition).

Bendich et al. [2] further showed that the translation rules in [5] hold at the level of the basis elements of the individual paths, which means the entire pyramid decomposes as a direct sum of (indecomposable) pointwise 1-dimensional diamond summands, flush with its boundaries, shown in blue in Figure 2(right). The decomposition of the levelset zigzag and extended persistence are simply slices through these flush diamond summands. Bauer et al. [1] study the Mayer–Vietoris Pyramid in the cohomological setting.

We call the bottom-most space in the support of a diamond its *apex*. The homology class assigned to the diamond in that space, an *apex class*, and any cycle that belongs to such a class, an *apex representative*. We note that just like the choice of the basis for

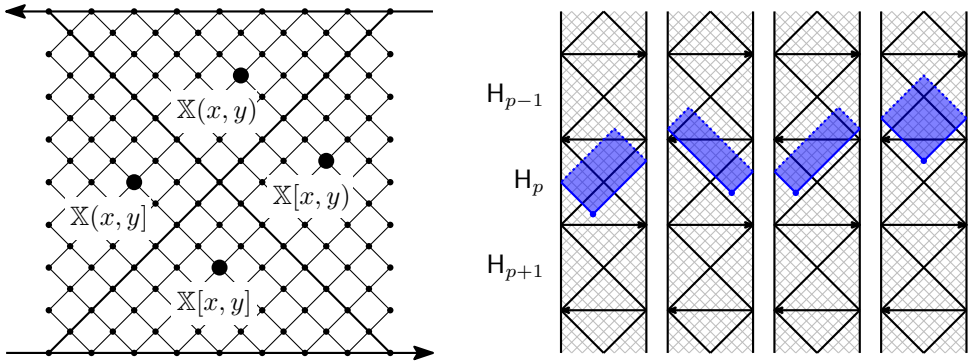


Figure 2 Left: Mayer–Vietoris Pyramid arranges spaces $\mathbb{X}[b,d], \mathbb{X}(b,d], \mathbb{X}[b,d), \mathbb{X}(b,d)$ in a way where every diamond in the pyramid belongs to the Mayer–Vietoris long exact sequence. Right: Four types of flush diamond indecomposables in the infinite strip of homology groups; the apex of each diamond is marked.

ordinary persistence is not unique, neither is the choice of the apex classes. An apex class is characterized by lying outside the image of the maps into the apex; specifically, for the four types of apexes their classes satisfy:

$$
\begin{aligned}
&[z] \in \mathsf{H}(\mathbb{X}[b,d]) \text{ s.t. } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}[b+\varepsilon,d]) \to \mathsf{H}(\mathbb{X}[b,d])) \text{ and } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}[b,d-\varepsilon]) \to \mathsf{H}(\mathbb{X}[b,d]))\\
&[z] \in \mathsf{H}(\mathbb{X}(b,d]) \text{ s.t. } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}(b-\varepsilon,d]) \to \mathsf{H}(\mathbb{X}(b,d])) \text{ and } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}(b,d-\varepsilon]) \to \mathsf{H}(\mathbb{X}(b,d]))\\
&[z] \in \mathsf{H}(\mathbb{X}[b,d)) \text{ s.t. } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}[b+\varepsilon,d)) \to \mathsf{H}(\mathbb{X}[b,d))) \text{ and } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}[b,d+\varepsilon)) \to \mathsf{H}(\mathbb{X}[b,d)))\\
&[z] \in \mathsf{H}(\mathbb{X}(b,d)) \text{ s.t. } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}(b+\varepsilon,d)) \to \mathsf{H}(\mathbb{X}(b,d))) \text{ and } [z] \notin \operatorname{im}(\mathsf{H}(\mathbb{X}(b,d-\varepsilon)) \to \mathsf{H}(\mathbb{X}(b,d)))
\end{aligned}
$$

Because there is a map from the apex to every space in the diamond, we can recover a representative in any space by simply mapping an apex representative forward. All the maps in the pyramid are inclusions, except for the boundary homomorphism connecting two consecutive dimensions of homology in the strip. We say more about this translation in Section 7, but the overarching point is that given an apex representative, it is straightforward to recover every other representative, and levelset zigzag representatives in particular.

3 Setup

Prism. Given a zigzag in Equation (1), we define a larger cell complex $\overline{K}$, called a *prism*:

$$\overline{K} = \{\sigma \times [i,i+1] \mid [i,i+1] \subseteq T(\sigma), i \in \mathbb{Z}\} \cup \{\sigma \times i \mid i \in T(\sigma), i \in \mathbb{Z}\}, \tag{3}$$

together with a function $\overline{f} : |\overline{K}| \to \mathbb{R}$ on its underlying space, defined by the projection onto the second component, $\overline{f}(x \times t) = t$. See Figure 3. We observe that the prism consists of two types of cells: *horizontal cells* $\sigma \times [i,i+1]$ and *vertical cells* $\sigma \times i$.

We define an integer interlevel set, for $i,j \in \mathbb{Z}$, $\overline{K}_i^j = \{\sigma \times T \in \overline{K} \mid T \subseteq [i,j]\}$. We note that its underlying space $|\overline{K}_i^j| = \overline{f}^{-1}[i,j]$. Then for any pair of integers b,d, we get four types of pairs of subspaces,

$$\overline{K}[b,d] = \overline{K}_b^d, \quad \overline{K}(b,d] = (\overline{K}_{-1}^d, \overline{K}_{-1}^b), \quad \overline{K}[b,d) = (\overline{K}_b^{n+1}, \overline{K}_d^{n+1}), \quad \overline{K}(b,d) = (\overline{K}, \overline{K}_{-1}^b \cup \overline{K}_d^{n+1}).$$

Applying homology to each pair, we get relative homology groups that appear in the four quadrants of the Mayer–Vietoris pyramid: $\mathsf{H}(\overline{K}[b,d]), \mathsf{H}(\overline{K}(b,d])), \mathsf{H}(\overline{K}[b,d)), \mathsf{H}(\overline{K}(b,d))$.

Because even-indexed spaces in Equation (1) include into odd-indexed spaces, $K_{2i-1} \leftarrow K_{2i} \to K_{2i+1}$, we have an isomorophism of homology groups $\mathsf{H}(\overline{K}[2i,2i+1]) \simeq \mathsf{H}(\overline{K}[2i+1,2i+1]) \simeq \mathsf{H}(\overline{K}[2i+1,2i+2])$, induced by deformation retractions. It follows that the

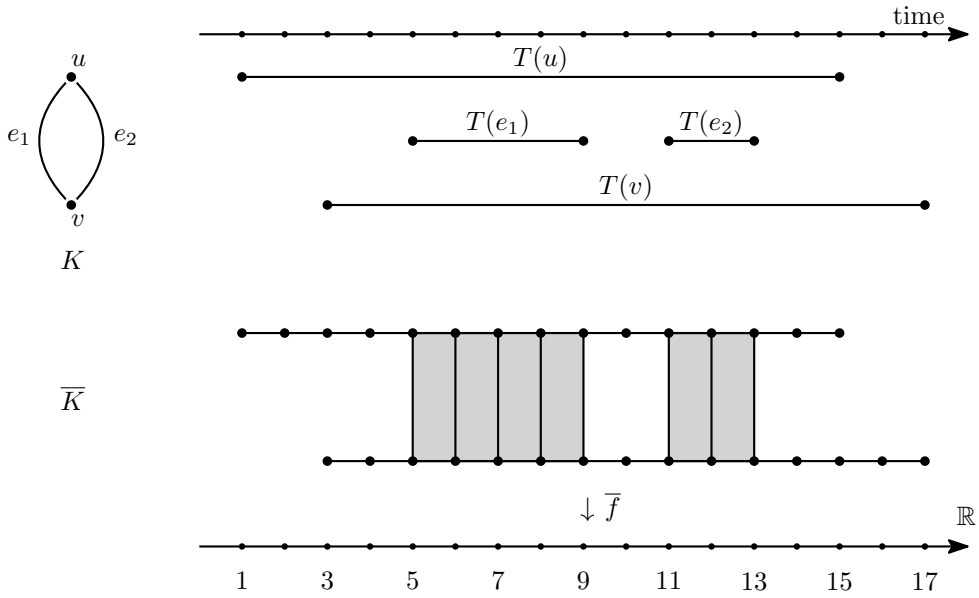


Figure 3 A Δ-complex K, with the times $T(\sigma)$ of its simplices illustrated, together with the corresponding prism $\overline{K}$ and its projection $\overline{f}$ onto the second coordinate.

levelset zigzag [5] of function $\overline{f}$,

$$\underbrace{\mathsf{H}(\overline{K}[0,0])}_{\mathsf{H}(K_0)} \to \underbrace{\mathsf{H}(\overline{K}[0,1]) \xleftarrow{\simeq} \mathsf{H}(\overline{K}[1,1]) \xrightarrow{\simeq} \mathsf{H}(\overline{K}[1,2])}_{\mathsf{H}(K_1)} \leftarrow \underbrace{\mathsf{H}(\overline{K}[2,2])}_{\mathsf{H}(K_2)} \to \ldots \leftarrow \underbrace{\mathsf{H}(\overline{K}[n,n])}_{\mathsf{H}(K_n)}$$

is isomorphic to our starting zigzag in Equation (2). Explicitly, the isomorphism is given by the following bijection between the four types of intervals:

$$\begin{array}{lcll}
(\mathsf{H}(K_{2i}), \mathsf{H}(K_{2j})) & \leftrightarrow & (\mathsf{H}(\overline{K}[2i,2i]), \mathsf{H}(\overline{K}[2j,2j])) & \text{(open-open)} \\
(\mathsf{H}(K_{2i}), \mathsf{H}(K_{2j+1})) & \leftrightarrow & (\mathsf{H}(\overline{K}[2i,2i]), \mathsf{H}(\overline{K}[2j+1,2j+2])) & \text{(open-closed)} \\
(\mathsf{H}(K_{2i+1}), \mathsf{H}(K_{2j})) & \leftrightarrow & (\mathsf{H}(\overline{K}[2i,2i+1]), \mathsf{H}(\overline{K}[2j,2j])) & \text{(closed-open)} \\
(\mathsf{H}(K_{2i+1}), \mathsf{H}(K_{2j+1})) & \leftrightarrow & (\mathsf{H}(\overline{K}[2i,2i+1]), \mathsf{H}(\overline{K}[2j+1,2j+2])) & \text{(closed-closed)}
\end{array}$$

Intervals. Because we perform computation on the input complex K, rather than the prism, we need a notation for different intervals and pairs of intervals. The connection between these and the corresponding intervals in $\overline{K}$ will be made clear in Section 6. We denote the sub- and super-level sets:

$$K^i_{-1} = \{\sigma \in K \mid \min(\sigma) \le i\}, \qquad K^{n+1}_j = \{\sigma \in K \mid \max(\sigma) \ge j\},$$

and define four pairs of spaces:

$$\begin{array}{ll}
K[b,d] = K^{n+1}_b \cap K^d_{-1}, & K(b,d] = (K^d_{-1}, K^b_{-1}), \\
K[b,d) = (K^{n+1}_b, K^{n+1}_d), & K(b,d) = (K, K^b_{-1} \cup K^{n+1}_d).
\end{array}$$

Because every step of the input zigzag is a simplicial complex, it follows that for any $\sigma, \tau \in K$,

$$\sigma \in \partial\tau \Rightarrow \min(\sigma) < \min(\tau) < \max(\tau) < \max(\sigma). \tag{4}$$

The following relationships between sub- and super-levelsets follow immediately:

$$\alpha \subseteq K^x_{-1} \Rightarrow \partial\alpha \subseteq K^x_{-1} \tag{5}$$

$$\alpha \subseteq K - K^{n+1}_x \Rightarrow \alpha \subseteq K^x_{-1} \tag{6}$$

Cone. To compute the persistence decomposition of the zigzag, we follow the algorithm of Dey and Hou [7], who use a construction similar to extended persistence [6] on the cone $\hat{K} = \omega * K$ over the input complex K, where $*$ indicates a join with the cone vertex ω. Every simplex σ in the base space K gets value $\min(\sigma)$ from the input zigzag. Every simplex $\hat{\sigma}$ in the cone $\hat{K} - K$ gets value $\max(\sigma)$ from the input zigzag. We then define a filtration, where the base space simplices come first, ordered by increasing value, and the cone simplices come second, ordered by decreasing value. Specifically,

$$0 \to \mathsf{H}(K^1_{-1}) \to \ldots \to \mathsf{H}(K^n_{-1}) = \mathsf{H}(K) \to \mathsf{H}(K, K^{n+1}_n) \to \ldots \to \mathsf{H}(K, K^{n+1}_1) \to \mathsf{H}(K, K) = 0,$$

which in reduced homology is isomorphic to

$$0 \to \mathsf{H}(K^1_{-1}) \to \ldots \to \mathsf{H}(K) \to \mathsf{H}(K \cup \omega * K^{n+1}_n) \to \ldots \to \mathsf{H}(K \cup \omega * K^{n+1}_1) \to \mathsf{H}(\hat{K}) = 0. \tag{7}$$

The latter is an ordinary filtration and we compute its persistence via the $R = DV$ decomposition of the boundary map of the cone $\hat{K}$. Crucially, the resulting bars are in one-to-one correspondence with the bars in the decomposition of the input zigzag in Equation (2).

4 Lifting

Because prism $\overline{K}$ has so many cells, it is expensive to process directly. We want to perform computation on the cone $\hat{K}$ instead. To recover the apex representatives for the prism, we need to lift cone cycles in $\hat{K}$ to prism cycles in $\overline{K}$. In Section 6, we explain what exact cycles to lift to get different apex representatives. Meanwhile, we first describe an algorithm that given a (relative) cycle z in KI produces a cycle $\overline{z}$ in $\overline{K}I$, where I is one of the four types of intervals, $(b, d]$, $[b, d)$, (b, d), $[b, d]$. Algorithm 1 in this section is a reinterpretation of an inefficient, but easier-to-understand Lift-Cycle-easy algorithm in the full version of this paper [8]. We encourage the reader to go through through that algorithm first.

▶ Remark. A p-chain $\overline{z}$ in $\overline{K}$ consists of two types of p-cells: $\tau \times t$, where τ is a p-simplex, and $\sigma \times [t_1, t_2]$, where σ is a $(p-1)$-simplex.

▶ Remark. We use an abridged notation, $c \cdot \sigma \times [i, j]$, to represent the chain $\sum_{k \in [i, j-1]} c \cdot \sigma \times [k, k+1]$. This choice is guided by computational efficiency: the former requires constant space vs. the $O(j - i)$ space required for the latter.

Algorithm 1 takes a p-cycle z, a direction expressed as a pair of values (s, f), and an initial $(p-1)$-cycle w_{init}. We call the pair (s, f) a direction because the order of the values specifies whether we process the cycle in increasing ($s < f$) or decreasing ($f < s$) order. The algorithm stretches the cycle from $\overline{K}[s, s]$ to $\overline{K}[f, f]$, covering $\overline{K}I$; see Figure 1.

Correctness. The correctness of Algorithm 1 follows from the following claim about the boundary structure, which will be important in Section 6. We let z^y_x be the restriction of cycle z to the simplices τ whose times t_τ lie in the interval $[x, y]$, i.e., $z^y_x = \sum_{\tau \in z, t_\tau \in [x,y]} \langle \tau, z \rangle \cdot \tau$.

▷ Claim 5. Given input cycle $z \in K$, the boundary of the lifted cycle $\overline{z}$ satisfies $\partial\overline{z} = w_{\text{init}} \times s + w_{\text{final}} \times f$, where $w_{\text{final}} = w_{\text{init}} + \partial z^d_b$.

Algorithm 1 LIFT-CYCLE(z, (s, f), w_{init}.)

Input:
- relative p-cycle z in $H(KI)$, where I is one of $(b, d]$, $[b, d)$, (b, d), $[b, d]$;
- direction from start to finish, s, f $(= b, d$ or $d, b)$;
- initial boundary $w_{\text{init}} = -\partial z \cap K[s, s] \subseteq K[s, s]$

Output:
- relative p-cycle $\overline{z}$ in $H(\overline{K}I)$

1: **for** *each* $\tau \in z$ **do**
2: $\quad t_\tau \leftarrow$ a value in $T(\tau) \cap [b, d]$ ▷ if $T(\tau) \cap [b, d] = \emptyset$, ignore τ
3: $\quad \overline{z} \leftarrow \overline{z} + \langle \tau, z \rangle \cdot (\tau \times t_\tau)$
4: **for** *each* σ s.t. $\exists \tau \in z, \sigma \in \partial\tau$ **do** ▷ each face of some simplex in z
5: $\quad c \leftarrow \langle \sigma, w_{\text{init}} \rangle$
6: $\quad l \leftarrow s$
7: $\quad$ **for** $(t_\tau, \tau) \in \{(t_\tau, \tau) \mid \tau \in \delta\sigma\}$ in order of t_τ from s to f **do**
8: $\quad\quad$ **if** $l \neq t_\tau$ **then**
9: $\quad\quad\quad \overline{z} \leftarrow \overline{z} + c \cdot (\sigma \times [l, t_\tau])$
10: $\quad\quad c \leftarrow c + \langle \tau, z \rangle \cdot \langle \sigma, \partial\tau \rangle$
11: $\quad\quad l \leftarrow t_\tau$
12: $\quad$ **if** $l \neq f$ **then**
13: $\quad\quad \overline{z} \leftarrow \overline{z} + c \cdot (\sigma \times [l, f])$

Proof. Simplex σ generates a cell $\sigma \times [s, l] \in \overline{z}$ iff $\langle \sigma, w_{\text{init}} \rangle \neq 0$ (Line 5). This cell contributes $\sigma \times s$ to the boundary of $\overline{z}$. Simplex σ generates a cell $\sigma \times [l', f] \in \overline{z}$ iff $\langle \sigma, w_{\text{init}} \rangle + \sum_{\tau \in \delta\sigma \cap z_b^d} \langle \sigma, \tau \rangle \neq 0$. By definition this is the case only for simplices in $w_{\text{final}} = w_{\text{init}} + \partial z_b^d$. ◁

It follows that if $w_{\text{init}} \subseteq K[s, s]$, then $\overline{z}$ is a cycle in $\overline{K}I$. What is not immediate is why the chains $\sigma \times [l, t_\tau]$ and $\sigma \times [l, f]$ in Lines 9 and 13 are in the prism $\overline{K}$.

▷ Claim 6. Chains $\sigma \times [l, t_\tau]$ and $\sigma \times [l, f]$ added in Lines 9 and 13 of Algorithm 1 are present in $\overline{K}$.

Proof. Without loss of generality, assume $s < f$. There are three types of chains:

1. $\sigma \times [s, t_{\tau_1}]$. σ has a non-zero coefficient on this interval iff $\langle \sigma, w_{\text{init}} \rangle \neq 0$ (Line 5), in which case $\sigma \in K[s, s]$. Existence of coface τ_1 implies $\min(\sigma) \leq s \leq t_{\tau_1} < \max(\sigma)$.
2. $\sigma \times [t_{\tau_i}, t_{\tau_{i+1}}]$. From Equation (4), $\min(\sigma) < t_{\tau_i} \leq t_{\tau_{i+1}} < \max(\sigma)$.
3. $\sigma \times [t_{\tau_i}, f]$. Suppose that this chain is not in $\overline{K}$. This means $x = \max(\sigma) \in [t_{\tau_i}, f)$. Equation (4) implies that σ has no cofaces in $[x, f]$. Therefore, $\sigma \in w_{\text{init}} + \partial z_s^x$ and $\sigma \notin \partial z_x^f$. Therefore, $\sigma \in \partial z$. Since we assumed z is a relative cycle in KI, it cannot have any boundary inside the interval itself – a contradiction. ◁

▶ Remark. It may seem strange that we state nothing about the relationship between z and $\overline{z}$. This is because we exploit additional properties of the algorithm in Section 6.

Running time. Let m be the size of the input p-cycle z. Algorithm 1 requires $O(p \cdot m \log m)$ time. It goes through all simplices σ that are faces of some simplex $\tau \in z$; there are at most $(p+1) \cdot m$ such $(p-1)$-simplices. For each one, the algorithm sorts its cofaces by their times t_τ. Although there is no bound on the number of cofaces of one simplex, the total number of

the cofaces to sort is again at most $(p+1)\cdot m$. Individual operations in the update take constant time, so the overall running time of Algorithm 1 is $O(p \cdot m \log m)$. The size of the lifted cycle is $O(p \cdot m)$.

5 Apex Representatives

We give a characterization of apex representatives in the prism.

▶ **Lemma 7** (Closed endpoint).

1. *If a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d])$, contains $\tau \times d$ with $\min \tau = d$, then*

$$[\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}(b,d-1]) \to \mathsf{H}(\overline{K}(b,d])\right).$$

2. *If a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}[b,d))$, contains $\tau \times b$ with $\max \tau = b$, then*

$$[\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}[b+1,d)) \to \mathsf{H}(\overline{K}[b,d))\right).$$

3. *If an absolute cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}[b,d])$, contains $\sigma \times b$ and $\tau \times d$ with $\max \sigma = b$ and $\min \tau = d$, then*

$$[\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}[b+1,d]) \to \mathsf{H}(\overline{K}[b,d])\right) \quad \textit{and} \quad [\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}[b,d-1]) \to \mathsf{H}(\overline{K}[b,d])\right).$$

Proof. All the statements follow the same proof, except for the endpoints of the intervals, so we only show the first. Suppose there is some class $[\alpha] \in \mathsf{H}(\overline{K}(b,d-1])$ that maps to $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d])$. Then $\overline{z} = \alpha + \partial\beta$ for some chain β. Since $\tau \in \overline{z}$ and $\tau \notin \alpha$ (since $\min \tau = d > d-1$), τ must be in $\partial\beta$. But from Equation (4), τ has no cofaces in K^d_{-1} – a contradiction. ◀

▶ **Lemma 8** (Open endpoint).

1. *If the boundary $\partial\overline{z}$ of a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d])$, contains $\sigma \times b$ with $\min \sigma = b$, then $[\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}(b-1,d]) \to \mathsf{H}(\overline{K}(b,d])\right)$.*
2. *If the boundary $\partial\overline{z}$ of a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}[b,d))$ contains $\sigma \times d$ with $\max \sigma = d$, then $[\overline{z}] \notin \operatorname{im}\left(\mathsf{H}(\overline{K}[b,d+1)) \to \mathsf{H}(\overline{K}[b,d))\right)$.*
3. *If the boundary $\partial\overline{z}$ of a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d))$, contains $\sigma \times b$ with $\min \sigma = b$ and $\tau \times d$ with $\max \tau = d$, then*

$$\overline{z} \notin \operatorname{im}\left(\mathsf{H}(\overline{K}(b-1,d)) \to \mathsf{H}(\overline{K}(b,d))\right) \quad \textit{and} \quad \overline{z} \notin \operatorname{im}\left(\mathsf{H}(\overline{K}(b,d+1)) \to \mathsf{H}(\overline{K}(b,d))\right).$$

Proof. All the statements follow the same proof, except for the endpoints of the intervals, so we only show the first. From the long exact sequence of the triple [14, p. 118], $K^{b-1}_{-1} \subseteq K^b_{-1} \subseteq K^d_{-1}$, we know that the image in the claim is equal to the kernel of the map induced by the boundary,

$$\operatorname{im}\left(\mathsf{H}_*(\overline{K}(b-1,d]) \to \mathsf{H}_*(\overline{K}(b,d])\right) = \ker\left(\mathsf{H}_*(\overline{K}(b,d]) \xrightarrow{\partial^*} \mathsf{H}_{*-1}(\overline{K}(b-1,b])\right).$$

For $[\overline{z}]$ to be in the kernel, $\partial\overline{z}$ needs to be a relative boundary in $\overline{K}(b-1,b]$, i.e., $\partial\overline{z} = \partial\beta + \gamma$ for some $\beta \subseteq K^b_{-1}$ and $\gamma \subseteq K^{b-1}_{-1}$. Since $\sigma \in \partial\overline{z}$ and $\sigma \notin \gamma$, σ must be in $\partial\beta$. But from Equation (4), σ has no cofaces in K^b_{-1} – a contradiction. ◀

Putting Lemmas 7 and 8 together, we get the following theorem.

▶ **Theorem 9.**

1. *If a relative cycle $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d])$, contains $\tau \times d$ with $\min \tau = d$, and its boundary $\partial\overline{z}$ contains $\sigma \times b$ with $\min \sigma = b$, then $\overline{z}$ is an apex representative.*

2. *If a relative cycle* $\overline{z}$*, with* $[\overline{z}] \in \mathsf{H}(\overline{K}[b,d))$*, contains* $\tau \times b$ *with* $\max \tau = b$*, and its boundary* $\partial\overline{z}$ *contains* $\sigma \times d$ *with* $\max \sigma = d$*, then* $\overline{z}$ *is an apex representative.*
3. *If an absolute cycle* $\overline{z}$*, with* $[\overline{z}] \in \mathsf{H}(\overline{K}[b,d])$*, contains* $\sigma \times b$ *and* $\tau \times d$ *with* $\max \sigma = b$ *and* $\min \tau = d$*, then* $\overline{z}$ *is an apex representative.*
4. *If the boundary* $\partial\overline{z}$ *of a relative cycle* $\overline{z}$*, with* $[\overline{z}] \in \mathsf{H}(\overline{K}(b,d))$*, contains* $\sigma \times b$ *with* $\min \sigma = b$ *and* $\tau \times d$ *with* $\max \tau = d$*, then* $\overline{z}$ *is an apex representative.*

6 Four Intervals

We assume $R = DV$ is the decomposition of the boundary matrix D of the cone $\hat{K}$ filtration in Equation (7). We use σ, τ to refer to simplices in the base space K, and $\hat{\sigma}, \hat{\tau}$ for the simplices in the cone $\hat{K} - K$.

Summary. The main content of this section is summarized in Table 1: lifting the stated cycles z in K, derived from the lazy reduction, with the given arguments to the algorithm Lift-Cycle, produces apex representatives in $\overline{K}$. The results of the section are more general, but also more verbose; they derive the expression for the cycles for any reduction.

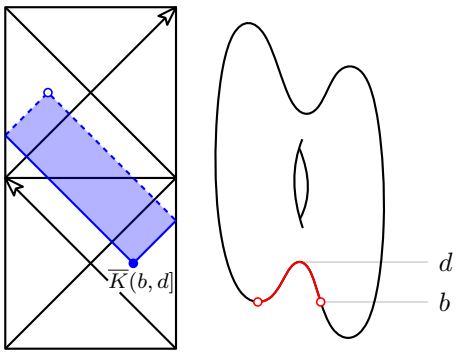
Table 1 Summary of the chains and arguments used in different cases of the Lift-Cycle algorithm, derived from the lazy reduction.

EP type	LZZ type	Apex	z	(s,f)	w_{init}
Ordinary	closed-open	$K(b,d]$	$V[\tau]$	(d,b)	0
Relative	open-closed	$K[b,d)$	$R[\hat{\tau}] \cap K$	(b,d)	0
Extended (d,b)	open-open	$K[b,d]$	$R[\hat{\tau}]$	(d,b)	0
Extended (b,d)	closed-closed	$K(b,d)$	$V[\hat{\tau}] \cap K$	(b,d)	$-R[\hat{\tau}]$

6.1 Ordinary (closed-open)

Throughout this subsection, we assume the following **setting**:

a pair (σ, τ) in the cone filtration, born at b and dying at d, i.e., $\min(\sigma) = b < d = \min(\tau)$.



▷ Claim 10. Let $z = V[\tau]$. Then $[z] \in \mathsf{H}(K(b,d])$. Furthermore, $\tau \in z$ and $\sigma \in \partial z = R[\tau]$.

Proof. Since V is upper-triangular with all diagonal entries non-zero, τ is the latest simplex in $V[\tau]$, i.e., $\max\{\min(\tau') \mid \tau' \in V[\tau]\} = \min(\tau) = d$. Therefore, $V[\tau] \subseteq K^d_{-1}$. By definition, the boundary $\partial V[\tau] = R[\tau]$. σ is its latest simplex, i.e., $\max\{\min(\sigma') \mid \sigma' \in R[\tau]\} = \min(\sigma) = b$. Therefore, $\partial V[\tau] \subseteq K^b_{-1}$. It follows that if $z = V[\tau]$, then $[z] \in \mathsf{H}(K^d_{-1}, K^b_{-1}) = \mathsf{H}(K(b,d])$. ◁

▷ Claim 11. Relative cycle $\overline{z}$ produced by the call Lift-Cycle$(V[\tau], (d,b), 0)$ is an apex representative in $\mathsf{H}(\overline{K}(b,d])$.

Proof. Let $z = V[\tau]$.

1. $\sigma \times b \in \partial\overline{z}$, $\min(\sigma) = b$.
   Because $\sigma \in \partial z$ (Claim 10) and $\min(\sigma) = b$, its coboundary $\delta\sigma \subseteq K^{n+1}_b$. Meanwhile, $z \subseteq K^d_{-1}$. Therefore, all cofaces of σ in z intersect $[b,d]$. Therefore, $\sigma \in \partial(z \cap [b,d])$. It follows from Claim 5 that

   $$\sigma \times b \in \partial\overline{z} = w_{\text{final}} \times b = \partial(z \cap [b,d]) \times b.$$

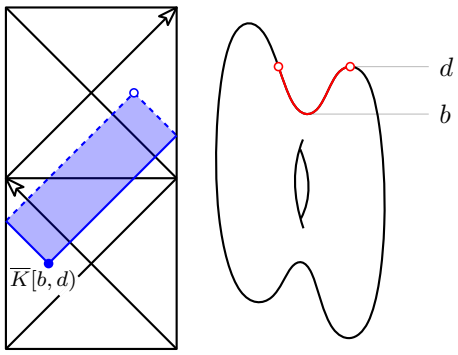
2. $\tau \times d \in \overline{z}$, $\min(\tau) = d$.
   This follows immediately since $\tau \in z$ (Claim 10) and $T(\tau) \cap [b,d] = \{d\}$.

It follows from Theorem 9 that $\overline{z}$ is an apex representative in $\mathsf{H}(\overline{K}(b,d])$. ◁

6.2 Relative (open-closed)

Throughout this subsection, we assume the following **setting**:

a pair $(\hat{\sigma}, \hat{\tau})$ in the cone filtration, born at d and dying at b, i.e., $\max(\sigma) = d > b = \max(\tau)$.



▷ Claim 12. Let chain z consist of the boundary of the cone simplices in $V[\hat{\tau}]$ restricted to the base space,

$$z = \partial\left(V[\hat{\tau}] \cap (\hat{K} - K)\right) \cap K.$$

Then $[z] \in \mathsf{H}(K[b,d))$. Furthermore, $\tau \in z$ and $\sigma \in \partial z$.

Proof. Let $\gamma = V[\hat{\tau}] \cap (\hat{K} - K)$, i.e., $z = (\partial\gamma) \cap K$.

1. $z \subseteq K^{n+1}_b$, $\tau \in z$.
   Because γ consists of only cone simplices, every simplex $\tau' \in z$ is in the boundary of some cone simplex $\hat{\tau}'$. Because $\hat{\tau}$ is the latest simplex in $V[\hat{\tau}]$, we have that $\hat{\tau}'$ is added before $\hat{\tau}$. Therefore, $\max(\tau') \geq \max(\tau) = b$. Because $\hat{\tau}$ is the only simplex in γ with τ in its boundary, $\tau \in \partial\gamma \cap K = z$.

2. $\partial z \subseteq K_d^{n+1}$.
   Let
   $$\alpha = \partial z = -\partial\left(\partial\gamma \cap (\hat{K} - K)\right)$$
   be the shared boundary between the base space and the cone parts of $\partial\gamma$. We note that because cone simplices can only be in the boundary of cone simplices, $(\partial V[\hat{\tau}]) \cap (\hat{K} - K) = \partial\gamma \cap (\hat{K} - K)$. Therefore, all the cone simplices $\hat{\sigma}' \in \partial\gamma \cap (\hat{K} - K))$ are such that $\max(\sigma') \geq \max(\sigma) = d$. Therefore, any simplex in the shared boundary α is in K_d^{n+1}. Putting the first two sub-claims together, $[z] \in \mathsf{H}(K_b^{n+1}, K_d^{n+1}) = \mathsf{H}(K[b,d))$.
3. $\sigma \in \partial z$.
   $(\hat{\sigma}, \hat{\tau})$ is a persistence pair, therefore, $\hat{\sigma} \in \partial V[\hat{\tau}]$. Because cone simplices can only be in the boundaries of cone simplices, $\hat{\sigma} \in \partial\gamma$. Because $\hat{\sigma}$ is the only simplex in $(\partial\gamma) \cap (\hat{K} - K)$ that has σ in its boundary, it follows that
   $$\sigma \in \partial z = \partial\left((\partial\gamma) \cap K\right) = \partial\left((\partial\gamma) \cap (\hat{K} - K)\right). \qquad \triangleleft$$

▶ Remark 13 (Lazy reduction). If $R = DV$ is obtained by lazy reduction, it follows from Corollary 3 that $V[\hat{\tau}]$ consists only of cone simplices. In other words, the chain in the prior claim simplifies to

$$z = (\partial V[\hat{\tau}]) \cap K = R[\hat{\tau}] \cap K.$$

▷ Claim 14. Relative cycle $\overline{z}$ produced by the call Lift-Cycle$(R[\hat{\tau}] \cap K, (b,d), 0)$ is an apex representative in $\mathsf{H}(\overline{K}[b,d))$.

Proof. Let $z = R[\hat{\tau}] \cap K$.

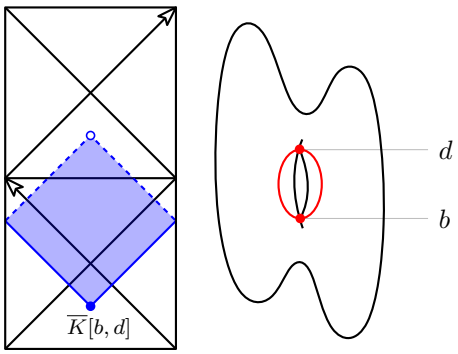
1. $\sigma \times d \in \partial\overline{z}$, $\max(\sigma) = d$.
   Because $\sigma \in \partial z$ (Claim 12) and $\max(\sigma) = d$, its coboundary $\delta\sigma \subseteq K_{-1}^d$. Meanwhile, $z \subseteq K_b^{n+1}$. Therefore, all cofaces of σ in z intersect $[b,d]$. Therefore, $\sigma \in \partial\left(z \cap [b,d]\right)$. It follows from Claim 5 that
   $$\sigma \times d \in \partial\overline{z} = w_{\text{final}} \times d = \partial\left(z \cap [b,d]\right) \times d.$$
2. $\tau \times b \in \overline{z}$, $\max(\tau) = b$.
   This follows immediately since $\tau \in z$ (Claim 12) and $T(\tau) \cap [b,d] = \{b\}$.

It follows from Theorem 9 that $\overline{z}$ is an apex representative in $\mathsf{H}(\overline{K}[b,d))$. $\triangleleft$

6.3 Extended (open-open)

Throughout this subsection, we assume the following **setting**:

a pair $(\sigma, \hat{\tau})$ in the cone filtration born at d, dying at b, i.e., $\min(\sigma) = d > b = \max(\tau)$.



▷ Claim 15. Let chain c consist of those simplices in $V[\hat{\tau}]$ that are either in the cone, or in K_b^{n+1}, i.e.,

$$c = V[\hat{\tau}] \cap (\hat{K} - K) + V[\hat{\tau}] \cap K_b^{n+1}.$$

Let $z = \partial c$; then $[z] \in \mathsf{H}(K[b,d])$. Furthermore, $\sigma, \tau \in z$.

Proof. We split the chain $V[\hat{\tau}]$ into three parts:

$$\alpha = V[\hat{\tau}] \cap (K - K_b^{n+1}), \qquad \beta = V[\hat{\tau}] \cap K_b^{n+1}, \qquad \gamma = V[\hat{\tau}] \cap (\hat{K} - K),$$

and re-write its boundary accordingly: $R[\hat{\tau}] = \partial V[\hat{\tau}] = \partial\alpha + \partial\beta + \partial\gamma$. $c = (\beta + \gamma) = V[\hat{\tau}] - \alpha$.

1. $z = \partial c \subseteq K_{-1}^d$, $\sigma \in z$.
   $\partial V[\hat{\tau}] \subseteq K_{-1}^d$ because σ is the latest simplex in $R[\hat{\tau}]$. Putting together Equations (5) and (6), $\alpha \subseteq K - K_b^{n+1}$ implies $\partial\alpha \subseteq K_{-1}^b \subseteq K_{-1}^d$. Therefore, $z = \partial V[\hat{\tau}] - \partial\alpha \subseteq K_{-1}^d$.
   Because $\sigma \in \partial V[\hat{\tau}]$, but $\sigma \notin \partial\alpha$, $\sigma \in z$.
2. $z \subseteq K_b^{n+1}$, $\tau \in z$.
   Since chain $\beta \subseteq K_b^{n+1}$, its boundary $\partial\beta \subseteq K_b^{n+1}$. Since boundaries $\partial V[\hat{\tau}], \partial\alpha, \partial\beta \subseteq K$, so is $\partial\gamma$. Every simplex in γ has the form $\omega * \tau$, where $\tau \in K_b^{n+1}$. Therefore, $\partial\gamma \subseteq K_b^{n+1}$. It follows that $z = \partial\beta + \partial\gamma \subseteq K_b^{n+1}$.
   Because $\max(\tau) = b$, it has no cofaces in K_b^{n+1}. Therefore, $\tau \notin \partial\beta$. Because $\hat{\tau}$ is the only simplex in γ that has τ in its boundary, $\tau \in \partial\gamma$. Therefore, $\tau \in z = \partial\beta + \partial\gamma$.

   Therefore, $[z] \in \mathsf{H}(K_b^{n+1} \cap K_{-1}^d) = \mathsf{H}(K[b,d])$. ◁

▶ Remark 16 (Lazy reduction). Furthermore, if $R = DV$ comes from the lazy reduction, then the next claim proves that in the previous claim, chain $c = V[\hat{\tau}]$. In other words, we can use $z = R[\hat{\tau}] = \partial V[\hat{\tau}]$ directly.

▷ Claim 17 (Lazy reduction). If $R = DV$ is obtained from the lazy reduction, then all the base space simplices in $V[\hat{\tau}]$ are in K_b^{n+1}, i.e., $V[\hat{\tau}] \cap K \subseteq K_b^{n+1}$.

Proof. Let τ' be a simplex in $V[\hat{\tau}] \cap K$. In this case, for the lazy reduction, because $V[\tau', \hat{\tau}] \neq 0$, from Corollary 3, taking $\sigma_j = \sigma$ and $\tau_i = \tau'$, we get

$$\max(\tau') > \min(\tau') > \min(\sigma) = d > b.$$ ◁

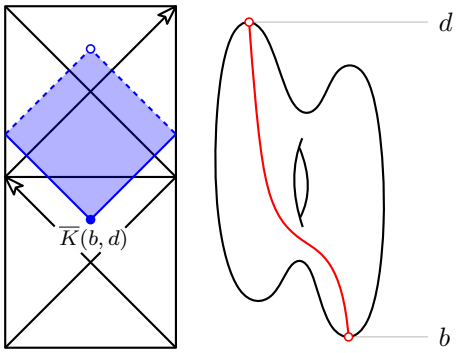
▷ Claim 18. Absolute cycle $\overline{z}$ produced by the call Lift-Cycle$(R[\hat{\tau}], (d,b), 0)$ is an apex representative in $\mathsf{H}(\overline{K}[b,d])$.

Proof. Let $z = R[\hat{\tau}]$. Because $\sigma, \tau \in z$ (Claim 15) and $\min(\sigma) = d$ and $\max(\tau) = b$, both simplices $\sigma \times d$ and $\tau \times b$ are in the lifted cycle $\overline{z}$. It follows from Theorem 9 that $\overline{z}$ is an apex representative in $\mathsf{H}(\overline{K}[b,d])$. ◁

6.4 Extended (closed-closed)

Throughout this subsection, we assume the following **setting**:

a pair $(\sigma, \hat{\tau})$ in the cone filtration, born at b, dying at d, i.e., $\min(\sigma) = b < d = \max(\tau)$.



▷ Claim 19. Let $z = V[\hat{\tau}] \cap K_b^{n+1}$. Then $[z] \in \mathsf{H}(K(b,d))$. Furthermore,

$$\sigma \in w_{\text{init}} = -\partial\left(V[\hat{\tau}] \cap (\hat{K} - K) + V[\hat{\tau}] \cap K_b^{n+1}\right) \subseteq K[b,b]$$

and $\tau \in (w_{\text{init}} + \partial z) = -\partial\left(V[\hat{\tau}] \cap (\hat{K} - K)\right)$.

Proof. We split the chain $V[\hat{\tau}]$ into three parts:

$$\alpha = V[\hat{\tau}] \cap (K - K_b^{n+1}), \qquad z = \beta = V[\hat{\tau}] \cap K_b^{n+1}, \qquad \gamma = V[\hat{\tau}] \cap (\hat{K} - K),$$

and re-write its boundary accordingly: $R[\hat{\tau}] = \partial V[\hat{\tau}] = \partial\alpha + \partial\beta + \partial\gamma$.

1. $\partial z \subseteq K_{-1}^b \cup K_d^{n+1}$.
   Putting together Equations (5) and (6), $\alpha \subseteq K - K_b^{n+1}$ implies $\partial\alpha \subseteq K_{-1}^b$. Because σ is the latest simplex in $R[\hat{\tau}]$ and $\min(\sigma) = b$, $R[\hat{\tau}] \subseteq K_{-1}^b$. It follows that $\partial(\beta+\gamma) = R[\hat{\tau}] - \partial\alpha \subseteq K_{-1}^b$. Since $\partial\gamma \subseteq K_d^{n+1}$, we have $\partial z = \partial(\beta+\gamma) - \partial\gamma \subseteq K_{-1}^b \cup K_d^{n+1}$. Therefore, $[z] \in \mathsf{H}(K(b,d))$.
2. $\sigma \in w_{\text{init}} = \partial(\beta+\gamma) \subseteq K[b,b]$.
   We already saw that $\partial(\beta+\gamma) \subseteq K_{-1}^b$. To show that it is also in K_b^{n+1}, we note that since $\beta \subseteq K_b^{n+1}$, its boundary $\partial\beta \subseteq K_b^{n+1}$. $\partial\gamma \subseteq K_d^{n+1} \subseteq K_b^{n+1}$. Therefore, $\partial(\beta+\gamma) \subseteq K_b^{n+1}$. Because $\min(\sigma) = b$, Equation (4) implies that σ cannot be in the boundary of any simplex $\tau \in \alpha$ since $\max(\tau) < b$. Since $\sigma \in R[\hat{\tau}]$ and $w_{\text{init}} = \partial(\beta+\gamma) = R[\hat{\tau}] - \partial\alpha$, it follows that $\sigma \in w_{\text{init}}$.
3. $\tau \in -\partial\gamma$.
   Because $\hat{\tau}$ is the only simplex in γ that has τ in its boundary, $\tau \in -\partial\gamma$. ◁

▷ Claim 20 (Lazy reduction). If $R = DV$ is obtained from the lazy reduction, then $z = V[\hat{\tau}] \cap K$ is in $K(b,d)$, $\sigma \in w_{\text{init}} = -R[\hat{\tau}] = -\partial V[\hat{\tau}] \subseteq K[b,b]$, and $\tau \in (w_{\text{init}} - \partial z)$.

Proof. We only need to show that, in the case of the lazy reduction, $\alpha = V[\hat{\tau}] \cap (K - K_b^{n+1}) = 0$, i.e., $V[\hat{\tau}] \cap K = V[\hat{\tau}] \cap K_b^{n+1}$. Then the claim follows from the previous Claim 19.

The proof is analogous to the proof of Claim 17. Let τ' be a simplex in $V[\hat{\tau}] \cap K$. In this case, for the lazy reduction, because $V[\tau', \hat{\tau}] \neq 0$, from Corollary 3, taking $\sigma_j = \sigma$ and $\tau_i = \tau'$, we get $\max(\tau') > \min(\tau') > \min(\sigma) = b$. ◁

▶ Remark 21. $V[\hat{\tau}] \cap K$ can be empty. For example, if K is a single vertex that appears at b and then disappears at d. This is the reason why we need to bootstrap Algorithm 1 with an initial cycle w_{init}.

▷ Claim 22. Relative cycle $\overline{z}$ produced by the call Lift-Cycle$(V[\hat{\tau}] \cap K, (b,d), -R[\hat{\tau}])$ is an apex representative in $\mathsf{H}(\overline{K}(b,d))$.

Proof.

1. $\sigma \times b \in \partial\overline{z}$, $\min(\sigma) = b$.
   Simplex σ, with $\min(\sigma) = b$, is in the initial cycle w_{init} (Claim 19). Therefore, $\sigma \times b$ is in the boundry $\partial\overline{z}$ of the lifted cycle.
2. $\tau \times d \in \partial\overline{z}$, $\max(\tau) = d$.
   Simplex τ, with $\max(\tau) = d$, is in $w_{\text{init}} + \partial z$ (Claim 19). Its coboundary $\delta\tau \subseteq K^d_{-1}$. Meanwhile, $z \subseteq K^{n+1}_b$ (Claim 19). Therefore, all cofaces of τ in z intersect $[b, d]$. Therefore, $\tau \in w_{\text{init}} + \partial\left(z \cap [b, d]\right) = w_{\text{final}}$. It follows from Claim 5 that

$$\tau \times d \in w_{\text{final}} \times d \subseteq \partial\overline{z}.$$

It follows from Theorem 9 that $\overline{z}$ is an apex representative in $\mathsf{H}(\overline{K}(b, d))$. ◁

7 Zigzag Representatives

To solve our original problem – to find zigzag representatives – all that remains is to map an apex representative into an appropriate space in the zigzag. Because we assumed in Remark 4 that no simplex is supported on a single space, the death and the birth for any given interval are distinct. Suppose we want to map an apex representative $\overline{z}$, with $[\overline{z}] \in \mathsf{H}(\overline{K}I)$, into a zigzag representative in $\mathsf{H}(\overline{K}[i, i]) \simeq \mathsf{H}(K_i)$.

If i is odd, then the p-cycle $\overline{z}$ may contain a vertical cell $\tau \times i$, which does not map into $(p-1)$-cells in the levelset. To sidestep this complication, we perturb i. Let $x = i \pm \varepsilon \in I$ be a real value near i that lies inside our birth-death interval. We assume $x = i + \varepsilon$; the other case is symmetric. We implicitly subdivide $\overline{K}$ at x and extend our previous notation to allow for the interlevel sets ending at x, e.g., $\overline{K}^x_i$. We note that $\overline{K}^x_i$ deformation retracts onto $\overline{K}^i_i$, with the homotopy following the second (time) coordinate, and $\overline{K}^x_x$ includes into $\overline{K}^i_i$ once the cell times are shifted from x to i. We want to compose three maps:

$$\mathsf{H}_p(\overline{K}I) \to \mathsf{H}_p(\overline{K}(i, x)) \xrightarrow{\partial^*} \mathsf{H}_{p-1}(\overline{K}[x, x]) \to \mathsf{H}_{p-1}(K_i).$$

Because $\overline{K}I$ is an apex of a diamond, and $\overline{K}(i, x)$ is a space in the diamond, the first map is an inclusion (of pairs), so $\overline{z}$ remains a relative cycle in $\overline{K}(i, x)$. Let

$$\overline{z} = \sum \alpha_\tau \cdot (\tau \times t_\tau) + \sum \alpha_\sigma \cdot (\sigma \times [t^1_\sigma, t^2_\sigma]),$$

then the second (boundary) map takes $\overline{z}$ to

$$\overline{z}[x, x] = \sum_{\substack{(\sigma \times [t^1_\sigma, t^2_\sigma]) \in \overline{z} \\ x \in [t^1_\sigma, t^2_\sigma]}} \alpha_\sigma \cdot (\sigma \times x),$$

which maps into K_i by dropping the second coordinate,

$$z(i) = \sum_{(\sigma \times x) \in \overline{z}[x,x]} \alpha_\sigma \cdot \sigma.$$

By storing the apex cycle $\overline{z}$ in an interval tree [11, 15], we can retrieve any zigzag representative of size C in time $O(\log m + C)$.

References

1. Ulrich Bauer, Magnus Bakke Botnan, and Benedikt Fluhr. Structure and interleavings of relative interlevel set cohomology. *arXiv [math.AT]*, August 2021. arXiv:2108.09298.
2. Paul Bendich, Herbert Edelsbrunner, Dmitriy Morozov, and Amit Patel. Homology and robustness of level and interlevel sets. *Homology, Homotopy and Applications*, 15(1):51–72, 2013.
3. Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, August 2010. doi:10.1007/s10208-010-9066-0.
4. Gunnar Carlsson, Vin de Silva, Sara Kališnik, and Dmitriy Morozov. Parametrized homology via zigzag persistence. *Algebraic & Geometric Topology*, 19(2):657–700, March 2019. doi:10.2140/agt.2019.19.657.
5. Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 247–256, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542408.
6. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of computational mathematics*, 9(1):79–103, February 2009. doi:10.1007/s10208-008-9027-z.
7. Tamal K. Dey and Tao Hou. Fast Computation of Zigzag Persistence. 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 43:1–43:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.43.
8. Tamal K Dey, Tao Hou, and Dmitriy Morozov. Apex representatives. *arXiv [cs.CG]*, February 2025. arXiv:2502.17704.
9. Tamal K. Dey, Tao Hou, and Dmitriy Morozov. A fast algorithm for computing zigzag representatives. 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 3530–3546. SIAM, 2025. doi:10.1137/1.9781611978322.116.
10. Herber 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.
11. Herbert Edelsbrunner. Dynamic data structures for orthogonal intersection queries. Technical Report F59, Inst. Information Process., Graz University of Technology, Austria, 1980. URL: https://pub.ista.ac.at/~edels/Papers/1980-01-OrthogonalIntersectionQueries.pdf.
12. Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010.
13. Herbert Edelsbrunner and Dmitriy Morozov. Persistent homology. In Jacob E Goodman, Joseph O'Rourke, and Csaba D Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, 2017. URL: https://www.csun.edu/~ctoth/Handbook/chap24.pdf.
14. Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.
15. Edward M. McCreight. Efficient algorithms for enumerating intersecting intervals and rectangles. Technical Report CSL-80-9, Xeox Palo Alto Research Center, 1980.
16. Dmitriy Morozov and Primoz Skraba. Persistent (co)homology in matrix multiplication time. *arXiv [math.AT]*, December 2024. doi:10.48550/arXiv.2412.02591.
17. Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *Discrete & computational geometry*, 72(1):310–344, July 2024. doi:10.1007/s00454-023-00613-x.

Decomposing Multiparameter Persistence Modules

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Jan Jendrysiak
Graz University of Technology, Austria

Michael Kerber
Graz University of Technology, Austria

Abstract

Dey and Xin (J.Appl.Comput.Top., 2022) describe an algorithm to decompose finitely presented multiparameter persistence modules using a matrix reduction algorithm. Their algorithm only works for modules whose generators and relations are distinctly graded. We extend their approach to work on *all* finitely presented modules and introduce several improvements that lead to significant speed-ups in practice. Our algorithm is fixed-parameter tractable with respect to the maximal number of relations of the same degree and with further optimisation we obtain an $O(n^3)$ time algorithm for interval-decomposable modules. In particular, we can decide interval-decomposability in this time. As a by-product to the proofs of correctness we develop a theory of parameter restriction for persistence modules. Our algorithm is implemented as a software library AIDA, the first to enable the decomposition of large inputs. We show its capabilities via extensive experimental evaluation.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Topological Data Analysis, Multiparameter Persistence Modules, Persistence, Decomposition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.41

Related Version *Full Version*: `http://arxiv.org/abs/2504.08119` [26]

Supplementary Material *Software (Algorithm)*: `https://github.com/JanJend/AIDA` [38]
archived at `swh:1:dir:c339cb432335e83758b23a807641b9c6a43525c8`
Software (Library): `https://github.com/JanJend/Persistence-Algebra` [39]
archived at `swh:1:dir:e41d23c6d743921e03d8a54e45ea3a4d36f7718b`

Funding *Tamal K. Dey*: Supported by NSF grants NSF grants CCF-2437030 and DMS-2301360.
Jan Jendrysiak: Supported by Austrian Science Fund (FWF) grant P 33765-N and W1230.
Michael Kerber: Supported by the Austrian Science Fund (FWF) grant P 33765-N.

Acknowledgements The second author wants to thank Klaus Lux for his insights and discussions about the Meat-Axe algorithm and Algorithmic Representation Theory, as well as Martin Kalck for pointers to relevant established theory.

1 Introduction

Multiparameter Persistence Modules (MPM) as introduced by Carlsson and Zomorodian [21, 22] make it possible to capture topological features of data with respect to more than one parameter. This, for example, allows detection of features in different scale-regions stably with respect to outliers [13]. One of the main challenges of dealing with MPM is that their indecomposable summands do not provide a discrete invariant, as in the case of finitely generated 1-parameter persistence modules, where this decomposition leads to the *barcode*.

In contrast, indecomposable MPM can be arbitrarily complex because the representation theory of the partially ordered set $\mathbb{N}^d$ is wild ([45, 49]). In light of this difficulty, researchers in TDA have come up with various invariants with varying degree of discrimination powers and ease of computability ([16, 24, 27, 32, 48, 51, 52, 58, 60]). Nevertheless, the begging question remains how to compute the indecomposable summands of MPM.

© Tamal K. Dey, Jan Jendrysiak, and Michael Kerber;
licensed under Creative Commons License CC-BY 4.0

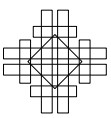


Figure 1 Philippe Chaperon's set design for Giuseppe Verdi's opera Aida, 1880.

A fast algorithm for this task has numerous benefits. Primarily, every additive invariant of persistence module can, by definition, be computed from its indecomposable summands, possibly reducing computation time. For example, the computation of the matching distance [10, 12] becomes more tractable for decomposed modules as the complexity of the barcode template [46] reduces, which we show empirically in Section 7.

Contributions. We present an algorithm inspired by the Dey-Xin algorithm [29], speeding up various steps and lifting the assumption that the module needs to be distinctly graded. Our methods are collected in the software package AIDA (Figure 1), which allows the decomposition of presentations arising from large data within seconds. We demonstrate the usefulness of our improvements in extensive experimental evaluation. Since the methods used in AIDA are of more general interest we implemented them in a stand-alone, header-only C++library PERSISTENCE-ALGEBRA. Both this library and the AIDA code are publicly available.[1]

While the distinctly graded case allows for an incremental column-by-column approach to compute a decompositon, the general case requires to consider all columns of the same degree at once, say k_i many at the i-th step. It does not suffice to consider all permutations of these columns [5]; instead arbitrary changes-of-base within the columns could be part of a correct solution. Our first approach requires a brute-force iteration over these change-of-basis matrices (which is only possible over a finite field) and is therefore not polynomial in k_i. Nevertheless, through a careful analysis, we limit the number of cases and provide a tractable solution if k_i is small. Since we can then use the poly-time strategy of [29], this yields a decomposition algorithm which is fixed-parameter tractable with respect to $k_{\max} := \max_i k_i$.

[1] `https://github.com/JanJend/AIDA` and `https://github.com/JanJend/Persistence-Algebra`

To further reduce the iteration – or even avoid it altogether – we analyse which kinds of matrix operations can actually help in the decomposition. This leads to another FPT-algorithm in a parameter κ_i smaller than k_i which promises to accelerate the computation for modules where k_i is too large. A modification of this idea also yields an algorithm which, for interval-decomposable modules, runs in $\mathcal{O}(n^3)$ time, independent of κ_i.

Related work. Multi-parameter persistent homology is an active research area with many theoretic, computational, and applied results in recent years [15]. In particular, efficient algorithms have been devised for the computation of a presentation of standard bifiltrations [4, 2, 17, 18, 25, 44], for minimizing presentations [29, 33, 47] and for efficient distance computation [10, 12, 28]. Our decomposition algorithm complements this pipeline.

In theory we can decompose any module over a finite dimensional algebra in polynomial time by decomposing its endomorphism algebra [23], but this algorithm is not practical. The algorithms in the C MeatAxe Package ([50], [57]), in the MAGMA system [14], and in the GAP system [34], [54], rely on the Holt-Rees extension [37] of the MeatAxe algorithm [53], both of which are *Las Vegas* algorithms and too slow for the large sparse matrices appearing in TDA. We remark that that the run-time complexity of these algorithms is sometimes not precisely cited in the TDA literature. For example the bounds in [15, Section 8.5] and [29, Introduction]) are a misinterpretation of the usage of the MeatAxe algorithm in this pipeline and use the *lower* bound given in [36] for the non-practical algorithm described in [55].

The only algorithm specifically designed for MPM is the aforementioned Dey-Xin algorithm [29] which has a time complexity of $\mathcal{O}(n^{2\omega+1})$, where $\omega < 2.373$ is the matrix multiplication constant. It only works for uniquely graded modules, a property that is satisfied by some, but not all modules that appear in practice. Moreover, an algorithm to compute graphcodes [43] yields a partial decomposition

It is well known that decompositions are not stable; every persistence module is even arbitrarily close to an indecomposable one in interleaving distance [9], but geometric bifiltrations typically do not create such pathologies. Moreover, Bjerkevik [11, Thm 5.4] constructs a module which captures *all* decompositions of an entire ϵ-interleaving-neighbourhood. His construction creates many generators and relations of the same degree so its decomposition requires an efficient algorithm we provide with AIDA.

Complicated indecomposables can arise already in relatively simple geometric examples [20, 19] and most modules in practice are not interval-decomposable (e.g., most examples from our benchmark set contain non-interval indecomposables; see also [3]). However, we expect that in practice most modules will mainly consist of small or structurally simple indecomposables (see [1]), and we designed our algorithm to work fast in such scenarios.

Outline. We review persistence modules and their presentations in Section 2. The algorithm by Dey and Xin [29], which is the basis of our work, is reviewed and modified in Sections 3 and 4. In Section 5, we describe how to decompose non-distinctly graded modules with an exhaustive approach. In Section 6, we present an algorithm that reduces the potentially restrictive iteration of the exhaustive algorithm. The detailed descriptions of our algorithms and proofs of correctness are deferred to the extended arXiv version [26].

2 Persistence Modules and their Presentations

Consider a simplicial complex S, discretely filtered with respect to d many parameters. That is for each $\alpha \in \mathbb{N}^d$ there is a subcomplex $S_\alpha \subset S$ and for each pair $\alpha \leq \beta$ there is an inclusion $S_\alpha \subset S_\beta$. Applying the simplicial homology functor $H_*(-;\mathbb{K})$, for a field $\mathbb{K}$ results in a commutative diagram of $\mathbb{K}$-vector spaces over $\mathbb{N}^d$. For $d = 2$, it has the following shape:

$$\begin{array}{ccccccc}
\vdots & & \vdots & & \vdots & & \\
\uparrow & & \uparrow & & \uparrow & & \\
H_*(S_{0,1}\,;\mathbb{K}) & \longrightarrow & H_*(S_{1,1}\,;\mathbb{K}) & \longrightarrow & H_*(S_{2,1}\,;\mathbb{K}) & \longrightarrow & \cdots \\
\uparrow & & \uparrow & & \uparrow & & \\
H_*(S_{0,0}\,;\mathbb{K}) & \longrightarrow & H_*(S_{1,0}\,;\mathbb{K}) & \longrightarrow & H_*(S_{2,0}\,;\mathbb{K}) & \longrightarrow & \cdots
\end{array}$$

The maps encode how classes in $H_*(S_\alpha, \mathbb{K})$ *persist* when parameters are increased.

▶ **Definition 1.** *A d-parameter* persistence module X *over $\mathbb{K}$ is a commutative diagram of $\mathbb{K}$-vector spaces over $\mathbb{N}^d$.*

Presentations. A persistence module can be succinctly represented by a *presentation*. This is a set of generators and a set of relations among these generators. If these sets are finite, they form a *presentation matrix*, where the row indices correspond to the generators, the column indices to the relations and an entry at (i, j) tells us which scalar coefficient in $\mathbb{K}$ the i-th generator appears with in the j-th relation.

▶ **Definition 2.** *A* graded matrix *is a matrix $M \in \mathbb{K}^{m\times n}$ together with functions $G\colon [m] \to \mathbb{N}^d$ and $R\colon [n] \to \mathbb{N}^d$ which decorate the rows and columns with degrees such that*

$$\forall (i,j) \in [m] \times [n] : \quad \text{If } R(j) \not\geq G(i) \text{ then } M_{i,j} = 0.$$

Given two functions G, R we write $\mathbb{K}^{G\times R}$ for the corresponding space of graded matrices.

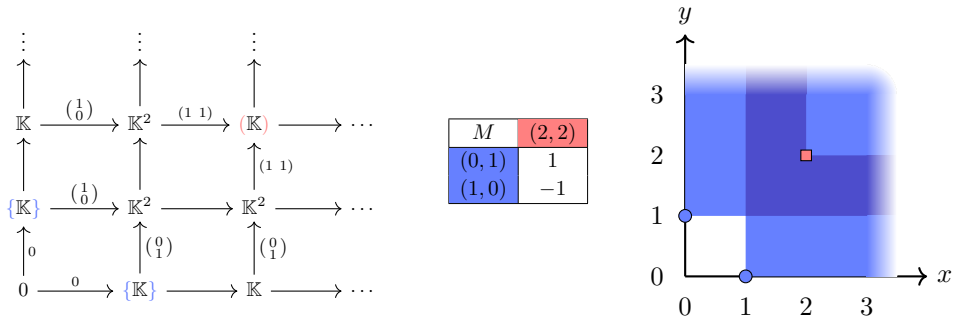


Figure 2 The matrix in the middle presents the module on the left. The location of generators are indicated by blue brackets, the relation by a red bracket. The picture on the right indicates the dimensions of the vector spaces over a continuous plane.

▶ **Definition 3.** *A finite presentation is called* minimal *if there is no presentation of the same module with less generators or less relations.*

▶ Remark 4. Every finitely generated (fg) persistence module has a finite presentation (Hilbert's syzygy theorem [35]). Furthermore, there are efficient algorithms to compute minimal presentations of persistence modules from simplicial bi-filtrations ([8, 42, 47]) and Schreyer's algorithm ([31]) for the general case.

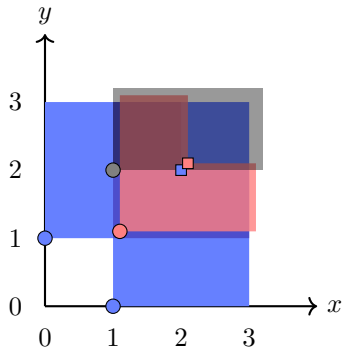
▶ **Theorem 5** (Krull-Remak-Schmidt-Azumaya [6]). *Every fg persistence module X has a finite decomposition into indecomposable submodules, $X = \sum_{i \in I} X_i$, and the isomorphism types of the submodules are independent of the decomposition.*

Decompositions of modules correspond to decompositions of their presentations, but this correspondence is only surjective, see [26, Example 2.30] for details.

▶ **Definition 6.** *Let $M \in \mathbb{K}^{m \times n}$ be a matrix. A* block *b is a pair (b_{rows}, b_{cols}) where $b_{rows} \subset [m]$, $b_{cols} \subset [n]$ and we write M_b for the corresponding submatrix. M is called (block-)* decomposed *if there is a set of blocks $\mathcal{B}$ which partition the row- and column-indices such that M is zero outside of the blocks. We write $M_{\mathcal{B}}$ for a matrix which is block-decomposed and if the matrix is graded, then G_b, R_b denote the corresponding degrees of b.*

▶ **Example 7.** A matrix decomposed over $(b = (\{1,2\},\{1\}),\ c = (\{3\},\{2\}),\ d = (\{4\},\{\}))$

M	$(2,2)$	$(2,2)$
$(0,1)$	1	0
$(1,0)$	1	0
$(1,1)$	0	1
$(1,2)$	0	0



▶ **Definition 8** ([29, Section 4.1]). *For $M \in \mathbb{K}^{G \times R}$, an elementary row- (column-) operation from index i to index j is called* admissible *if $G(i) \geq G(j)$ ($R(i) \leq R(j)$).*

Admissible operations preserve the isomorphism type of the presented module.

▶ **Proposition 9** ([26, Sec.2.1]). *If M, $M' \in \mathbb{K}^{G \times R}$ minimally present isomorphic modules, then there is a sequence of admissible operations transforming M into M'. Equivalently, there are invertible graded matrices $Q \in \mathbb{K}^{G \times G}$, $P \in \mathbb{K}^{R \times R}$ such that $M' = QMP$.*

In particular any decomposition of a module lifts to a change-of-basis between a presentation matrix and a block-decomposed presentation matrix.

▶ Remark 10. Proposition 9 points to the main difficulty in writing a matrix-elimination type algorithm. Assume that we want to bring a presentation M into a certain shape, i.e. with zeroes in a block b. This is equivalent to finding invertible graded matrices Q, P such that $(QMP)_b = 0$. A priori this is a *quadratic* system of equations in the entries of Q and P.

3 The Generalized Persistence Algorithm

The Generalized Persistence (GP) Algorithm introduced by Dey and Xin [29] finds an indecomposable decomposition of a presentation matrix $M \in \mathbb{K}^{G \times R}$ assuming that all relations have pairwise distinct degrees. We review the core functionality of their algorithm.

The GP-Algorithm loops over the relations in an order compatible with the product order on $\mathbb{N}^d$, similarly to the Persistence Algorithm for 1-parameter persistence modules [30, 7]. It manipulates the matrix via admissable row and column operations and maintains the invariant that after having handled the i-th column, the matrix consisting of the first i columns is block-decomposed into indecomposables. We must therefore decompose a graded matrix that is already decomposed except for its last column. This is reflected in the next definition; note that the GP algorithm assumes N to consist of a single column, but we already define the structure for N of arbitrary size to prepare our extension of the algorithm.

▶ **Definition 11.** *Let $\alpha \in \mathbb{N}^d$ and $[M_{\mathcal{B}}\ N] \in \mathbb{K}^{G\times(R\cup k\cdot\alpha)}$ be a presentation, where N contains k columns, all of degree α. It is called* α-decomposed, *if it is* minimal, *$M_{\mathcal{B}}$ is an indecomposable decomposition, and the degree of no column of $M_{\mathcal{B}}$ is larger than α. For $b \in \mathcal{B}$ we denote by N_b the submatrix $N_{b_{rows}\times k\cdot\alpha}$ of N with the row indices of b.*

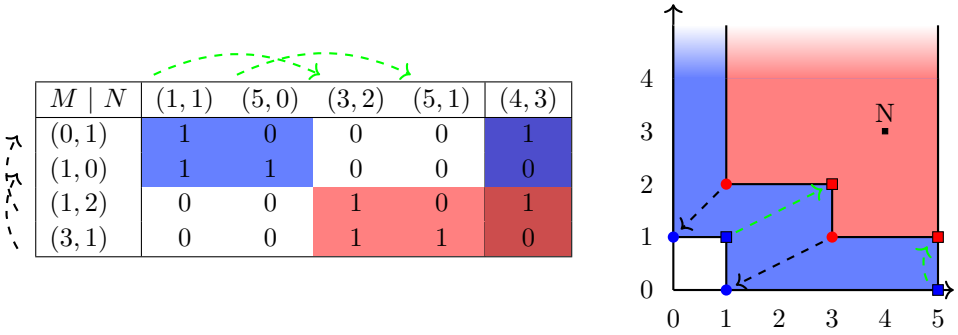


Figure 3 A (4,3)-decomposed presentation. There are two blocks in $M_{\mathcal{B}}$ and they present the interval modules on the right. Some admissible operations have been indicated by arrows both in the matrix and between the corresponding generators and relations.

`BlockReduce`. If we can decompose α-decomposed presentations, then by iterating over the relations we can decompose all finite presentations. If N has only one column this means we have to zero out every sub-column N_b of N whenever possible to reach the block-decomposition with the most blocks. This is done by a subroutine which Dey and Xin called `BlockReduce`, a name we will keep.

▶ **Example 12.** We illustrate `BlockReduce` on the example given in Figure 3. Let us call the (blue) upper block b and the lower (red) block c. We will use two types of admissible operations to try to zero out N_b:

(a) Column operations from M_b to N, e.g. adding the first column to the fifth column.
(b) Row operations from M_c to M_b, e.g. adding the fourth row to the second row.

Adding the third row to the first would delete all entries in N_b, but this would also destroy the block-decomposition of $M_{\mathcal{B}}$, placing a 1 at position $M_{1,3}$:

$M \mid N$	(1,1)	(5,0)	(3,2)	(5,1)	(4,3)
(0,1)	1	0	1	0	0
(1,0)	1	1	0	0	0
(1,2)	0	0	1	0	1
(3,1)	0	0	1	1	0

$M \mid N$	(1,1)	(5,0)	(3,2)	(5,1)	(4,3)
(0,1)	1	0	1	0	0
(1,0)	1	1	1	1	0
(1,2)	0	0	1	0	1
(3,1)	0	0	1	1	0

However, if we also add the fourth to the second row and then use the two column operations indicated by the 2 (green) arrows (right side), we can revert the alteration of $M_{\mathcal{B}}$.
We thus need a third type of admissible operations: **(c)** Column operations from b to c.

The operations of type (a), (b), and (c) have a pleasant feature: their effect on the matrix does not change if the order of their application changes. Dey-Xin call such sets of operations *independent*. In the setting of Remark 10, when restricting the entries of Q and P to those belonging to independent operations, the system of equations becomes *linear*.

Dey and Xin observed that following this strategy, repeated application of the operations (a)–(c) actually suffices to decompose an α-decomposed presentation with $k = 1$, but assumed that the generators are also uniquely graded. We prove a more general statement, which therefore also lifts this assumption from the GP Algorithm.

▶ **Lemma 13** ([26, Lemma 5.10]). *Let $[M_{\mathcal{B}}\ N]$ be α-decomposed, then after a suitable sequence of elementary column operations on N, there is a sequence of independent admissible operations which finds an indecomposable decomposition of $[M_{\mathcal{B}}\ N]$ without altering $M_{\mathcal{B}}$.*

At last assume again that $k = 1$. After having zeroed out as much as possible of N, the blocks b for which N_b are still non-zero must all *merge* into a new block, since their non-zero entries now overlap in this last column. This finishes the loop of the GP Algorithm.

Extending `BlockReduce`. Consider an α-decomposed presentation $[M_{\mathcal{B}}\ N]$ and let $k \geq 1$ now be arbitrary, where we recall that k is the number of columns of N. At first, we are still interested in deleting the entire sub-matrices N_b for every $b \in \mathcal{B}$ wherever this is possible. Consider now *sets* of admissible operations of the types (a)-(c) in Example 12, encoded as graded matrices. Using $M_{b,c} \coloneqq M_{b_{\text{rows}} \times c_{\text{cols}}}$, their effect on $(M_{\mathcal{B}}\ N)$ can be described:

(a) A graded matrix $U \in \mathbb{K}^{R_b \times k\cdot\alpha}$ and $N_b \leftarrow N_b + M_b U$.
(b) A graded matrix $Q \in \mathbb{K}^{G_b \times G_c}$ and $N_b \leftarrow N_b + QN_c, \quad M_{b,c} \leftarrow M_{b,c} + QM_c$.
(c) A graded matrix $P \in \mathbb{K}^{R_b \times R_c}$ and $M_{b,c} \leftarrow M_{b,c} + M_b P$.

Finding Q, P such that after performing (b) and (c) the matrix $M_{\mathcal{B}}$ has not changed is equivalent to demanding that $M_{b,c} = 0$, which is a *linear* system $(*)$. Finding U, Q such that performing (a), (b) zeroes out N_b is another one $(**)$.

`BlockReduce` (General Version). `Input`: $[M_{\mathcal{B}}\ N] \in K^{G \times (R \cup k\cdot\alpha)}$, $b \in \mathcal{B}$.

Find $U \in \mathbb{K}^{R_b \times k\cdot\alpha}$ and $\forall c \neq b \in \mathcal{B}$: $\quad Q_c \in \mathbb{K}^{G_b \times G_c}$, $P_c \in \mathbb{K}^{G_b \times R_c}$, such that

$$Q_c M_c + M_b P_c = 0 \quad (*) \quad \text{and} \quad N_b + \sum Q_c N_c + M_b U = 0 \quad (**) \tag{1}$$

In [29], Dey and Xin write these equations in a large matrix which they call S-matrix.

Run time of `BlockReduce`. Let $(M_{\mathcal{B}}\ N)$ be α-decomposed with M of size $m \times n$, then the size of every block is also bounded by $m \times n$. It follows that $(*)$ contains at most $(m^2 + n^2)$ variables and mn equations and $(**)$ contains at most $m^2 + nk$ variables and mk equations. Let $\omega < 2.373$ denote the exponent for matrix multiplication.

▶ **Proposition 14** (cf. Algorithm 3 in [29]). *The time complexity of solving $(*)$ is in $\mathcal{O}\left((nm)^{\omega-1}\left(m^2+n^2\right)\right)$ and of solving $(**)$ is in $\mathcal{O}\left((mk)^{\omega-1}\left(m^2+nk\right)\right)$.*

4 Modifications

AIDA implements many modifications to `BlockReduce` which make this subroutine much faster and enable the *Automorphism-invariant* α-decomposition in Section 6.

Column-Sweeps. When processing a new set of columns N of degreee α_i, we first try to zero out each N_b only with column-operations (i.e., ignoring row-operations first), which is often sufficient and, if not, will maybe reduce the number of non-zero entries. To this end we reduce the sub-matrix $M_b^{\leq \alpha_i}$ of M_b containing the columns of degree $\leq \alpha_i$. Importantly, this matrix can be stored and sometimes even updated when the next set of columns is considered in the same way as in the standard persistence algorithm for one parameter.

`BlockReduce` (Hom-Variant). The following change is subtle but crucial:

1. Find a basis $(Q_i, -P_i)_{i\in I}$ of the solution-space of $(*)$ for every c and store it.
2. Search for the solution of $(**)$ in the vector spaces spanned by $(Q_i, -P_i)_{i\in I}$ for each c.

This split makes no computational difference asymptotically, and its overhead is negligible. In subsequent algorithms, we will call `BlockReduce` recursively without changing $M_{\mathcal{B}}$ (thanks to Lemma 13). By solving and storing $(*)$, we thus avoid recomputations during the recursion.

Morphisms of Presentations. Let $M_c \in \mathbb{K}^{G_1\times R_1}$, $M_b \in \mathbb{K}^{G_2\times R_2}$ present persistence modules X_c, X_b. Up to sign, the solutions of `BlockReduce` $(*)$ form the $\mathbb{K}$-vector space

$$\operatorname{Hom}(M_c, M_b) \coloneqq \left\{(Q, P) \in \mathbb{K}^{G_2\times G_1} \times \mathbb{K}^{R_2\times R_1} \mid QM_c = M_bP\right\}$$

and there is a natural $\mathbb{K}$-linear surjection $\operatorname{Hom}(M_c, M_b) \twoheadrightarrow \operatorname{Hom}(X_c, X_b)$.

So when solving `BlockReduce` $(*)$ for each pair $c \neq b \in \mathcal{B}$, we are actually computing, akin to elementary matrices, the generators of a subgroup of $\operatorname{Aut}(M_{\mathcal{B}})$. Additionally, the property of a morphism $(Q, P) \in \operatorname{Hom}(M_c, M_b)$ to be part of a solution of `BlockReduce` $(**)$ depends only on the map $\widetilde{Q}\colon X_c \to X_b$ which it induces.

Fast Hom-computation. Let X, Y be persistence modules, then $\operatorname{Hom}(X, Y)$ can be computed faster if Y is pointwise "small". This is because we can restrict the target space of each generator of X. We will explore this thoroughly in future research and also point to the following known result for "smallest" modules, also due to Dey and Xin.

▶ **Proposition 15** ([28], Proposition 16). *Let X and Y be interval modules with n generators and relations, then $\operatorname{Hom}(X, Y)$ can be computed in $\mathcal{O}(n)$ time.*

Furthermore, not all homomorphisms are needed to find a decomposition.

▶ **Observation 16.** *The map $\widetilde{Q}\colon X_c \to X_b$ of persistence modules induces a linear map of vector spaces $\left(\widetilde{Q}\right)_\alpha : (X_c)_\alpha \to (X_b)_\alpha$. If this map is the zero map then, since $\operatorname{Im} N_c$ is generated at α, QN_c is a linear combination of the columns of $M_b^{\leq\alpha}$ (cf. Column-Sweeps).*

▶ **Definition 17.** *For persistence modules X, Y we define the space of α-*homomorphisms

$$\operatorname{Hom}^\alpha(X, Y) \coloneqq \operatorname{Hom}(X, Y) \Big/ \{f\colon X \to Y \mid f_\alpha\colon X_\alpha \to Y_\alpha = 0\}.$$

For presentations M_c, M_b one defines $\operatorname{Hom}^\alpha(M_c, M_b)$ analogously.

Thus, instead of finding a basis for the whole solution space of `BlockReduce` $(*)$ we actually only need to compute a set of solutions which span $\operatorname{Hom}^\alpha(X_c, X_b)$. This is a simple reduction based on Observation 16 and will decrease the size of the system $(**)$, but more importantly for Section 6 it will sometimes reduce this solution space to zero.

Interval Modules. For X, Y interval modules we can describe this reduction precisely. By [28, Proposition 16] the vector space $\mathrm{Hom}(X,Y)$ has a basis consisting of "valid" overlaps:

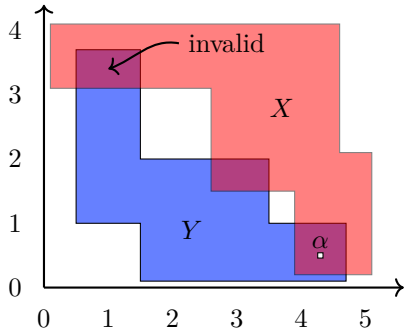


Figure 4 We compute that $\mathrm{Hom}(X,Y) \simeq \mathbb{K}^2$, but $\mathrm{Hom}^{\alpha}(X,Y) \simeq \mathbb{K}$.

At most one of the valid overlaps can contain α and so $\dim_K \mathrm{Hom}^{\alpha}(X,Y) \leq 1$. Define a relation $X \to_{\alpha} Y :\Leftrightarrow \mathrm{Hom}^{\alpha}(X,Y) \neq 0$. This relation is a *partial order* on any set of isomorphism classes of interval modules ([26, Lemma 6.16]). Additionally, an interval module can have at most dimension 1 at α. Therefore, when we consider a set of new relations N in the algorithm, if M_b presents an interval module, we can reduce every sub-column $N_{b,i}$ to have a single non-zero entry. Together with the transitivity of α-maps, this observation allows Gaussian (row-)elimination on N avoiding `BlockReduce` $(**)$ (6, [26, Algorithm B.1]).

5 Decomposing General Presentations

One might hope that we could process the columns of N in some order and just use `BlockReduce`. Unfortunately this does not work: Counterexamples are [5, Example 5.14] and also [26, Example 5.2] where we construct a 2-parameter module over $\mathbb{F}_2$ which is smallest for presentations over $\mathbb{F}_2$ which cannot be decomposed with this method. If on the other hand we allow column operations which do not follow some order, the set of blocks for which N_b can be partially deleted is no long unique up to permutation [26, Example 5.3].

Exhaustive Decomposition. One immediate difference to the distinctly graded case is that a decomposition might introduce several new blocks. Instead of finding all of them at once, we will try to split the presentation into two summands and use recursion. We invoke Lemma 13 again to find invertible matrices, which leave $M_{\mathcal{B}}$ unchanged and such that

$$Q\,[M_{\mathcal{B}}\ N]\begin{bmatrix} P & U \\ 0 & T \end{bmatrix} = [QM_{\mathcal{B}}P \quad Q\,(M_{\mathcal{B}}U + NT)\,] = \begin{bmatrix} M_{\mathcal{B}_1} & 0 & N_1 & 0 \\ 0 & M_{\mathcal{B}_2} & 0 & N_2 \end{bmatrix}.$$

It follows that there is an invertible matrix $T \in \mathbb{K}^{k\times k}$ and a partition $[k] = \{1,\dots,j\} \cup \{j+1,\dots,k\}$ of the columns of N, which allow us to decompose the whole presentation by zeroeing out submatrices with `BlockReduce`: After multiplying N with T from the right, for every block $b \in \mathcal{B}$ we can delete either $N_{b,[j]}$ or $N_{b,[j]^{\complement}}$. Here we see that we would have to solve $(*)$ again, but since we stored its solution this is not necessary and $(**)$ is efficiently solved. Since we work over a finite field, this immediately implies a brute-force algorithm:

Exhaustive α-decomposition. For each invertible matrix T over $\mathbb{F}_q$ and $j \in [k]$ set $N_1 = (NT)_{\{1,\dots,j\}}$, $N_2 = (NT)_{\{j+1,\dots,k\}}$ and decompose each separately with `BlockReduce`.

This is computationally unfeasible even for smaller k: The maximal number of iterations of this loop for $q = 2$ and $k \in \{2, \dots, 6\}$ are 6, 342, 60 822, 40 058 262, 100 833 607 062 and more generally $|\mathbf{GL}_k(\mathbb{F}_q)| \in q^{k^2+\mathcal{O}(k)}$. Fortunately, we can be much smarter about the matrices and partitions we use: Assume that T is as above so that NT decomposes into two parts, supported on $[j]$ and $[j]^{\complement}$, after using `BlockReduce` $(**)$. The two parts also induce a partition of $T = [T_1\ T_2]$. If T_1', T_2' are matrices with the same column-spaces as T_1, T_2 respectively then $N\,[T_1'\ T_2']$ could be brought to the same decomposition with `BlockReduce`. That is, finding a decomposition with this strategy depends only on the *decomposition* of $\mathbb{F}_q^k$ as a vector space which $T = [T_1\ T_2]$ represents. Additionally, once we have deleted as many parts of NT_1 as possible, we can update the block structure and do the same again for NT_2, but now also being able to use column-operations from the columns of NT_1 to those of NT_2 without breaking linearity of the system. That is, we also do not need to consider a decomposition $[T_1\ T_2]$, if a decomposition has been considered already that is of the form $[T_1\ T_2' + T_1 S]$, for any matrix S and any T_2' with the same column-space as T_2. But this is the case for *every* other invertible matrix of the form $[T_1\ T_2']$, so that we only need to iterate over all proper subspaces $\operatorname{span}(T_1)$ of $\mathbb{F}_q^k$, i.e. the Grassmanian $\mathbf{Gr}_l(\mathbb{F}_q^k)$ for every $l < k$. These matrices can be generated fast by leveraging the Schubert cell decomposition [56]:

Algorithm `Generate` $\mathbf{Dec}_q(k, l)$. For each set $P \in \binom{[k]}{l}$ construct a $k \times l$ matrix T_1 in column-echelon form with pivots given by P. Then construct all subspaces in the corresponding cell by changing the entries that are irrelevant for the echelon form and not in the same row as a pivot. For each T_1 generated this way, choose any T_2 with opposite pivots and store the pairs $[T_1\ T_2]$ in a set $\mathbf{Dec}_q(k, l)$.

Avoiding Redundant Counting. Since exchanging T_1 and T_2 in $\mathbf{Dec}_q(k, l)$ would not make a difference for the decomposition we can actually ignore the sets $\mathbf{Dec}_q(k, l)$ for $l > k/2$. In the case where k is even and $l = k/2$ we can avoid double counting even more. Consider the graph on $\mathbf{Gr}_{k/2}(\mathbb{F}_q^k)$ where $U \sim V$ if they form a decomposition (also called the q-Kneser graph). By symmetry we only need to find a minimal vertex cover of this graph and use these subspaces for the computation above. This can be done by considering in `Generate` $\mathbf{Dec}_q(k, k/2)$ only those pivot sets $P \in \binom{[k]}{k/2}$ which contain the first position. Let $\mathbf{Dec}_q(k) \coloneqq \bigcup_{l \le k/2} \mathbf{Dec}_q(k, l)$, then we can iterate over $\mathbf{Dec}_q(k)$ instead of $[k] \times \mathbf{GL}_k(\mathbb{F}_q)$ for the exhaustive decomposition. The first terms of $|\mathbf{Dec}_2(k)|$ are 1, 2, 7, 43, 186, 1965, 14605, 297181, ... and the asymptotic growth is in $q^{k^2/4+\mathcal{O}(k)}$.

At last, when calling the recursion on a sub-matrix we can pull back decompositions we have already tried to the corresponding sub-space to avoid double counting again.

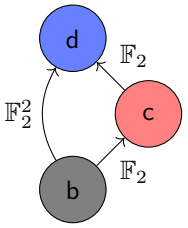
▶ **Theorem 18.** *Let M be a minimal presentation of a persistence module X over $\mathbb{F}_q$ with n generators and relations and at most k relations of the same degree. Then the exhaustive decomposition algorithm has a worst-case running time in $\mathcal{O}\left(n^{2\omega+1} + n^{\omega+2} q^{k^2/4+\mathcal{O}(k)}\right)$.*

Proof. For at most n times we need to compute `BlockReduce` $(*)$ and then solve `BlockReduce` $(**)$ at most $q^{k^2/4+\mathcal{O}(k)}$ many times. Using Proposition 14 gives the term above. ◀

6 Automorphism-invariant α-decomposition

While exhaustive α-decomposition will comfortably work only as long as, say, $k \le 7$, we will encounter presentations where this approach is unsustainable. We will see in the next example that iteration over subspaces as in Section 5 can often be avoided.

▶ **Example 19.** Consider the $(1,3)$-decomposed presentation extending Example 7. We calculate: $\mathrm{Hom}^{\alpha}(M_b, M_c) \simeq \mathbb{K}$, $\mathrm{Hom}^{\alpha}(M_c, M_d) \simeq \mathbb{K}$, and $\mathrm{Hom}^{\alpha}(M_b, M_d) \simeq \mathbb{K}^2$ and all other Hom^{α}-sets are trivial. We depict this information in a digraph structure on $\mathcal{B}$.



M N	$(2,2)$	$(2,2)$	$(1,3)$	$(1,3)$
$(0,1)$	1	0	0	1
$(1,0)$	1	0	0	0
$(1,1)$	0	1	0	1
$(1,2)$	0	0	1	1

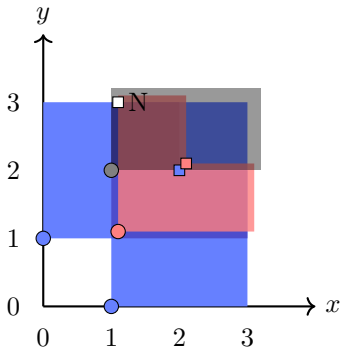


Figure 5 The digraph on $\mathcal{B}$ tells us which sub-modules are *invariant* under α-base-changes. If $\mathcal{B}$ is acyclic, then the group we have computed is *unipotent*.

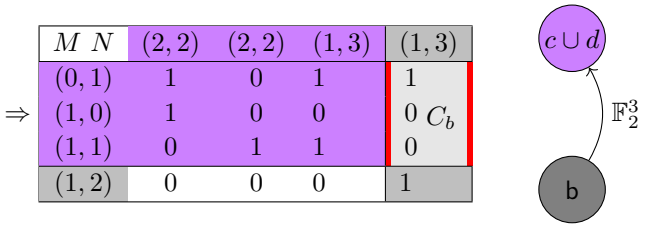
There are no maps to b and so if we want to change N_b then we can only use admissible column operations. If we consider the last row as a presentation itself, it is not decomposed maximally: The two 1s form a block, which can be made smaller by adding the 4th to the 3rd column. That is, we are *minimising* this sub-presentation, reaching the following presentation on the left and the last column index must become a part of the block b.

M N	$(2,2)$	$(2,2)$	$(1,3)$	$(1,3)$
$(0,1)$	1	0	1	1
$(1,0)$	1	0	0	0
$(1,1)$	0	1	1 $N_c^{\{0\}}$	1 C_b
$(1,2)$	0	0	0	1

$\Rightarrow$

M N	$(2,2)$	$(2,2)$	$(1,3)$	$(1,3)$
$(0,1)$	1	0	1 C_c	1
$(1,0)$	1	0	0	0
$(1,1)$	0	1	1	0
$(1,2)$	0	0	0	1

Next we turn to the second block. The subbatch N_c has a part $N_c^{\{0\}}$ sitting above the cleared part of N_b and a part C_b in the new column of b. Now if we were again to add the 4th column to the 3rd, then we could not delete the 1 that would appear at $M_{4,3}$ with any row-operations, because there are no maps to M_b! So the non-existence of maps *to* a block restricts the column operations on N away *from* the block.
This implies that we can not use any operations from outside of c to decompose $\left[M_c \; N_c^{\{0\}}\right]$ and since we cannot reduce $N_c^{\{0\}}$ further, the 3rd column must become part of the block c. Then C_b can be zeroed out using again `BlockReduce` $(**)$ and we reach the presentation above on the right. Next we try to reduce N_d and start with the part C_c in the new column of c, because the column-operation from fourth to third is still impossible. We cannot zero out C_c with admissible operations, so the two blocks c and d need to *merge*:



We update the $\mathrm{Hom}^\alpha(-,-)$ spaces to the new blocks and can continue with C_b.

Extensions concentrated at α. We have seen in Example 19, that if $\mathcal{B}$ is acyclic with respect to $\mathrm{Hom}^\alpha \neq 0$, it is possible to process N in a way so that all admissible operations we need to consider only flow in the direction of the graph. Consequently there is no iteration over subspaces needed. In particular the algorithm would then also work for any field $\mathbb{K}$, not only a finite one. Let us formalise the structure of the matrix we need to decompose:

▶ **Definition 20.** *An α-decomposed presentation* $\begin{bmatrix} M_{\mathcal{C}} & 0 & N_{\mathcal{C}} & C \\ 0 & M_{\mathcal{D}} & 0 & N_{\mathcal{D}} \end{bmatrix}$ *is called* α-decomposed extension *if* $[M_{\mathcal{C}}\ N_{\mathcal{C}}]$ *and* $[M_{\mathcal{D}}\ N_{\mathcal{D}}]$ *are minimal indecomposable decompositions. We denote by κ the number of columns of C.*

When $\mathcal{B}$ is acyclic, we have seen in Example 19 that $\mathcal{C}$ and $\mathcal{D}$ contain just a single block and there are no (α-)maps from $M_{\mathcal{C}}$ to $M_{\mathcal{D}}$. We argued that in this case there are also no column operations from $N_{\mathcal{D}}$ to the left which need to be considered and so we can use `BlockReduce` (**) (on the extended presentations) to decide if C can be zeroed out.

▶ Remark 21. Let X, Y be the persistence modules presented by the two submatrices $[M_{\mathcal{C}}\ N_{\mathcal{C}}]$ and $[M_{\mathcal{D}}\ N_{\mathcal{D}}]$ in Definition 20 and Z the module presented by the entire matrix $[M\ N]$. The inclusion of $[M_{\mathcal{C}}\ N_{\mathcal{C}}]$ into $[M\ N]$ induces a short exact sequence $0 \to X \to Z \to Y \to 0$. To this sequence corresponds a cocycle in $\mathrm{Ext}^1(Y, X)$ which is exactly given by $[C]$.

The correctness of the algorithm follows from our main theorem by lifting isomorphisms:

▶ **Theorem 22.** *Let $0 \to X \xrightarrow{f} Z \xrightarrow{g} Y \to 0$ be induced by an α-decomposed extension. If $\mathrm{Hom}^\alpha(M_{\mathcal{C}}, M_{\mathcal{D}}) = 0$ and Z is decomposable, then so is the entire short exact sequence.*

Proof. [26, Theorem 7.25]. ◀

Notice that the condition means that X is *almost invariant* under the action of $\mathrm{Aut}(Z)$: No automorphism could alter X_α and therefore also not the cocycle $C\colon\ A(-\alpha)^\kappa \to X$.

What if $\mathcal{B}$ is not acyclic? In general there is no order in which we can traverse the blocks, such that maps to the current block come only from parts of the matrix which already form an indecomposable decomposition. The solution here is to contract all strongly connected components of $\mathcal{B}$ forming the *condensation* $\mathcal{T}$ of this digraph.

The α-decomposed extension we need to decompose in each step may then have many blocks in $\mathcal{C}$ and $\mathcal{D}$ and thus cannot be reduced by a linear system anymore. We employ another algorithm `ExtensionDecomp` ([26, Algorithm B.4]) which again loops over subspaces, but of $\mathbb{K}^\kappa$, the column-space of C. The resulting algorithm is FPT in $\kappa_{\max}$, the largest occurrence of κ, and called the Automorphism-invariant Iterative Decomposition Algorithm.

Interval-Decomposables. Assume that the input module is interval-decomposable. Then in Section 4 we have seen that $\mathcal{B}$ is transitive and each $N_{b,i}$ can be assumed to be a single entry. Also, for every α-decomposed extension we need to reduce, the matrices $[M_{\mathcal{C}}\ N_{\mathcal{C}}]$ and $[M_{\mathcal{D}}\ N_{\mathcal{D}}]$ must each present a set of isomorphic intervals. Then all row and column operations on C are possible and we can use normal matrix reduction ([26, Algorithm B.1]).

▶ **Theorem 23.** *The method outlined above decomposes interval decomposable modules in $\mathcal{O}(n^3)$ time where n is the number of generators and relations.*

7 Experiments

We have implemented our decomposition algorithm in a stand-alone C++ library AIDA. AIDA takes as input a minimal presentation in the form of a SCC2020 file[2] which can be generated, for instance, using the MPFREE library [42]. 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 GNU/Linux (Ubuntu 20.04.2) and gcc 9.4.0.

Test instances. We considered a large collection of data files [41] including chain complex data generated by various bifiltrations, including function-Rips, lower-star bifiltrations, and multi-covers – see [33]. Also included are chain complexes of function-alpha bifiltrations for various choices of point clouds and density functions as described in the paper of Alonso et al. [4]. We additionally generated test cases which are not based on bifiltration data by simple random processes that can create interval-decomposable or random presentation inspired by the Erdös-Renyi-model of random graphs ([26, Appendix A]). In total, the benchmark set contains around 1300 different minimal presentation files. Our benchmarks scripts, the version of AIDA that we have used, and the results of our experiments are available at [40].

Most instances in the above collection are distinctly graded, that is $k_{\max} = 1$. This is perhaps surprising because, for instance, function-alpha bifiltrations are generally not distinctly graded simplex-wise. Yet apparently their homology classes have unique degrees. A few instances also have $k = 2$ or $k = 3$, but the only instances with large values of k are the one coming from multi-cover bilfiltrations as described in [25].

Effectiveness of the speed-ups. We implemented our code such that some optimizations can be disabled to gauge their importance. Our first finding is that the column sweep optimization as described at the beginning of Section 3 has the most visible effect on practical performance. In Table 1, we display the running time for some representative instances. We observe that the improvement factor depends on the instance type and even varies a lot for instances of the same type (as the third and fourth row of the table indicate). However, the improvement was consistent over all considered examples.

We next consider the effect of the optimized computation of the system $(*)$. In many instances, this technique does not speed up the computation or even slightly slows it down, due to the technical overhead of solving $(*)$ first. However, the slow-down was never more than a factor of 2 in the experiments. On the other hand, for some instances, a speed-up is achieved; most significantly, this is the case for the off-datasets, which are triangular meshes in 3D bi-filtered along two of the coordinate axis (bottom half of Table 1). This happens

[2] `https://bitbucket.org/mkerber/chain_complex_format/src/master/`

Table 1 Running times (in seconds) for the algorithm in different configurations: "vanilla" means that no modification is enabaled. "+sweep" means that column sweep is enabled, "+homset" means that the separate and optimized computation of $(*)$ is enabled.

type	#Gens	#Rels	vanilla	+sweep	+homset
fun-alpha (kde)	10000	14014	118	0.39	0.42
fun-alpha (random)	6460	11154	7.64	0.37	0.33
random	800	782	6.85	2.12	3.31
random	800	776	54.0	2.54	3.03
off (hand)	595	596	508	0.96	0.02
off (eros)	2819	2820	>3600	22.7	0.40
off (dragon)	8229	8230	>3600	833	9.4
off (raptor)	7203	7186	>3600	1588	32.7

because this filtration tends to create the large but pointwise low-dimensional summands which we have optimized the computation for.

Empirical complexity. We demonstrate in Table 2 that AIDA is able to decompose presentations with hundreds of thousands of generators and relations within seconds in many practical cases. This is also true for non-distinctly graded cases: the bottom half of the table displays multi-cover instances where $k_{\max}$ is larger than 1 ($k_{\max} = 4$ on the left, $k_{\max} = 6$ on the right of the table). We also display the progression of AIDA's time and memory consumption when the input size (roughly) doubles. We can see that in some cases, AIDA scales almost linearly with the size of the input presentation (the upper-left and lower-right part of the table) whereas in other cases the time grows rather quadratically (the upper-right and lower-left part). We also observed instances with a super-quadratic growth in our experiments; nevertheless, the empirical behavior is much better than what the asymptotic bound predicts. This is not surprising for realistic sparse input matrices, in analogy to the persistence algorithm for one parameter, because matrix reduction is generally faster. Also the summands typically stay small, so that $(*)$ can be solved fast.

Table 2 Running time and memory of AIDA for large data sets. The top half are function-alpha bifiltrations for point clouds of a torus with kde function value, in homology dimension 0 and 1. The bottom half are multi-cover bifiltration of a random point sample, with presentations in homology dimension 2 and 1. All numbers are averaged over 5 instances.

dim	#Gens	#Rels	time	mem	dim	#Gens	#Rels	time	mem
0	5000	7431	0.11	17 MB	1	12410	19890	22.3	108 MB
	10000	14701	0.22	29 MB		25681	40288	80.3	277 MB
	20000	29030	0.47	53 MB		53250	81093	562	928 MB
	40000	57691	1.23	101 MB		112229	165769	1952	1.55 GB
2	15707	31414	0.47	50 MB	1	6856	13711	0.12	23 MB
	32251	64503	1.25	97 MB		14570	29140	0.27	46 MB
	71166	142332	6.36	209 MB		33303	66606	0.76	99 MB
	154400	308800	17.6	447 MB		74462	148924	1.95	218 MB

Comparison with other software. We are not aware of any other implementation for the special case of multi-graded persistence modules, but there exist some libraries that include a decomposition algorithm for modules over finite dimensional algebras, for example the

algorithm by Lux and Szöke [50] in the C-MeatAxe library. Since using this algorithm would require too many preprocessing steps, we instead used the decomposition algorithm for quiver representations in the QPA pacakge [54] for GAP [34] and implemented (also in the AIDA package) an algorithm to convert MPM to quiver representations. Our tests show that the QPA algorithm already needs several minutes for presentations with around 100 generators and relations and times out for all presentations we have considered for testing AIDA.

Speeding up barcode template computation. We demonstrate how decomposing a module can speed up computational tasks in multi-parameter persistence: barcode templates are a common tool to capture all combinatorial barcodes that arise from a 2-parameter module through restrictions to one parameter. Such a barcode template is computed via the arrangement of lines in the plane, arising from the joins of generators and relations of a module via point-line duality. It is evident that this construction is compatible with decomposition, meaning that if a module $M = M_1 \oplus M_2$ decomposes, the arrangements of M_1 and M_2 can be computed separately and then be overlaid.

We implemented a small test program that computes the barcode template using the arrangement package of CGAL [59] once for the undecomposed modules and once after applying AIDA. Table 3 shows that the arrangement size as well as the computation time drastically reduces when using AIDA. We remark that the variance of the aida arrangement in size was very large, leading to the effect that the average arrangment size even decreases in our sample. In either case, we speculate that incorporating AIDA would speed-up the visualization of persistence modules in RIVET and is a necessary preprocessing step in the exact computation of matching distances in practice.

Table 3 Running time for computing the barcode template of function-alpha bifiltrations of randomly sampled point in the unit square using random function values. The columns show the number of lines generated that define the arrangement, the number of vertices in the final arrangment and the time to compute it. All number are averages of 5 randomly generated instances.

			With AIDA			Without AIDA		
#points	#Gens	#Rels	#lines	V	time	#lines	V	time
20	18.4	28	24.6	210	0.002	176	6639	0.034
40	59	97	144	6295	0.026	1877	692K	3.50
60	99	166	382	39996	0.19	5096	5.076M	27.1
80	137	224	346	33301	0.16	8801	14.3M	81.8

8 Conclusion

AIDA is the first practical algorithm to decompose multi-parameter persistence modules. We emphasise that it works for any number of parameters (although all experiments were on 2 parameters) and should be adaptable to any other poset or acyclic quiver.

Our improvements are a consequence of a better understanding of MPM on the algebraic level. More speed-ups are possible; for instance, the faster computation of $\mathrm{Hom}(X, Y)$ that we employ is not the best possible and we plan to improve it further. We also believe that the algorithm from Section 6 should show improvement over the exhaustive algorithm of Section 5 for sparse input with larger $k_{\max}$ and non-sparse input also for smaller $k_{\max}$.

References

1 Ángel Javier Alonso and Michael Kerber. Decomposition of zero-dimensional persistence modules via rooted subsets. In *International Symposium on Computational Geometry (SoCG)*, 2023. doi:10.4230/LIPICS.SOCG.2023.7.

2 Ángel Javier Alonso, Michael Kerber, and Siddharth Pritam. Filtration-domination in bifiltered graphs. In *Algorithm Engineering and Experiments (ALENEX)*, 2023. doi:10.1137/1.9781611977561.CH3.

3 Ángel Javier Alonso, Michael Kerber, and Primoz Skraba. Probabilistic analysis of multiparameter persistence decompositions into intervals. In *International Symposium on Computational Geometry (SoCG)*, 2024. doi:10.4230/LIPIcs.SoCG.2024.6.

4 Ángel Javier Alonso, Michael Kerber, Tung Lam, and Michael Lesnick. Delaunay bifiltrations of functions on point clouds. In *Symposium on Discrete Algorithms (SODA)*, 2024. doi:10.1137/1.9781611977912.173.

5 Hideto Asashiba, Mickaël Buchet, Emerson G. Escolar, Ken Nakashima, and Michio Yoshiwaki. On interval decomposability of 2d persistence modules. *Computational Geometry*, 105-106:101879, 2022. doi:10.1016/j.comgeo.2022.101879.

6 Gorô Azumaya. Corrections and supplementaries to my paper concerning Krull-Remak-Schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

7 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat–persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017. doi:10.1016/J.JSC.2016.03.008.

8 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient Two-Parameter Persistence Computation via Cohomology. In *International Symposium on Computational Geometry (SoCG)*, 2023. doi:10.4230/LIPIcs.SoCG.2023.15.

9 Ulrich Bauer and Luis Scoccola. Generic two-parameter persistence modules are nearly indecomposable, 2022. arXiv:2211.15306.

10 Silvia Biasotti, Andrea Cerri, Patrizio Frosini, and Daniela Giorgi. A new algorithm for computing the 2-dimensional matching distance between size functions. *Pattern Recognition Letters*, 32(14):1735–1746, 2011. doi:10.1016/J.PATREC.2011.07.014.

11 Håvard Bakke Bjerkevik. Stabilizing decomposition of multiparameter persistence modules. *Foundations of Computational Mathematics*, January 2025. doi:10.1007/s10208-025-09695-w.

12 Håvard Bakke Bjerkevik and Michael Kerber. Asymptotic improvements on the exact matching distance for 2-parameter persistence. *Journal of Compututational Geometry*, 14(1):309–342, 2023. doi:10.20382/JOCG.V14I1A12.

13 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.

14 Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997. doi:10.1006/jsco.1996.0125.

15 Magnus Bakke Botnan and Michael Lesnick. An introduction to multiparameter persistence, 2023. arXiv:2203.14289.

16 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.

17 Mickaël Buchet, Bianca B. Dornelas, and Michael Kerber. Sparse higher order Čech filtrations. In *International Symposium on Computational Geometry (SoCG)*, 2023. doi:10.4230/LIPICS.SOCG.2023.20.

18 Mickaël Buchet, Bianca B. Dornelas, and Michael Kerber. Sparse higher order Čech filtrations. *Journal of the ACM*, 71(4):28:1–28:23, 2024. doi:10.1145/3666085.

19 Mickaël Buchet and Emerson G. Escolar. Realizations of indecomposable persistence modules of arbitrarily large dimension. In *International Symposium on Computational Geometry (SoCG)*, 2018. doi:10.4230/LIPICS.SOCG.2018.15.

20 Mickaël Buchet and Emerson G. Escolar. Realizations of indecomposable persistence modules of arbitrarily large dimension. *Journal of Computational Geometry*, 13(1):298–326, 2022. doi:10.20382/jocg.v13i1a12.

21 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.

22 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.

23 Alexander Chistov, Gábor Ivanyos, and Marek Karpinski. Polynomial time algorithms for modules over finite dimensional algebras. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, 1997. doi:10.1145/258726.258751.

24 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.

25 René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. Computing the multicover bifiltration. In *International Symposium on Computational Geometry (SoCG)*, 2021. doi:10.4230/LIPICS.SOCG.2021.27.

26 Tamal K. Dey, Jan Jendrysiak, and Michael Kerber. Decomposing multiparameter persistence modules. *Arxiv report 2504.08119*, 2025. doi:10.48550/arXiv.2504.08119.

27 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.

28 Tamal K. Dey and Cheng Xin. Computing Bottleneck Distance for 2-D Interval Decomposable Modules. In *International Symposium on Computational Geometry (SoCG)*, 2018. doi:10.4230/LIPIcs.SoCG.2018.32.

29 Tamal K. Dey and Cheng Xin. Generalized persistence algorithm for decomposing multiparameter persistence modules. *Journal of Applied and Computational Topology*, 6:271–322, 2022. doi:10.1007/s41468-022-00087-5.

30 Herbert Edelsbrunner and John Harer. *Computational Topology: an Introduction*. American Mathematical Society, 2010.

31 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.

32 Marc Fersztand, Emile Jacquard, Vidit Nanda, and Ulrike Tillmann. Harder–Narasimhan filtrations of persistence modules. *Transactions of the London Mathematical Society*, 11(1), 2024. doi:10.1112/tlm3.70003.

33 Ulderico Fugacci, Michael Kerber, and Alexander Rolle. Compression for 2-parameter persistent homology. *Computational Geometry*, 109, 2023. doi:10.1016/j.comgeo.2022.101940.

34 The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.13.1*, 2024. URL: https://www.gap-system.org.

35 David Hilbert. Über die Theorie der algebraischen Formen. *Mathematische Annalen*, 36:473–534, 1890. URL: http://eudml.org/doc/157506.

36 Derek F. Holt. The meataxe as a tool in computational group theory. In *The Atlas of Finite Groups - Ten Years On*, pages 74–81. Cambridge University Press, 1998. doi:10.1017/CBO9780511565830.011.

37 Derek F. Holt and Sarah Rees. Testing modules for irreducibility. *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics*, 57(1):1–16, 1994. doi:10.1017/S1446788700036016.

38 Jan Jendrysiak. AIDA. Software, version 0.2., FWF P 33765-N, swhId: swh:1:dir:c339cb432335e83758b23a807641b9c6a43525c8 (visited on 2025-05-30). URL: https://github.com/JanJend/AIDA, doi:10.4230/artifacts.23282.

39 Jan Jendrysiak. Persistence Algebra. Software, version 0.1., FWF P 33765-N, swhId: swh:1:dir:e41d23c6d743921e03d8a54e45ea3a4d36f7718b (visited on 2025-05-30). URL: https://github.com/JanJend/Persistence-Algebra, doi:10.4230/artifacts.23283.

40 Jan Jendrysiak and Michael Kerber. Aida benchmarks SoCG 2025 [data set]. Graz University of Technology, 2025. doi:10.3217/ag750-fq560.

41 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.

42 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.

43 Michael Kerber and Florian Russold. Representing two-parameter persistence modules via graphcodes, 2025. arXiv:2503.07368.

44 Michael Kerber and Matthias Söls. The localized union-of-balls bifiltration. In *International Symposium on Computational Geometry (SoCG)*, 2023. doi:10.4230/LIPICS.SOCG.2023.45.

45 Mark Kleiner. Partially ordered sets of finite type. *Journal of Soviet Mathematics*, 3(5):607–615, May 1975. doi:10.1007/BF01084663.

46 Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules, 2015. arXiv:1512.00180.

47 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.

48 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.

49 Michèle Loupias. Indecomposable representations of finite ordered sets. In *Representations of Algebras*, pages 201–209. Springer, 1975.

50 Klaus M. Lux and Magdolna Szőke. Computing Decompositions of Modules over Finite-Dimensional Algebras. *Experimental Mathematics*, 16(1):1–6, 2007. doi:10.1080/10586458.2007.10128988.

51 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.

52 Dmitriy Morozov and Amit Patel. Output-sensitive computation of generalized persistence diagrams for 2-filtrations, 2023. arXiv:2112.03980.

53 Richard A. Parker. The computer calculation of modular characters (the meat-axe). In *Computational Group Theory*, pages 267–274. Academic Press, 1984.

54 The QPA-team. *QPA - Quivers, path algebras and representations - a GAP package, Version 1.33*, 2022. URL: https://folk.ntnu.no/oyvinso/QPA/.

55 Lajos Rónyai. Computing the structure of finite algebras. *Journal of Symbolic Computation*, 9:355–373, 1990. doi:10.1016/S0747-7171(08)80017-X.

56 Hermann Schubert. Anzahl-Bestimmungen für lineare Räume beliebiger Dimension. *Acta Mathematica*, 8:97–118, 1900. doi:10.1007/BF02417085.

57 Magdolna Szőke. *Examining Green Correspondents of Weight Modules*. PhD thesis, RWTH Aachen, 1998.

58 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21:1–38, 2020. URL: http://jmlr.org/papers/v21/19-054.html.

59 Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. 2D arrangements. 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#PkgArrangementOnSurface2`.

60 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`.

Range Counting Oracles for Geometric Problems

Anne Driemel
University of Bonn, Germany

Morteza Monemizadeh
Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands

Eunjin Oh
Department of Computer Science and Engineering, POSTECH, Pohang, South Korea

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

In this paper, we study estimators for geometric optimization problems in the sublinear geometric model. In this model, we have oracle access to a point set with size n in a discrete space $[\Delta]^d$, where queries can be made to an oracle that responds to orthogonal range counting requests. The query complexity of an optimization problem is measured by the number of oracle queries required to compute an estimator for the problem. We investigate two problems in this framework, the Euclidean Minimum Spanning Tree (MST) and Earth Mover Distance (EMD). For EMD, we show the existence of an estimator that approximates the cost of EMD with $O(\log \Delta)$-relative error and $O(\frac{n\Delta}{s^{1+1/d}})$-additive error using $O(s \operatorname{polylog} \Delta)$ range counting queries for any parameter s with $1 \leq s \leq n$. Moreover, we prove that this bound is tight. For MST, we demonstrate that the weight of MST can be estimated within a factor of $(1 \pm \varepsilon)$ using $\tilde{O}(\sqrt{n})$ range counting queries.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Range counting oracles, minimum spanning trees, Earth Mover's Distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.42

Related Version *Full Version*: `https://arxiv.org/abs/2504.15292`

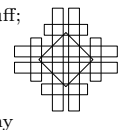
Funding *Eunjin Oh*: Supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00440239, Sublinear Scalable Algorithms for Large-Scale Data Analysis) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00358505).
David P. Woodruff: supported in part by Office of Naval Research award number N000142112647 and a Simons Investigator Award.

Acknowledgements This research was initiated during the Workshop "Massive Data Models and Computational Geometry" held at the University of Bonn in September 2024 and funded by DFG, German Research Foundation, through EXC 2047 Hausdorff Center for Mathematics and FOR 5361 KI-FOR Algorithmic Data Analytics for Geodesy (AlgoForGe). We thank the organizers and participants for the stimulating environment that inspired this research.

1 Introduction

In recent years, the size of data encountered in various applications has grown exponentially. While classical algorithms are designed to process the entire input to get exact or approximate solutions, the increasing demand for *efficiency* in massive datasets necessitates approaches that can provide useful results without examining all of the input data. Motivated by this, a wide range of sublinear-time algorithms has been extensively studied over the past

© Anne Driemel, Morteza Monemizadeh, Eunjin Oh, Frank Staals, and David P. Woodruff;
licensed under Creative Commons License CC-BY 4.0



few decades from various subfields in theoretical computer science. For instance, there are sublinear-time algorithms for the longest increasing subsequence and edit distance problems on strings [32, 34, 29], the triangle counting and vertex cover problems on graphs [10, 26, 38], Earth Mover's Distance on probability distributions [9], and minimum spanning tree on general metric spaces [22]. Moreover, there are sublinear-time algorithms for several fundamental geometric problems [16, 20, 21, 31, 35]. For further information, refer to [24, 39].

To obtain sublinear-time algorithms, we need an *oracle* to access the input data. There are various sublinear-time oracle models in the geometric setting although there are common oracles for strings, graphs and metric spaces. Chazelle et al. [16] presented several geometric algorithms assuming that the input is given without any preprocessing. In the case of point inputs, the only thing one can do is uniform sampling. In the case of polygons, one can also check if two edges are adjacent. Although they showed that several problems admit sublinear-time algorithms in this model, it seems not strong enough to solve a wider range of fundamental geometric problems such as the clustering problems, the Earth Mover's Distance problem, and the minimum spanning problem. Subsequently, many researchers developed sublinear-time algorithms for several different models. Examples are models in which the oracle can answer orthogonal range emptiness queries and cone nearest point queries [20], orthogonal range counting queries [21, 35], or separation queries [31].

In this paper, we study the *range counting oracle model* for geometric optimization problems. Here, we do not have direct access to the input point set, but instead we use an orthogonal range counting data structure for access. More specifically, given an orthogonal range as a query, it returns the number of input points contained in the query range. In fact, many database software systems, such as Oracle[1] and Amazon SimpleDB[2], provide built-in support for such range queries. These queries allow users to compute aggregates, such as the count, sum, or average of records that fall within a specified range of values for a set of attributes, which is crucial for a wide array of applications. For example, in data analytics, range queries are used to calculate the number of sales within a specific date range or determine the total revenue from products within a given price interval. In geographic information systems (GIS), range queries help aggregate spatial data points within certain coordinate bounds, such as counting the number of locations within a specific radius of a given point. Furthermore, in machine learning, range queries are employed during data preprocessing to summarize statistics over selected subsets, which supports tasks such as data filtering and dimensionality reduction.

The following are desirable properties for a reasonable oracle model: the queries to the oracle should be efficient to implement, it is preferable for it to be supported by lots of well-known databases, and it should allow sublinear-time algorithms for fundamental geometric problems. First, this model clearly satisfies the first property; there are numerous works on data structures for range counting queries and their variants [2, 7, 14, 15, 13, 25, 40]. In particular, one can preprocess a set of n points in $\mathbb{R}^d$ in $O(n \log^{d-1} n)$ time to construct a data structure of size $O(n \log^{d-1} n)$ so that the number of points inside a query range can be found in $O(\log^{d-1} n)$ time. Second, range counting queries are already supported by well-known public databases as we mentioned earlier. Therefore, without any special preprocessing, we can apply sublinear-time algorithms designed on the range counting oracle for such databases in sublinear time. Third, we show that several fundamental problems can be solved in sublinear time on the range counting oracle model in this paper.

[1] `https://docs.oracle.com/en/database/other-databases/nosql-database/24.3/sqlreferencefornosql/operator1.html`

[2] `https://docs.aws.amazon.com/AmazonSimpleDB/latest/DeveloperGuide/RangeQueriesSelect.html`

Table 1 Summary of our results. Here, n denotes the number of points, and Δ denotes the size of the domain. For EMD, a parameter s determines a trade-off between the additive error and the query complexity.

		Additive Error	Multiplicative Error	Query Complexity
EMD	UB	$O(n\Delta/s^{1+1/d})$	$O(\log \Delta)$	$\tilde{O}(s)$
	LB	$\Omega(n\Delta/s^{1+1/d})$	–	$\tilde{O}(s)$
Cell Sampling	UB	–	$1 \pm \varepsilon$	$\tilde{O}(\sqrt{n})$
	LB	–	$O(1)$	$\Omega(\sqrt{n})$
MST	UB	–	$1 \pm \varepsilon$	$\tilde{O}(\sqrt{n})$
	LB	–	$O(1)$	$\Omega(n^{1/3})$

We focus on the following three fundamental geometric problems: the Earth Mover's Distance problem, the minimum spanning tree problem, and the cell sampling problem. The Earth Mover's Distance $\mathsf{EMD}(R, B)$ between two point sets R and B in $\mathbb{R}^d$ and the cost $\mathsf{MST}(P)$ of a minimum spanning tree of a point set $P \in \mathbb{R}^d$ are defined as follows.

$$\mathsf{EMD}(R, B) = \min_{\pi: R \to B} \sum_{r \in R} \|r - \pi(r)\| \quad \text{and} \quad \mathsf{MST}(P) = \min_{T} \sum_{uv \in E(T)} \|u - v\|,$$

where π ranges over all one-to-one matchings between R and B, and T ranges over all spanning trees of P. If we have direct access to the input point set(s), then $\mathsf{EMD}(R, B)$ can be computed exactly in near quadratic time in $\mathbb{R}^2$ [4], and approximately within $(1 + \varepsilon)$-relative error in near linear time [3]. Similarly, $\mathsf{MST}(P)$ can be computed exactly in near quadratic time by computing the complete graph on P and applying Kruskal's algorithm, and computed approximately in near linear time [30]. Note that EMD can be used for measuring the similarity between two point sets, and MST can be used for summarizing the distribution of a point set. Thus these problems have various applications in multiple areas of computer science including computer vision [11], machine learning [27] and document similarity [33].

The cell sampling problem is defined as follows. Let P be a set of points in a discrete space $[\Delta]^d$. Given a value r with $1 \leq r \leq \Delta$, let $\mathcal{G}$ be the grid imposed on $[\Delta]^d$ whose cells have side length r. We say a grid cell is *non-empty* if it contains a point of P. Our goal is to sample one non-empty cell almost uniformly at random from the set of all non-empty cells of $\mathcal{G}$. If we have direct access to P, this problem is trivial as we can compute all non-empty cells explicitly. However, in our sublinear model, we cannot compute all non-empty cells if the number of non-empty cells exceeds our desired query complexity. We believe that cell sampling can be considered as a fundamental and basic primitive in sublinear models as sublinear algorithms rely on efficient sampling to extract meaningful information from large data sets. In fact, this problem has been studied in the dynamic streaming model, and has been used as a primitive for several geometric problems [28, 36] and graph problems [19].

Our results. We present sublinear-time algorithms for EMD, Cell Sampling, and MST in the orthogonal range counting oracle model. Our results are summarized in Table 1.

- For EMD, we present an algorithm for estimating the Earth Mover's Distance between two point sets of size n in $[\Delta]^d$ within a multiplicative error of $O(\log \Delta)$ in addition to an additive error of $O(n\Delta/s^{1+1/d})$. Our algorithm succeeds with a constant probability, and uses $\tilde{O}(s)$ range counting queries for any parameter s. Notice that the optimal solution can be as large as $\Theta(n\Delta)$, and in this case, the additive error is relatively small. Furthermore, we show that this trade-off is tight for any parameter s.

- For Cell Sampling, we present an algorithm that samples a non-empty cell in a grid so that the probability that a fixed non-empty cell is selected is in $[(1-\varepsilon)/m, (1+\varepsilon)/m]$ for any constant $0 < \varepsilon < 1$. Here m denotes the number of non-empty cells. We also show that the query complexity here is tight. Moreover, if we sample $\tilde{O}(\sqrt{n}\log(n))$ non-empty cells, we can estimate the number of non-empty cells within $(1 \pm \varepsilon)$-factor.
- For MST, we present an algorithm for estimating the cost of a minimum spanning tree of a point set of size n in $[\Delta]^2$ within a multiplicative error of $1 \pm \varepsilon$. Our algorithm again succeeds with constant probability, and uses $\tilde{O}(\sqrt{n})$ range counting queries. Also, we show that any randomized algorithm using $o(n^{1/3})$ queries has at least a constant multiplicative error.

Related work. Both EMD and MST have been extensively studied across various sublinear models over the past few decades. The work most closely related to ours is the one by Czumaj et al. [20]. They studied the Euclidean minimum spanning tree problem in a different model. In particular, their oracle supports *cone nearest* queries, *orthogonal range emptiness* queries, and *point sampling* queries.[3] Here, given a cone nearest query consisting of a cone C with apex p, the oracle returns $(1+\delta)$-approximate nearest neighbor of p in $C \cap P$, where P is the set of input points. Using $\tilde{O}(\sqrt{n})$ queries, their algorithm estimates the cost of the MST of n points within a relative error of $(1 \pm \varepsilon)$. Moreover, they showed that if the oracle supports orthogonal emptiness queries only, then any deterministic algorithm for estimating the cost of MST within $O(n^{1/4})$-relative error uses $\Omega(\sqrt{n})$ orthogonal range emptiness queries. Sublinear time algorithms for computing a metric minimum spanning tree problem have also been studied [23]. Here, the oracle supports *distance* queries: Given two points, it returns their distance in the underlying space. The most popular sublinear-*space* model is the streaming model. In this model, the input arrives as data stream, and an algorithm is restricted to using a sublinear amount of memory relative to the input size. Both EMD and MST are studied extensively in this model [5, 18, 28], showcasing the challenges and techniques for processing large datasets in the steaming model.

Although no sublinear algorithms for EMD and MST were known in the range counting oracle model prior to our work, other geometric optimization problems have been studied in this model. Monemizadeh [35] presented an algorithm for the facility location problem using $\tilde{O}(\sqrt{n})$ queries, and Czumaj and Sohler [21] studied clustering problems and map labeling problems. To the best of our knowledge, these are the only work done in the range counting oracle model. However, there are several closely related models such as the range-optimization model. For instance, Arya et al. [8] studied the range MST problem, where the goal is to construct a data structure for computing the cost of MST within a query orthogonal range. In this setting, clustering problems also have been considered [1, 37].

Preliminaries. We make use of quadtrees for both EMD and MST. Consider a discrete space $[\Delta]^2$. For $i \in \{0, 1, \ldots, \log \Delta\}$, we use $\mathcal{G}_i$ to denote a grid over the discrete space $[\Delta]^2$ consisting of cells with side lengths of 2^i. A quadtree $\mathcal{Q}$ on $[\Delta]^2$ is then constructed as follows. The root of the quadtree corresponds to the unique cell of $\mathcal{G}_{\log \Delta}$. For each node v of the quadtree of depth i has four equal sized children, each corresponding to a cell in $\mathcal{G}_{i-1}$ contained in the cell corresponding to v. Note that the depth of $\mathcal{Q}$ is $O(\log \Delta)$. If it is clear from the context, we use a node of the quadtree and its corresponding cell interchangeably.

[3] More precisely, the oracle supports cone nearest queries and orthogonal range queries only, but they are also given the access to the input point set, which allow them to sample a vertex uniformly at random.

Let P be a point set in $[\Delta]^d$. For both EMD and MST we will use binary search in combination with range counting queries to find the k^{th} smallest point in dimension $i \leq d$. Furthermore, we will often want to sample a point from P uniformly at random. We can do so using $O(\log \Delta)$ range counting queries in a procedure called *telescoping sampling* [35].

▶ **Lemma 1** (Telescoping Sampling [35]). *We can select a point of P uniformly at random using $O(\log \Delta)$ range counting queries.*

Throughout this paper, we use $\tilde{O}(f(s))$ to denote $O(f(s) \operatorname{polylog} \Delta)$. Due to space restrictions, some proofs are omitted; they can be found in the full version of the paper.

2 Sublinear algorithms for the Earth Mover's Distance problem

In this section, we present several sublinear-time algorithms that approximate the cost of the Earth Mover's Distance between two point sets in $[\Delta]^d$ for an integer $d \geq 1$. The approximation bounds we obtain are tight up to a factor of $O(\log \Delta)$ for any integer $d \geq 1$.

In a one-dimensional space, we know the configuration of an optimal matching: the kth leftmost red point is matched with the kth leftmost blue point for all $1 \leq k \leq n$. Thus it suffices to estimate the cost of this matching. On the other hand, we do not know the exact configuration of an optimal matching in a higher dimensional space. Thus instead of considering an optimal matching, we consider a matching of cost $O(\mathsf{OPT} \log \Delta)$, where OPT is the Earth Mover's Distance between two point sets. This is where the multiplicative error of $O(\log \Delta)$ comes from. Apart from this, we use the same approach for all dimensions $d \geq 1$.

2.1 A sublinear algorithm for points in 1D

In this section, we present a sublinear algorithm using $\tilde{O}(s)$ range counting queries in a one-dimensional space. More precisely, we prove the following theorem. The optimal solution can be as large as $\Theta(n\Delta)$, and in this case, the additive error is relatively small.

▶ **Theorem 2.** *Given two sets R and B of size n in a discrete space $[\Delta]$ and a parameter $s > 1$, we can estimate the cost of the Earth Mover's Distance between R and B within $O(1)$ multiplicative error in addition to $O(\frac{\Delta n}{s^2})$ additive error with probability at least $2/3$ using $\tilde{O}(s)$ range counting queries.*

If no two points coincide, we have $\mathsf{OPT} \geq n$. Therefore, we have the following corollary.

▶ **Corollary 3.** *Given two sets R and B of size n in a discrete space $[\Delta]$ such that no two points of $R \cup B$ coincide, and a parameter $s > 1$, we can estimate the cost of the Earth Mover's Distance between R and B within $O(\frac{\Delta}{s^2})$ multiplicative error with probability at least $2/3$ using $\tilde{O}(s)$ range counting queries.*

Let $R = \{r_1, r_2, \ldots, r_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ be two sets in $[\Delta]$ sorted along the axis. Let M be an optimal matching between R and B. We separate two cases: *long* edges and *short* edges. An edge of M is *long* if its length is at least Δ/s, and it is *short*, otherwise. Let OPT be the Earth Mover's Distance between R and B.

We employ the following *win-win* strategy: For a long edge, a slight perturbation of its endpoints introduces only a constant relative error. Thus, we move its endpoints to the center of a grid, enabling us to estimate the total length efficiently. For short edges, we can disregard cases where the number of such edges is small, as this results in only a small additive error. Consequently, we may assume that the number of short edges is large,

allowing us to estimate their total length using sampling. To further reduce the additive error, we partition the set of short edges into $\log(\Delta/s)$ subsets based on edge length and then consider each subset separately.

Long edges. For long edges, we subdivide $[\Delta]$ into segments each of length $\Delta/(2s)$. For each long edge (r_i, b_i), imagine that we move r_i and b_i to the centers of the segments containing them. Since (r_i, b_i) is long, this introduces an additive error of $O(\Delta/s)$, resulting in a constant relative error. Using this fact, we estimate the total length of all long edges within a constant relative error using $\tilde{O}(s)$ range counting queries. Let L_{long} be the resulting estimator.

▶ **Lemma 4.** *We can estimate the total length of the long edges in M within $O(1)$ multiplicative error using $\tilde{O}(s)$ range counting queries.*

Proof. We subdivide $[\Delta]$ into s segments each of length $\Delta/(2s)$. Here, the number of segments is $O(s)$. Let $\mathcal{S}$ be the set of the resulting segments. Then a long edge contains at least one segment. Imagine that we move every point p incident to a long edge to the center of the segment containing p. For each segment S, we compute the number $n_r(S)$ of red points contained in S and the number $n_b(S)$ of blue points contained in S, respectively, using $O(s)$ queries. Then we construct another point set P' by adding $n_r(S)$ red points to the center of S and adding $n_b(R)$ blue points to the center of S. We can do this without using queries. For a point $p \in R \cup B$, let p' be the its corresponding point in P'. Then consider the matching $M' = \{(r', b') \mid (r, b) \in M\}$. We can compute M' explicitly without using any queries. Let M'_{long} be the set of edges of M' containing at least one segment of $\mathcal{S}$. A long edge of M has its corresponding edge in M'_{long}. However, some short edge of M might have its corresponding edge in M'_{long}. Here, observe that the short edges of M having their corresponding edges in M'_{long} have length at least $\Delta/(2s)$, thus the cost induced by those edges are also within a constant factor of OPT. They will be also counted for short edges, but this also increases the total cost by a constant factor. Therefore, the lemma holds. ◀

Short edges. For short edges, we use random sampling. More specifically, we subdivide the set of short edges into $t = \log(\Delta/s)$ subsets with respect to their lengths: $E_1, E_2, \ldots, E_t$, where E_i is the set of all edges of M of lengths in $[2^{i-1}, 2^i)$. Then we estimate the *number* of edges of E_i for every index i. To do this, we choose a random sample of R of size $x = O(s \log^2 \Delta)$. For each chosen point, we can find its mate. Let S_i denote the set of all edges between the sampled points and their mates. If $|E_i| \geq n/(s \log \Delta)$ then the number of edges in $E_i \cap S_i$ would be a good estimator for $|E_i|$. Otherwise, we can ignore them as E_i induces an additive error of at most $\Delta n/(s^2 \log \Delta)$. This is because every short edges has length at most Δ/s. Let $\ell_i = |E_i \cap S_i| \cdot (n/x)$. Then we have the following lemma.

▶ **Lemma 5.** *If $|E_i| \geq n/(s \log \Delta)$, we have $\ell_i = \Theta(|E_i|)$ with a constant probability. Otherwise, we have $\ell_i \leq O(|E_i|)$ with a constant probability.*

Proof. Let Y_j be the random variable where $Y_j = 1$ if the jth edge in S_i is contained in E_i, and $Y_j = 0$ otherwise. Let $Y = Y_1 + \ldots + Y_x$, where $x = s \log^2 \Delta$ is the number of sampled points. By definition, we have $\mathbb{E}[Y_j] = \frac{|E_i|}{n}$, and $\mathbb{E}[Y] = x \cdot \frac{|E_i|}{n}$. Now we analyze the failure probability. If $|E_i| \geq n/(s \log \Delta)$, we use Chernoff bounds. Then the probability that $|\mathbb{E}[Y] - Y| \geq \varepsilon \mathbb{E}[Y]$ is at most $2\exp(-\varepsilon^2 \cdot p \cdot (s \log^2 \Delta)/3)$, where p is the probability that $Y_j = 1$. In our case, $p = |E_i|/n \geq 1/(s \log \Delta)$. Thus the failure probability is at most $1/\Delta$ in this case. If $|E_i| < n/(s \log \Delta)$, we use Markov's inequality. Then the probability that $Y_i \geq c|E_i|$ is less than a small constant for a large constant c. Therefore, the second part of the lemma also holds. ◀

To amplify the success probability of Lemma 5, we repeat the procedure for computing ℓ_i $O(\log \Delta)$ times. Then we take the minimum, and let $L = \sum_{i=1}^{t} 2^i \cdot \ell_i + L_{\text{long}}$.

▶ **Lemma 6.** *With a constant probability,* $(1/4) \cdot \mathsf{OPT} - n\Delta/s^2 \leq L \leq 4 \cdot \mathsf{OPT} + n\Delta/s^2$.

Proof. Since we can estimate the total length of long edges with an additive error of $2 \cdot \mathsf{OPT}$, it suffices to focus on short edges. For each edge of E_i, we round up its length to 2^i. This induces the additive error of $2 \cdot \mathsf{OPT}$. Now we use Lemma 5. Consider the event that we have the desired bounds for all indices i in Lemma 5. Due to the amplification of success probability we made before, this total success probability is a constant. If $|E_i| \geq (n/(s \log \Delta)$, the estimator ℓ_i has an error of $O(1) \cdot \mathsf{OPT}$ in total for all such indices. If $|E_i| < n/(s \log \Delta)$, we do not have a lower bound guarantee for ℓ_i. However, since the number of such edges is n/s in total, and the length of each such edge is Δ/s, this induces the additive error of $n\Delta/s^2$. Therefore, the total additive error is $n\Delta/s^2$. ◀

2.2 A sublinear algorithm for higher dimensions

In this section, we present a sublinear algorithm that approximates the cost of the Earth Mover's Distance between two point sets in $[\Delta]^d$ for an integer $d \geq 2$. More specifically, we prove the following theorem. Again, notice that the Earth Mover's Distance between two point sets can be as large as $\Theta(\Delta n)$ in the worst case.

▶ **Theorem 7.** *Given two sets R and B of size n in a discrete space $[\Delta]^d$ and a parameter $s > 1$, we can estimate the cost of the Earth Mover's Distance between R and B within $O(\log \Delta)$-relative error in addition to $O(\frac{\Delta n}{s^{1+1/d}})$ additive error with probability at least $2/3$ using $\tilde{O}(s)$ range counting queries.*

If no two points coincide, we have $\mathsf{OPT} \geq n$. Therefore, we have the following corollary.

▶ **Corollary 8.** *Given two sets R and B of size n in a discrete space $[\Delta]^2$ and a parameter $s > 1$ such that no two points in $R \cup B$ coincide, we can estimate the cost of the Earth Mover's Distance between R and B within $O(\max\{\log \Delta, \frac{\Delta}{s^{1+1/d}}\})$-relative error with probability at least $2/3$ using $\tilde{O}(s)$ range counting queries.*

Let OPT be the Earth Mover's Distance between R and B. This algorithm is essentially the same as the algorithm for the one-dimensional case. The only difference is that we do not know the configuration of an optimal matching in a higher dimensional space. Instead, we can use a matching obtained from the quadtree on $[\Delta]^d$, which has cost of $O(\log \Delta) \cdot \mathsf{OPT}$. Then as before, we separate two cases. If an edge has length at least $\Delta/s^{1/d}$, it is *short*. Otherwise, it is *long*. Then we can compute all long edges exactly, and we can estimate short edges using sampling. Details can be found in the full version of this paper.

3 Lower bounds for the Earth Mover's Distance

In this section, we show that for any parameter s, any randomized algorithm that approximates the Earth Mover's Distance between two sets within an additive error of $O(n\Delta/s^{1+1/d})$ requires at least s orthogonal range counting queries. More specifically, we prove:

▶ **Lemma 9.** *For any parameter s and any integers $d \geq 1$ and $\Delta \geq 1$, any randomized algorithm for approximating the Earth Mover's Distance between two point sets of size n in a discrete space $[\Delta]^d$ within additive error $O(n\Delta/s^{1+1/d})$ uses $\Omega(s)$ range counting queries.*

(a) (b)

Figure 1 (a) All segments use the near gadget. The cost of this instance is n. (b) The gray segment uses the far gadget. The cost of this instance is $\Theta(n\Delta/s^2)$.

If points from $B \cup R$ may coincide, achieving any multiplicative approximation factor is impossible. Even when no two points of $B \cup R$ coincide, Lemma 9 shows that no randomized algorithm with a multiplicative error of $O(\Delta/s^{1+1/d})$ uses $o(s)$ range counting queries.

3.1 A lower bound for points in 1D

In this section, we prove Lemma 9 in the one-dimensional space. This will be extended for a higher dimensional space later. To analyze the lower bound on the additive error of a Monte Carlo algorithm, we use the following version of Yao's Minmax Theorem [6]: Assume that there exists a randomized algorithm using s range counting queries with success probability at least $2/3$. Then for any distribution μ of instances, there exists a deterministic algorithm using s range counting queries that returns desired answers on instances chosen from μ with probability at least $2/3$. Here, the success probability of a randomized algorithm is the probability that the output is within our desired bound.

Thus it suffices to construct a distribution μ of instances such that no deterministic algorithm using s range counting queries has success probability at least $2/3$. For this, we define two types of gadgets on a segment S of length $\Delta/(8s)$: the *near* gadget and *far* gadget. See Figure 1. For the far gadget, we put $4n/s$ red points near the left endpoint of S and $4n/s$ blue points near the right endpoint of S. For the near gadget, we put $2n/s$ red points and $2n/s$ blue points alternately starting from the left endpoint of S so that the distance between any two consecutive points is one, and the starting point lies in the left endpoint of S. Additionally, we put $2n/s$ red points and $2n/s$ blue points alternately so that the distance between any two consecutive points is one, and the last point lies in the right endpoint of S. Notice that the cost induced by the near gadget is $\Theta(n/s)$, but the cost induced by the far gadget is $\Theta(n\Delta/s^2)$. Our strategy is to place $O(s)$ copies of the near gadget inside $[\Delta]$ and to hide one far gadget inside $[\Delta]$ with probability $1/2$. In this way, we can construct two types of instances, one with cost n and one with cost $\Theta(n\Delta/s^2)$.

More specifically, we define a distribution μ of instances as follows. We partition $[\Delta]$ into $8s$ segments each of length $\Delta/(8s)$. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_{8s}\}$ be the set of resulting segments along the axis. See Figure 1. With probability $1/2$, we let $t = 0$. Then with probability $1/2$, we choose one index t from $1, 2, \ldots, s$ uniformly at random. That is, the probability that a fixed index i is chosen is $1/(16s)$ for $i = 1, 2, \ldots, 8s$. Then for each index $i \neq t$, we place the near gadget on S_i. For index $i = t$, we place the far gadget on S_i. Notice that we do not place the far gadget anywhere if $t = 0$. Thus it suffices to show that, for any deterministic algorithm using s range counting queries, the probability that it estimates the cost of OPT within an additive error of $O(n\Delta/s^2)$ on the instances chosen from μ is less than $2/3$.

▶ **Lemma 10.** *For any deterministic algorithm using s range counting queries, the probability that it estimates the cost of an instance chosen from μ within an additive error of $O(n\Delta/s^2)$ is less than $2/3$.*

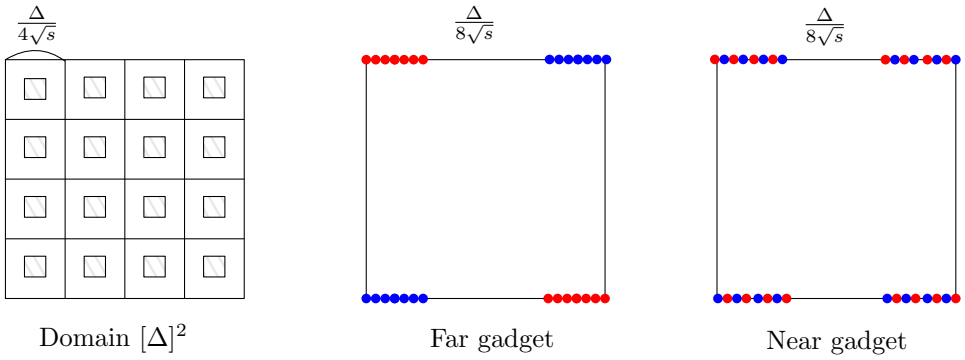


Figure 2 We partition $[\Delta]^2$ into $16s$ squares. In each square, we place either the far gadget or the near gadget. The cost of the far gadget is at least $n\Delta/s^2$, and the cost of the near gadget is $\Theta(n/s)$.

3.2 Lower bounds for dimensions 2 and higher

We now extend the construction of the 1D case to the 2D case. We again use Yao's Minmax Theorem. Thus it suffices to construct a distribution μ of instances such that no deterministic algorithm using s range counting queries has success probability at least $2/3$.

For this, we define two types of gadgets on a square S of side length $\Delta/(8\sqrt{s})$: the *near* gadget and *far* gadget. See Figure 2. For the far gadget, we put two copies of the far gadget constructed from the 1D case, one on the upper side of S and one on the lower side of S, so that a red point comes first in the upper side, and a blue point comes first in the lower side. For the near gadget, we do the same: Put two copies of the near gadget constructed from the 1D case, one on the upper side and one on the lower side, so that a red point comes first in the upper side, and a blue point comes first in the lower side. In this way, every gadget has $8n/s$ red points and $8n/s$ blue points.

Now define a distribution μ of instances as follows. We construct the grid on $[\Delta]^2$ consisting of $16s$ cells each of side length $\Delta/(4\sqrt{s})$. See Figure 2. We choose a cell W as follows. With probability $1/2$, we let $W = \emptyset$. Then with probability $1/2$, we choose one cell from the $16s$ cells uniformly at random, and let it W. Then for each cell in the grid, except for W, we place the near gadget in the middle of the cell. Then we place the far gadget in the middle of W. Notice that we do not place the far gadget anywhere if $W = \emptyset$. Thus it suffices to show that, for any deterministic algorithm using s range counting queries, the probability that it estimates the cost of OPT within an additive error of $O(n\Delta/s^{1.5})$ on an instance chosen from μ is less than $2/3$. We can prove this similarly to Lemma 10.

▶ **Lemma 11.** *For any deterministic algorithm using s range counting queries, the probability that it estimates the cost of an instance chosen from μ within an additive error of $O(n\Delta/s^{1.5})$ is less than $2/3$.*

Extension to a higher dimensional space. The construction of an input distribution used in the two-dimensional case can be easily extended to a higher dimensional space. We again define two gadgets on a d-dimensional cube S of side length $\Delta/(4s^{1/d})$. For the far gadget, we put two copies of the far gadget we constructed from the $(d-1)$-dimensional case, one on one facet of S and one on its parallel facet of S. We do the same for the near gadget using two copies of the near gadget we constructed from the $(d-1)$-dimensional case. Then we can define a distribution of instances as we did for the two-dimensional space. Details can be found in the full version of this paper.

4 Sampling a non-empty cell uniformly at random

In this section, we show how to sample a grid cell (almost) uniformly at random from all grid cells containing points of P. Let $\mathcal{G}$ be a grid of certain side length r imposed on the discrete space $[\Delta]^2$. Let $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ denote the set of all grid cells of $\mathcal{G}$ containing points of P. We say such cells are *non-empty*. For each cell c_i, we let n_i denote the number of points contained in c_i. Here, we say sampling is *c-approximate* uniform if the probability that each non-empty cell is chosen is in $[1/(cm), c/m]$ for a constant c with $c > 1$. Or, we simply say sampling is *almost* uniform.

One might want to use the telescoping sampling of Lemma 1 directly: In particular, we choose a point v of P uniformly at random, and return the cell of $\mathcal{G}$ containing v. However, the probability that a cell c_i is chosen in this way is n_i/n, not $1/m$. So, instead, we use a two-step sampling procedure as stated in Algorithm 1. Here, we are required to check if it has at most $\sqrt{n}$ non-empty cells. If so, we need to find all of them explicitly. This can be done $O(\sqrt{n}\log\Delta)$ range counting queries:

▶ **Lemma 12.** *If the number of non-empty cells is $O(\sqrt{n})$ we can compute all of them using $O(\sqrt{n}\log\Delta)$ range counting queries.*

Algorithm 1 CellSampling(r).

1: Let $\mathcal{G}$ be the grid of side length r imposed on the discrete space $[\Delta]^2$.
2: **if** the number of non-empty cells of $\mathcal{G}$ is at most $\sqrt{n}$ **then**
3: Let $\mathcal{C}$ be the set of non-empty cells of $\mathcal{G}$.
4: Return a cell chosen from $\mathcal{C}$ uniformly at random.
5: **else**
6: Select a set T of $x = \sqrt{n}\log n$ points from P uniformly at random.
7: Let n_p be the number of points of P contained in the cell of $\mathcal{G}$ containing p.
8: Assign weight $w(p) = \frac{n}{xn_p}$ to p.
9: Let $m' = \sum_{p\in T} w(p)$ denote the total weight of all points in T.
10: Sample a point p from T with probability $\frac{w(p)}{m'}$.
11: Return the cell of $\mathcal{G}$ containing p.

▶ **Lemma 13.** CellSampling(r) *selects a cell almost uniformly at random from all non-empty cells of $\mathcal{G}$ using $\tilde{O}(\sqrt{n})$ range counting queries.*

▶ **Corollary 14.** *We can estimate the number of non-empty cells within an $O(1)$-relative error using $\tilde{O}(\sqrt{n})$ range counting queries.*

Lower bound. Now, we show that any randomized algorithm that can perform c-approximate uniform sampling for non-empty cells requires at least $\Omega(\sqrt{n}/c)$ range counting queries for any parameter $c \geq 1$. Our lower bound holds even in a discrete one-dimensional space $[\Delta]$ with $\Delta \geq n$. We assume that we have an interval L of length $\Delta \geq n$ consisting of n cells each of length Δ/n. We subdivide L into $\sqrt{n}/(4c)$ segments each of length $4c\Delta/\sqrt{n}$. Let $\mathcal{S}$ be the set of the segments for the line segment L. In this way, each segment in S contains $(4c\Delta/\sqrt{n})/(\Delta/n) = 4c\sqrt{n}$ cells. We construct a set $\mathcal{I}$ of $\sqrt{n}/(4c) + 1$ instances as follows.

We construct one *uniform* instance of L. In this instance, for each segment $S \in \mathcal{S}$, we identify its leftmost cell and place $4c\sqrt{n}$ points in that cell. Therefore, each uniform instance contains $\sqrt{n}/(4c)$ non-empty cells (one corresponding to each segment) while maintaining a total of n points per uniform instance.

Next, we construct $\sqrt{n}/(4c)$ *non-uniform* instances of L. For the i^{th} non-uniform instance, we select the i^{th} segment, which we refer to as the *witness segment*. In the leftmost cell of this witness segment, we place $2c\sqrt{n}$ points. Additionally, in each of the next $2c\sqrt{n}$ cells following the leftmost cell, we place one point per cell. For each of the remaining $\sqrt{n}/(4c)-1$ non-witness segments, we place $4c\sqrt{n}$ points in the leftmost cell of each segment, as we did for the uniform instance. In this way, every non-uniform instance has $2c\sqrt{n}+1+\sqrt{n}/(4c)-1 = 2c\sqrt{n}+\sqrt{n}/(4c)$ non-empty cells and contains $4c\sqrt{n}\cdot(\sqrt{n}/(4c)-1)+2c\sqrt{n}+2c\sqrt{n} = n$ points in total. Thus, the ratio of the number of non-empty cells between a non-uniform instance and a uniform instance is $(2c\sqrt{n}+\sqrt{n}/(4c))/(\sqrt{n}/(4c)) = 8c^2+1$. For both instances see Figure 3.

▶ **Lemma 15.** *Given an instance I from $\mathcal{I}$, any randomized algorithm that determines whether I is uniform with a success probability of at least $2/3$ requires $\Omega(\sqrt{n}/c)$ range counting queries.*

▶ **Lemma 16.** *Any algorithm that can perform c-approximate uniform sampling for non-empty cells requires $\Omega(\sqrt{n}/c)$ range counting queries for any parameter $c \geq 1$.*

5 Estimating the cost of a minimum spanning tree

Let $P \subseteq [\Delta]^2$ be a set of n points. In this section, we present a $(1+\varepsilon)$-approximation algorithm for estimating the cost OPT of a minimum spanning tree of P that uses $\tilde{O}(\sqrt{n})$ range counting queries. The algorithm is randomized and succeeds with a constant probability. We then show that any randomized constant-factor approximation algorithm requires $\Omega(n^{1/3})$ range counting queries.

5.1 Algorithm for the minimum spanning tree problem

For general graphs, Chazelle, Rubinfeld, and Trevisan [17] presented a $(1+\varepsilon)$-approximation algorithm for estimating the cost of a minimum spanning tree of a graph G. This so-called CRT algorithm uses two types of oracles: we need to sample a vertex of G uniformly at random, and we need to access the list of the neighbors of each vertex. If the average degree of G is d, and the edge weights of G come from $\{1,\ldots,w\}$, then the running time of this algorithm is $O(dw\varepsilon^{-2}\log(dw/\varepsilon))$. Our algorithm is based on the CRT algorithm. As it works for graphs with bounded average degree, we use a quadtree-based $(1+\varepsilon)$-spanner S of P

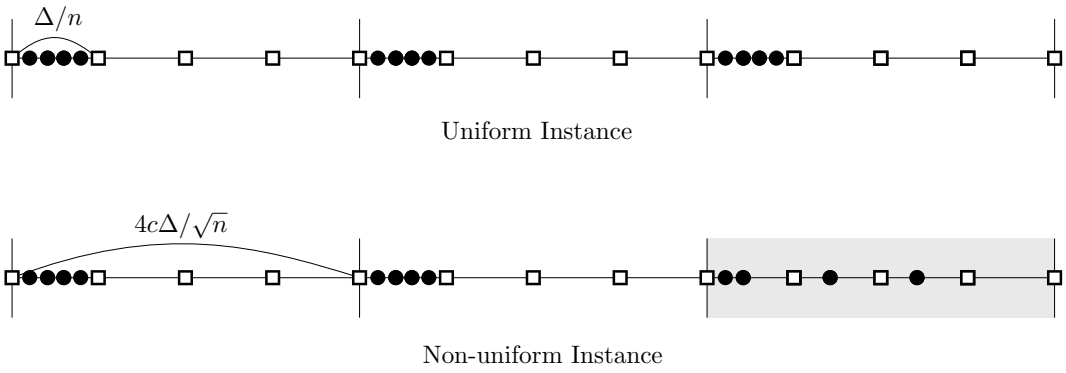


Figure 3 Illustration for the uniform instance and a non-uniform instance. The gray segment is the witness of the non-uniform instance.

introduced by [12]. Here, the maximum degree of S is $O(\log \Delta)$, and the cost of a minimum spanning tree of P is within $(1+\varepsilon)$ to the cost of a minimum spanning tree of S as shown in [30]. Thus it suffices to estimate the cost of a minimum spanning tree of S.

However, a main issue here is that the running time depends on the maximum edge weight, which is Δ in our case. To handle this, we need to look at the CRT algorithm more carefully. As we will see later, the problem reduces to the problem of estimating the number c_i of components of S_i for every $i \in \{0, 1, \ldots, \log_{1+\varepsilon}(2\Delta)\}$, where S_i denotes the subgraph of G induced by edges of length less than $(1+\varepsilon)^i$. To do this, the CRT algorithm samples ε^{-2} vertices from $V(S_i)$, and computes the number of edges contained in the component of S_i containing each sampled vertex v. If v is contained in a small component, we can traverse S_i until we visit all vertices of the component. If v is contained in a large component, we cannot compute this. In this case, we ignore it. This induces the additive error of $\varepsilon(1+\varepsilon)^i \cdot (n/t)$ to the final estimator, where t is the size threshold between large components and small components. To get an approximation factor of $1+\varepsilon$, we need to set $t = \Delta$, which is too large for our purpose. Note that we need to traverse a component of size t, which requires at least t queries. Here, the factor of n/t in the additive error indeed the number of components we cannot traverse using $O(t)$ queries.

Our strategy here is to traverse S_i in a *cell-by-cell* manner. More specifically, we consider a grid $\mathcal{G}$ of side length $(1+\varepsilon)^i/2$, and contract all vertices of S_i in a single cell of $\mathcal{G}$ into a vertex. Then using the same number of queries, we can visit more of the original vertices. By a careful analysis, we can reduce the additive error to $\varepsilon(n/t) \leq \varepsilon \cdot \mathsf{OPT}$. This idea was already used by [28] for maintaining the cost of the minimum spanning tree in the dynamic streaming setting.[4] However, this raises a new issue: we need to sample a *cell containing a point of P* uniformly at random. Here, we can use Lemma 13 to choose a random sample from vertices of S'_i.

Then we have the following lemma. Details can be found in the full version of this paper.

▶ **Theorem 17.** *Given a set P of size n in a discrete space $[\Delta]^2$, we can estimate the cost of a minimum spanning tree of P within a factor of $(1+\varepsilon)$ with a constant probability using $\tilde{O}(\sqrt{n})$ range counting queries.*

5.2 Lower bound for the minimum spanning tree problem

In this section, we show that any randomized constant-factor approximation algorithm for the Euclidean minimum spanning tree problem uses $\Omega(n^{1/3})$ range counting queries. For this, we construct a distribution μ of instances where any randomized algorithm using $o(n^{1/3})$ queries fails to obtain a constant-factor approximate solution with probability at least $1/3$.

Let $[\Delta]^2$ be a discrete domain with $\Delta = O(n)$. We subdivide $[\Delta]^2$ into $16n^{1/3}$ equal-sized cells where each has side length $4n^{5/6}$. See Figure 4. In the middle of each cell, we place either the *strip* gadget or the *uniform* gadget. Each gadget is defined on the domain $[n^{5/6}]^2$ which we further subdivide into $n^{4/3}$ finer cells of side length $n^{1/6}$. For the strip gadget, we put one point to each finer cell on one diagonal of $[n^{5/6}]^2$. For the uniform gadget, we put one point to the cell in the i-th row and the $((i \cdot n^{1/3}) \mod n^{2/3})$-th column for $i = 0, 1, 2, \ldots, n^{2/3}$. In total, each gadget contains $n^{2/3}$ points. Notice that the cost of a

[4] As they did, we can use the Euclidean complete graph on P instead of a spanner S of P. However, this requires additional work: Given two cells c and c' and a value r, we need to check if there are two points $p \in c$ and $p' \in c'$ with $\|p - p'\| = r$. We cannot do this using $O(\log \Delta)$ range counting queries. Instead, we can resolve this by defining another distance function. But we feel that working with spanners makes the analysis simpler.

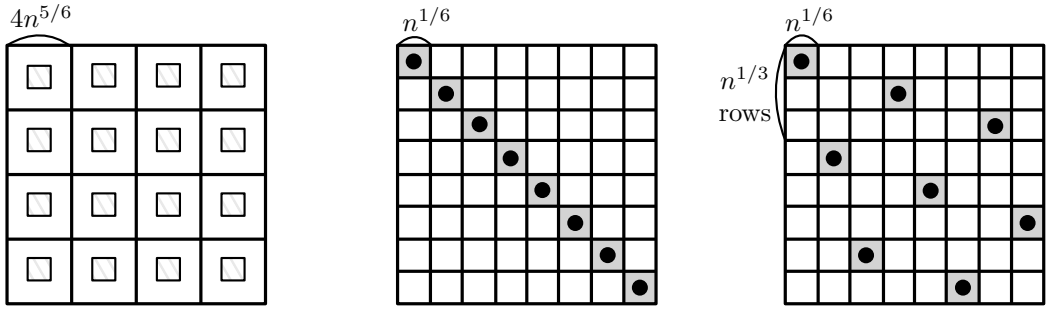


Figure 4 The domain $[\Delta]$ is partitioned into $16n^{1/3}$ cells. Each cell contains the strip gadget or the uniform gadget. The strip gadget has cost $\Theta(n^{5/6})$ while the uniform gadget has cost $\Theta(n^{7/6})$.

minimum spanning tree of the points inside the strip gadget is $\Theta(n^{5/6})$, and the cost of a minimum spanning tree of the points inside the uniform gadget is $\Theta(n^{7/6})$. Now we define a distribution μ of instances. First, with probability $1/2$, we place copies of the strip gadget in all cells of the domain $[\Delta]^2$. Then with probability $1/2$, we pick one cell c of the domain $[\Delta]^2$ uniformly at random. We place a copy of the uniform gadget in c, and place copies of the strip gadget in the other cells. Then we have the following lemma.

▶ **Lemma 18.** *Let I and I' be two instances of μ such that I has the uniform gadget, but I' does not use the uniform gadget. Then $\mathsf{MST}(I') \leq 2 \cdot \mathsf{MST}(I)$, where $\mathsf{MST}(\cdot)$ denotes the cost of a minimum spanning tree.*

Proof. Let M be a minimum spanning tree of I. Observe that the cost of M is at least $2 \cdot n^{7/6}$. This is because M contains at least $n^{1/3}$ edges connecting two points contained different cells in the domain $[\Delta]^2$. Their total length is at least $2 \cdot n^{7/6}$.

Now we analyze an upper bound on the cost of a minimum spanning tree of I'. We can construct a spanning tree of I' from M as follows. The two instances are the same, except that I has a uniform gadget in a cell, say c, and I' has a strip gadget in c. By the cut property, there are at most $O(1)$ edges of M having one endpoint in c and one endpoint lying outside of c. Moreover, such edges have length at most $8n^{5/6}$. For each such edge, we reconnect it with any vertex in c. Then we remove all edges having both endpoints in c from M, and add the edges of the minimum spanning tree of the strip gadget. In total, the cost of M decreases by at most $n^{7/6} - O(1) \cdot n^{5/6}$. Here, the term $n^{7/6}$ is the cost of the uniform gadget. Therefore, we have $\mathsf{MST}(I') - \mathsf{MST}(I) \leq n^{7/6} - O(1) \cdot n^{5/6} \leq (1/2) \cdot n^{7/6} \leq \mathsf{MST}(I)$. ◀

▶ **Lemma 19.** *Any randomized constant-factor approximation algorithm for the minimum spanning tree problem on a point set of size n in a discrete space $[\Delta]^d$ requires $\Omega(n^{1/3})$ range counting queries.*

References

1 Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Range-Clustering Queries. In *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 5:1–5:16, 2017. doi:10.4230/LIPICS.SOCG.2017.5.

2 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting in three and higher dimensions. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 149–158, 2009. doi:10.1109/FOCS.2009.58.

3 Pankaj K Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, pages 1052–1065, 2022. doi:10.1145/3519935.3519977.

4 Pankaj K. Agarwal, Hsien-Chih Chang, and Allen Xiao. Efficient algorithms for geometric partial matching. In *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019)*, pages 6:1–6:14, 2019. doi:10.4230/LIPICS.SOCG.2019.6.

5 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 324–330, 2009. doi:10.1109/FOCS.2009.25.

6 Sanjeev Arora and Boaz Barak. Computational complexity: a modern approach, 2009.

7 Sunil Arya and David M Mount. Approximate range searching. *Computational Geometry*, 17(3-4):135–152, 2000. doi:10.1016/S0925-7721(00)00022-5.

8 Sunil Arya, David M. Mount, and Eunhui Park. Approximate geometric mst range queries. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34, pages 781–795, 2015. doi:10.4230/LIPICS.SOCG.2015.781.

9 Khanh Do Ba, Huy L Nguyen, Huy N Nguyen, and Ronitt Rubinfeld. Sublinear time algorithms for earth mover's distance. *Theory of Computing Systems*, 48:428–442, 2011. doi:10.1007/S00224-010-9265-8.

10 Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Sublinear time algorithms and complexity of approximate maximum matching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 267–280, 2023. doi:10.1145/3564246.3585231.

11 Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pages 1–12, 2011.

12 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.

13 Timothy M Chan. Orthogonal range searching in moderate dimensions: kd trees and range trees strike back. *Discrete & Computational Geometry*, 61:899–922, 2019. doi:10.1007/S00454-019-00062-5.

14 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 (SoCG 2011)*, pages 1–10, 2011.

15 Timothy M Chan and Konstantinos Tsakalidis. Dynamic orthogonal range searching on the RAM, revisited. *Journal of Computational Geometry*, 9(2):45–66, 2018. doi:10.20382/JOCG.V9I2A5.

16 Bernard Chazelle, Ding Liu, and Avner Magen. Sublinear geometric algorithms. In *Proceedings of the thirty-fifth annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 531–540, 2003. doi:10.1145/780542.780620.

17 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. doi:10.1137/S0097539702403244.

18 Xi Chen, Vincent Cohen-Addad, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Streaming Euclidean MST to a constant factor. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 156–169, 2023. doi:10.1145/3564246.3585168.

19 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ESA.2017.29.

20 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 J. Comput.*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

21 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *Proceedings of the 9th Annual European Symposium on Algorithms (ESA 2001)*, pages 266–277, 2001. doi:10.1007/3-540-44676-1_22.

22 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC 2004)*, pages 175–183, 2004. doi:10.1145/1007352.1007386.

23 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM J. Comput.*, 39(3):904–922, 2009. doi:10.1137/060672121.

24 Artur Czumaj and Christian Sohler. Sublinear-time algorithms. *Property testing: current research and surveys*, pages 41–64, 2010. doi:10.1007/978-3-642-16367-8_5.

25 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. Computational geometry: Algorithms and applications, 2008.

26 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 722–734, 2018. doi:10.1145/3188745.3188810.

27 Rémi Flamary, Marco Cuturi, Nicolas Courty, and Alain Rakotomamonjy. Wasserstein discriminant analysis. *Machine Learning*, 107:1923–1945, 2018. doi:10.1007/S10994-018-5717-1.

28 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Proceedings of the twenty-first annual symposium on Computational geometry (SoCG 2025)*, pages 142–149, 2005. doi:10.1145/1064092.1064116.

29 Elazar Goldenberg, Robert Krauthgamer, and Barna Saha. Sublinear algorithms for gap edit distance. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 1101–1120, 2019. doi:10.1109/FOCS.2019.00070.

30 Sariel Har-Peled. *Geometric approximation algorithms*. Number 173 in Mathematical Surveys and Monographs. American Mathematical Soc., 2011.

31 Sariel Har-Peled, Mitchell Jones, and Saladi Rahul. Active-learning a convex body in low dimensions. *Algorithmica*, 83:1885–1917, 2021. doi:10.1007/S00453-021-00807-W.

32 Tomasz Kociumaka and Barna Saha. Sublinear-time algorithms for computing & embedding gap edit distance. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1168–1179, 2020. doi:10.1109/FOCS46700.2020.00112.

33 Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *Proceedings of 32rd International Conference on Machine Learning (ICML 2015)*, pages 957–966, 2015. URL: http://proceedings.mlr.press/v37/kusnerb15.html.

34 Michael Mitzenmacher and Saeed Seddighin. Improved sublinear time algorithm for longest increasing subsequence. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1934–1947, 2021. doi:10.1137/1.9781611976465.115.

35 Morteza Monemizadeh. Facility location in the sublinear geometric model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275, pages 6:1–6:24, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.6.

36 Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error lp-sampling with applications. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1143–1160. SIAM, 2010. doi:10.1137/1.9781611973075.92.

37 Eunjin Oh and Hee-Kap Ahn. Approximate range queries for clustering. In *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99, pages 62:1–62:14, 2018. doi:10.4230/LIPICS.SOCG.2018.62.

38 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA 2012)*, pages 1123–1131, 2012. doi:10.1137/1.9781611973099.88.

39 Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011. doi:10.1137/100791075.

40 Cheng Sheng and Yufei Tao. New results on two-dimensional orthogonal range aggregation in external memory. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS 2011)*, pages 129–139, 2011. doi:10.1145/1989284.1989297.

On Spheres with k Points Inside

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Alexey Garber
School of Mathematical and Statistical Sciences, University of Texas Rio Grande Valley, Brownsville, TX, USA

Morteza Saghafian
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Abstract

We generalize a classical result by Boris Delaunay that introduced Delaunay triangulations. In particular, we prove that for a locally finite and coarsely dense generic point set A in $\mathbb{R}^d$, every generic point of $\mathbb{R}^d$ belongs to exactly $\binom{d+k}{d}$ simplices whose vertices belong to A and whose circumspheres enclose exactly k points of A. We extend this result to the cases in which the points are weighted, and when A contains only finitely many points in $\mathbb{R}^d$ or in $\mathbb{S}^d$. Furthermore, we use the result to give a new geometric proof for the fact that volumes of hypersimplices are Eulerian numbers.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Triangulations, higher-order Delaunay triangulations, hypertriangulations, Delone sets, k-sets, Worpitzky's identity, hypersimplices

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.43

Related Version *Full Version*: `https://arxiv.org/abs/2410.21204`

Funding *Herbert Edelsbrunner*: partially supported by the Wittgenstein Prize, Austrian Science Fund (FWF), grant no. Z 342-N31, and by the DFG Collaborative Research Center TRR 109, Austrian Science Fund (FWF), grant no. I 02979-N35.
Alexey Garber: partially supported by the Simons Foundation.
Morteza Saghafian: partially supported by the Wittgenstein Prize, Austrian Science Fund (FWF), grant no. Z 342-N31, and by the DFG Collaborative Research Center TRR 109, Austrian Science Fund (FWF), grant no. I 02979-N35.

1 Introduction

In the seminal paper [5], Boris Delaunay (also spelled Delone) introduced the Delaunay triangulation of a finite point sets using simplices with empty circumspheres. His construction can be reformulated as follows: for a (finite and generic) point set, $A \subseteq \mathbb{R}^d$, the simplices with vertices in A that contain no points of A inside their circumspheres cover the convex hull of A in one layer. In this paper, we generalize Delaunay's construction and prove similar properties for simplices with circumspheres that enclose exactly k points of A, for some fixed non-negative integer k. We call these simplices the k-hefty simplices of A.

We introduce the main concepts we will work with. A set $A \subseteq \mathbb{R}^d$ is *locally finite* if every closed ball contains at most a finite number of the points of A, and it is *coarsely dense* if every closed half-space contains at least one and therefore infinitely many of the points of A. If A has both properties, we call it a *thin Delone set*; compare with the more restrictive class of *Delone sets*, which are *uniformly discrete* and *relatively dense*, meaning the smallest inter-point distance is bounded away from 0, and the radius of the largest empty ball is bounded away from ∞. We call A *generic* if no $d+1$ of its points lie on a common hyperplane, and no $d+2$ of its points lie on a common $(d-1)$-sphere. Any $(d-1)$-sphere

© Herbert Edelsbrunner, Alexey Garber, and Morteza Saghafian;
licensed under Creative Commons License CC-BY 4.0

bounds a closed d-ball and thus partitions $\mathbb{R}^d$ into points *inside* the sphere (in the interior of the ball), points *on* the sphere, and points *outside* the sphere (in the complement of the ball). Assuming A is generic, there is a unique $(d-1)$-sphere passing through any $d+1$ points of A, which we call the *circumscribed sphere* of the d-simplex spanned by the points.

▶ **Main Definition.** *Let k be a non-negative integer and $A \subseteq \mathbb{R}^d$ a generic thin Delone set or a generic finite set. A d-simplex with vertices in A is* k-hefty *if exactly k points of A lie inside the circumsphere of the d-simplex.*

For example, the 0-hefty simplices are the top-dimensional simplices in the Delaunay triangulation of A, and k-hefty simplices with $k > 0$ are related to the cells in higher-order Delaunay triangulations [2, 7]. Our main result is Theorem 2.2, which we restate here in less technical terms:

▶ **Main Theorem.** *Let k be a non-negative integer and $A \subseteq \mathbb{R}^d$ a generic thin Delone set. Then the k-hefty simplices of A cover $\mathbb{R}^d$ exactly $\binom{d+k}{d}$ times.*

More specifically, almost every point of $\mathbb{R}^d$ is covered by exactly $\binom{d+k}{d}$ k-hefty simplices, while boundary points of k-hefty simplices may be contained in more than this number of such simplices. We also prove versions of this theorem for finitely many points, points on the d-dimensional sphere, and weighted points. In addition, we apply the covering multiplicities to get a new proof that the volumes of hypersimplices are Eulerian numbers, and to get new proofs for some bounds on k-sets.

The paper is organized as follows. In Section 2, we introduce the main definitions, prove the main result for thin Delone sets (Theorem 2.2) and finite sets (Corollary 2.3), and formulate their local versions (Theorem 2.4). In Section 3, we apply the results to obtain a new proof for the fact that volumes of hypersimplices are Eulerian numbers and new proofs for old bounds on k-sets. In the concluding Section 4, we discuss extensions of the results to points in hyperbolic and spherical spaces and to points with real weights in Euclidean space.

2 Hefty Simplices in Euclidean Space

This section presents the main result of this paper, which is stated for infinite and finite point sets in Euclidean space. We begin with the main technical lemma before stating and proving the main theorem.

2.1 Main Technical Lemma

For technical reasons we first show that the k-hefty simplices of a thin Delone set A are "locally uniform" in size. Specifically, we prove an upper bound for the radii of spheres that enclose a fixed point, $x \in \mathbb{R}^d$, as well as at most k points of A. To this end, we write $B(x, R)$ for the closed ball with center x and radius R, and note that the number of points of A in this ball goes to infinity when R goes to infinity.

▶ **Lemma 2.1.** *Let $A \subseteq \mathbb{R}^d$ be coarsely dense, k a non-negative integer, and $x \in \mathbb{R}^d$. Then there exists $R = R(x, A, k)$ such that if x is inside a sphere that is not fully contained in $B(x, R)$, then there are at least $k+1$ point of $A \cap B(x, R)$ inside this sphere.*

Proof. Without loss of generality, assume $x = 0$. For every unit vector, $u \in \mathbb{S}^{d-1}$, the open halfspace of points y that satisfy $(y, u) > 0$ contains infinitely many points of A. It follows that the function $f_u \colon (0, \infty) \to \mathbb{Z}$ that maps $r > 0$ to the number of points of A inside the sphere with center ru and radius r is non-decreasing and unbounded.

We introduce $g\colon \mathbb{S}^{d-1} \to \mathbb{R}$ defined by $g(u) = \inf\{r > 0 \mid f_u(r) \geq k+1\}$ and claim that g is bounded. To derive a contradiction, suppose g is unbounded, and let $u_1, u_2, \ldots$ be a sequence of unit vectors with $g(u_n) \geq n$. Since $\mathbb{S}^{d-1}$ is compact, there is a subsequence that converges to a vector $u_0 \in \mathbb{S}^{d-1}$. Let S be the sphere with radius $g(u_0)+1$ and center $(g(u_0)+1)u_0$. By construction, there are at least $k+1$ points of A inside S. Since these points are (strictly) inside the sphere, there is a sufficiently small $\varepsilon > 0$ such that moving the center of the sphere by at most ε while adjusting its radius so the origin remains on the sphere, retains at least $k+1$ point of A inside the sphere. But this contradicts the unboundedness of g as there are points u_i in the subsequence that are within distance ε from u_0 with $g(u_i)$ much larger than $g(u_0)+1$.

Since g is bounded, $M = \sup\{g(u) \mid u \in \mathbb{S}^{d-1}\}$ is finite and, by construction of g, there are at least $k+1$ points of A inside any sphere with center y and radius $\|y\|$ as long as $\|y\| \geq M$. Setting $R = 2M$, every sphere with center y that encloses the origin and is not contained in $B(0,R)$ has radius $r > M$. This sphere encloses the ball with center $M\frac{y}{\|y\|}$ and radius M, so there are at least $k+1$ points of A inside this sphere that all belong to $B(0,R)$. ◀

As an immediate consequence of Lemma 2.1, the circumsphere of any k-hefty simplex of A that encloses x is completely contained in $B(x,R)$, in which R depends on x, A, and k.

2.2 Global Covering

Our first goal is to generalize the classic result of Delaunay that the 0-hefty simplices of every thin Delone set cover $\mathbb{R}^d$ in one layer; that is: every point of $\mathbb{R}^d$ is contained in at least one 0-hefty simplex and almost every point of $\mathbb{R}^d$ is contained in exactly one 0-hefty simplex. Specifically, we show that for every generic thin Delone set, $A \subseteq \mathbb{R}^d$, the family of k-hefty simplices covers $\mathbb{R}^d$ $\binom{d+k}{d}$ times. We call $\binom{d+k}{d}$ the *k-th covering number* and note that it depends on the dimension, d, and the parameter, k, but not on the set A. We call $x \notin A$ *generic with respect to* A if $A \cup \{x\}$ is generic. Almost every point $x \in \mathbb{R}^d$ is generic with respect to a generic thin Delone set, A. To see this, observe that by local finiteness of A there are only countably many hyperplanes spanned by d points each or spheres spanned by $d+1$ points each, so the union of these hyperplanes and spheres has Lebesgue measure zero.

▶ **Theorem 2.2.** *Let k be a non-negative integer and $A \subseteq \mathbb{R}^d$ a generic thin Delone set. Then any point $x \in \mathbb{R}^d$ that is generic with respect to A belongs to exactly $\binom{d+k}{d}$ k-hefty simplices of A.*

Proof. The case $d = 1$ is obvious since every k-hefty simplex of A is an interval with endpoints in A and exactly k points between the two endpoints. Every point that is generic with respect to A, i.e. in $\mathbb{R} \setminus A$, is contained in exactly $k+1$ such intervals. For $d \geq 2$, the proof splits into three steps.

Step 1. Letting k be a non-negative integer and $A \subseteq \mathbb{R}^d$ a generic thin Delone set, we prove that there is a constant $c = c(k,A)$ such that any point that is generic with respect to A is contained in exactly c k-hefty simplices of A. Write $cover_k(x,A)$ for the number of k-hefty simplices of A that contain x. By Lemma 2.1, $cover_k(x,A)$ is finite. Indeed, every k-hefty simplex of A that contains x must select its vertices from the finitely many points inside the ball $B(x,2M)$. To show that $cover_k(x,A)$ is the same for all generic points, we move x continuously from one point to another. The only time $cover_k(x,A)$ can change is when x passes through the boundary of a k-hefty simplex. It suffices to show that for every $(d-1)$-simplex, Δ, with vertices in A, the number of k-hefty simplices with facet Δ is the same on both sides of Δ.

Consider the line, L, that consists of all points equidistant to the vertices of Δ and mark each point $y \in L$ with the number of points of A inside the sphere with center y that passes through the vertices of Δ. This partitions L into labeled intervals, and since A is generic, the labels of two consecutive intervals differ by exactly one. Fix a *left to right* direction on L, move y in this direction, and observe that the portion of space inside the sphere centered at y that lies to the left of the hyperplane spanned by Δ shrinks, while the portion to the right of this hyperplane grows. The transitions from an interval labeled $k+1$ to another labeled k are in bijection with the k-hefty simplices with facet Δ to the left of Δ. Indeed, as y makes the transition, there is a point of A that passes from inside to outside the sphere centered at y, so this point is to the left of Δ. Similarly, the transitions from an interval labeled k to another labeled $k+1$ are in bijection with the k-hefty simplices with facet Δ to the right of Δ. There are equally many transitions of either kind because the labels go to infinity on both sides. This proves that $cover_k(x, A)$ does not depend on x.

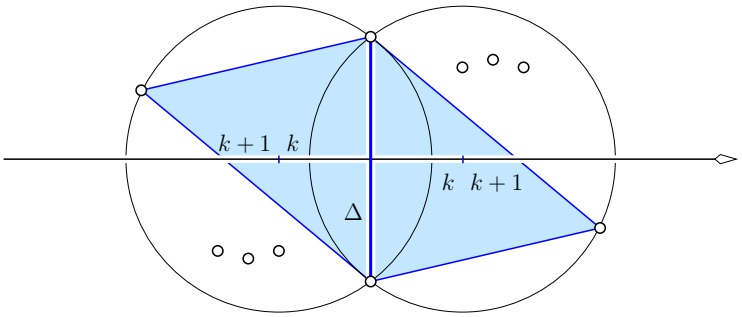


Figure 1 Two circles in the 1-parameter family of circles that pass through the endpoint of the edge Δ. Both are the circumcircles of k-hefty triangles, with $k = 3$ in the case displayed. As we move the center from left to right, every point that leaves the inside of the circle lies to the left of Δ, and every point that enters the inside of the circle lies to the right of Δ.

Step 2. We strengthen the claim proved in Step 1 by showing that the constant depends on d and k but not on A. Specifically, we prove that for every dimension d and non-negative integer k, there exists a number $f(d, k)$ such that for any generic thin Delone set, $A \subseteq \mathbb{R}^d$, any point $x \in \mathbb{R}^d$ generic with respect to A belongs to exactly $f(d, k)$ k-hefty simplices of A.

It suffices to show that for two thin Delone sets, A and A', and two points, $x, x' \in \mathbb{R}^d$, that are generic with repect to both sets, $cover_k(x, A) = cover_k(x', A')$. By Lemma 2.1, there exists $R > 0$ such that if a sphere encloses x and a point outside $B(x, R)$, then there are at least $k+1$ points of $A \cap B(x, R)$ inside this sphere. It follows that the circumspheres of all k-hefty simplices that enclose x are contained in $B(x, R)$. Similarly, let R' be the constant from Lemma 2.1 for x' and A'. We construct a new thin Delone set, A'': picking points y and y' at distance larger than $R + R'$ from each other, we let $A'' \cap B(y, R)$ be a translate of $A \cap B(x, R)$ and $A'' \cap B(y', R')$ a translate of $A' \cap B(x', R')$. We perturb the points if necessary to achieve genericity, and add more points outside $B(y, R)$ and $B(y', R')$ until A'' is thin Delone. By choice of R and R' we have $cover_k(x, A) = cover_k(y, A'')$, because every k-hefty simplex of A whose circumsphere encloses x translates into a k-hefty simplex of A'' whose circumsphere encloses y, and vice versa. Similarly, we have $cover_k(x', A') = cover_k(y', A'')$. Finally, $cover_k(x, A) = cover_k(x', A')$ because $cover_k(y, A'') = cover_k(y', A'')$ as proved in Step 1.

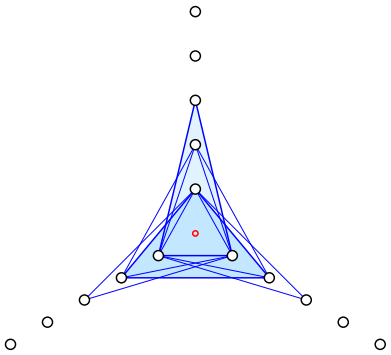


Figure 2 Before perturbation, the points of A lie on three half-lines emanating from the origin. We show six 2-hefty triangles, and emphasize two of them by shading.

Step 3. We provide an explicit example of a generic thin Delone set, $A \subseteq \mathbb{R}^d$, and a point, $x \in \mathbb{R}^d$, with $cover_k(x, A) = \binom{d+k}{d}$. Specifically, we prove that for every dimension d and non-negative integer k, there exists a generic thin Delone set $A \subseteq \mathbb{R}^d$, and a point x generic with respect to A, such that $cover_k(x, A) = \binom{d+k}{d}$.

Let S be a regular d-simplex with vertices $v_0, v_1, \ldots, v_d$ and barycenter $0 = \frac{1}{d+1}\sum_i v_i$, and A the set of points of the form iv_j, for integers $i \geq 1$ and $j = 0, 1, \ldots, d$. We call A a *radial* set and the points with fixed j a *direction* of A; see Figure 2. Set $x = 0$. Every k-hefty simplex of A that contains 0 has exactly one vertex from each direction. Let $i_0v_0, i_1v_1, \ldots, i_dv_d$ be the vertices of such a simplex. The number of points of A inside its circumsphere is $\sum_j(i_j - 1) = k$. Enumerating these simplices is the same as writing k as an ordered sum of $d + 1$ non-negative integers, and there are exactly $\binom{d+k}{d}$ ways to do that. To complete the proof, we perturb A while making sure that $d + 1$ points span a simplex that contains 0 before the perturbation iff they span such a simplex after the perturbation. This completes the proof of our main result. ◀

2.3 The Finite Case

For a finite set, $A \subseteq \mathbb{R}^d$, the covering multiplicity of Theorem 2.2 holds sufficiently deep inside the set but acts only as an upper bound near the fringes. To formalize these claims, we introduce a parametrized generalization of the convex hull: for each integer $k \geq 0$, the *k-hull* of A, denoted $H_k(A)$, is the common intersection of all closed half-spaces that miss at most k of the points in A. Clearly, $H_0(A) = \operatorname{conv} A$, and $H_k(A) \supseteq H_{k+1}(A)$ for every k. It is also easy to see that $H_k(A)$ is contained in the convex hull of A after removing at most k of its points, and indeed $H_k(A)$ *is* the intersection of all such convex hulls. By the Centerpoint Theorem of discrete geometry [6, Section 4.1], $H_k(A) \neq \emptyset$ if $k < \frac{n}{d+1}$, in which $n = \#A$ is the number of points in A.

▶ **Theorem 2.3.** *Let $A \subseteq \mathbb{R}^d$ be finite and generic, $k \geq 0$ an integer, and $x \in \mathbb{R}^d$ generic with respect to A. Then x is covered by at most $\binom{d+k}{d}$ k-hefty simplices of A, with equality iff $x \in H_k(A)$.*

Proof. To prove the upper bound, we add points outside all circumspheres of $d + 1$ points in A to construct a thin Delaunay set $A' \subseteq \mathbb{R}^d$. This is possible because the union of balls bounded by such spheres is bounded. Hence, $A \subseteq A'$, and any k-hefty simplex of A is also a k-hefty simplex of A'. Assuming x is generic with respect to A', Theorem 2.2 implies that

exactly $\binom{d+k}{d}$ k-hefty simplices of A' cover x. If all of them are also k-hefty simplices of A, then x is covered by exactly $\binom{d+k}{d}$ k-hefty simplices of A, but if not, then x is covered by fewer than $\binom{d+k}{d}$ k-hefty simplices of A.

"$\Longleftarrow$". We prove $x \in H_k(A)$ implies that every k-hefty simplex of A' covering x is also a k-hefty simplex of A. Since A' has exactly $\binom{d+k}{d}$ k-hefty simplices that cover x, this will imply that A has the same number of such simplices. Consider such a k-hefty simplex of A', and let $B \subseteq A'$ be the points inside its circumsphere. Since $x \in H_k(A)$, every closed half-space that misses at most k points of A contains x. But $\#B = k$, so x belongs to the convex hull of $A \setminus B$. The top-dimensional simplices of $\mathrm{Del}(A \setminus B)$ cover the convex hull once, hence there is a unique 0-hefty simplex of $A \setminus B$ that covers x. All points of $A' \setminus A$ lie outside its circumsphere, which implies that the same simplex is the unique 0-hefty simplex of $A' \setminus B$ that covers x. Therefore $B \subseteq A$ because the vertices of the simplex are in A, and all vertices we removed must come from A as well. So this is a k-hefty simplex of A.

"$\Longrightarrow$". Assuming $x \notin H_k(A)$, we modify the construction of A' to show that there is a set $A'' \supseteq A$ such that all k-hefty simplices of A are k-hefty simplices of A'' but at least one k-hefty simplex of A'' that covers x is not a k-hefty simplex of A. In contrast to A', the set A'' will be finite. Since $x \notin H_k(A)$, there are d points in A such that the open half-space bounded by the hyperplane passing through these points that contains x contains at most k points of A. Let Δ be the $(d-1)$-simplex spanned by the d points, write Δ^+ for the open half-space that contains x, and let $B \subseteq A$ be the points in Δ^+. By assumption, $\#B \leq k$. We get A'' by adding $1 + k - \#B$ points to A as follows. First, we add $k - \#B$ points in Δ^+ but outside all circumspheres of $d+1$ points in A. After that, we add a point $y \in \mathbb{R}^d$ so that the d-simplex that is the pyramid with apex y and base Δ covers x, and all other k points in Δ^+ are inside the circumsphere of this d-simplex. We also require that y is outside all circumspheres of $d+1$ points in A. It is clear that such a point y exists on a sufficiently large sphere that passes through all vertices of Δ. As proved above, there are at most $\binom{d+k}{d}$ k-hefty simplices of A'' that cover x. The pyramid with apex y and base Δ is such a simplex, but it is not a k-hefty simplex of A. Hence, the number of k-hefty simplices of A that cover x is strictly less than $\binom{d+k}{d}$, as claimed. ◀

▶ Remark. Observe that our arguments in part "$\Longleftarrow$" imply that once we fix a subset $B \subseteq A$ of size k, then the k-hefty simplices of A that have exactly these points inside their circumspheres have disjoint interiors. This is because they are 0-hefty simplices of $A \setminus B$.

2.4 Local Covering

By Theorem 2.2, the k-hefty simplices of a thin Delone set cover $\mathbb{R}^d$ without gap an integer number of times. However, beyond one dimension, it is generally not possible to split such a cover into sub-covers (separate layers) that enjoy the same property. As an example, consider the thin Delone set of which Figure 3 shows one point surrounded by five others. Assuming the remaining points are sufficiently far from the six, the point at the center is incident to five 1-hefty triangles, as shown. Globally, the 1-hefty triangles cover any small neighborhood of this point three times, but it is not possible to decompose this cover into three; see [9] for indecomposable coverings in other contexts. It is however possible to split it locally into two. Such a local split exists for every $k \geq 1$, and we prove that the hefty simplices incident to the point cover any small neighborhood a specific number of times.

▶ **Theorem 2.4.** *Let $A \subseteq \mathbb{R}^d$ be a generic thin Delone set and k a non-negative integer. Then the k-hefty simplices of A incident to a vertex $a \in A$ cover any sufficiently small neighborhood of a exactly $\binom{d+k-1}{d-1}$ times.*

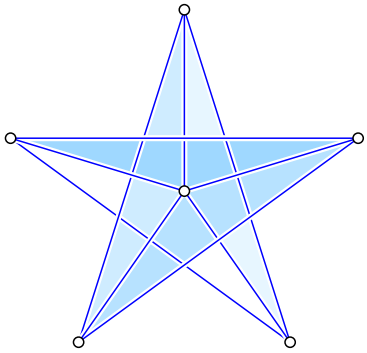


Figure 3 Six points of a Delone set, in which one is surrounded by five others, and the remaining points lie outside the circumcircles defined by these points. The five 1-hefty triangles incident to the surrounded point cover any small neighborhood of that point twice, but they cannot be split into two disjoint covers of such a neighborhood.

Proof. The case $k = 0$ is that of the Delaunay triangulation of A. Its top-dimensional simplices cover $\mathbb{R}^d$ exactly once, and thus also every neighborhood of $a \in A$ exactly once. We therefore assume $k > 0$. Let $A' = A \setminus \{a\}$. Observe that A' is also a generic thin Delone set and a is generic with respect to A'. We can therefore apply Theorem 2.2 to the $(k-1)$-hefty simplices of A' that contain $a \in \mathbb{R}^d$. There are only finitely many such simplices and each of them contains a small neighborhood of a.

Let y be a point in a sufficiently small neighborhood of a that is inside the intersection of all $(k-1)$-hefty simplices of A' that contain a, and assume that y is generic with respect to A. By Theorem 2.2, there are exactly $\binom{d+k}{d}$ k-hefty simplices of A that cover y. They split into simplices incident and not incident to a. The latter are the $(k-1)$-hefty simplices of A' that contain a. Indeed, since y is sufficiently close to a, a k-hefty simplex of A with vertices in A' contains a in its interior iff it contains y in its interior. The circumsphere of each such simplex encloses exactly $k-1$ points of A' as we removed a from A to get A'. This implies that the number of k-hefty simplices of A that contain y and are not incident to a is $\binom{d+k-1}{d}$. Thus, the number of k-hefty simplices of A that contain y and are incident to a is

$$\binom{d+k}{d} - \binom{d+k-1}{d} = \binom{d+k-1}{d-1}, \tag{1}$$

as claimed. ◀

Similar to Theorem 2.3, we establish an inequality for finite sets that follows from Theorem 2.4. We are satisfied with a less refined statement than formulated for the global case in Section 2.3.

▶ **Corollary 2.5.** *Let $A \subseteq \mathbb{R}^d$ be finite and generic and k be a non-negative integer. Then every generic point in a small neighborhood of any point $a \in A$ belongs to at most $\binom{d+k-1}{d-1}$ k-hefty simplices of A incident to a.*

3 Applications

We discuss two applications of Theorem 2.2: its relation to volumes of hypersimplices and Eulerian numbers in Section 3.1, and the connection to k-sets and k-facets in Section 3.2.

3.1 Worpitzky's Identity for Eulerian Numbers

As an application of our covering results, we show how the multiplicities from Theorem 2.2 are related to the volumes of hypersimplices and to Eulerian numbers. Specifically, we show that de Laplace's relation for hypersimplices [4] implies Worpitzky's identity for Eulerian numbers [14], and vice versa.

We begin by introducing the three main concepts we need in this subsection. Letting d be a positive integer, the number of *descents* in a permutation $j_1, j_2, \ldots, j_d$ of $1, 2, \ldots, d$ is the number of indices $1 \leq i \leq d-1$ such that $j_i > j_{i+1}$. For $0 \leq k \leq d-1$, the *Eulerian number* for d and k is the number of permutations of $1, 2, \ldots, d$ with exactly k descents. For example, $A(d,0) = A(d,d-1) = 1$, for every d, and $\sum_{k=0}^{d-1} A(d,k) = d!$. Less obvious is Worpitzky's identity [14] between two polynomials from more than a century ago:

$$\sum\nolimits_{k=0}^{d-1} A(d,k) \binom{x+k}{d} = x^d. \tag{2}$$

Next, let $x_0, x_1, \ldots, x_d$ be $d+1$ affinely independent points and write $\Delta = \operatorname{conv}\{x_0, x_1, \ldots, x_d\}$ for the d-simplex they span. A *p-fold sum* is obtained by selecting p of the $d+1$ points, considering them as vectors, and returning the point that corresponds to the sum of these vectors. For each $1 \leq p \leq d$, the convex hull of the p-fold sums of the $d+1$ points is a d-dimensional convex polytope referred to as a *d-hypersimplex of order p*, denoted $\Delta_d^{(p)}$. Clearly, $\Delta_d^{(1)} = \Delta$, and more generally, $\Delta_d^{(p)}$ is a homothetic copy of the convex hull of the barycenters of all $(p-1)$-dimensional faces of Δ. Since the barycenter of a $(p-1)$-simplex is $1/p$ times the sum of its vertices, the volume of that polytope is $1/p^d$ times the volume of the homothetic hypersimplex. Define the *relative volume* of $\Delta_d^{(p)}$ as $v(d,p) = \operatorname{vol}_d(\Delta_d^{(p)})/\operatorname{vol}_d(\Delta)$, and observe that it does not depend on the choice of Δ. Again more than a century ago, de Laplace [4] proved that these relative volumes are Eulerian numbers:

$$v(d,k+1) = A(d,k); \tag{3}$$

see also the combinatorial proof of the same equation by Stanley [11]. The third concept is the dual of the order-n Voronoi tessellation of a finite set $A \subseteq \mathbb{R}^d$, introduced in 1990 by Aurenhammer [2]. Referring to this dual as the *order-n Delaunay mosaic* of A, denoted $\operatorname{Del}_n(A)$, it is defined by its d-cells, each the convex hull of a collection of averages of n points selected from A. Specifically, for each $1 \leq p \leq d$ and every $(n-p)$-hefty simplex, take all sets of cardinality n that contain all $n-p$ points inside the circumsphere together with any p points on the circumsphere. E.g. for $p=1$, we get $d+1$ averages whose convex hull is a homothetic copy of the original d-simplex, and for $p=2$, we get a homothetic copy of the convex hull of the midpoints of the edges of the d-simplex. Collecting these polytopes, we get $\operatorname{Del}_n(A)$. For $n=1$, we have $\operatorname{Del}_1(A) = \operatorname{Del}(A)$, and more generally $\operatorname{Del}_n(A)$ has a d-cell for every $(n-p)$-hefty simplex in which p varies from 1 to d. Since the vertices are averages of n points, each d-cell in $\operatorname{Del}_n(A)$ has volume $1/n^d$ times the volume of the corresponding hypersimplex. It is now easy to prove the following relation for the relative volumes of the hypersimplices.

▶ **Theorem 3.1.** *For integers $d, n \geq 1$, the relative volumes of the hypersimplices satisfy*

$$\sum\nolimits_{p=1}^{d} v(d,p) \binom{n+d-p}{n-p} = n^d, \tag{4}$$

in which $\binom{n+d-p}{n-p} = 0$ whenever $n-p < 0$.

Proof. Let A be any Delone set – and not just a thin Delone set – in $\mathbb{R}^d$, so that every ball whose radius exceeds some given constant contains at least one point of A. Let $R > 0$ be sufficiently large and consider all d-cells in $\mathrm{Del}_n(A)$ that are contained in $[-R, R]^d$. Setting $n' = \max\{0, n-d\}$, there are $n - n' \leq d$ different types of d-cells to be considered, namely homothetic copies of hypersimplices of orders 1 to $n - n'$ defined by $(n-p)$-hefty simplices for $1 \leq p \leq n - n'$. The total volume of these d-cells is $(2R)^d + \mathcal{O}\left(R^{d-1}\right)$, since we miss only a constant width neighborhood of each facet of the hypercube.

Consider now an $(n-p)$-hefty simplex of A. Generically, its circumsphere passes through $d+1$ points and encloses $n-p$ of the points in A. By definition of Delaunay set, the radius of this circumsphere is bounded from above by a constant times $n-p$. Furthermore, the $(n-p)$-hefty simplex contains every d-cell in $\mathrm{Del}_n(A)$ it may determine since the vertices of the latter are averages of the vertices of the $(n-p)$-hefty simplex. By Theorem 2.2, the $(n-p)$-hefty simplices that define d-cells of $\mathrm{Del}_n(A)$ inside the hypercube therefore cover most of the hypercube exactly $\binom{n+d-p}{n-p}$ times. It follows that the total volume of these p-hefty simplices is $\binom{n+d-p}{n-p}(2R)^d + \mathcal{O}\left(R^{d-1}\right)$. By definition of relative volume, the total volume of the corresponding hypersimplices thus is $v(d,p)\binom{n+d-p}{n-p}(2R)^d/n^d + \mathcal{O}\left(R^{d-1}\right)$. Taking the sum for $1 \leq p \leq n - n'$, dividing by $(2R)^d$, and taking the limit as R goes to infinity, we get the claimed relation. ◀

To see that Theorem 3.1 together with de Laplace's relation implies Worpitzky's identity and together with Worpitzky's identity implies de Laplace's relation, we change the summation index in (4) from p to $k = d - p$ and apply $\binom{n+k}{n-d+k} = \binom{n+k}{d}$ to get

$$\sum_{k=0}^{d-1} v(d, d-k)\binom{n+k}{d} = n^d. \tag{5}$$

Since this relation holds for every positive integer, n, it also holds if we treat $\binom{n+k}{d}$ and n^d as polynomials of degree d in n. Substituting $A(d,k) = A(d, d-k-1)$ for $v(d, d-k)$ using de Laplace's relation (3), we get Worpitzky's identity (2). To see the other direction, we observe that the polynomials given by the binomial coefficients are linearly independent, so there is only one way to write n^d as their linear combination, and it is given by Worpitzky's identity. Comparing (5) with (2), we get $v(d, d-k) = A(d,k) = A(d, d-k-1)$, which is (3).

3.2 *k*-sets and *k*-facets

In this section, we briefly discuss connections between the covering numbers studied in Section 2 and the k-sets and k-facets studied in discrete geometry; see e.g. [8, Chapter 11]. Letting A be a generic set of n points in $\mathbb{R}^d$, a *k-set* is a set of k points, $B \subseteq A$, such that B and $A \setminus B$ can be separated by a hyperplane, and a *k-facet* is a set of d points, $\Delta \subseteq A$, such that the hyperplane passing through these points partitions $A \setminus \Delta$ into k and $n-d-k$ points on its two sides. We refer to [12, Section 2.2] for a discussion of the relation between these two concepts.

We present alternative proofs of two well-established results, which we state in terms of k-facets. Both proofs make use of the *inversion* of A through the unit sphere centered at a point $x \in \mathbb{R}^d \setminus A$, which maps a point $a \in \mathbb{R}^d \setminus \{x\}$ to $\iota_x(a) = x + (a-x)/\|a-x\|^2$. It is not difficult to see that the image under this inversion of a hyperplane that avoids x is a sphere that passes through x. Similarly, the image of the open half-space that does not contain x is the open ball bounded by this sphere. If the hyperplane passes through the points of a k-facet, $\Delta \subseteq A$, and separates x from the k-set on the other side, then the d-simplex spanned by x and the points in $\iota_x(\Delta)$ is a k-hefty simplex of $A' = \iota_x(A) \cup \{x\}$ incident to x.

Write $F_k(A)$ for the number of k-facets of A. We first reprove the following 2-dimensional result by Alon and Győri [1] for k smaller than a third of the number of points.

▶ **Proposition 3.2.** *Let A be a generic set of n points in $\mathbb{R}^2$ and $k < \frac{n}{3}$ a non-negative integer. Then $\sum_{j=0}^{k} F_j(A) \leq (k+1)n$.*

Proof. Recall that the k-hull of A is the intersection of all closed half-spaces that miss at most k points of A, denoted $H_k(A)$. By the Centerpoint Theorem of discrete geometry, $H_k(A)$ has a non-empty interior if $k < \frac{n}{3}$; see e.g. [6, Section 4.1]. Let x be a point in the interior of $H_k(A)$ and set $A' = \iota_x(A) \cup \{x\}$. Let $j \leq k$. As explained above, the inversion through the unit circle centered at x maps every j-facet of A to a j-hefty triangle of A' incident to x. By Corollary 2.5, the j-hefty triangles incident to x cover a small neighborhood of x at most $j+1$ times, so if we consider all j between 0 and k, we get the neighborhood of x covered at most $1 + 2 + \ldots + (k+1) = \binom{k+2}{2}$ times.

To continue, we draw a half-line emanating from x through every point in $\iota_x(A)$, thus splitting the neighborhood of x into n angles. Every hefty triangle incident to x covers a contiguous sequence of these angles, and for each $i \geq 1$, there are at most n triangles that cover exactly i of these angles. Each angle is covered some integer number of times, and we take the sum of these numbers over all angles. If $\sum_{j=0}^{k} F_j(A) > (k+1)n$, then this sum is strictly greater than $n(1 + 2 + \ldots + (k+1)) = n\binom{k+2}{2}$. This implies that one of the angles is covered more than $\binom{k+2}{2}$ times, which contradicts Corollary 2.5. ◀

We continue with Lovász Lemma – see [3] but also [8, Lemma 11.3.2] – which is a crucial ingredient in many arguments about k-sets; see e.g. [10]. This lemma gives an asymptotic upper bound on the number of $(d-1)$-simplices spanned by the points of k-facets a line can intersect. We give a short proof of a more general version by Welzl [13]. To state the lemma, we say a directed line, L, *enters* a k-facet, $\Delta \subseteq A$, if L intersects the $(d-1)$-simplex spanned by Δ at an interior point and moves from the side with k points to the side with $n-d-k$ points as it passes through the $(d-1)$-simplex.

▶ **Proposition 3.3.** *Let $A \subseteq \mathbb{R}^d$ be a generic finite set, k a non-negative integer, and L a directed line. Then L enters at most $\binom{d+k-1}{d-1}$ k-facets of A.*

Proof. We may assume that L passes through the convex hull of A and let $x \in L$ be a point reached after passing through $\operatorname{conv} A$. The inversion through the unit sphere centered at x maps every k-facet entered by L to a k-hefty simplex incident to x such that L passes through this simplex before it reaches x. By Corollary 2.5, there are at most $\binom{d+k-1}{d-1}$ such k-hefty simplices. ◀

Since a line can be directed in two different ways, the number of $(d-1)$-simplices spanned by k-facets of the set A one orientation of this line or the other can enter is at most $2\binom{d+k-1}{d-1}$.

4 Concluding Remarks

The results on covering numbers presented in Section 2 extend to other settings of which we mention three:

- locally finite sets in hyperbolic space;
- finite sets on the sphere;
- weighted points in Euclidean space.

A claim about global covering analogous to Theorem 2.2 holds in all three settings, while one about local covering analogous to Theorem 2.4 holds only for the first two. The first setting is most similar to the situation in Euclidean space studied in Section 2: the proofs are almost verbatim the same, with the main difference being a modified radial set since the design illustrated in Figure 2 is not thin Delone in hyperbolic space. For the second setting, there is an ambiguity in the definition of a k-hefty simplex since any $(d-1)$-sphere on the d-sphere bounds two complementary d-balls. We thus require that the finite set is *k-balanced*, by which we mean that every open hemisphere contains at least $k+1$ points. With this assumption, only the smaller of the two open d-balls has a chance to contain k of the points, so we consider it the inside of the $(d-1)$-sphere. The proofs of the extensions of Theorems 2.2 and 2.4 to k-balanced sets on the sphere are similar to those for thin Delone sets in Euclidean space. Alternatively, we may use stereographic projection to reduce the spherical to the Euclidean setting.

The generalization to points with real weights is akin the generalization of Voronoi to power tessellations or diagrams; see e.g. [6, Chapter 13]. We interpret a point with weight $w \in \mathbb{R}$ as a sphere with squared radius w (which may be negative), and we generalize the circumsphere of a d-simplex to the *orthosphere*, which in the generic case is the unique sphere orthogonal to the $d+1$ spheres that are the vertices of the d-simplex. The orthosphere *encloses* a weighted point if the two centers are strictly closer than required for the two spheres to be orthogonal to each other, and the d-simplex is *k-hefty* if it encloses exactly k of the weighted points. With these adaptations of the definitions, a claim analogous to Theorem 2.2 holds also in the weighted case, provided the weights satisfy a mild condition, which is for example satisfied if the weights are bounded. On the other hand, Theorem 2.4 fails to extend. Indeed, the k-hefty simplices incident to a weighted point cover the neighborhood of the point some integer number of times, but depending on the point and its surrounding, this integer varies between 0 and $\binom{d+k}{d}$.

References

1. Noga Alon and Ervin Győri. The number of small semispaces of a finite set of points in the plane. *Journal of Combinatorial Theory, Series A*, 41:154–157, 1986. doi:10.1016/0097-3165(86)90122-6.
2. Franz Aurenhammer. A new duality result concerning voronoi diagrams. *Discrete & Computational Geometry*, 5:243–254, 1990. doi:10.1007/BF02187788.
3. Imre Bárány, Zoltán Füredi, and László Lovász. On the number of halving planes. *Combinatorica*, 10:175–183, 1990. doi:10.1007/BF02123008.
4. Pierre-Simon de Laplace. *Œuvres complètes de Laplace, vol. 7 (1878–1912)*. Gauthier-Villars, 1886.
5. Boris Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskii i Estestvennykh Nauk*, 7:793–800, 1934.
6. Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.
7. Herbert Edelsbrunner and Georg Osang. The multi-cover persistence of euclidean balls. *Discrete & Computational Geometry*, 65:1296–1313, 2021. doi:10.1007/s00454-021-00281-9.
8. Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.
9. János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. In Jin Akiyama, Wen-Yang Chen, Mikio Kano, Xiaodong Li, and Qinglin Yu, editors, *Discrete Geometry, Combinatorics and Graph Theory*, volume 4381 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 2007. doi:10.1007/978-3-540-70666-3_15.

10 Natan Rubin. Improved bounds for point selections and halving hyperplanes in higher dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 4464–4501, 2024. arXiv preprint. doi:10.48550/arXiv.2403.00412.

11 Richard P. Stanley. Eulerian partitions of a unit hypercube. In Martin Aigner, editor, *Higher Combinatorics*, page 49. Reidel, 1977.

12 Uli Wagner. k-sets and k-facets. In Eli Goodman, János Pach, and Ricky Pollack, editors, *Discrete and Computational Geometry – 20 Years Later*, volume 453 of *Contemporary Mathematics*, pages 443–514. American Mathematical Society, 2008.

13 Emo Welzl. Entering and leaving j-facets. *Discrete & Computational Geometry*, 25:351–364, 2001. doi:10.1007/s004540010085.

14 Julius Worpitzky. Studien über die Bernoullischen und Eulerischen Zahlen. *Journal für die reine und angewandte Mathematik*, 94:203–232, 1883.

A Minor-Testing Approach for Coordinated Motion Planning with Sliding Robots

Eduard Eiben
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Robert Ganian
Algorithms and Complexity Group, TU Wien, Austria

Iyad Kanj
School of Computing, DePaul University, Chicago, IL, USA

M. S. Ramanujan
Department of Computer Science, University of Warwick, UK

Abstract

We study a variant of the Coordinated Motion Planning problem on undirected graphs, referred to herein as the Coordinated Sliding-Motion Planning (CSMP) problem. In this variant, we are given an undirected graph G, k robots $R_1, \ldots, R_k$ positioned on distinct vertices of G, $p \leq k$ distinct destination vertices for robots $R_1, \ldots, R_p$, and $\ell \in \mathbb{N}$. The problem is to decide if there is a serial schedule of at most ℓ moves (i.e., of makespan ℓ) such that at the end of the schedule each robot with a destination reaches it, where a robot's move is a free path (unoccupied by any robots) from its current position to an unoccupied vertex. The problem is known to be NP-hard even on full grids. It has been studied in several contexts, including coin movement and reconfiguration problems, with respect to feasibility, complexity, and approximation. Geometric variants of the problem, in which congruent geometric-shape robots (e.g., unit disk/squares) slide or translate in the Euclidean plane, have also been studied extensively.

We investigate the parameterized complexity of CSMP with respect to two parameters: the number k of robots and the makespan ℓ. As our first result, we present a fixed-parameter algorithm for CSMP parameterized by k. For our second result, we present a fixed-parameter algorithm parameterized by ℓ for the special case of CSMP in which only a single robot has a destination and the graph is planar. A crucial new ingredient for both of our results is that the solution admits a succinct representation as a small labeled topological minor of the input graph.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases coordinated motion planning on graphs, parameterized complexity, topological minor testing, planar graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.44

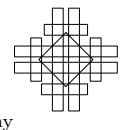
Related Version *Full Version*: https://arxiv.org/abs/2502.21175

Funding *Robert Ganian*: Projects No. 10.55776/Y1329 and 10.55776/COE12 of the Austrian Science Fund (FWF), Project No. 10.47379/ICT22029 of the Vienna Science Foundation (WWTF).
M. S. Ramanujan: Supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/V044621/1.

1 Introduction

In the coordinated motion planning problem, we are given an integer $\ell \in \mathbb{N}$, an undirected graph G, k robots $R_1, \ldots, R_k$ positioned on distinct starting vertices of G and $p \leq k$ distinct destination vertices for robots $R_1, \ldots, R_p$. In each step, one robot (or a subset of robots in other variants of the problem in which robots may move in parallel) may move from its

© Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan;
licensed under Creative Commons License CC-BY 4.0



current vertex to an adjacent unoccupied vertex. The goal is to decide if there is a schedule (i.e., a sequence of moves) of length at most ℓ, at the end of which each robot $R_1, \ldots, R_p$ has reached its destination vertex without colliding with other robots along the way. The problem and its variants (including the common variant where $p = k$) have been extensively studied, with respect to various motion types and graph classes, due to their ubiquitous applications in artificial intelligence (planning), robotics, computational geometry and more generally theoretical computer science [3, 6, 10, 11, 16, 17, 23, 24, 28, 31–38]. Moreover, it generalizes several well-known puzzles, including the famous $(n^2 - 1)$-puzzle and the Rush-Hour puzzle. Most of these variants are NP-hard, including the aforementioned (n^2-1)-puzzle [12,29]. Due to its numerous applications, the coordinated motion planning problem on (full rectangular) grids was posed as the SoCG 2021 Challenge [18].

We consider a variant of the coordinated motion planning problem on undirected graphs in which robots move serially one at a time, and a robot's move consists of a move along a free/unobstructed path (i.e., a path in which no other robot is located at the time of the move) as opposed to a single edge; we refer to such a move as a *sliding* move, analogous to the type of motion in geometric variants of the coordinated motion planning problem, where robots are geometric shapes (e.g., disks or rectangles) that move by translating/sliding in the plane. These geometric variants have been extensively studied since the 1980s [24, 28, 31–33]. The goal of this variant is to decide whether there is a schedule of makespan at most ℓ, where here the makespan of the schedule is simply the number of moves in it. We denote our general problem of interest as CSMP. We will also consider the special case of CSMP where G is planar and $p = 1$ – that is, only a designated robot has a destination and the remaining robots merely act as "blockers". We refer to this special case as Planar-CSMP-1.

Contributions and Techniques. We present a fixed-parameter algorithm for CSMP parameterized by the number k of robots, and a fixed-parameter algorithm for Planar-CSMP-1 parameterized by the makespan ℓ. A crucial common ingredient for both results is showing that the solution to a problem instance admits a succinct representation, whose size is a function of the parameter, as a topological minor of the input graph. To the best of our knowledge, this is a novel approach in the context of coordinated motion planning problems.

CSMP. Showing that the solution admits a succinct representation entails first bounding the makespan in an optimal solution by a function of the number k of robots. Towards this end, we preprocess the graph by shortening (to $\mathcal{O}(k)$) sufficiently long paths comprising degree-2 vertices and focus on two cases. In one case, where the graph has bounded[1] treedepth, we show that the instance can be reduced iteratively without altering the solution. This reduction process ultimately results in an instance of bounded size, implying the existence of a solution with a bounded makespan.

In the other case, we use ideas from [10] to define a notion of *havens*, which are subgraphs of bounded size with the following property: if a robot has to pass through a haven, then with a bounded number of additional steps, we can enable it to pass through the haven regardless of the other robots in the haven (i.e., without collision), effectively "untangling" all the robots' traversals within the haven. Building on this, we construct a schedule in which each robot interacts with only a bounded number of havens, ultimately leading to a bounded-makespan schedule.

[1] In the rest of this description, we simply say *bounded* to refer to values bounded by a function of k.

After bounding the makespan in an optimal solution, we show that it essentially describes a graph H of bounded size as a topological minor in the input graph. Here, we have to work with *rooted* topological minors since all robots have specified starting points and some have specified destinations. For us, the roots are just the starting and ending vertices, as we have boundedly-many roots. Towards our goal, we partition the vertices of the graph induced by the edges participating in an optimal solution into two sets: important and unimportant vertices. We show that the number of important vertices is bounded, and that the unimportant vertices form paths whose internal vertices have degree 2. This structure enables us to show that the graph induced by the edges in the solution (with terminals as roots) is a realization of some rooted graph with a bounded number of vertices as a topological minor in the input graph. We also prove the converse, namely that from *any* realization of H as a topological minor in G, one can obtain an optimal solution for the given instance. Towards proving this, we show that if a robot follows a path that is devoid of important vertices, then it slides on that path without stopping. Finally, we leverage known results on topological minor containment to verify the existence of the correct rooted graph.

Planar-CSMP-1. It is easy to see here that the solution admits a succinct representation since the parameter itself is the makespan. The difficulty is that the number of robots in the instance can be very large compared to the parameter ℓ, and we cannot find a realization of the succinct solution which avoids the "blocker" robots that do not contribute to the realization. To cope with this hurdle, we exploit the planarity of the graph to reduce its diameter, thus reducing the instance to an instance on a graph with bounded treewidth. The fixed-parameter tractability of the problem will then follow from Courcelle's Theorem [2, 8].

The heart of our reduction method is finding an "irrelevant edge" in the graph that can be safely contracted to produce an equivalent instance of the problem. While the method of finding an irrelevant edge and contracting it has been employed in several results in the literature to exclude large grids, thus reducing the treewidth of the graph, in our setting the grid approach seems unworkable. Instead, we show that a sufficiently-large component C of free vertices (i.e., vertices with no blockers on them) must contain an irrelevant edge. We reduce all the cases to one where C is formed by a skeleton consisting of a "long" path of degree-2 vertices (the degree is taken in C), plus a "small" number of additional vertices. We then show that if the instance is a YES-instance of the problem, then it admits a solution that interacts nicely with this skeleton, in the sense that we can identify a subgraph of the representation of the solution, i.e., a topological minor that is essentially a roadmap of the solution, separated from the rest of the solution by a small cut. We can enumerate each possible representation of the part of the solution that interacts with the skeleton, and for each possible representation, test whether it exists near the skeleton; if it does, we mark the vertices in the graph that realize this representation. Since the skeleton is long, the total number of marked vertices on the skeleton is small, and an edge with both endpoints unmarked – and hence can be contracted – must exist.

Related Work. The CSMP problem, with $p = k$, has been studied in [7] on several graph classes, and with respect to various settings, based on whether or not the destinations of the robots are distinguished (referred to as labeled or unlabeled). The graph classes that were considered include trees, grids, planar graphs, and general graphs, and it was shown that the problem is NP-hard even for (full rectangular) grids. Upper and lower bounds on the makespan of a solution, as well as computational complexity and approximation results, were derived for the various settings considered in [7].

A related problem to CSMP is that of moving unit disks (or objects [16]) in the plane, which has been studied in the context of reconfiguring/moving coins in the plane [1, 4], and was shown to be NP-hard [1]. We also mention the related work on coordinated pebble motion on graphs (and permutation groups), which was studied in [25] motivated by the work on the $(n^2 - 1)$-puzzle. The variant of coordinated motion planning in which only one designated robot is required to reach its destination, that is $p = 1$, and each move is along a single edge, was studied as Graph Motion Planning with 1 Robot in [27].

Despite the tremendous amount of work on coordinated motion planning problems, not much work has been done from the parameterized complexity perspective. The work in [10] studied the parameterized complexity of the non-sliding version of CSMP, that is, where each motion step is along a single edge, parameterized by the makespan in the serial setting. Another recent work [17] studied the parameterized complexity of the classical coordinated motion planning problem on grids, presenting parameterized algorithms for various objective measures (makespan, travel length). The work in [20] established the W[1]-hardness of the classical coordinated motion planning problem parameterized by the makespan in the parallel motion setting, and also showed the NP-hardness of this problem on trees, among other results. Finally, the parameterized complexity of a geometric variant of the PSPACE-complete Rush-Hour problem, which itself was shown to be PSPACE-complete [21], was studied [19]; in particular, that problem was shown to be FPT when parameterized by either the number of robots (i.e., cars) or the number of moves.

2 Terminology and Problem Definition

We assume basic familiarity with parameterized complexity theory, including *fixed-parameter tractability*, *parameterized reductions*, and the class W[1] [9, 15]. We also assume knowledge of standard graph-theoretic notions such as planarity, graph diameter and vertex degree [13]. We write $[n]$, where $n \in \mathbb{N}$, for $\{1, \ldots, n\}$.

Treewidth is a fundamental graph parameter which can be seen as a measure of how similar a graph is to a tree; trees have treewidth 1, while the complete n-vertex graph has treewidth $n - 1$. A formal definition of treewidth will not be necessary to obtain our results; we will make use of the fact that every planar graph with diameter r has treewidth at most $3r + 1$ [5, 30] and of Courcelle's Theorem [2, 8], which in particular provides a linear-time procedure to identify vertex subsets satisfying properties definable in *Monadic Second Order Logic* (MSO) on graphs of constant treewidth. We also use the notion of *treedepth* [26].

Rooted graphs. A rooted graph is a graph where some vertices are labeled – formally:

▶ **Definition 1** (Rooted graphs). *A* rooted graph *is a graph G with a set $\partial(G) \subseteq V(G)$ of distinguished vertices called* roots *and a labeling $\lambda_G : \partial(G) \to \mathbb{N}$. The set $\partial(G)$ is the* root set *of G, and the* label set *of G is $\Lambda(G) = \{\lambda_G(v) \mid v \in \partial(G)\}$.*

▶ **Definition 2** (Topological minors of rooted graphs [22]). *Let G and H be two rooted graphs with labelings λ_H and λ_G for their root sets. We say that H is a* topological minor *of G if there exist injective functions $\phi : V(H) \to V(G)$ and $\varphi : E(H) \to \mathsf{paths}(G)$ such that*

- *for every $e = \{h, h'\} \in E(H)$, the endpoints of $\varphi(e)$ are $\phi(h)$ and $\phi(h')$,*
- *for every distinct $e, e' \in E(H)$, the paths $\varphi(e)$ and $\varphi(e')$ are internally vertex-disjoint,*
- *there does not exist a vertex v in the image of ϕ and an edge $e \in E(H)$ such that v is an internal vertex on $\varphi(e)$, and*
- *for every $v \in \partial(H)$, $\lambda_H(v) = \lambda_G(\phi(v))$.*

We say that (ϕ, φ) *is a* realization *of* H *as a topological minor in* G*. Moreover, the subgraph of* G *induced by the union of the edges in the paths in* $\varphi(e)$ *for* $e \in E(H)$ *and the vertices in* $\phi(h)$ *for* $h \in V(H)$ *is called a* topological minor model *of* H*.*

Note that for $\partial(G) = \emptyset$ the above coincides with the usual definition of topological minors.

▶ **Proposition 3.** ([22]) *There is an algorithm that, given two rooted graphs* G *and* H*, runs in time* $f(|H|) \cdot \mathcal{O}(|G|^3)$ *for some computable function* f *and decides whether* H *is a topological minor of* G*.*

Problem Formalizations. In our problems of interest, we are given an undirected graph G and a set $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ of k robots where $\mathcal{R}$ is partitioned into two sets $\mathcal{M}$ and $\mathcal{F}$. Each $R_i \in \mathcal{M}$, has a starting vertex s_i and a destination vertex t_i in $V(G)$ and each $R_i \in \mathcal{F}$ is associated only with a starting vertex $s_i \in V(G)$. We refer to the elements in the set $\{s_i \mid i \in [k]\} \cup \{t_i \mid R_i \in \mathcal{M}\}$ as *terminals*. We assume that all the s_i are pairwise distinct and that all the t_i are pairwise distinct. We use a discrete time frame $[0, q]$, $q \in \mathbb{N}$, to reference the sequence of moves of the robots. In each time step $x \in [0, q]$, exactly one robot moves and the rest remain stationary.

In Coordinated Sliding-Motion Planning (CSMP), we are given a tuple $(G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$, where $G, \mathcal{R}$ and k are as described in the last paragraph and $\ell \in \mathbb{N}$. The goal is to decide whether there is a sequence of $q \leq \ell$ moves such that robots move serially one at a time, where a robot's move consists of a move along a free/unobstructed path (i.e., a path in which no other robot is located at the time of the move), and such that each $R_i \in \mathcal{R}$ starts in s_i at time step 0 and each $R_i \in \mathcal{M}$ is positioned on t_i at the end of time step q. The second problem that we consider is a restriction of CSMP to the case where G is planar and $\mathcal{M}$ contains exactly one robot, referred to as the *main* robot. We refer to this restriction as Planar-CSMP-1, and for simplicity represent its instances as tuples of the form $(G, s, t, \ell, \mathcal{R})$, where s and t are the starting and destination vertices of the main robot, respectively, and $\mathcal{R}$ is the set of all robots.

We next introduce some notation that will be used in our technical sections. A *route* for robot R_i is a tuple $W_i = (u_0^i, P_1^i, u_1^i, \ldots, P_q^i, u_q^i)$ where each u_j^i is a vertex in G and each P_j^i is a simple path in G, such that (i) $u_0^i = s_i$ and $u_q^i = t_i$ if $R_i \in \mathcal{M}$ and (ii) $\forall j \in [q]$, P_j^i is a u_{j-1}^i-u_j^i path in G (if $u_{j-1}^i = u_j^i$, then P_j^i is the singleton path comprising the same vertex).

Put simply, each W_i corresponds to a "walk" in G. If vertices of W_i at two consecutive time steps $j-1$ and j are identical, then j is a *waiting time step* for the robot R_i. Moreover, each R_i begins at its starting vertex at time step 0, and if $R_i \in \mathcal{M}$ then R_i is at its destination vertex at time step q. Though we work with undirected graphs and the paths described above are undirected paths, each path P_j^i has a natural *starting vertex* and *ending vertex* designated by the time step j.

▶ **Definition 4.** Two routes $W_i = (u_0^i, P_1^i, \ldots, P_q^i, u_q^i)$ and $W_j = (u_0^j, P_1^j, \ldots, P_q^j, u_q^j)$, where $i \neq j \in [k]$, are *non-conflicting* if $\forall r \in \{0, \ldots, q\}$, the path P_r^i is disjoint from u_{r-1}^j and the path P_r^j is disjoint from u_{r-1}^i. Otherwise, we say that W_i and W_j *conflict*.

In particular, in the above definition, for every $i \neq j \in [k]$ and $\forall r \in \{0, \ldots, q\}$, $u_r^i \neq u_r^j$.

A *schedule* S for $\mathcal{R}$ is a set of pairwise non-conflicting routes $W_i, i \in [k]$, during a time interval $[0, q]$, where exactly one robot moves in each time step. The *number of moves* in a route $W_i = (u_0^i, P_1^i, \ldots, P_q^i, u_q^i)$ or the number of moves of its associated robot is the number of time steps j such that $u_j^i \neq u_{j+1}^i$. The *total number of moves* (or *makespan*) in a schedule is the sum of the number of moves in its routes. A schedule of minimum total moves is called an *optimal schedule*.

A vertex v is *occupied* at time step i if some robot is located at v at time step i; otherwise, v is said to be *free* (or *unoccupied*). If the time step is not specified, it is assumed to be time step 0 (i.e., in the initial state). At each time step, a robot may either move to a different vertex, or stay at its current vertex. Finally, we always assume that the input graph is connected since otherwise, we can work on individual connected components.

Remarks on Feasibility and Constructiveness. When only considering feasibility, CSMP is equivalent to the version where each move is along a single edge instead of an unobstructed path. Hence, feasibility testing is linear-time solvable [10, 39] and we can work under the assumption that every instance of CSMP is feasible (otherwise, we can reject the instance).

3 FPT Algorithm for CSMP Parameterized by the Number of Robots

In this section, we show that CSMP is FPT parameterized by the number k of robots in the input. Note that we do not assume any restrictions on the input graph.

Our strategy has three high-level steps: (i) We first show that if a solution exists, then there is an optimal schedule whose makespan is bounded[2]. (ii) We then show that such a solution is essentially a realization of a rooted graph H of bounded size as a topological minor in a rooted graph G' obtained by assigning unique labels to the terminals in G. We also prove the converse, that is, from any realization of H as a topological minor in G', one can obtain an optimal solution for the given instance. (iii) Finally, we leverage known results on topological minor containment to verify the existence of the correct rooted graph. Let us now give more detail on Steps (i) and (ii).

Step (i): To prove the existence of a solution with boundedly many moves, we preprocess the graph by applying a reduction rule that shortens degree-2 paths to a bounded length. This simplification allows us to focus on two cases for the graph (assuming it is connected):

Case (a): Bounded Treedepth. If the graph has bounded treedepth, we show that instances with bounded treedepth and a bounded number of robots can be reduced iteratively without altering the solution. This reduction process results in an instance of bounded size, implying the existence of a solution with a bounded makespan.

Case (b): Long Paths. Suppose every vertex is the endpoint of a sufficiently long path. We use ideas from [10] to define a suitable notion of *havens* for our setting; in particular, havens are subgraphs of bounded size that can efficiently handle collisions. Here, by the "efficient handling" of collisions, we mean that given any pair of "configurations" of robots in a haven, we can move the robots from one configuration to another without any conflicts using only boundedly many moves. Thus, with a small number of time steps, a robot can pass through a haven without collision, regardless of other robots in the haven. Building on this fact, we construct a schedule where each robot interacts with only a bounded number of havens. Within each haven, the number of moves used by a robot is bounded and we show that outside the havens, the robot only makes boundedly many moves as well. This yields a bound on the makespan of the schedule.

Step (ii): Consider the graph induced by the edges in the solution. We partition the vertices of this graph into two sets: important and unimportant vertices. The important vertices are those with degree at least three, the terminals, and any vertex where a robot waits at any time step. We show that the number of important vertices is bounded by a function of k (say, $\zeta(k)$). Moreover, by definition, the unimportant vertices form paths with only degree-2 vertices as internal vertices. Additionally, every robot that visits a vertex in such a path

[2] We recall that by bounded, we mean "bounded by a function of the parameter" – in this case, k.

slides without stopping on any vertex of this path. This structure enables us to show that the graph induced by the edges in the solution (with terminals as roots) is a realization of some rooted graph with at most $\zeta(k)$ vertices as a topological minor in the input graph.

3.1 Bounding the Makespan in an Optimal Schedule

Our first task is to apply a reduction rule to the given instance that enforces useful structural properties on the graph without affecting the existence of a solution.

▶ **Reduction Rule 1.** *Let $I = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* CSMP*. If there is a path P in G such that every internal vertex of P is disjoint from the set of terminals and has degree exactly two in G, then shorten P to length $min\{|P|, 2k + 1\}$.*

We say that an instance I is *irreducible* if Reduction Rule 1 cannot be applied. In the rest of this section, we work with such instances. Our goal now is to show that every YES-instance has a schedule in which the makespan is bounded by a function of the number of robots.

▶ **Lemma 5.** *Let $I = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* CSMP*. If I is a* YES*-instance, then I has an optimal schedule with $2^{\mathcal{O}(k^4)}$ moves.*

The proof strategy of Lemma 5 is as follows. We show that for a YES-instance, either the treedepth of the input graph is bounded or a certain locality property holds (roughly speaking, every vertex is sufficiently close to a vertex of degree at least three). We then show that in either case, there is a solution with boundedly-many moves. Specifically, when the treedepth is bounded, we show how to reduce the graph to a bounded-size graph without affecting the existence of a solution (thus implicitly bounding the makespan of an optimal solution) and otherwise, we show how the aforementioned locality property can be used along with a greedy strategy to produce a schedule with boundedly many moves (again, bounding the makespan of an optimal solution).

3.2 Succinct Representation of a Solution

In what follows, fix an optimal schedule $S = \{W_i \mid i \in [k]\}$ for an instance $I = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$. By Lemma 5, we may assume, without loss of generality, that S has length $2^{\mathcal{O}(k^4)}$. We also assume that no robot moves in two consecutive time steps since otherwise, the movements of such a robot could be done in a single time step while the remaining robots wait. Moreover, we say that an edge e of G is *traversed by* S if there is an $i \in [k]$ and a time step j such that the path P^i_j contains the edge e. That is, some robot R_i uses the edge e in some time step j. We denote by G_S, the subgraph of G induced by the edges traversed by S. For a time step j, we denote by $G_{S,j}$ the subgraph of G_S induced by the edges traversed by S up to time step j. A vertex v in G_S is a *waiting vertex* within time interval $[0, j]$ if there is a robot R_i and a time step $j' \leq j$ such that R_i is at v at time steps $j' - 1$ and j. We say that v is a waiting vertex if it is a waiting vertex within some time interval. A vertex v in G_S is an *intersection* vertex within time interval $[0, j]$ if its degree in $G_{S,j}$ is at least three. We say that v is an intersection vertex if it is an intersection vertex within some time interval. A vertex v in $G_{S,j}$ is an *important* vertex within time interval $[0, j]$ if it is either a terminal vertex, or within time interval $[0, j]$ it is a waiting vertex or an intersection vertex. Otherwise, v is *unimportant* within time interval $[0, j]$. We say that v is an important (unimportant) vertex if it is important (resp. unimportant) within some time interval.

▶ **Observation 6.** *The number of waiting vertices within time interval $[0, j]$ is at most $k + j$.*

▶ **Observation 7.** *The unimportant vertices within any time interval $[0, j]$ induce a disjoint union of paths and any endpoint of any of these paths has degree 2 in $G_{S,j}$.*

For a time step j, we denote by $\mathcal{P}_{S,j}$, the set of all x-y paths P of non-zero length such that x and y are distinct important vertices within time interval $[0, j]$ and every internal vertex of P is an unimportant vertex within time interval $[0, j]$.

For two paths P and P' in G, their *crossing points* are those vertices that have degree at least three in the graph induced by the edges in $E(P) \cup E(P')$.

We say that S is a *special schedule up to time step* j if, for every robot R_i and $j > 0$ such that R_i moves in time step j, the path P^i_j has at most four crossing points with any path in $\mathcal{P}_{S,j'}$ where $j' \leq j - 1$. Note that every schedule is trivially special at time step 1 since only a single path has been taken at this point. Moreover, $\mathcal{P}_{S,1}$ has size at least one because a move has been made and so, $\mathcal{P}_{S,j}$ has size at least one for every j. We say that S is a *special schedule* if it is a special schedule up to the final time step.

We say that two schedules are *equivalent at time step* j if the positions of all the robots at the end of time j is the same in both schedules, that is, robot R_i is on vertex v after time step j in one schedule if and only if the same happens in the other schedule. We say that two schedules are equivalent if they are equivalent at every time step. In particular, two equivalent schedules take the same number of time steps.

▶ **Lemma 8.** *There is a solution* S' *that is a special schedule.*

Henceforth, we assume that the solution S is a special schedule.

▶ **Lemma 9.** *The number of important vertices in* G_{S} *is bounded by* $2^{2^{\mathcal{O}(k^4)}}$.

3.3 Leveraging Succinct Representations and Topological Minor Containment

In what follows, fix a schedule $S = \{W_i \mid i \in [k]\}$ for an instance $I = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$.

We define the *representation* of S to be the rooted graph (denoted by H_S) whose vertices are the important vertices of G_S, and in which there is an edge between two vertices u and v if and only if there is a u-v path in G_S whose internal vertices have degree 2 in G_S and are unimportant. Moreover, the terminals receive unique labels (assume that the terminals are labeled with $[p]$ for some $p \leq 2k$) and the remaining vertices receive the same label which is distinct from those of the terminals, say $p + 1$. An illustration is provided in Figure 1. From Lemma 9, we directly obtain:

▶ **Lemma 10.** *If* S *is a special schedule, then its representation has* $2^{2^{\mathcal{O}(k^4)}}$ *vertices.*

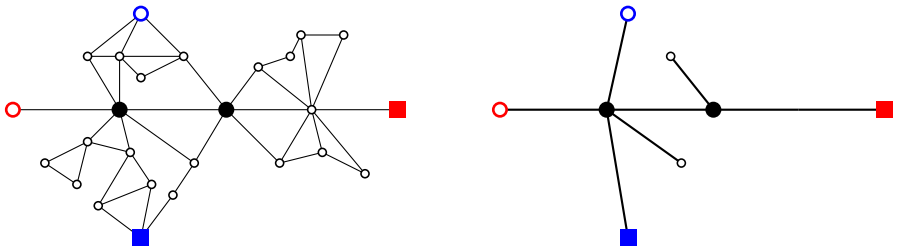


Figure 1 Left: an instance of CSMP with two robots with destinations (red and blue, with destinations marked as squares) and two blocker robots (both marked as solid black circles). Right: A representation of one particular optimal schedule for this instance, where we first move the two blockers and then the remaining two robots.

For the graph G, we let G' denote the rooted graph where the terminals get their unique labels and the rest of the vertices get a specific label $p+1$, where p is the number of terminals.

▶ **Lemma 11.** *H_S is a topological minor of G' and moreover, any realization of H_S as a topological minor in G' implies a schedule with length at most that of* S.

▶ **Theorem 12.** CSMP *is fixed-parameter tractable parameterized by the number of robots.*

Proof. For a hypothetical optimal solution S, we guess (i.e., use brute-force branching to determine) the representation H_S. By Lemmas 8 and 10, there are at most $2^{2^{2^{\mathcal{O}(k^4)}}}$ choices and these can be enumerated in FPT time. For each H_S, we do the following:

1. Decide whether the guess is feasible by brute-forcing over it. More precisely, we guess the configurations of the robots at each time step and verify that this guess corresponds to a set of non-conflicting routes with at most ℓ moves in total. Reject guesses that do not pass this check. Clearly, this step takes time $f(k)$ for some computable function f.
2. If H_S is not rejected, then invoke Proposition 3 to decide whether H_S is a topological minor of G' and return "yes" if and only if at least one of the invocations returns "yes".

Fixed-parameter tractability follows from the fact that Proposition 3 is used at most $f(k)$ times. The correctness follows from Lemma 10 and Lemma 11. ◀

4 An FPT Algorithm for Planar-CSMP-1 Parameterized by the Makespan

We start by giving a high-level description of the result. Our aim is to prove that Planar-CSMP-1 is fixed-parameter tractable parameterized by ℓ. The crux of our technique is showing that a sufficiently large component of free vertices must contain an edge that can be safely contracted. These contractions eventually result in bounding the size of each free component in the graph, and consequently the diameter and hence the treewidth of the graph (due to planarity). The fixed-parameter tractability of the problem will then be established using Courcelle's Theorem [2, 8].

To show that each large component C of free vertices contains an edge that can be safely contracted, we first show that this is the case unless C is formed by a skeleton consisting of a sufficiently long path of degree-2 vertices (the degree is taken in C), plus a sufficiently small number of additional vertices. We then focus on the skeleton of this component. We show that if the instance is a YES-instance of the problem, then it admits a solution S that interacts "nicely" with this skeleton. Recall from the previous section that a solution can be succinctly represented as a rooted topological minor of size $\sigma(\ell)$ for some computable function σ. This is because at most ℓ robots move in the solution. Now, we can identify a subgraph S of the representation of the solution (i.e., a topological minor that is essentially a roadmap of the solution), that is separated from the rest of the solution by a small cut. The realization of this subgraph uses most of the skeleton and the remaining part of this realization consists of small subgraphs that are close to the skeleton. Therefore, we can assume that all edges in these parts of the subgraphs are realized by edges in the graph.

Thus, it suffices to enumerate each possible representation of the part of the solution that interacts with the skeleton (which can be done in FPT-time), and test whether it exists near the skeleton; if it does, we mark the vertices in the graph that realize this representation. Since the skeleton is long, the total number of marked vertices on the skeleton is small, and an edge with both endpoints unmarked that can be contracted must exist. We remark that in many of the arguments that we make, we rely on the crucial fact that, except for the main robot, the robots are "indistinguishable" from each other. We now proceed to the details.

Refer to a vertex on which a robot other than the main robot is located as a *blocker* vertex. Let the *blocker-distance* between a vertex v and a vertex w, denoted $blockd(v, w)$, be the minimum integer p such that there is a v-w path in G containing p blocker vertices. The

reason why the blocker-distance is relevant is that we show that vertices which have high blocker-distance from s and t can be removed from the graph. Observe that, for any two vertices v, w, *blockd*(v, w) can be computed in polynomial time via, e.g., Dijkstra's algorithm.

Let $\mathcal{I} = (G, s, t, \ell, \mathcal{R})$ be an instance and let $F \subseteq V(G)$ be the set of free vertices in G. A *free component* is a connected component of $G[F]$. Free components will be central to our proof – in particular, observe that the blocker-distance between s and t is at most $\ell - 1$ and we can safely remove all vertices from G at blocker distance at least $\ell + 1$ from s. Hence, if there are no large free components, then the diameter of G is bounded. As G is planar, also its treewidth is bounded and we can use Courcelle's Theorem [2, 8] to prove the following:

▶ **Lemma 13.** Planar-CSMP-1 *can be solved in time $f(\ell, \lambda) \cdot |\mathcal{I}|$, where f is a computable function and λ is the size of the largest free component in the input instance $\mathcal{I}$.*

The Structure of Free Components. The following observation is straightforward:

▶ **Observation 14.** *Let Q be a path of free vertices and let x, y be arbitrary vertices on Q such that the subpath of Q between x and y has length at least ℓ. If there exists an x-y path P_{xy} whose internal vertices are disjoint from Q such that the number of blockers on P_{xy} is at most $\ell - \mathrm{blockd}(s, x) - \mathrm{blockd}(y, t) - 1$, then the instance $\mathcal{I}$ is a* YES*-instance and we can output a solution for $\mathcal{I}$ in polynomial time.*

We refer to a path where the above observation does not result in solving the instance as a *resilient* path. In other words, a path Q is resilient if and only if for every pair of vertices x, y in $V(Q)$ whose distance on Q is at least ℓ, there is no x-y path Z in $G - Q$ with at most $\ell - \mathit{blockd}(s, x) - \mathit{blockd}(y, t) - 1$ blockers. Our goal is to show that every "sufficiently large" free component contains a "still sufficiently long" path with certain special properties that, intuitively, allow the path to be "cleanly separated" from a hypothetical solution. This is formalized in the following definition and lemma, where the property of being "still sufficiently long" is given by the function $g(\ell) = 32\ell^6 \cdot 2^{14\ell^2} + 1$.

▶ **Definition 15.** *Let Q be a path in a free component C with endpoints u', v'. We say that Q is* ℓ-clean *if it satisfies the following conditions:*

1. *every vertex in Q has degree precisely 2 in C;*
2. *$|V(Q)| > g(\ell)$;*
3. *if $\mathcal{I}$ admits a solution that intersects $V(Q)$, then there are two vertices $u, v \in V(Q)$, each at distance at most $\ell^2 + 1$ from u' and v', respectively, and a solution S for $\mathcal{I}$ such that: $\{u, v\}$ is a cut in G_{S} between $\{s, t\}$ and the u-v subpath of Q; we denote by Q^{S}_{uv} the connected subgraph of G_{S} that contains u, v and all the connected components of $G_{\mathrm{S}} - \{u, v\}$ that do not contain s or t; and*
4. *Q is resilient.*

▶ **Lemma 16.** *There is a polynomial-time procedure which takes as input a free component C in an instance $\mathcal{I} = (G, s, t, \ell, \mathcal{R})$ such that $|V(C)| \geq (g(\ell) + 1) \cdot (\ell - 1) + 3(\ell + 2)$, and either solves the instance, or outputs an instance equivalent to $\mathcal{I}$ obtained by contracting an edge in C, or outputs an ℓ-clean path Q in C.*

4.1 Finding Irrelevant Edges

For the following considerations, it will be useful to proceed with a fixed choice of C and an ℓ-clean path Q. Next, we will show that we can make an even stronger assumption about the hypothetical solutions "passing through" Q.

▶ **Lemma 17.** *Assume that $\mathcal{I}$ admits a solution that intersects $V(Q)$. Then there exist vertices u, v each at distance at most ℓ^2+1 from u' and v', respectively, and a solution S^* for $\mathcal{I}$ satisfying properties 3-4 in Definition 15, and furthermore satisfying the following property 5: $|V(Q^{S^*}_{uv}) - V(Q)| \leq \ell$.*

For the following, we will refer to a solution satisfying properties 3-5 as (Q,u,v)-*canonical*, or simply Q-canonical in brief when the identities of u and v do not matter. From the properties of canonical solutions, we can directly establish the following:

▶ **Lemma 18.** *Let S^* be a Q-canonical solution. Then there is a $(2\ell^2)$-vertex rooted graph $U^{S^*}_{uv}$ (with roots u, v and* `occupied`*), such that $U^{S^*}_{uv}$ is a rooted topological minor of G (with roots u, v and* `occupied`*) and there is a realization (ϕ,ψ) of $U^{S^*}_{uv}$ that maps $U^{S^*}_{uv}$ to $Q^{S^*}_{uv}$. Moreover, waiting vertices are in $\phi(U^{S^*}_{uv})$ and for every edge $e \in E(U^{S^*}_{uv})$ either $\psi(e)$ is a single edge or a subpath of Q.*

Crucially, below we provide a procedure that can efficiently test for $U^{S^*}_{uv}$ as defined above.

▶ **Lemma 19.** *Let Q be an ℓ-clean u'-v' path and let u, v be the vertices specified in property 3 of Definition 15. Let $p \leq 2\ell$ and U be a connected p-vertex rooted graph with roots u, v and "*`occupied`*" such that U contains at most $\ell - \text{blockd}(s,x) - \text{blockd}(y,t) - 1$ occupied vertices. There is a procedure which runs in time $2^{\mathcal{O}(\ell^2)} \cdot |V(G)||V(Q)|$ and either outputs a realization (ϕ,ψ) of U which results in a subgraph $\Gamma \subseteq G$ such that:*

- *u,v separate Γ from u' and v', and*
- *for every edge $e \in E(U)$ either $\psi(e)$ is a single edge or a subpath of Q, or*

correctly determines that no such $(\Gamma,(\phi,\psi))$ exists.

Proof Sketch. As the first step, we use the ℓ-cleanness of Q to establish that if Γ exists, then it must have a "necklace-like" structure. In particular, such a Γ must consist of the u-v subpath of Q along with at most p connected subgraphs $\Gamma_1,\ldots,\Gamma_j$, each of size $p+p\ell$, each intersecting the u-v subpath of Q but otherwise being pairwise disjoint, and having pairwise distance at least $\ell+1$ on $G[Q]$. We further may assume that $\Gamma_1,\ldots,\Gamma_j$ are ordered based on their distance to u.

With this knowledge, we exhaustively branch to consider every possible choice of rooted subgraphs $\Gamma_1,\ldots,\Gamma_j$ (up to isomorphism) to construct a set of *signatures* of a hypothetical tuple $(\Gamma,(\phi,\psi))$; crucially, the number of such signatures can be upper-bounded by $\ell^{\mathcal{O}(\ell)}$. We can now show that to test whether there exists some tuple $(\Gamma,(\phi,\psi))$, it suffices to consider each signature individually and see if there is a "leftmost" Γ matching that signature, i.e., a Γ where the subgraphs $\Gamma_1,\ldots,\Gamma_j$ are pushed as close to u as possible (while adhering to the properties of the given signature). This then allows us to process the u-v subpath of Q from u to v and test, at each vertex x, whether it could be the "first vertex" of Γ_1. Once we find the leftmost such x, we can jump ahead by $\ell+1$ and proceed to Γ_2, and so forth.

A key point here is that since each of the subgraphs $\Gamma_1,\ldots,\Gamma_j$ has bounded size, it suffices to perform each such test on the $(p+p\ell)$-neighborhood of x. Since this graph is planar and has bounded diameter (by a function of ℓ), it also has bounded treewidth [5,30], thus allowing us to employ well-known subgraph-isomorphism testing algorithms [14] to determine whether each x can indeed be part of a specified subgraph Γ_i. We remark that performing this test on each vertex individually is critical: the subgraph of G induced on the neighborhood of all of Q may have arbitrarily large treewidth. ◀

We call the subgraph Γ identified above a *host* for the *roadmap* U.

▶ **Reduction Rule 2.** *Let Q be a given ℓ-clean u'-v' path. There is a reduction rule which enumerates:*

- *the family $\mathcal{U}$ of all connected rooted graphs (with roots u, v and* `occupied`*) containing at most $(2\ell^2)$-many vertices, and*
- *the set Z of all vertex pairs (u, v) at distance at most $\ell^2 + 1$ from the respective endpoints of Q,*

and then for each choice of $U \in \mathcal{U}$ and $(u, v) \in Z$, applies Lemma 19 to determine whether there is a host of U w.r.t. (u, v) on Q. If there is, we mark *every vertex in $\phi(U) \cap V(Q)$ where (ϕ, ψ) is the computed realization.*

Afterwards, if there is an edge e on Q with two unmarked endpoints, we contract e.

▶ **Lemma 20.** *Reduction Rule 2 is safe. Moreover, for each ℓ-clean path Q, it can be executed in time at most $2^{\mathcal{O}(\ell^2)} \cdot |V(G)| \cdot |V(Q)|$, and is guaranteed to perform a contraction if Q contains more that $32\ell^6 \cdot 2^{14\ell^2} + 1$ many vertices.*

▶ **Theorem 21.** Planar-CSMP-1 *is fixed-parameter tractable parameterized by ℓ.*

Proof. We begin by applying Lemma 16 to each free component in G of size at least $\big(g(\ell) + 1) \cdot (\ell - 1)\big) + 3(\ell + 2)$. If this solves the instance, we are done; otherwise, it is guaranteed to either produce an equivalent instance obtained by an edge contraction (in which case we repeat), or identify an ℓ-clean path Q of length more than $g(\ell) = 32\ell^6 \cdot 2^{14\ell^2} + 1$. In the third case, we apply Reduction Rule 2 to Q to produce an equivalent instance obtained by an edge contraction in Q – as guaranteed by Lemma 20 – and restart.

The exhaustive application of the above procedure either correctly solves the input instance, or results in an equivalent instance such that each free component in G has size at most $\big(g(\ell)+1)\cdot(\ell-1)\big)+3(\ell+2)-1$. At this point, we invoke Lemma 13 to solve the instance. The overall running time is upper-bounded by $2^{\mathcal{O}(\ell^2)} \cdot |\mathcal{I}|^{\mathcal{O}(1)} + \zeta(\ell) \cdot |\mathcal{I}|$, where the former term is the running time of the preprocessing steps and ζ is a computable function that results from the application of Lemma 13 (and specifically of Courcelle's Theorem within that lemma) on an instance with the aforementioned bound on the size of free components. ◀

5 Conclusion

In this paper, we designed FPT algorithms for two important variants of the Coordinated Motion Planning problem on undirected graphs, CSMP and Planar CSMP-1, in which robots move by sliding along unobstructed paths. Our results take a different perspective and contribute new insights to the design of FPT algorithms for coordinated motion planning problems on graphs, developing and employing new techniques that combine topological graph theory, fixed-parameter tractability and computational logic. Our work opens up several interesting questions pertaining to the parameterization by the makespan ℓ.

1. Is CSMP parameterized by the makespan W[1]-hard on general graphs?
2. Can Theorem 21 be generalized to solve CSMP on planar graphs?
3. What is the parameterized complexity of CSMP and CSMP-1 on grids where each sliding-move/path is either horizontal or vertical?

References

1 Manuel Abellanas, Sergey Bereg, Ferran Hurtado, Alfredo García Olaverri, David Rappaport, and Javier Tejel. Moving coins. *Computational Geometry*, 34(1):35–48, 2006. doi:10.1016/J.COMGEO.2005.06.005.

2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991. doi:10.1016/0196-6774(91)90006-K.

3 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 *SoCG*, volume 224, pages 12:1–12:16, 2022. doi:10.4230/LIPIcs.SoCG.2022.12.

4 Sergey Bereg, Adrian Dumitrescu, and János Pach. Sliding disks in the plane. *International Journal of Computational Geometry & Applications*, 18(05):373–387, 2008. doi:10.1142/S0218195908002684.

5 Therese Biedl. On triangulating k-outerplanar graphs. *Discret. Appl. Math.*, 181:275–279, 2015. doi:10.1016/J.DAM.2014.10.017.

6 Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Solomon Eyal Shimony. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In *IJCAI*, pages 740–746, 2015. URL: http://ijcai.org/Abstract/15/110.

7 Gruia Călinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM Journal on Discrete Mathematics*, 22(1):124–138, 2008. doi:10.1137/060652063.

8 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 A. Deligkas, E. Eiben, R. Ganian, I. Kanj, and M. S. Ramanujan. Parameterized algorithms for coordinated motion planning: Minimizing energy. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 53:1–53:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.53.

11 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM Journal on Computing*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.

12 Erik D. Demaine and Mikhail Rudoy. A simple proof that the $(n^2 - 1)$-puzzle is hard. *Theoretical Computer Science*, 732:80–84, 2018.

13 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

14 Frederic Dorn. Planar subgraph isomorphism revisited. In Jean-Yves Marion and Thomas Schwentick, editors, *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, volume 5 of *LIPIcs*, pages 263–274. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPICS.STACS.2010.2460.

15 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

16 Adrian Dumitrescu. Motion planning and reconfiguration for systems of multiple objects. In Sascha Kolski, editor, *Mobile Robots*, chapter 24. IntechOpen, Rijeka, 2007.

17 Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *Proceedings of the 39th International Symposium on Computational Geometry*, volume 258 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.28.

18 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The CG: SHOP challenge 2021. *ACM Journal on Experimental Algorithmics*, 27:3.1:1–3.1:12, 2022. doi:10.1145/3532773.

19 Henning Fernau, Torben Hagerup, Naomi Nishimura, Prabhakar Ragde, and Klaus Reinhardt. On the parameterized complexity of the generalized rush hour puzzle. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 6–9, 2003. URL: http://www.cccg.ca/proceedings/2003/22.pdf.

20 Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. Exact algorithms and lowerbounds for multiagent pathfinding: Power of treelike topology. *CoRR*, abs/2312.09646, 2023. doi:10.48550/arXiv.2312.09646.

21 Gary W. Flake and Eric B. Baum. Rush hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2):895–911, 2002. doi:10.1016/S0304-3975(01)00173-6.

22 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488. ACM, 2011. doi:10.1145/1993636.1993700.

23 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.

24 Iyad Kanj and Salman Parsa. On the parameterized complexity of motion planning for rectangular robots. 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 65:1–65:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.65.

25 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

26 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

27 Christos H. Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph (extended abstract). In *STOC*, pages 511–520, 1994. doi:10.1109/SFCS.1994.365740.

28 Geetha Ramanathan and Vangalur S. Alagar. Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal obstacles. In *ICRA*, volume 2, pages 514–522, 1985. doi:10.1109/ROBOT.1985.1087248.

29 Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.

30 Neil Robertson and Paul D. Seymour. Graph minors. III. planar tree-width. *J. Comb. Theory B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

31 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2:46–75, 1983.

32 Jacob T. Schwartz and Micha Sharir. On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983.

33 Jacob T. Schwartz and Micha Sharir. On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298–351, 1983.

34 Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. doi:10.1016/J.ARTINT.2014.11.006.

35 Irving Solis, James Motes, Read Sandström, and Nancy M. Amato. Representation-optimal multi-robot motion planning using conflict-based search. *IEEE Robotics Automation Letters*, 6(3):4608–4615, 2021. doi:10.1109/LRA.2021.3068910.

36 Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015. doi:10.1016/J.ARTINT.2014.11.001.

37 Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, pages 1443–1449, 2013. doi:10.1609/AAAI.V27I1.8541.

38 Jingjin Yu and Steven M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016. doi:10.1109/TRO.2016.2593448.

39 Jingjin Yu and Daniela Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *WAFR*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 729–746, 2014. doi:10.1007/978-3-319-16595-0_42.

Higher Order Bipartiteness vs Bi-Partitioning in Simplicial Complexes

Marzieh Eidi[1]
Center for Scalable Data Analytics and Artificial Intelligence, Leipzig University, Germany
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

Sayan Mukherjee
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
Center for Scalable Data Analytics and Artificial Intelligence, Leipzig University, Germany
Duke University, Durham, NC, USA

Abstract

Bipartite graphs are a fundamental concept in graph theory with diverse applications. A graph is bipartite iff it contains no odd cycles, a characteristic that has many implications in diverse fields ranging from matching problems to the construction of complex networks. Another key identifying feature is their Laplacian spectrum as bipartite graphs achieve the maximum possible eigenvalue of graph Laplacian. However, for modeling higher-order connections in complex systems, hypergraphs and simplicial complexes are required due to the limitations of graphs in representing pairwise interactions. In this article, using simple tools from graph theory, we extend the cycle-based characterization from bipartite graphs to those simplicial complexes that achieve the maximum Hodge Laplacian eigenvalue, known as disorientable simplicial complexes. We show that a N-dimensional simplicial complex is disorientable if its down dual graph contains no simple odd cycle of distinct edges and no twisted even cycle of distinct edges. Furthermore, we see that in a N-simplicial complex without twisting cycles, the fewer the number of (non-branching) simple odd cycles in its down dual graph, the closer is its maximum eigenvalue to the possible maximum eigenvalue of Hodge Laplacian. Similar to the graph case, the absence of odd cycles plays a crucial role in solving the bi-partitioning problem of simplexes in higher dimensions.

2012 ACM Subject Classification Mathematics of computing → Combinatorics

Keywords and phrases Bipartite graphs, Simplicial complex, Disorientability, Hodge Laplacian, odd cycles, Twisted cycles, down-dual graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.45

Related Version *Full Version*: `https://arxiv.org/abs/2409.00682`

Funding *Sayan Mukherjee*: would like to acknowledge partial funding from NSF DMS 17-13012, NSF 30 BCS1552848, NSF DBI1661386, NSF IIS 15-46331, NSF DMS 16-13261, as well as the high-performance computing partially supported by grant 2016-IDG-1013 from the North Carolina Biotechnology Center as well as the Alexander von Humboldt Foundation, the BMBF and the Saxony State Ministry for Science.

Acknowledgements We would like to thank the anonymous reviewers for their comments which helped us to improve the manuscript. The first author also thanks Jürgen Jost for his valuable feedback on the first draft of this paper.

[1] corresponding author

© Marzieh Eidi and Sayan Mukherjee;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Bipartite graphs are useful tools in diverse domains, from matching problems and coding theory to social networks and biomedical applications such as cancer detection[1, 18, 15]. Exploring the properties of bipartite graphs, and their applications not only has enriched graph theory but also has opened the door to innovative solutions in challenging applications [1]. In the past few years, there has been a trend in understanding cycles in networks in terms of function, dynamics, and synchronizability [10]. Cycles and bipartite graphs are closely related. A graph is bipartite if and only if it has no odd cycles. Cycle-based characterization of bipartite graphs is essential in various domains, such as clustering, coloring and matching problems as well as complex network construction and analysis where detecting quasi-bipartite clusters, i.e. graphs with a few number of even cycles, is needed [7, 8]. Moreover, this characterization as well as some of its applications have been extended to directed graphs where a strongly connected digraph is bipartite if and only if it has no cycle of odd length [12].

One the other side, one can test for a graph being bipartite by observing the spectrum of the graph Laplacian, a matrix computed based on the incidence relations between vertices and edges of the graph. For the normalized Laplacian, the graph is bipartite if and only if two is the maximum eigenvalue of its Laplacian. Consider the spectrum of the normalized Laplacian of an unweighted graph in increasing order, $0 = \lambda_1 \leq ... \leq \lambda_n \leq 2$; the graph is connected if and only if λ_2 is bigger than zero and is bipartite iff λ_n is equal to two. While the down side of the spectrum tells the number of connected components (i.e. graph's topology), the up side reveals bipartiteness. Both the top and bottom eigenvalues then can help us to partition the vertices of the graph into two sets in two different ways; if λ_2 is positive but small we can partition the vertices into two sets with very few edges connecting the sets while elements of each set are highly connected to each other and they form clusters. And when λ_n is close to two we can partition the set of the vertices into two sets in a way that for each of these sets, there are few connections, and almost all of the edges, connect the vertices of one set to the other.

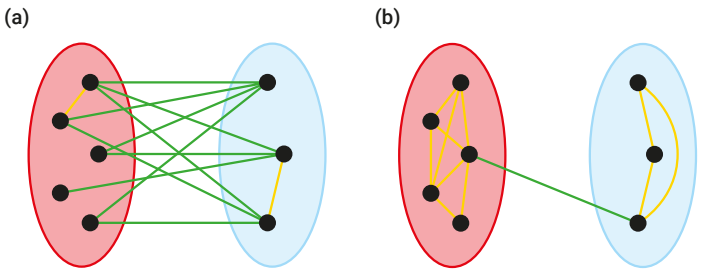


Figure 1 Partitioning vertices based on the Laplacian spectrum. In (a) vertices are partitioned based on λ_n and in (b) based on the λ_2.

Many empirical networks and complex systems however incorporate higher-order relations between elements and therefore are modeled as hypergraphs and/or simplicial complexes, rather than graphs. Simplicial complexes are generalizations of graphs where there are not only vertices and edges but also possibly triangles, tetrahedrons, and so on. To develop a systematic tool for the structural analysis of simplicial complexes, different methods, and theories have been extended from graphs to these higher-order structures. An important

question is that can the graph partitioning scheme that we described, be generalized to simplicial complexes? The main starting point to answer this question is the discrete "Hodge" Laplacian which is a generalization of the graph Laplacian for simplicial complexes. There have been many advances in our understanding of the spectrum of discrete Hodge Laplacian on simplicial complexes in the past few years [17, 11, 9]. Moreover, Laplacian-based methods have become popular for detecting the structure and dynamics of complex networks modeled by simplicial complexes in the past few years [3]. It is well known, due to Eckmann [5], that for any fixed d where $N \geq d \geq 0$ (N is the dimension of the complex), the minimum eigenvalue of the d-Laplacian can tell us about the topology of the complex in dimension d. This theorem has led to many theoretical findings regarding the minimum eigenvalues of the Laplacian on simplicial complexes [11, 9] as well as clustering methods for d-simplexes in such a way that the clusters represent the topology of the complex in dimension d [4, 16]. This is a generalization of what was described at the beginning based on the down side of the Laplacian spectrum for clustering the vertices of a graph. However, there is very little known about the maximum eigenvalue of the Laplacian and specifically the higher dimensional analogues for bipartiteness. A higher-order notion of bipartiteness called "disorientability", was introduced in [14] as a structure that the spectrum of the (up) Hodge Laplacian achieves its maximum possible value; the graph is the simplest setting and a one-dimensional simplicial complex is disoriantable if and only if it is bipartite. It is known that simple random walks on the vertices of a connected bipartite graph are periodic. It has been shown that the same is true when the random walks are on higher dimensional simplexes of a general simplicial complex and such random walks are periodic if and only if the complex is disorientable [14, 13, 6]. Random walks on graphs and simplicial complexes are the main tools in tackling diverse complex real-world problems from ranking web pages in Google page rank algorithm [2] to signal processing and flow network decomposition [16]. Considering the numerous theoretical implications and practical applications that bipartite graphs have, it is very desirable to develop our understanding of disorientability in simplicial complexes not just in terms of the spectrum of the Laplacian, but in terms of the combinatorial structure and shape of the complex and the parity of length of higher order cycles and see if we can have similar simplified characterization for recognizing whether a simplicial complex is disorientable and does disorientability give some kind of bi-portioning of the simplicies of higher dimensions in a similar manner like bipartite graphs? Similarly, can we detect quasi-bipartite clusters in higher dimensions? Since more data sets and some complex problems are nowadays represented by simplicial complexes and not graphs it seems necessary to develop the applied partitioning methods for partitioning the simplexes of dimensions higher than zero and explore further the applications, and devise algorithms. In this manuscript, we state conditions under which the simplicial complex is disorientable, in terms of the length of the higher dimensional cycles in the complex. Consequently, any simplicial complex can become disorientable by a finite number of splittings of some of the maximum dimensional simplexes; note that this is very similar to the graph case where by dividing some of the edges into two, and playing with the parity of the length of their cycles, we can always make the graph bipartite. And the fewer number of splittings needed to make the graph bipartite, the closer the maximum eigenvalue would be to the possible maximum eigenvalue of the graph Laplacian. We elaborate this for graphs and simplicial complexes in more details in the next sections (Lemma 19 and Theorem 20).

To the best of our knowledge, this is the first full characterization of disorientability from the combinatorial structure of the higher-dimensional cycles in simplicial complexes.

First, we recall some preliminary notions.

2 Background

▶ **Definition 1** (Simplicial complex). *A simplicial complex K on a vertex set $V = \{v_1, ..., v_n\}$ consists of a collection of simplices, that is, subsets of V with the requirement that all these subsets are closed under inclusion. A subset that contains $d+1$-vertices, is called a d-simplex. A 0-simplex is simply a vertex, a 1-simplex is an edge, and a 2-simplex is a triangle. The dimension of the complex is the maximum d, where we have at least one d-simplex in K.*

For computational purposes, we need to define an orientation for each d-simplex when $d \geq 1$.

▶ **Definition 2** (Oriented Simplex). *An orientation of a d-simplex is an equivalence class of orderings of its vertices, where two orderings are equivalent if they differ by an even permutation. For simplicity, we can choose the reference orientation of the simplices induced by the ordering of the vertex labels.*

A 0-simplex (a node) can have only one orientation. Hence, issues of orientation do not arise in graph-theoretic settings. Let S_d be the set of all d-simplexes in K with $0 \leq d \leq N$ and $[S_d]$ be the set of all oriented d-simplexes. For any d-simplex σ ($d \neq 0$) we have two opposite orientations, clockwise and counterclockwise (note that both are in $[S_d]$). A face of a d-simplex σ is a subset of σ with cardinality d, i.e., with one element of σ omitted. If ρ is a face of a d-simplex σ_d, then σ_d is called a co-face of ρ_{d-1}. The degree of a d-simplex is the number of its co-faces (i.e., of dimension $d+1$). A d-simplex is called branching if its degree is bigger than 2.

▶ **Definition 3** (Cycle, Simple and twisted Cycles). *A d-cycle of length l ($0 \leq d \leq N-1$) is a chain of d simplexes, $\sigma_d^0 ..., \sigma_d^l$ such that for each i, σ^i and σ^{i+1} are upper adjacent (i. e. they share a co-face) and $\sigma^0 = \sigma^l$. A d-cycle is called non-twisted (or simple) if there exist an assignment of orientations on its $d+1$-simplexes such that the induced orientation of every two $d+1$-simplex on their common d-face cancel each other (namely they are opposite). Otherwise we call it twisted. We call the simplicial complex non-twisting if it does not have a twisted cycle and otherwise, we call it twisting.*

▶ **Example 4.** A discrete cylinder and torus with respectively simple 0-cycle(s) and 1-cycle(s) are examples of non-twisting simplicial complexes and discrete Mobius strip and Klein bottle are twisting as they have respectively a 0-twisting cycle and a 1-twisting cycle (see Fig 4).

Note that graphs can not have twisted cycles and this just can happen in higher dimensions. Twisting cycles do not naturally arise in geometric and topological data analysis (TDA) when constructing Simplicial complexes from a set of data points. However, we present it here for the sake of completeness of our characterization.
We can extend face/co-face relations to the oriented d-simplexes with the help of the boundary/coboundary operators. The d-th chain group $C_d(K)$ of K is a vector space with real coefficients with the basis S_d.

▶ **Definition 5** (Boundary and co-boundary matrices). *The* boundary map $\partial_d : C_d(K) \to C_{d-1}(K)$ *is a linear operator from oriented d-simplices to oriented $(d-1)$-simplices defined by*

$$\partial_d[i_0, ...i_d] = \sum_{j=0}^{d} (-1)^j [i_0, .i_{j-1}, i_{j+1}..i_d]. \tag{1}$$

After choosing a basis for C_d, the boundary operator ∂_d can be represented by a matrix B_d, which enables us to simply perform computations via matrix calculus. For graphs, the matrix B_1 is the node-to-edge incidence matrix. Likewise, the higher-order boundary maps can be interpreted as higher-order incidence matrices between simplices and their (co-)faces for each B_i. The transpose of the boundary matrix B_i denoted by B_i^T represents the co-boundary map $\partial_d^T : C_d \to C_{d+1}$. This enables us to define (higher order) Laplacian:

▶ **Definition 6** (Hodge Laplacian). *We can define the d-th discrete (combinatorial) Hodge Laplacian as*

$$L_d = B_d^T B_d + B_{d+1} B_{d+1}^T. \tag{2}$$

For $d = 0$, $L_0 = B_1 B_1^T$ and for $d = N$, $L_N = B_N^T B_N$. $B_d^T B_d$ and $B_{d+1} B_{d+1}^T$ are respectively called the down and up Laplacians.

▶ Remark 7. For every d the non-zero spectrum of L_d^{up} equals to the non-zero spectrum of L_{d+1}^{down} [9].

We can extend Laplacian definitions and properties to arbitrary positive weight functions on the set of simplices (note that in the above case all the weights are one); another special case is the "normalized Laplacians" where for each simplex σ such that σ is not a facet, the degree of σ equal to its weight and the weights of all the facets are one.
The Laplacian spectrum and in particular its maximum eigenvalue can be obtained using the Rayleigh quotient. For a graph G we have:

$$\lambda_n = Sup_f \frac{\sum_{u \sim v} (f(u) - f(v))^2}{\sum_v f^2(v) d(v)}$$

where λ_n is the maximum eigenvalue of the normalized Laplacian and the supremum is taken over real-valued functions on the vertex set and $d(v)$ is the degree of the vertex v. This formula can be extended to weighted graphs as well as higher dimensions. For a N-dimensional simplicial complex $\lambda_n \leq N + 1$ [11].
We can create dual graphs of simplicial complexes based on the upper adjacency and/or lower adjacency connections between the simplexes. For this purpose, we consider a N-simplicial complex K and we fix a number d ($0 \leq d \leq N$).

▶ **Definition 8** (Dual graphs). *The up-dual graph G of K in dimension d is constructed as follows: each d-simplex in K becomes a vertex in the dual graph G, and there is an edge between two vertices in G if their corresponding d-simplexes in K share a co-face. Similarly, the down-dual graph G of K in dimension d is a graph G that its vertices are the d-simplexes of K but there is an edge between two vertices in G if their corresponding d-simplexes in K share a face. Particularly, the down-dual graph of a graph is called a line graph. We call a N-complex d-connected if the up dual graph in dimension d is connected.*

There are two types of generalizations for bipartiteness for a N-simplicial complex; a notion which has been investigated more, as it is connected to random walks [14], is called disorientability.

▶ **Definition 9** (Higher order bipartiteness). *A disorientation of a N-complex K is a choice of orientations of its N-simplexes, that whenever two arbitrary N-simplexes intersect in a $(N-1)$-simplex, they induce the same orientation on it (see Fig.2.). If K has a disorientation it is said to be disorientable. A graph is disorientable iff it is bipartite [14].*

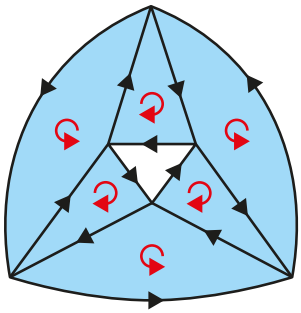


Figure 2 Example of a 2-d disorientable simplicial complex.

Another natural analogue is "$(d+1)$-partiteness": having some partition $A_0, ..., A_d$ of V so that every d-simplex contains one vertex from each A_i. A $(d+1)$-partite complex is seen to be disorientable, but the opposite does not necessarily hold for $d \geq 2$ [14]. Since this second case is not related to the spectrum of the Laplacian and periodicity of random walks and does not have the theoretical implications that disorientability has we will omit it here.

3 Main Results

To explore disorientable N-dimensional simplicial complexes we use their down-dual graphs and we start with graphs as 1-d complexes. As already mentioned in [14] bipartite graphs are disorientable and in fact, these two are equivalent in graphs. We elaborate on this from a signs perspective and the line graph of a graph.
If a graph is disorientable, there is a choice of orientations on its edges that adjacent edges induce the same orientation on their common vertex. Equivalently, there is an assignment of $\pm$ to vertices (note that these are just signs and not orientations) such that no two adjacent vertex have the same sign, namely we have a bipartite graph where each partition is labeled with one of these signs representing the head/tails of the oriented edges. Now let's look at the line graph of a bipartite graph. We have the following simple observation:

When going to line graphs of a general graph, its cycles and the parity of their length are preserved. But we might get some more cycles in the line graph that did not exist before due to the existence of branching vertices, i.e. those vertices that have degree bigger or equal than 3. We call these cycles branching cycles as they correspond to the branching vertices in the original graph. For any such vertex v with degree $\alpha \geq 3$ we would have a sequence of cycles with lengths α, $\alpha - 1$,.., 3 in the line graph. So if there exists no branching vertex, there is a one-to-one correspondence between the cycles in the graphs and its dual (line graph). Therefore if the graph is bipartite and non-branching its line graph is also bipartite.

Therefore a non-branching disoientable graph is a graph where we can assign $+$ and $-$ to the edges of its line graph such that no two adjacent edges have the same sign. But what if the bipartite graph has some branching vertices? In this case, the line graph is not bipartite and has (fundamental) odd cycles that for every such cycle, all of its edges relate to the same corresponding branching vertex. We note that if we consider general (not necessarily fundamental) cycles the existing odd cycles include more than one edge corresponding to that branching vertex. From graph theory, recall that a fundamental cycle is a cycle formed by combining a path within a spanning tree with an edge not in that tree.

Therefore, the graph is disorientable iff it is bipartite or equivalently iff its line graph has no odd cycle of distinct edges, namely they correspond to different vertices in the original graph. In the language of signs, a graph is disorientable iff in its line graph, we can assign $+$ and $-$ to the edges such that no two adjacent edges that correspond to different vertices in the original graph, get the same sign.

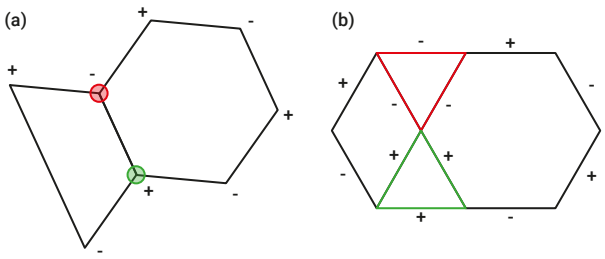


Figure 3 A bipartite graph and its line graph. (a) is bipartite and (b) is the line graph of (a) and has two odd cycles of length 3 corresponding to the branching red and green vertices in (a).

This simple but fundamental change of perspective from graphs to their line graphs helps us to explore bipartiteness in general simplicial complexes. We start with the simplest cases: non-branching and non-twisting simplicial complexes and we gradually develop the idea for the general case.

1. If the simplicial complex is non-branching and non-twisting:

▶ **Lemma 10.** *A non-branching, non-twisting simplicial complex is disorientable if and only if in its down dual graph, we can assign + and - to the edges in such a way that no two adjacent edges have the same sign. We note that this is equivalent to the condition that the down-dual graph has no odd cycle which is the same as bipartiteness.*

Proof. If the simplicial complex is disorientable, by definition there is an assignment of orientations on N-simplexes such that they induce the same orientation on their common face. Therefore, in its corresponding down dual graph, we would be able to have a single assignment of $+/-$ to each edge representing the induced orientations on the common $(N-1)$-faces (in the simplicial complex) such that no two adjacent edges get the same sign. This means that the down-dual graph has no odd cycle. Therefore it is bipartite. For the reverse, if there is such assignment of $+/-$ to the edges of the down-dual graph, by definition the simplicial complex is disorientable as each such sign for each edge can be interpreted as the unique induced sign on the common $(N-1)$-simplex (from its oriented N-simplexes cofaces). ◀

▶ Remark 11. The above lemma can be considered as a higher-dimensional version of the two-coloring problem on graphs. We note that in this case, we can color the max-dimensional simplexes with two colors in such a way that no two adjacent simplexes that share a face have the same color. Recall that a graph can be colored by two colors if and only if it is bipartite. Also planer graphs (graphs that can be drawn without any of their edges crossing) can be colored using at most four colors, such that no two adjacent nodes have the same color.

2. The simplicial complex is branching and non-twisting:
▶ **Lemma 12.** *A branching non-twisting simplicial complex is disorientable if and only in its down dual graph, either all of the edges of an odd cycle correspond to a branching $(N-1)$-simplex and/or that odd cycle includes more than one edge corresponding to that branching simplex.*

Proof. Similar to the branching bipartite graphs that are already described, for every branching $(N-1)$-simple α, in the down dual graph we would have a (branching) cycle of length= degree α, as well as all (branching) cycles with length between three and degree of α, such that all of the edges of these cycles correspond to α. Therefore, all of the edges corresponding to all of these cycles in the down dual graph are adjacent and get the same sign and when moving to another adjacent edge (that corresponds to another $(N-1)$-simple), the sign is changed. Therefore, we might have odd cycles, but they can just be for the branching $(N-1)$-simplexes, and no other fundamental odd cycle will exist. Also, when considering general odd cycles (not necessarily fundamental ones) they might include more than one edge corresponding to that branching simplex. ◀

3. The simplicial complex is non-branching and twisting:
▶ **Lemma 13.** *A non-branching simplicial complex is disorientable if and only if in its down dual graph, all the twisting cycles have odd length and all the non-twisting cycles have even length.*

Proof. As we saw, in non-branching (disoriantable) simplicial complexes, the down dual graph has no simple cycle of odd length. We should show every twisted cycle in the down dual graph should be odd to assure disorientability. We note that twisted cycles, by definition, always include two N-simplexes inducing the same orientation on their common face. We recall that this happens whenever we try to assign orientations on the N-simplexes in such a way that lower-adjacent N-simplexes induce opposite orientation on their common face. We call this three simplexes the twisted part. Lets assume the twisted cycle is as following:
$(N-1)$-cycle of length l, $\sigma^0_{N-1}, \beta^1_N ..., \beta^l_N, \sigma^l_{N-1}$
where $\sigma^0 = \sigma^l$ and $\sigma^0(\sigma^l)$, β^1_N and β^l_N is the twisting part.
If we start assigning orientations on N-simplexes, starting from β^1_N in a compatible manner where every two adjacent N-simplex in the cycle induce the same orientation on their common face, if l is even, due to the twisting part, the induced orientation of β^l_N on σ^l would be opposite of orientation of σ^0 induced by β^1_N (see Fig 4.a). Therefore we need odd number of N-simplexes in the cycle to make compatible induced orientations on all common faces along the cycle and in particular on the $\sigma^0 = \sigma^l$ (see Fig 4.c). Roughly speaking, the twisted part acts as an extra (hidden) N-dimensional simplex where the twisted connection corresponds to two faces of this simplex, with opposite orientations. Therefore, every twisted cycle as opposed to simple cycles should have odd length to induce the dissorientability condition along the cycle. The rest of the proof is as before and we omit it here. ◀

As examples of types 1 and 3, see Figure 4 in Example 17. Also, a single edge which is the face of three triangles is a simple example of a branching and non-twisting complex.

In conclusion, we have the following theorem that enables us to check the dissorientability of a general simplicial complex based on the length of the cycles of its down-dual graph.

▶ **Theorem 14.** *A simplicial complex is dissorientable if and only if in its down dual graph, the (possible) odd cycles only correspond to the branching* $(N-1)$*-simplexes and/or twisted cycles and there is no twisted cycle of even length.*

Proof. This can be simply obtained by combining Lemma 3.1, Lemma 3.3 and Lemma 3.4. ◀

▶ Remark 15. A similar result for the special case of this theorem on the normalized Laplacian in [9] was brought to our attention by the reviewers; there they use a detailed analysis of the spectrum of the up-Laplacian in simplicial complexes, while our approach is fully combinatorial and based on the shape of the down-dual graph where we show the results for the maximum eigenvalue of the Laplacian, independent of the weight function on the simplexes. This approach then helps us to obtain the following main result for general Laplacians.

▶ **Theorem 16.** *Every simplicial complex can become dissorientable by a finite number of splittings of some of the* N*-dimensional simplexes into two* N*-dimensional simplexes.*

Proof. We orient all of N-simplexes one by one in a compatible manner, namely in such a way that the adjacent N-simplexes induce the same orientation on their common face. If we can achieve this goal globally and for all of the lower-adjacent N-simplexes, we are done; i.e., the simplicial complex is disorientable. If not, there are at least two N-simplexes, A and B, such that they induce opposite orientations on their common face. We note that this means A and B are adjacent vertices in the down dual graph and based on the previous theorem both are included in a cycle of odd (even for the twisting cycles) length. We chose one of them arbitrarily (A) and we split it into two N-simplexes A' and A'', where A'' and B have non-empty intersections. We orient A' based on the orientation of A in such a manner that A and A' have exactly the same orientation on the shared face. We then orient A'' in a compatible manner with A' where they induce the same orientation on their common face. Then trivially A'' will have a compatible orientation with B, meaning they induce the same orientation on their common face. If we repeat this process for any two non-compatible N-simplex we get our desired result. We note that by splitting A, we make the length of the simple cycle in the down-dual graph including A', A'', B even, and the length of the twisted cycles will become odd. Also if the incompatibility is happening in a branching $(N-1)$-simplex, we might need to split more than one of the cofaces of such simplex; this splitting of course will not change the number of branching and consequently will not affect the length of its corresponding cycle in the down-dual graph. ◀

▶ Remark 17. Note that if splitting one/some of the N-simplex(es) divides its (their) free $(N-1)$-face(s) into two, then this will not have an effect on the other N-simplexes. But if we split the N-simplex in such a way that it divides its non-free face into two, due to the simplicial structure, this will also affect all other N-dimensional cofaces of the divided $(N-1)$ simplex, namely those which are lower adjacent to the original N-simplex (as shown in the next example).

▶ **Example 18.** In the presented example in Figure 4, we show how to make simplicial complexes disorintable and the effect on the cycles of their corresponding down-dual graphs.

We have two main rows of simplicial complexes (with blue) and the down-dual graph of each example is drawn at the bottom of each complex. The top row is the discrete Mobius strip and the bottom row corresponds to a tetrahedron.

▶ **Lemma 19.** *The fewer odd cycles a graph has, the fewer splittings are required to make the graph bipartite; in other words, the closer the maximum eigenvalue is to 2.*

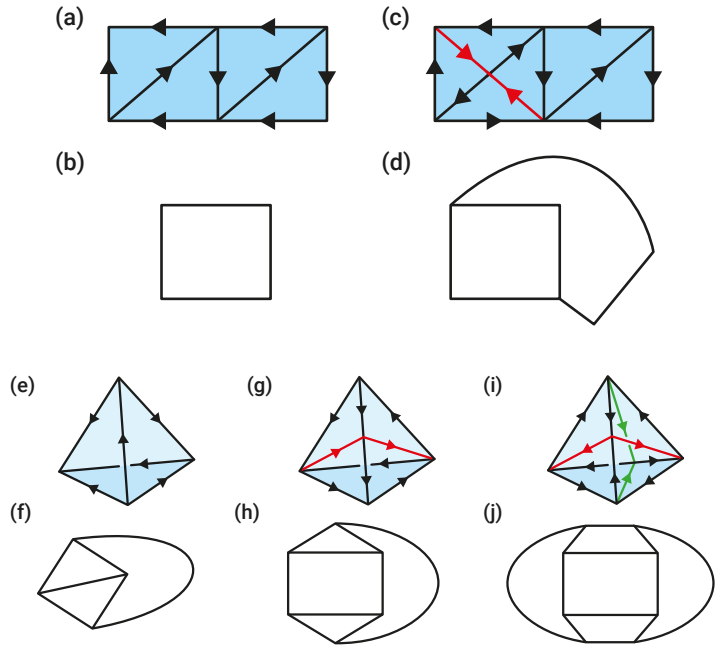


Figure 4 Making simplicial complexes disorintable and the effect on the cycles of their corresponding down-dual graphs for Mobius strip and tetrahedron. The down-dual graph of each complex is drawn in its bottom. (a) is a non-disorientable Mobius strip as (b) has a twisting even cycle. (c) is obtained by splittings of two simplexes of (a), presented by red arrows, which is disorientable as in (d) twisting cycle(s) are even and simple cycle(s) are odd. Similarly, (e) and (g) are non-disorientable as based on respectively (f) and (h) they have odd simple cycles. (i) is obtained from (e) by splitting of all its four 2-simplexes and is disorientable as (j) has no odd simple simple.

Proof. We should show that dividing an edge in an odd cycle of a graph into two by adding a vertex increases the maximum eigenvalue of the graph Laplacian. This can be demonstrated using the Rayleigh quotient, where an arbitrary real-valued function f on the graph's vertex set is considered. If O is an arbitrary odd cycle, there is always an edge xy in the cycle that the function value at x and y has the same sign. By creating a new vertex z (its degree is 2) between x and y with an opposite function sign, the Rayleigh quotient increases, as by adding z the denominator would increase by a factor of $2f^2(z)$ while the nominator increases by a factor of $(f(z) - f(x))^2 + (f(z) - f(y))^2$ which is triviality bigger. Since this happens for any arbitrary real-valued function, the Rayleigh quotient as well as its supremum which is the maximum eigenvalue of the Laplacian increases and it becomes closer to the maximum possible value (i.e. two) and that is obtained by bipartite graphs with no odd cycles. ◀

▶ **Theorem 20.** *In a N-simplicial complex without twisting cycles, the fewer the number of (non-branching) simple odd cycles in its down dual graph, the fewer splittings of N- simplexes are required to make the complex disorientable; namely the closer is its maximum eigenvalue to the possible maximum eigenvalue (which for the Normalized Laplacian is $N + 1$).*

Proof. The proof is very similar to the one of the above Lemma. Here we use the down-dual graph (in dimension N) of the complex and we turn every simple (non-branching) odd cycle to even cycle one by one by adding one vertex as described before. Using Rayleigh quotient of the Laplacian in dimension $N - 1$ and exact argument as above gives the result.

As mentioned before, both of L^{up}_{N-1} and L^{down}_{N} have the same maximum obtained by the supremum of the Rayleigh quotient of functions defined on N-simplexes (i.e. vertices of the down dual graph). ◀

4 Conclusion

We have fully characterized disorientability of general simplicial complexes in terms of the parity of the length of cycles in their down dual graphs. As a direct result, every simplicial complex can become dissorientable by a finite number of splittings of some of the N-dimensional simplexes into two N-dimensional simplexes in a similar manner that any graph can become bipartite by a finite number of splitting of some of its edges into two (i.e. making the odd edges even). Such splittings do not change the topology of the complex and the lower the number of splittings needed to make the complex disoriantable, the closer is the maximum eigenvalue of its Hodge Laplacian spectrum to the possible maximum eigenvalue.

We hope that this new perspective allows the extension of a range of theoretical and applied cycle-based methods from bipartite graphs to higher dimensional disorientable simplicial complexes.

References

1 Armen S. Asratian, Tristan M. J. Denley, and Roland Häggkvist. *Bipartite Graphs and their Applications*. Cambridge Tracts in Mathematics. Cambridge University Press, 1998. doi:10.1017/CBO9780511984068.

2 Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. Proceedings of the Seventh International World Wide Web Conference. doi:10.1016/S0169-7552(98)00110-X.

3 Timoteo Carletti, Lorenzo Giambagli, and Ginestra Bianconi. Global topological synchronization on simplicial and cell complexes. *Phys. Rev. Lett.*, 130:187401, May 2023. doi:10.1103/PhysRevLett.130.187401.

4 Stefania Ebli and Gard Spreemann. A notion of harmonic clustering in simplicial complexes. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1083–1090. IEEE, 2019. doi:10.1109/ICMLA.2019.00182.

5 Beno Eckmann. Harmonische funktionen und randvertanfgaben in einem komplex. *Commentarii Math. Helvetici*, 17:240–245, 1944-1945.

6 Marzieh Eidi and Sayan Mukherjee. Irreducibility of markov chains on simplicial complexes, the spectrum of the discrete hodge laplacian and homology, 2023. arXiv:2310.07912.

7 Ernesto Estrada, Desmond J. Higham, and Naomichi Hatano. Communicability and multipartite structures in complex networks at negative absolute temperatures. *Phys. Rev. E*, 78:026102, August 2008. doi:10.1103/PhysRevE.78.026102.

8 Ernesto Estrada and Juan A. Rodríguez-Velázquez. Spectral measures of bipartivity in complex networks. *Phys. Rev. E*, 72:046105, October 2005. doi:10.1103/PhysRevE.72.046105.

9 Danijela Horak and Jürgen Jost. Spectra of combinatorial laplace operators on simplicial complexes. *Advances in Mathematics*, 244:303–336, 2013. doi:10.1016/j.aim.2013.05.007.

10 Siyang Jiang, Jin Zhou, Michael Small, Jun-an Lu, and Yanqi Zhang. Searching for key cycles in a complex network. *Phys. Rev. Lett.*, 130:187402, May 2023. doi:10.1103/PhysRevLett.130.187402.

11 Jürgen Jost and Dong Zhang. Cheeger inequalities on simplicial complexes, 2023. arXiv:2302.01069.

12 Gregory Z. Gutin Jørgen Bang-Jensen. *Digraphs Theory, Algorithms and Applications*. Cambridge Tracts in Mathematics. Springer London, 2008. doi:10.1007/978-1-84800-998-1.

13 Sayan Mukherjee and John Steenbergen. Random walks on simplicial complexes and harmonics. *Random Structures & Algorithms*, 49(2):379–405, 2016. doi:10.1002/rsa.20645.

14 Ori Parzanchevski and Ron Rosenthal. Simplicial complexes: Spectrum, homology and random walks. *Random Structures & Algorithms*, 50(2):225–261, 2017. doi:10.1002/rsa.20657.

15 Georgios A Pavlopoulos, Panagiota I Kontou, Athanasia Pavlopoulou, Costas Bouyioukos, Evripides Markou, and Pantelis G Bagos. Bipartite graphs in systems biology and medicine: a survey of methods and applications. *GigaScience*, 7(4):giy014, February 2018. doi:10.1093/gigascience/giy014.

16 Michael T. Schaub, Austin R. Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, January 2020. doi:10.1137/18m1201019.

17 Max Wardetzky. Discrete laplace operators. *An Excursion Through Discrete Differential Geometry: AMS Short Course, Discrete Differential Geometry*, pages 1–57, 2007. URL: http://geometry.cs.cmu.edu/ddgshortcourse/notes/01_DiscreteLaplaceOperators.pdf.

18 William H. Weir, Peter J. Mucha, and William Y. Kim. A bipartite graph-based expected networks approach identifies ddr genes not associated with tmb yet predictive of immune checkpoint blockade response. *Cell Reports Medicine*, 3(5):100602, 2022. doi:10.1016/j.xcrm.2022.100602.

Non-Euclidean Erdős–Anning Theorems

David Eppstein
University of California, Irvine, CA, USA

Abstract

The Erdős–Anning theorem states that every point set in the Euclidean plane with integer distances must be either collinear or finite. More strongly, for any (non-degenerate) triangle of diameter δ, at most $O(\delta^2)$ points can have integer distances from all three triangle vertices. We prove the same results for any strictly convex distance function on the plane, and analogous results for every two-dimensional complete Riemannian manifold of bounded genus and for geodesic distance on the boundary of every three-dimensional Euclidean convex set. As a consequence, we resolve a 1983 question of Richard Guy on the equilateral dimension of Riemannian manifolds. Our proofs are based on the properties of additively weighted Voronoi diagrams of these distances.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases integer distances, additively weighted Voronoi diagrams, convex distance functions, Riemannian manifolds, geodesic distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.46

Related Version *Full Version*: `https://arxiv.org/2401.06328`

Funding *David Eppstein*: Research supported in part by NSF grant CCF-2212129.

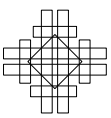
1 Introduction

Many incidence properties of lines or curves, seemingly mysterious on their own, become obvious when viewed through the lens of computational geometry, where the curves may be features of Voronoi diagrams or related constructions. Thus, for instance, to see that the three perpendicular bisectors of a triangle's sides meet at a single point, observe that these bisectors contain the edges of the Voronoi diagram of the triangle vertices, and these Voronoi edges meet at the Voronoi vertex. In this work we apply the same principle to the Erdős–Anning theorem on integer distances in the plane, conventionally proven using the algebraic intersection properties of hyperbolae. By reinterpreting these hyperbolae and their intersection points as the cell boundaries and vertices of additively-weighted Voronoi diagrams, and by analyzing these diagrams geometrically and topologically rather than algebraically, we greatly generalize this theorem, to several wide classes of two-dimensional metric spaces. In doing so, we resolve a 1983 question of Richard Guy on the equilateral dimension of Riemannian manifolds, the maximum number of points that can be placed at equal distances from each other in these spaces.

The Erdős–Anning theorem states that every point set in the Euclidean plane with integer distances must be either collinear or finite [1, 9]. More strongly, for any (non-degenerate) triangle of diameter δ, at most $O(\delta^2)$ points can have integer distances from all three triangle vertices. When the whole point set has diameter D, its number of points is $O(D)$ [25], and if in addition it is non-collinear its number of points is $D^{O(1/\log\log D)}$ [11]. In contrast, as Euler already observed, there exist infinite non-collinear point sets of bounded diameter with all distances rational, for instance, dense subsets of a unit circle [10, 13].

There are many metrics other than Euclidean distance for which we might ask similar questions. One obstacle to generalizing the Erdős–Anning theorem to other metrics is that its usual proof is heavily based on algebraic geometry. In the Euclidean plane, the locus of points whose distances to two given points differ by a fixed integer is a branch of a hyperbola.

© David Eppstein;
licensed under Creative Commons License CC-BY 4.0



Therefore, the points with integer distances to three non-collinear points lie at the intersection points of two finite families of hyperbolas. Hyperbolas are algebraic curves of degree two, and by Bezout's theorem two hyperbolas have at most four crossings, so these two finite families have finitely many intersection points. But it is not obvious how to extend this argument to metrics for which the corresponding curves are more complicated or non-algebraic. Erdős and Anning originally used a different argument using trigonometric inequalities [1], but its generalization is also non-obvious.

A second obstacle is that analogous statements are untrue for some metrics. For the L_1 or L_∞ distances in the plane, the integer lattice provides a familiar example of an infinite set that is not collinear and has only integer distances. Thus, generalizations of the theorem to other families of metrics must avoid families that include L_1 or L_∞.

In this paper we prove the following generalized versions of the Erdős–Anning theorem:

> *For any strictly convex distance function on $\mathbb{R}^2$, every point set with integer distances must be either collinear or finite. For any (non-degenerate) triangle of diameter δ, at most $O(\delta^2)$ points can have integer distances from all three triangle vertices. If a set with integer distances has diameter D, it has at most $O(D)$ points.* (Section 3)
>
> *For any two-dimensional complete Riemannian manifold of finite genus g, every point set with integer distances must either be finite or be contained in a geodesic (a locally-shortest curve) that contains every shortest curve between pairs of the points. If some three points do not lie on a single shortest curve, and have diameter δ, at most $O((g+1)\delta^2)$ points can have integer distances from these three points.* (Section 4)
>
> *For geodesic distance on the boundary of a convex set in $\mathbb{R}^3$, every point set with integer distances must either be finite or be contained in a geodesic that contains every shortest curve between pairs of the points. If some three points do not lie on a single shortest curve, and have diameter δ, at most $O(\delta^2)$ points can have integer distances from these three points. If a set with integer distances has diameter D, it has at most $O(D^{4/3})$ points.* (Section 5)

Our proofs replace the intersection properties of low-degree algebraic curves with the geometric and topological properties of additively weighted Voronoi diagrams. Our results for Riemannian manifolds resolve a question posed in 1983 by Richard Guy [12], on the *equilateral dimension* of Riemannian manifolds, the maximum number of points at unit distance from each other in these manifolds. Guy asked whether the equilateral dimension could be bounded by a function of dimension. For complete Riemannian 2-manifolds of bounded genus, we bound the equilateral dimension by a function of the genus (Corollary 12) but for locally Euclidean incomplete Riemannian 2-manifolds, for complete Riemannian 2-manifolds of unbounded genus, and for complete Riemannian metrics on $\mathbb{R}^3$, we show that the equilateral dimension is unbounded (Example 6 and Example 7).

For space reasons we defer many lemmas and other material to the full version at `https://arxiv.org/2401.06328`.

2 Preliminaries: Voronoi diagrams and star-shaped cells

A *Voronoi diagram* of a finite set of point *sites* in a metric space (X, d) subdivides the space according to which site is closest. Each site s_i has a (closed) cell V_i associated with it, where

$$V_i = \left\{ p \in X \;\middle|\; d(p, s_i) \le \min_{j \neq i} d(p, s_j) \right\}$$

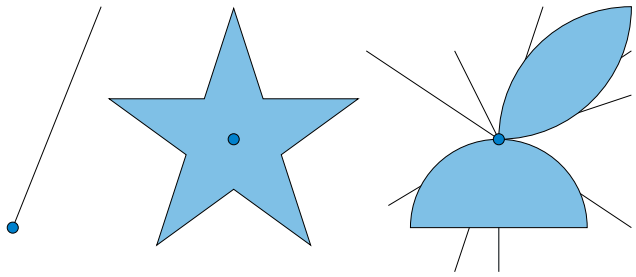


Figure 1 Three star-shaped sets, each shown with a point in its kernel.

is the set of points whose distance to site s_i is less than or equal to the distance to any other site s_j. We will only use Voronoi diagrams of finite sets (in fact, of at most three sites), so we only define and discuss the properties of these diagrams for finite sets of sites.

The intersection of any collection of Voronoi cells is the locus of points having equal distances to their sites. When X has the topology of the plane, the *bisector* of a pair of sites (the intersection of their Voronoi cells) is often (but not always) a curve, and a non-empty intersection of three or more sites is often (but not always) an isolated point or points, called *Voronoi vertices*. We will use *additively weighted* Voronoi diagrams, in which each site s_i has a weight w_i associated with it, and in which the distances in the definition of cells are modified by adding these weights:

$$V_i = \left\{ p \in X \;\middle|\; d(p, s_i) + w_i \le \min_{j \neq i} d(p, s_j) + w_j \right\}.$$

Voronoi diagrams have been the object of intensive study, including Voronoi diagrams for convex distance functions [6, 15, 17], for points on a sphere [3, 21, 23], for points in the hyperbolic plane [4, 8, 24], orbifolds [20], and for more general Riemannian manifolds [18]. Additively weighted Voronoi diagrams are also a standard concept [5, 16], and confusingly, are sometimes called *hyperbolic Dirichlet tessellations* despite their use of Euclidean distance [2], because their cells are bounded by hyperbolas (the same hyperbolas from the Erdős–Anning proof). We did not find works combining additive weights and non-Euclidean distances.

The *kernel* of a subset S of the plane is defined as the set of points p such that, for every point q in S, the line segment pq belongs to S. S is convex if and only if its kernel coincides with S. Weakening this formulation of convexity, S is defined to be *star-shaped* if its kernel is non-empty. (See Figure 1. Note that, according to this definition, star-shaped sets are not required to form topological disks, and the empty set is not star-shaped.) It is known that, for instance, the Voronoi cells of unweighted convex distance functions are star-shaped topological disks [6]. For additively weighted Voronoi diagrams, we should be a little more careful, because of the possibility that a Voronoi cell can degenerate to a line segment, a single point, or the empty set, but in the Euclidean case they are again known to be star-shaped when non-empty [2].

3 Convex distance functions

A *convex distance function* on $\mathbb{R}^d$ can be defined from any centrally symmetric compact convex body K. The resulting function $d_K(p, q)$ is the smallest scale factor s such that a copy of K, scaled by s and centered at p, contains q. Any convex distance function defines a norm on $\mathbb{R}^d$, $||p||_K = d_K(0, p)$. In the reverse direction, the closed unit ball $\{p \mid ||p|| \le 1\}$

of any norm $||\cdot||$ can be used to define a convex distance function. A convex set is *strictly convex* if its boundary does not contain any line segment, and a *strictly convex distance function* is a convex distance function defined from a strictly convex set. Convex distance functions obey all the requirements of a metric space, including the triangle inequality. For strictly convex distance functions and non-collinear triples of points p, q, r, they obey a strict form of the triangle inequality [6]:

$$d_K(p,r) < d_K(p,q) + d_K(q,r).$$

Examples of strictly convex distance functions include the Minkowski L_p distances for $1 < p < \infty$, obtained from the convex body

$$K_p = \left\{ (x_1, x_2, \ldots x_d) \;\middle|\; \sum_{i=1}^{d} x_i^p \le 1 \right\}.$$

For $p = 1$ and for the limiting case $p = \infty$ one obtains distance functions that are convex but not strictly convex. For $p = 2$ we recover the familiar Euclidean distance. Convex distance functions define *geodesic metric spaces* in which the distance between any two points equals the length of a shortest curve between the points. For strictly convex distance functions, this shortest curve is unique: it is just the *line segment* between its two points.

In unweighted Voronoi diagrams of strictly convex distance functions, every cell is a topological disk (or the whole space) with its site in its interior. However, additively weighted Voronoi diagrams may have additional complications, which we examine more carefully in the full version. In brief: A site may have zero or negative distance to another site, and therefore belong to the cell of another site; we call such sites *degenerate*. The Voronoi cell of a degenerate site may be empty, a single point, or a line segment or ray extending away from the site in the direction opposite its nearest non-degenerate neighbor. The cells of non-degenerate sites are strictly star-shaped with respect to their sites, interior-disjoint, and cover the plane. From these properties we prove:

▶ **Lemma 1.** *For the additively weighted Voronoi diagram of three non-collinear points, the three cells have at most two points of common intersection.*

The proof is a case analysis that considers how many sites are degenerate. The two degenerate cells of two degenerate sites lie on rays diverging from the non-degenerate site, and cannot intersect. With one degenerate site, there can be only one point of common intersection, the degenerate site (if its cell is a single point) or the opposite endpoint of its line segment cell. The main case has no degenerate sites. If there could be three points of common intersection, the line segments connecting these points to the three sites would lie entirely within the cells of their sites, and could not cross. Therefore, these segments would form a planar drawing of the nonplanar graph $K_{3,3}$, a contradiction.

3.1 Erdős–Anning theorems for strict convex distance functions

▶ **Theorem 2.** *Let s_1, s_2, and s_3 be three non-collinear points of a 2-dimensional strictly convex distance function d. Then the number of points that can have integer distances to all three of s_1, s_2, and s_3 is $O\big(d(s_1,s_2)\cdot d(s_1,s_3)\big)$.*

Proof. Let p be a point with integer distances to all three of s_1, s_2, and s_3. Consider the additively weighted Voronoi diagram of these three sites, with weights $w_i = d(s_1,p) - d(s_i,p)$. Then at p, the three additively weighted distances all equal $d(s_1,p)$, so all three sites are equal nearest neighbors. This means that p must belong to the triple intersection of Voronoi cells $V_1 \cap V_2 \cap V_3$.

Weight w_1 is identically zero, and by the triangle inequality the remaining two weights have absolute values $|d(s_1,p) - d(s_2,p)|$ and $|d(s_1,p) - d(s_3,p)|$ that are at most $d(s_1,s_2)$ and $d(s_1,s_3)$ respectively. Therefore, there are at most $\big(2d(s_1,s_2)+1\big) \cdot \big(2(d(s_1,s_3)+1\big)$ combinations of weights defining an additively weighted Voronoi diagram for which p might be a triple intersection point, and at most two triple intersection points per diagram by Lemma 1, giving a total of at most $2\big(2d(s_1,s_2)+1\big) \cdot \big(2(d(s_1,s_3)+1\big)$ possible points that can have integer distances to all three of s_1, s_2, and s_3. ◀

The following generalization of the Erdős–Anning theorem follows immediately:

▶ **Theorem 3.** *If a set S of points of a 2-dimensional strictly convex distance function has the property that all distances between points of S are integers, then S is finite or collinear.*

Proof. If S is not collinear, it has three non-collinear points to which we apply Theorem 2. ◀

3.2 Diameter bounds

The number of points in a set with integer distances can also be bounded linearly in the diameter of the set, generalizing a result of Solymosi [25] for Euclidean distance.

▶ **Theorem 4.** *If a set S of points of a 2-dimensional strictly convex distance function has diameter D (measured according to that distance function), and all distances in S are integers, then $|S| = O(D)$, with a constant of proportionality that depends on the distance function.*

Proof. We may assume without loss of generality that the unit ball of the given distance function d is contained in a Euclidean unit circle. For, if it does not, let ξ be the maximum Euclidean distance from the origin to any point of the unit d-ball, and scale both S and d by $1/\xi$, leaving distances between the scaled points unchanged and allowing the d-ball to fit within a Euclidean unit circle. Let ρ be the minimum Euclidean distance from the origin of the unit circle of d; note that, if we treat the distance function as fixed, then ρ is a constant. Then the function measured by d is lower bounded by the Euclidean distance, and upper bounded by $\frac{1}{\rho}$ times the Euclidean distance. From the lower bound on d, the Euclidean diameter of S is at most its d-diameter, D.

Enclose S in a square bounding box of (Euclidean) side length D, and center this within a larger square B of side length $3D$, so that all points of S are at Euclidean distance at least D from the boundary of B. Construct the (Euclidean) Voronoi diagram of S, and let s_1 be the point of S whose Voronoi cell V_1, intersected with B, has the smallest Euclidean area, $\leq 9D^2/n = O(D^2/n)$. Let s_2 be the Euclidean nearest neighbor of s_1 in S, let m be the midpoint of s_1 and s_2, and construct a right triangle ps_1m with right angle at s_1, contained in $V_1 \cap B$, with the largest possible Euclidean area. (See Figure 2 for an example.) We distinguish two cases:

- If p is on the boundary of B, as shown in Figure 2, it is at Euclidean distance $\geq D$ from s_1. In order for right triangle ps_1s_2 to have area $O(D^2/n)$, at most the area of the set $V_1 \cap B$ that contains it, edge s_1s_2 must have Euclidean length $O(D/n)$, and therefore also $d(s_1,s_2) = O(D/n)$ (larger than the Euclidean length by at most the constant factor $\frac{1}{\rho}$). For this distance to be a nonzero integer, n must be $O(D)$.
- Otherwise, p is on the boundary of V_1, and equidistant (in Euclidean distance) from s_1 and some site s_3 which cannot be collinear with s_1s_2. Therefore, the Euclidean distance from s_1 to p is at least half the Euclidean distance from s_1 to s_3. Combining this inequality

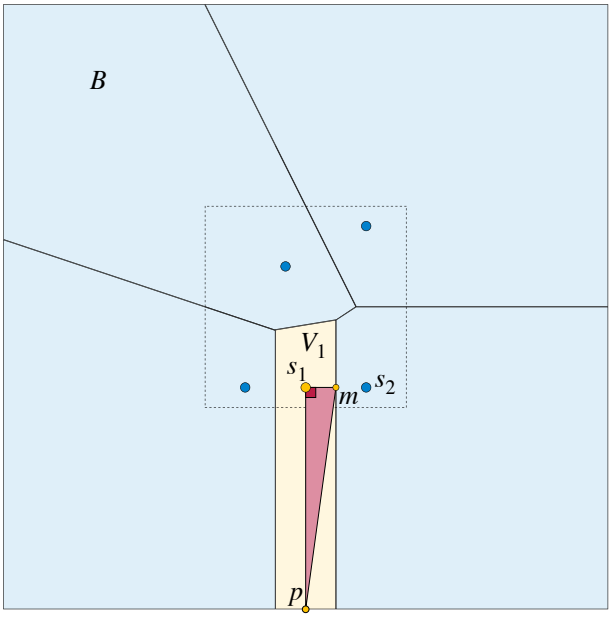


Figure 2 Figure for the proof of Theorem 4: After enclosing the given points in a bounding box B of side length three times the diameter D, s_1 is the point whose Euclidean Voronoi cell V_1 has the smallest intersection with B, s_2 is its nearest neighbor, m is their midpoint, and right triangle ps_1m is chosen to have as large an area as possible within $V_1 \cap B$. In the case shown, p is on the boundary of B, at distance greater than D from s_1, from which it follows that $d(s_1, s_2) = O(D/n)$.

with the area of the triangle and the $\frac{1}{\rho}$ factor by which d can increase over Euclidean distance gives the bound $d(s_1, s_2) \cdot d(s_1, s_3) = O(D^2/n)$. It follows from Theorem 2 that $n = O(D^2/n)$ and therefore that $n = O(D)$. ◀

The examples of $n \times n$ integer grids for the L_1 or L_∞ distance show that the assumption of strict convexity is necessary for this result.

4 Riemannian manifolds

In this section we prove a form of the Erdős–Anning theorem for two-dimensional complete Riemannian manifolds. These can be defined intrinsically (without respect to an embedding) as smooth manifolds equipped with a smoothly varying inner product structure on their tangent space. "Completeness" in this context means that all Cauchy sequences converge; for instance, puncturing the Euclidean plane by removing the origin would produce a space that is not complete. Examples of such spaces include the Euclidean plane itself, the surface of any smooth convex body in three dimensions such as a sphere, and the hyperbolic plane.

One complication in these spaces is that their usual analogues of straight lines, *geodesics*, are only locally shortest curves between their points: two or more points may belong to a common geodesic that does not contain the shortest curve between them. Indeed, in some cases a geodesic may cross itself or form a dense subset of its surface. For this reason, we will generally work with shortest curves (between some two points of the space) rather than geodesics. These curves may not be unique: for instance, for antipodal points on a sphere, there are infinitely many shortest curves, covering the whole sphere. By the Hopf–Rinow theorem [14], every complete Riemannian manifold has a shortest curve between every two points. Thus, these are geodesic metric spaces.

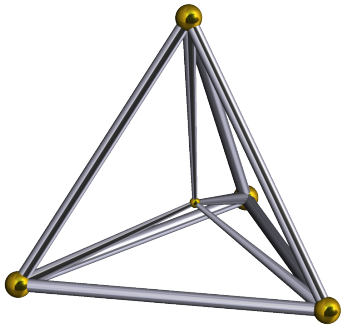


Figure 3 Five small spheres connected in pairs by cylindrical tubes. Smoothing the sphere-cube edges produces a complete Riemannian manifold. Public domain image from `https://commons.wikimedia.org/wiki/File:Schlegel_wireframe_5-cell.png`.

We will not assume that the surfaces we consider are orientable. One way to parameterize a two-dimensional surface, without assuming orientability, is by its *Euler genus*, the number $2 - \chi$ where χ is the Euler characteristic. It is possible for a Riemannian manifold to have infinite genus, but (except for Example 7, which motivates this limitation) we will only work with surfaces of bounded genus.

4.1 Examples

▶ **Example 5.** Consider any finite set of three or more points S on a unit sphere, no three of which belong to a great circle. Then each two points of S have a unique shortest curve which does not pass through other points of S. Perturb the sphere by placing a small smooth bump along each shortest curve, increasing the distance between its two points to a rational number without changing the other distances, and then scale the result to clear the denominators of these rational distances. The result is a complete Riemannian manifold of Euler genus zero with an arbitrary finite number of points at integer distances from each other.

It is necessary to assume that our Riemannian manifolds are complete, because of the following example:

▶ **Example 6.** Consider the points coordinatized by two real numbers r and θ, with r positive and θ arbitrary, and with the standard locally Euclidean geometry of the polar coordinate system. This can be thought of as a cone with an infinite angle at its removed apex [26]. Its Euler genus is zero (topologically it is just an open halfplane). When two of its points have θ-coordinates that differ by at least π, their distance equals the sum of their r-coordinates (distances from the removed apex). Therefore, the points $\{(\frac{1}{2}, i\pi) \mid i \in \mathbb{Z}\}$ form an infinite set at unit distance from each other. They have no shortest curves, because any such curve would pass through the missing apex point.

As the next example shows, to obtain a generalization of the Erdos–Anning theorem to complete Riemannian manifolds, we will also need to assume bounded genus, and our bounds on numbers of points in integer distance sets will need to depend on the genus.

▶ **Example 7.** For any n, one may form a complete Riemannian manifold from n spheres, connected by cutting out disks from pairs of spheres, attaching a cylindrical tube between the boundaries of the two disks, and smoothing the parts of the surface where the disks meet

the tubes. This construction may be performed abstractly as a topological space; it is not necessary to embed the spheres and tubes into three-dimensional space. Let S consist of one point per sphere, for spheres small enough that all distances within one sphere are less than $\frac{1}{3}$; then by adjusting the lengths of all the tubes, all distances between points of S can be made equal to one. This construction produces arbitrarily large finite sets of points at unit distance from each other, with genus $O(n^2)$. Because they all have the same distance, no three of these points belong to a shortest curve.

The connected sum of an infinite sequence of examples of this type for $n = 4, 5, 6, \ldots$, using tubes to connect the example for each n to the examples for $n-1$ and $n+1$, forms a single complete Riemannian manifold of infinite genus in which one can find arbitrarily large (but nevertheless finite) sets of points that are all at unit distance from each other.

Although the topological structure of this example is complicated, it can be embedded in a distance-preserving way into a Riemannian metric on $\mathbb{R}^3$. For each n, choose the spheres and tubes of the n-point example to have a small enough radius that they can all be smoothly embedded within a unit ball of $\mathbb{R}^3$, coiling the tubes to cause them to have the desired lengths. Form the connected sum of these examples as before and embed it smoothly in $\mathbb{R}^3$ using disjoint unit balls for each term in the connected sum. Thicken the resulting smoothly embedded surface S by a small amount (varying smoothly over the surface) chosen to be small enough that the thickened set remains smoothly embedded in $\mathbb{R}^3$, and give the resulting thickened set a Riemannian metric that agrees with the metric on S (and the Euclidean distance of $\mathbb{R}^3$) in directions parallel to S, but that is longer in the direction perpendicular to S by a factor that varies smoothly from very large for points on S to one at the boundary of this thickened set. Then, for the remaining points of $\mathbb{R}^3$, outside the thickened set, use the usual Euclidean distance. By choosing this perpendicular lengthening factor large enough, it is possible to ensure that escaping from S to the boundary of the thickened set takes more than unit distance, and that distances within S are preserved.

Richard Guy asked whether the *equilateral dimension* of Riemannian manifolds, the maximum cardinality of a set of points in the manifold for which all distances are one, can be bounded by a function of the dimension of the manifold [12]. A positive answer was known only for the case of complete Riemannian manifolds of bounded curvature [19]. Example 6 provides a counterexample for locally Euclidean and topologically simple but incomplete Riemannian 2-manifolds. Example 7 provides a counterexample for complete Riemannian 2-manifolds of unbounded genus and for topologically simple complete Riemannian 3-manifolds. On the other hand, our generalization of the Erdős–Anning theorem to complete Riemannian 2-manifolds of bounded genus also provides a bound on the equilateral dimension of these manifolds, as a function of their genus (Corollary 12).

Our proof of the Erdős–Anning theorem for strictly convex distance functions relied on the non-planarity of $K_{3,3}$. For complete Riemannian manifolds of bounded genus we need to reach further for graphs that cannot be embedded, as the following example shows.

In the full version we prove properties of additively weighted Voronoi diagrams for Riemannian manifolds analogous to those of convex distance functions, culminating in a bound of $2g + 2$ on the number of intersections of the cells of three sites with none of the three belonging to a geodesic between the other two.

4.2 Erdős–Anning theorems for Riemannian manifolds

▶ **Theorem 8.** *Let s_1, s_2, and s_3 be three points of a 2-dimensional complete Riemannian manifold of Euler genus g, with distance function d. Suppose also that there is no shortest curve containing all three of these points. Then the number of points that can have integer distances to all three of s_1, s_2, and s_3 is $O\big((g+1)\cdot d(s_1,s_2)\cdot d(s_1,s_3)\big)$.*

Except for the factor of g coming from our bound on triple points of Voronoi diagrams from the full version, the proof is the same as for Theorem 2. For this theorem, the possibility that all triples of points lie on shortest curves (preventing the theorem from being applied) corresponds to the possibility of all points being collinear in the Euclidean plane or for convex distance functions. It is a strengthening of the more obvious generalization of collinearity, that all points of S lie on a common geodesic:

▶ **Lemma 9.** *Let S be a set of three or more points of a complete Riemannian manifold such that each three points of S belong to a shortest curve. Then S is a subset of a geodesic, and every shortest curve between pairs of points in S lies along this geodesic.*

Proof. Let s_1 be any of the points, and consider the system of rays containing shortest curves to all the other points. Suppose for a contradiction that two rays R_2 and R_3 in this system of rays contain shortest curves from s_1 to points s_2 and s_3 in S, and that these two rays are distinct and non-opposite. Then if s_1 were on a shortest path from s_2 to s_3, then a shortest curve from s_2 to s_3 would follow R_2 from s_2 to s_1 and then follow R_3 from s_1 to s_3. This cannot happen, because a shortest curve that turns from one ray to another at s_1 could be shortcut by a different curve in a neighborhood of s_1. Alternatively, if s_2 were on a shortest curve from s_1 to s_3, then we could get a shortest curve from s_1 to s_3 that follows R_2 from s_1 to s_2, turns at s_2, and then continues along the R_3 from s_2 to s_3, again an impossibility. The case that s_3 is on a shortest curve from s_1 to s_2 is symmetric. So none of the three points can be the middle point of a shortest curve containing all three points. This contradiction to the assumption of the lemma implies that rays R_2 and R_3 cannot exist, and that the system of rays contains either a single ray or two opposite rays. In either case, these two rays, and therefore all points of S, lie on a single geodesic.

If some two points s_2 and s_3 had a shortest curve that did not lie along this geodesic, then we could apply this same argument to the rays at s_2 containing the shortest curve to s_1 (along the geodesic through all points of S) and the ray containing the shortest curve from s_2 to s_3 (not along this geodesic), obtaining the same contradiction. Therefore all shortest curves lie on the same geodesic. ◀

The conclusion of this lemma cannot be strengthened to state that each pair of points in S has a unique shortest curve. For instance, consider an even number of points equally spaced on the equator of a sphere. In this example, each point has two shortest curves to its opposite point. A set of points whose shortest curves all lie on a single geodesic might still not have the property that each three points belong to a shortest curve: consider, for instance, three equally spaced points on the equator of a sphere.

Combining Theorem 8 with Lemma 9 gives us the following generalization of the Erdős–Anning theorem:

▶ **Theorem 10.** *If a set S of points of a 2-dimensional complete Riemannian manifold of bounded genus has the property that all distances between points of S are integers, then either S is finite or there is a single geodesic of the manifold containing all points of S and connecting each pair of points by a shortest curve.*

We also have the following diameter bound, weaker than in the Euclidean or convex distance function case:

▶ **Theorem 11.** *If a set S of points of a 2-dimensional complete Riemannian manifold of bounded genus g has integer distances and diameter D, then $|S| = O(gD^2)$.*

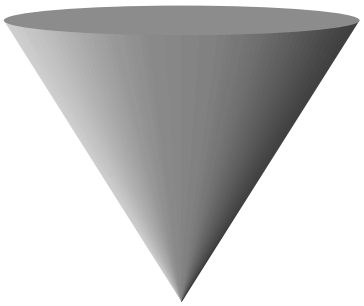


Figure 4 A solid cone.

Proof. If S has three points not all on a single shortest curve, this follows from Theorem 8. Otherwise, the points all have shortest curves along a geodesic. Let C be a shortest curve of length D between some two points of S. If C covers all points of S, there are at most $D+1$ points. Otherwise, let s_1 and s_2 be the two endpoints of C, and let s_3 be any point not covered by C. For s_1, s_2, and s_3 to be part of a shortest curve, the curve must have s_1 and s_2 as endpoints, for otherwise we would have a shortest curve that includes a segment from s_1 to s_2, causing it to be longer than D. In this case, the only possibility is that s_1 and s_2 have two shortest paths, and are at distance D from each other along a geodesic of length $2D$, so there are at most $2D$ points in S. ◀

▶ **Corollary 12.** *The equilateral dimension of a 2-dimensional complete Riemannian manifold of bounded genus g is $O(g)$.*

5 Convex surfaces

In this section we extend our results on Riemann manifolds to non-smooth surfaces of non-negative curvature. The restriction to non-negative curvature is motivated by Example 6, which shows that even a single cone point of negative curvature can be an obstacle to Erdős–Anning theorems. For concreteness, in this section we usually define a *convex surface* to mean the boundary of a convex subset of $\mathbb{R}^3$ with nonempty interior. However, we also allow a special case, which can be thought of as the limit of convex cylinders as the cylinder height goes to zero: the double covering of a convex subset K of $\mathbb{R}^2$, obtained by gluing two copies of K at corresponding boundary points. For instance, the Euclidean plane is a convex surface in the first of these two senses: it is the boundary of a halfspace. By results of Aleksei Pogorelov, a more abstract class of non-smooth convex manifolds, topologically equivalent to a sphere and with total Gaussian curvature 4π, can be characterized as being either boundaries of convex bounded subsets of $\mathbb{R}^3$ or double covers of convex bounded subsets of $\mathbb{R}^2$ [7]. We include double covers in our definition of convex surfaces to be consistent with this result of Pogorelov, but we do not require the convex sets from which they are defined to be bounded. We measure the distance on a convex surface as the infimum of lengths of curves on the surface. By a compactness argument, each pair of points on the surface can be connected by a curve of length equal to this distance, so this is a geodesic metric space [22], and we can apply arguments involving the Lipschitz property of distances. Every convex surface, in the sense described here, is topologically a disk (like the Euclidean plane), a sphere (like a geometric sphere, polyhedron, or double-covered topological disk), or an annulus (like the boundary of an infinite cylinder).

The boundary of a (bounded) solid cone (Figure 4) exhibits several of the common geometric features of a convex surface. The base of the cone is a flat disk; the sides of the cone have a flat surface metric (locally isometric to a subset of the Euclidean plane) even though they are curved in three-dimensional space. The circle where the base meets the sides is non-smooth; each point has a neighborhood that approaches a Euclidean metric (equivalent to that of a folded piece of paper) in the limit. The apex of the cone, however, is not Euclidean, even in the limit: it has a positive angular deficiency that can be measured by comparing the radius and circumference of small metric disks around it. The total curvature of this surface is 4π, by the Gauss–Bonnet theorem, but in this case this curvature is concentrated at the cone point (where it equals the angular deficit of this point) and along the circle where the base and sides meet.

In general, when a shortest curve on a convex surface passes through a locally Euclidean point, it must pass straight through it, in the local Euclidean geometry of a neighborhood of a point. A curve that does not do this could be shortcut to bypass the point. It is not possible for a shortest curve to pass through a cone point (any point of positive angular deficiency): any curve through such a point can be shortcut. Thus, we need only consider such points as endpoints of shortest curves, not as their interior points.

▶ **Lemma 13.** *For every point p on a convex surface, the disks of radius r centered on that point, in the limit as $r \to 0$, approach a cone for which p either has angular deficit zero (p is locally Euclidean) or a well-defined positive angular deficit (p is locally a cone point).*

Proof. For a point p on a convex surface, let $C_r(p)$ denote the cone formed by positive combinations of p with the points at distance r from p on the surface. Then C_r is monotone in r: for $r' < r$, $C'_r(p) \supset C_r(p)$, by convexity. By compactness of the space of cones at a point p, in the limit as r goes to zero, these cones approach a limiting cone; call this limit $C_0(p)$. Let B be the curve obtained by intersecting the boundary of $C_0(p)$ with a unit sphere centered at p. Then B is convex, and lies in a half-sphere, again by convexity of the convex surface. Thus, its length is at most 2π. If it is exactly 2π, then p is locally Euclidean; otherwise, it is locally equivalent to a cone point with deficit equal to the difference between 2π and the length of B. ◀

We defer the detailed analysis of Voronoi diagrams for these metrics to the full version.

5.1 Erdős–Anning theorems for convex surfaces

▶ **Theorem 14.** *Let s_1, s_2, and s_3 be three points of a convex surface, with distance function d. Suppose also that there is no shortest curve containing all three of these points. Then the number of points that can have integer distances to all three of s_1, s_2, and s_3 is $O\big(d(s_1, s_2) \cdot d(s_1, s_3)\big)$.*

Proof. Same as for Theorem 2 and Theorem 8, substituting the properties of additive Voronoi diagrams for convex surfaces in place of the corresponding properties of the diagrams for convex distance functions and Riemannian manifolds. ◀

▶ **Lemma 15.** *Let S be a set of three or more points of a convex surface such that each three points of S belong to a shortest curve. Then S is a subset of a geodesic, and every shortest curve between pairs of points in S lies along this geodesic.*

Proof. Same as for Lemma 9, again substituting the properties of Voronoi diagrams on these surfaces. ◀

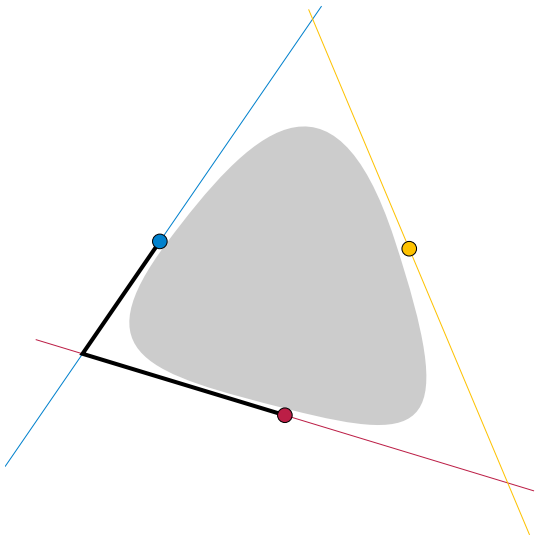


Figure 5 Lemma 17: a short path connecting two of three points separated by a convex set.

▶ **Theorem 16.** *If a set S of points of a convex surface has the property that all distances between points of S are integers, then either S is finite or all triples of points in S are subsets of shortest curves along a single geodesic.*

Proof. Same as for Theorem 10, again substituting the properties of Voronoi diagrams on these surfaces. ◀

Our diameter bound seems unlikely to be tight. It uses the following preliminary lemmas.

▶ **Lemma 17.** *Suppose that three given points in $\mathbb{R}^d$ are disjoint from an open convex set U. Then two of the points can be connected by a curve disjoint from U of length at most twice their Euclidean distance.*

Proof. Consider the plane containing the three points; if U does not intersect this plane then the points can be connected directly. Within this plane draw a line through each point, disjoint from U. The three lines form a triangle or unbounded three-sided region containing U; if all three points lie its sides, one of its internal angles is $\geq 60°$, and the curve connecting two points through this angle has length at most twice the distance between the points (Figure 5; worst case: U is an equilateral triangle and the three points are its edge midpoints). If one of the three points is not on a side of the region containing U, it can see the entire line through another of the points, and be connected directly to that point. ◀

▶ **Lemma 18.** *If n points on a convex surface have diameter D then some two of the points have geodesic distance $O(D/\sqrt{n})$.*

Proof. Let the set of points be S, and let the convex surface be the boundary of a set K (or the double cover of a set K), embedded in $\mathbb{R}^3$. By assumption, S lies within a ball B of radius D in $\mathbb{R}^3$. Subdivide the ball by a grid of cubes, of side length $O(D/\sqrt{n})$, positioned so that none of the given points lies on a cube boundary, and let $\square$ denote the set of cubes in this grid whose intersection with $B \cap K$ is non-empty and includes at least one non-vertex point of the cube. Each point in S is contained in at least one cube $C \in \square$. This cube C must touch (at least at a corner) a grid cube that is outside $\square$, because if all eight cubes corner-adjacent

to C belonged to $\square$, the intersection points of these eight cubes with $B \cap K$ would have a convex hull that entirely encloses C, preventing C from containing any boundary points of K. Therefore, C is within $O(1)$ grid positions of a boundary square of the union of cubes $\cup\square$. But $\cup\square$ is orthogonally convex – any axis-parallel line can cross only two boundary squares (one in each direction) – so it has $O(n)$ boundary squares. Therefore there are also $O(n)$ cubes near these squares that contain points of S. By choosing the grid size appropriately, we can adjust the constant factors in this $O(n)$ bound and ensure that fewer than $n/2$ cubes contain points of S. With this adjustment, some cube $C \in \square$, of side length $O(D/\sqrt{n})$, contains at least three points of S. We consider two cases:

- If the convex surface is non-degenerate (it bounds a convex set with non-empty interior), then by Lemma 17, two of these three points are connected by a path in space, disjoint from the interior of the given convex set, of length $O(D/\sqrt{n})$, and hence are at geodesic distance $O(D/\sqrt{n})$ on the convex surface.
- Otherwise, the convex surface is a double-covered two-dimensional convex set, two of the three points are on the same side of the double cover, and these two points can be connected directly within C by a geodesic segment of length $O(D/\sqrt{n})$.

In either case, we have two points at geodesic distance $O(D/\sqrt{n})$. ◀

The example of a square grid of $\sqrt{n} \times \sqrt{n}$ points on a flat plane, scaled to have diameter D, shows that Lemma 18 is tight.

▶ **Theorem 19.** *If a set S of points of a convex surface has integer distances and diameter D, then $|S| = O(D^{4/3})$.*

Proof. Let n be the size of S, and let its two closest such points be p and q, with distance $O(D/\sqrt{n})$ by Lemma 18. We distinguish two cases:

- Suppose that all triples of points in S that include both p and q belong to shortest curves. Each point in S must belong either to a ray from p through q or a ray from q through p, with the ray containing a shortest curve from p to q. Further, this shortest curve (and hence these two rays) must be unique, for if there were two shortest curves between p and q, one of them could be combined with part of a shortest curve from p or q to a third point r to form a shortest curve that passes through q or p at a non-straight angle, an impossibility. Thus, two rays contain all remaining points in S, and these points lie at integer spacing along these two rays, giving $O(D)$ points in S in total.
- Otherwise, some point r does not lie on a shortest curve with p and q. By Theorem 14, there can be $O(D^2/\sqrt{n})$ points in S. That is, $n = O(D^2/\sqrt{n})$. Multiplying both sides by $\sqrt{n}$ and taking the 2/3 power gives the result. ◀

6 Discussion

We have shown that integer-distance point sets are finite or collinear in three broad classes of metric spaces each generalizing Euclidean distance in the plane. The next natural direction to look for a further extension of these results is in higher dimensions. Erdős and Anning state without proof that their results extend to d-dimensional Euclidean space [1]. However, the obvious attempt to generalize our proof to higher dimensions, by using intersection patterns of $(d+1)$-tuples of Voronoi cells, does not work for convex distance functions: there exist three-dimensional convex distance functions for which some 4-tuple of points is equidistant from arbitrarily many other points [15]. For Riemannian manifolds, the theorem does not generalize, as Example 7 shows. Nevertheless, it would be interesting to study similar questions for higher-dimensional uniform metrics, such as hyperbolic space. It would also be of interest to strengthen the bound on diameter in Theorem 11 and Theorem 19.

References

1. Norman H. Anning and Paul Erdős. Integral distances. *Bulletin of the American Mathematical Society*, 51(8):598–600, 1945. doi:10.1090/S0002-9904-1945-08407-9.
2. Peter F. Ash and Ethan D. Bolker. Generalized Dirichlet tessellations. *Geometriae Dedicata*, 20(2):209–243, 1986. doi:10.1007/BF00164401.
3. Jeffrey M. Augenbaum and Charles S. Peskin. On the construction of the Voronoi mesh on a sphere. *Journal of Computational Physics*, 59(2):177–192, 1985. doi:10.1016/0021-9991(85)90140-8.
4. Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. Hyperbolic Delaunay complexes and Voronoi diagrams made practical. *Journal of Computational Geometry*, 5(1):56–85, 2014. doi:10.20382/JOCG.V5I1A4.
5. L. Paul Chew and Robert L. (Scot) Drysdale, III. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–87, 1981. doi:10.1137/0210006.
6. L. Paul Chew and Robert L. (Scot) Drysdale, III. Voronoi diagrams based on convex distance functions. In Joseph O'Rourke, editor, *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5–7, 1985*, pages 235–244. Association for Computing Machinery, 1985. doi:10.1145/323233.323264.
7. Robert Connelly. *Convex Polyhedra* by A. D. Alexandrov. *SIAM Review*, 48(1):157–160, March 2006. doi:10.1137/SIREAD000048000001000149000001.
8. Olivier Devillers, Stefan Meiser, and Monique Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proceedings of the 4th Canadian Conference on Computational Geometry (CCCG '92)*, pages 263–268, 1992. URL: https://cccg.ca/proceedings/1992/Paper43.pdf.
9. Paul Erdős. Integral distances. *Bulletin of the American Mathematical Society*, 51:996, 1945. doi:10.1090/S0002-9904-1945-08490-0.
10. Leonhard Euler. Fragmenta arithmetica ex Adversariis mathematicis deprompta, C: Analysis Diophantea. In *Opera postuma*, volume I, pages 204–263. Eggers, Petropolis, 1862. Theorem 65, p. 229. URL: https://scholarlycommons.pacific.edu/euler-works/806/.
11. Rachel Greenfeld, Marina Iliopoulou, and Sarah Peluse. On integer distance sets. Electronic preprint arxiv:2401.10821, 2024.
12. Richard K. Guy. An olla-podrida of open problems, often oddly posed. *The American Mathematical Monthly*, 90(3):196–200, 1983. doi:10.2307/2975549.
13. Heiko Harborth. Integral distances in point sets. In *Charlemagne and his Heritage: 1200 Years of Civilization and Science in Europe, Vol. 2 (Aachen, 1995)*, pages 213–224. Brepols, Turnhout, Belgium, 1998. doi:10.1484/M.STHS-EB.4.2017040.
14. H. Hopf and W. Rinow. Ueber den Begriff der vollständigen differentialgeometrischen Fläche. *Commentarii Mathematici Helvetici*, 3(1):209–225, 1931. doi:10.1007/BF01601813.
15. Christian Icking, Rolf Klein, Ngọc-Minh Lê, and Lihong Ma. Convex distance functions in 3-space are different. *Fundamenta Informaticae*, 22(4):331–352, 1995. doi:10.3233/FI-1995-2242.
16. Menelaos I. Karavelas and Ioannis Z. Emiris. Root comparison techniques applied to computing the additively weighted Voronoi diagram. In *Proceedings of the Fourteenth Annual ACM–SIAM Symposium on Discrete Algorithms, January 12–14, 2003, Baltimore, Maryland, USA*, pages 320–329. ACM and SIAM, 2003. URL: https://dl.acm.org/citation.cfm?id=644108.644161.
17. Ngọc-Minh Lê. On non-smooth convex distance functions. *Information Processing Letters*, 63(6):323–329, 1997. doi:10.1016/S0020-0190(97)00136-1.
18. Greg Leibon and David Letscher. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In Siu-Wing Cheng, Otfried Cheong, Pankaj K. Agarwal, and Steven Fortune, editors, *Proceedings of the Sixteenth Annual Symposium on Computational Geometry, Clear Water Bay, Hong Kong, China, June 12–14, 2000*, pages 341–349. ACM, 2000. doi:10.1145/336154.336221.

19 Jeremy Mann. Equilateral dimension of Riemannian manifolds with bounded curvature. *Rose-Hulman Undergraduate Mathematics Journal*, 14(1):Article 8, 2013. URL: `https://scholar.rose-hulman.edu/rhumj/vol14/iss1/8/`.

20 M. Mazón and T. Recio. Voronoi diagrams on orbifolds. *Computational Geometry: Theory & Applications*, 8(5):219–230, 1997. `doi:10.1016/S0925-7721(96)00017-X`.

21 R. E. Miles. Random points, sets and tessellations on the surface of a sphere. *Sankhyā, Series A*, 33:145–174, 1971. URL: `https://www.jstor.org/stable/25049720`.

22 S. B. Myers. Arcs and geodesics in metric spaces. *Transactions of the American Mathematical Society*, 57:217–227, 1945. `doi:10.1090/S0002-9947-1945-0011792-9`.

23 Hyeon-Suk Na, Chung-Nim Lee, and Otfried Cheong. Voronoi diagrams on the sphere. *Computational Geometry, Theory and Applications*, 23(2):183–194, 2002. `doi:10.1016/S0925-7721(02)00077-9`.

24 Frank Nielsen and Richard Nock. Hyperbolic Voronoi diagrams made easy. In Bernady O. Apduhan, Osvaldo Gervasi, Andrés Iglesias, David Taniar, and Marina L. Gavrilova, editors, *Proceedings of the 2010 International Conference on Computational Science and Its Applications, ICCSA 2010, Fukuoka, Japan, March 23–26, 2010*, pages 74–80. IEEE Computer Society, 2010. `doi:10.1109/ICCSA.2010.37`.

25 József Solymosi. Note on integral distances. *Discrete & Computational Geometry*, 30(2):337–342, 2003. `doi:10.1007/s00454-003-0014-7`.

26 Matthias Weber, David Hoffman, and Michael Wolf. An embedded genus-one helicoid. *Proceedings of the National Academy of Sciences*, 102(46):16566–16568, 2005. `doi:10.1073/pnas.0508008102`.

Shelling and Sinking Graphs on the Sphere

Jeff Erickson
University of Illinois Urbana-Champaign, IL, USA

Christian Howard
University of Illinois Urbana-Champaign, IL, USA

Abstract

We describe a promising approach to efficiently morph spherical graphs, extending earlier approaches of Awartani and Henderson [*Trans. AMS* 1987] and Kobourov and Landis [*JGAA* 2006]. Specifically, we describe two methods to morph shortest-path triangulations of the sphere by moving their vertices along longitudes into the southern hemisphere; we call a triangulation *sinkable* if such a morph exists. Our first method generalizes a longitudinal shelling construction of Awartani and Henderson; a triangulation is sinkable if a specific orientation of its dual graph is acyclic. We describe a simple polynomial-time algorithm to find a longitudinally shellable rotation of a given spherical triangulation, if one exists; we also construct a spherical triangulation that has no longitudinally shellable rotation. Our second method is based on a linear-programming characterization of sinkability. By identifying its optimal basis, we show that this linear program can be solved in $O(n^{\omega/2})$ time, where ω is the matrix-multiplication exponent, assuming the underlying linear system is non-singular. Finally, we pose several conjectures and describe experimental results that support them.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases morphing, planar graphs, spherical graph drawing, longitudinal shelling

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.47

Related Version *Full Version*: `https://arxiv.org/abs/2504.05098` [24]

Acknowledgements The authors thank Eli Kujawa for early helpful discussions and the anonymous reviewers for their helpful comments and suggestions.

1 Introduction

A *morph* between two planar straight-line graphs is a continuous deformation from one to the other, such that all edges in all intermediate graphs are interior-disjoint line segments. Planar graph morphing has many applications in graphics, animation, visualization, and geometric modeling, as well as connections to fundamental questions in low-dimensional topology.

There are two state-of-the-art approaches for morphing planar graphs. The first is the barycentric interpolation method of Floater, Gotsman, and Surazhsky [26, 29, 31, 56–58], which is based on an extension by Floater [27, 28] of Tutte's classical spring-embedding theorem [60]. These algorithms compute an implicit representation of a smooth morph, any intermediate stage of which can be constructed in $O(n^{\omega/2}) = O(n^{1.1864})$ time, where $\omega < 2.3728$ is the matrix-multiplication exponent. The second method combines an edge-collapsing strategy originally proposed by Cairns [11,12,35,59] with more recent algorithms for convex hierarchical graph drawing [13, 17, 36, 39, 40]. This method has led to several efficient algorithms for constructing explicit representations of piecewise-linear morphs [2, 25, 39, 40]. Algorithms are also known for several variants of planar morphing [5, 6, 10, 15, 43, 61].

Both planar morphing approaches have recently been generalized to geodesic graphs on the flat torus [14, 25, 44], and very recently, Luo, Wu, and Zhu generalized the barycentric interpolation method to arbitrary negative-curvature surfaces of higher genus [45, 46].

© Jeff Erickson and Christian Howard;
licensed under Creative Commons License CC-BY 4.0

1.1 What about the sphere?

In light of the long history of results for morphing planar graphs and their generalizations to higher-genus surfaces, it is surprising how little is known about morphing graphs on the sphere. Although embeddings on the sphere are *topologically* equivalent to embeddings in the plane, the different geometries of the two surfaces induce significantly different behavior.

In 1944, Cairns [12] proved that any two isomorphic shortest-path triangulations of the sphere are connected by a continuous family of shortest-path triangulations, using essentially the same edge-contraction strategy that he used for planar triangulations (but with more complex case analysis). A direct translation of Cairns's spherical morphing proof leads to an exponential-time algorithm; unlike his planar result, this is still the fastest algorithm known. In fact, as far as we are aware, this is the *only* algorithm known for morphing arbitrary spherical triangulations.

Morphing algorithms are known for a few special cases of spherical triangulations. Almost all of these algorithms operate by moving vertices along longitudes into the southern hemisphere, projecting the resulting "southern" triangulation into the plane, and applying a planar morphing algorithm. Awartani and Henderson [4] describe an algorithm to morph spherical triangulations that satisfy a certain **longitudinal shelling** condition using this strategy. They also prove that any triangulation with a *longitudinal seam*, meaning there is a longitude that does not cross the interior of any edge, satisfies their shelling condition. (We study Awartani and Henderson's shelling condition in more detail in Section 3.) Kobourov and Landis [41] describe an algorithm to morph between *Delaunay* triangulations via longitudinal morphs; their algorithm easily generalizes to **coherent** triangulations, which are central projections of arbitrary convex polyhedra. The same method is also implicit in Richter-Gebert's proof [47, Theorem 13.3.3] of classical theorems of Eberhard [21] and Steinitz [50] describing morphs between isomorphic convex polyhedra.

We call a sphere triangulation **sinkable** if it can be morphed along longitudes into the southern hemisphere. Every coherent triangulation is longitudinally shellable, and every longitudinally shellable triangulation is sinkable. Awartani and Henderson [4] describe an example of an unsinkable triangulation; it is not possible to move the vertices of their triangulation into the southern hemisphere without inverting at least one face. The simplest examples of unsinkable triangulation are the central projections of sufficiently twisted *Schönhardt polyhedra* [48], where the rotated faces are normal to the z-axis; see Figure 1.1. (Supnick proved that Schönhardt triangulations are the simplest non-coherent triangulations [54, 55]; our algorithm in Section 5 can be used to determine which of them are sinkable.)

The definitions of longitudinal shellability and sinkability depend on a specific choice of antipodal points as the north and south poles; neither property is invariant under rotations of the sphere. For example, a 90° rotation around the x-axis makes every Schönhardt triangulation in Figure 1.1 longitudinally shellable, and therefore sinkable.

1.2 Our Results

We describe a promising approach to efficiently morph shortest-path embeddings of graphs on the sphere, extending the previous results of Awartani and Henderson [4] and Kobourov and Landis [41]. At a high level, we propose finding suitable rotations of the sphere that allow the source and target embeddings to be pushed into the southern hemisphere along longitudes, thereby reducing to a planar morphing problem.

We emphasize that we do *not* develop this strategy into a complete algorithm; the existence of an efficient spherical morphing algorithm remains an open problem. We also note that sinking is not *required* to construct spherical morphs; for example, we can directly morph any Schönhardt triangulation into any other by twisting one face.

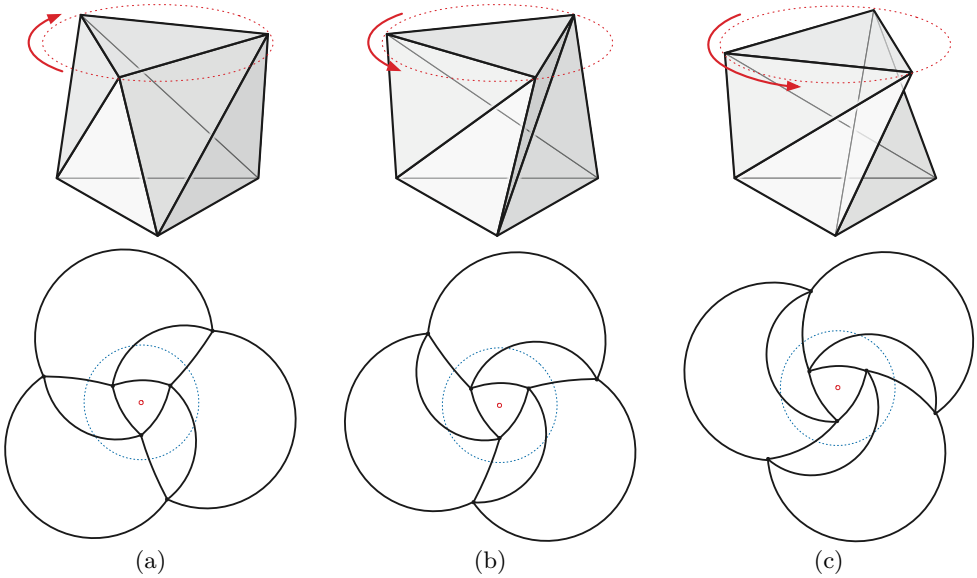


Figure 1.1 Three Schönhardt polyhedra and stereographic projections of the corresponding spherical triangulations. Depending on the twisting angle, these triangulations are (a) coherent and therefore longitudinally shellable, (b) sinkable but not longitudinally shellable, or (c) not sinkable.

We begin in Section 2 by reviewing relevant definitions, describing our proposed morphing strategy in detail, and proving some preliminary results. In Section 3, we formally define the *longitudinal shelling* condition introduced by Awartani and Henderson and provide several combinatorial characterizations, each of which can be tested in $O(n)$ time. We also reiterate Awartani and Henderson's proof that every longitudinally shellable triangulations is sinkable. In Section 4, we present an algorithm that either finds a rotation of a given sphere triangulation Γ that is longitudinally shellable, or reports correctly that no such rotation exists, in $O(n^{5/2}\log^3 n)$ time. We also construct a spherical triangulation that has no longitudinally shellable rotation. In Section 5, we describe an efficient algorithm to determine whether a given sphere triangulation is *sinkable*. First we show that sinkability is equivalent to the feasibility of a certain linear program with $O(n)$ variables and constraints. By identifying the optimal basis for this linear program, we show that it can be solved in $O(n^{\omega/2})$ time, assuming the underlying linear system is non-singular.

Due to space constraints, we defer several proofs and related results to the full version [24]. In particular, we describe the results of lightweight experiments, which suggest that "in practice" one can find a shellable rotation of a spherical triangulation by testing a small number of random rotations. We also present several open problems for further research, including several conjectures suggested by our experimental results.

2 Background and Preliminary Results

2.1 Cartography

We consider drawings of graphs on the unit sphere $S^2 = \{(x, y, z) \mid x^2 + y^2 + z^2 = 1\}$. The **north pole** is the point $(0, 0, 1)$; the **south pole** is the point $(0, 0, -1)$; the **equator** is the intersection of the unit sphere with the plane $z = 0$. A **longitude** is an open great-circular arc whose endpoints are the north and south poles. Every point except the poles lies on a unique longitude.

Throughout the paper, we fix an arbitrary simple maximal planar graph G with $n \geq 4$ vertices, arbitrarily indexed from 1 to n. An **embedding** of G on the sphere is an injective map from G to S^2, or less formally, a drawing of G where edges are arbitrary simple curves that intersect only at their shared endpoints. Every embedding of G on the sphere is a **triangulation**, meaning every face is bounded by a cycle of three distinct edges.

Unless explicitly indicated otherwise, we consider only **generic shortest-path** triangulations, meaning (1) every edge is a great-circular arc with length less than π; (2) no great circle contains more than two vertices, and (3) no great circle through the poles contains more than one vertex. Condition (1) is the natural spherical analogue of straight-line embeddings in the plane. Condition (3) implies that vertices lie on distinct longitudes.

Classical theorems of Steinitz [33, 49, 50] and Whitney [8, 62] imply that G has a unique embedding on the sphere, up to homeomorphism, which is a generic shortest-path triangulation. A **face of G** is a face of this essentially unique embedding. We sometimes identify each face as a triple (i, j, k) of vertices (or their indices) in counterclockwise order around its interior.

Every generic shortest-path triangulation has a unique **north face** whose interior contains the north pole and a unique **south face** whose interior contains the south pole. The remaining **non-polar** faces are of two types: An **up-face** has one vertex (called its **apex**) directly north of its opposite edge (called its **base**); symmetrically, a **down-face** has one vertex (its apex) directly south of its opposite edge (its base). The **legs** of a non-polar face f are the edges of f incident to the apex of f. The faces immediately north and south of any vertex i are respectively denoted $\boldsymbol{f_{\mathsf{above}}(i)}$ and $\boldsymbol{f_{\mathsf{below}}(i)}$. For each vertex i, $f_{\mathsf{above}}(i)$ is either the north face or a down-face, and $f_{\mathsf{below}}(i)$ is either the south face or an up-face. See Figure 2.1.

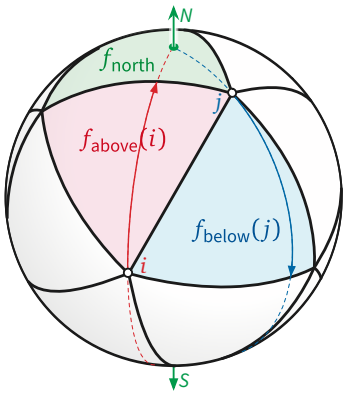


Figure 2.1 An up-face $f_{\mathsf{above}}(i)$, a down-face $f_{\mathsf{below}}(j)$, and the north face $f_{\mathsf{north}} = f_{\mathsf{above}}(j)$ of a spherical triangulation. Edge ij is a leg of both $f_{\mathsf{above}}(i)$ and $f_{\mathsf{below}}(j)$.

A face of a (generic) triangulation is **everted** if it has area greater than 2π, or equivalently, if it contains an open hemisphere. Every triangulation contains at most one everted face. We call a triangulation **full** if no face is everted, and **southern** if every vertex lies below the equator (which implies that the north face is everted).

A connected region R on the sphere is **θ-monotone** if every longitude is either disjoint from R or has connected intersection with R.

2.2 Coordinates

Throughout the paper, we represent spherical triangulations *implicitly* as projections of certain polyhedra onto the unit sphere. Specifically, we represent each vertex using signed homogeneous coordinates [51–53] – for any scalar $\lambda > 0$, the coordinate vectors (x, y, z) and $(\lambda x, \lambda y, \lambda z)$ represent the same point on the sphere. We can freely rescale the vertices, each independently, without changing the underlying spherical triangulation; although it is never actually necessary, this rescaling is sometimes convenient for purposes of discussion and intuition. For example, we can implicitly interpret any southern triangulation as a *planar* triangulation by scaling vertices onto the tangent plane $z = -1$; this scaling is equivalent to the *gnomonic* or *central* projection $(x, y, z) \mapsto (-x/z, -y/z, -1)$. Awartani and Henderson [4] analyze longitudinal morphing (defined below) by implicitly scaling vertices onto the unit cylinder $x^2 + y^2 = 1$.

2.3 Longitudinal Morphing

Two embeddings $\Gamma_0 \colon G \to S^2$ and $\Gamma_1 \colon G \to S^2$ are **isomorphic** if they define the same rotation system, or equivalently, if there is an orientation-preserving homeomorphism $H \colon S^2 \to S^2$ such that $\Gamma_1 = h \circ \Gamma_0$. Two isomorphic embeddings are **θ-equivalent** if each vertex of G lies on the same longitude in both embeddings.

A **homotopy** between two embeddings $\Gamma_0 \colon G \to S^2$ and $\Gamma_1 \colon G \to S^2$ is a continuous function $H \colon [0, 1] \times G \to S^2$ such that $H(0, \cdot) = \Gamma_0$ and $H(1, \cdot) = \Gamma_1$. Any pair of isomorphic embeddings Γ_0 and Γ_1 are connected by an **isotopy**, which is a homotopy H such that every intermediate drawing $\Gamma_t = H(t, \cdot)$ is an embedding. (The edges of these intermediate embeddings can be arbitrary simple curves, not necessarily shortest paths.) A **morph** is an isotopy in which the edges of every intermediate embedding Γ_t are shortest paths. Finally, a **longitudinal morph** is a morph in which vertices move only along their longitudes.

▶ **Lemma 2.1.** *Any two θ-equivalent triangulations are connected by a longitudinal morph.*

Proof sketch. Let Γ_0 and Γ_1 be two θ-equivalent triangulations. By scaling coordinate vectors as described in Section 2.2 if necessary, we can assume that each vertex has the same x- and y-coordinates in both embeddings. We define a longitudinal morph by linearly interpolating the z-coordinates of the vertices. The signed volume spanned by the vertices of any face is a linear function of the interpolation parameter t. Each non-polar face has the same orientation in Γ_0 and Γ_1, and therefore in every intermediate embedding. (See the full version [24] for a more detailed proof.) ◀

▶ **Corollary 2.2.** *A triangulation Γ is sinkable if and only if there is a southern triangulation that is θ-equivalent to Γ.*

We extend this argument slightly by considering **weak triangulations**, which are drawings of G that may include degenerate faces, but no inverted faces or crossing edges. (Weak triangulations are special cases of *weak embeddings* studied by Akitaya, Fulek, and Tóth [1] and Fulek and Kyncl [30].) A weak triangulation Γ_1 is θ-equivalent to a triangulation Γ_0 if vertices lie on the same longitudes in Γ_0 and Γ_1 and every non-degenerate non-polar face of Γ_i has the same orientation as the corresponding face of Γ_0.

▶ **Corollary 2.3.** *A triangulation Γ is sinkable if any only if there is a southern weak triangulation that is θ-equivalent to Γ.*

2.4 Morphing Strategy

Recall that a triangulation is **sinkable** if it can be longitudinally morphed to a θ-equivalent triangulation with all vertices below the equator. A **rotation** of a spherical triangulation Γ is the triangulation $\rho \circ \Gamma$ for some rigid motion $\rho\colon S^2 \to S^2$ of the sphere.[1] Our strategy rests on the following (admittedly optimistic) conjecture:

▶ **Conjecture 2.4.** *Every shortest-path triangulation Γ of the sphere can be rotated to obtain a sinkable triangulation.*

Conjecture 2.4 implies that one can morph any triangulation Γ_0 into any isotopic target triangulation Γ_1 through an intermediate coherent triangulation $\Gamma_{1/2}$. Steinitz's theorem [33, 49, 50] implies that a coherent embedding $\Gamma_{1/2}$ of G exists. Given any coherent triangulation, we can project its underlying convex polyhedron from a point on the positive z-axis just outside the polyhedron to the plane $z = -1$, and then centrally project the resulting planar triangulation back to the southern hemisphere, to obtain a θ-equivalent southern triangulation. It follows that *every* rotation of a coherent triangulation is sinkable.

We construct a morph from Γ_0 to $\Gamma_{1/2}$ in several stages as follows. First, we rotate Γ_0 so that the resulting triangulation Γ'_0 is sinkable (as guaranteed by Conjecture 2.4) and then longitudinally morph Γ'_0 to a southern triangulation Γ''_0.[2] Similarly, we rotate $\Gamma_{1/2}$ so that the resulting triangulation $\Gamma'_{1/2}$ has the same north face as Γ'_0 and Γ''_0, and then longitudinally morph $\Gamma'_{1/2}$ to a southern triangulation $\Gamma''_{1/2}$. Projection from the center of the sphere maps Γ''_0 and $\Gamma''_{1/2}$ to isomorphic straight-line triangulations in the plane $z = -1$, each with the same triangular outer face. We can construct a morph between these planar triangulations using any of several algorithms [2, 7, 11, 25, 29, 34, 40]; lifting this planar morph back to the sphere yields a spherical morph from Γ''_0 to $\Gamma''_{1/2}$. Our final morph is the concatenation of the rotation $\Gamma_0 \rightsquigarrow \Gamma'_0$, the longitudinal morph $\Gamma'_0 \rightsquigarrow \Gamma''_0$, the lifted planar morph $\Gamma''_0 \rightsquigarrow \Gamma''_{1/2}$, the reverse of the longitudinal morph $\Gamma'_{1/2} \rightsquigarrow \Gamma''_{1/2}$, and the reverse of the rotation $\Gamma_{1/2} \rightsquigarrow \Gamma'_{1/2}$. See Figure 2.2 for an example (without the initial and final rotations).

We construct a morph $\Gamma_1 \rightsquigarrow \Gamma_{1/2}$ using the same strategy. The final morph $\Gamma_0 \rightsquigarrow \Gamma_1$ is the concatenation of $\Gamma_0 \rightsquigarrow \Gamma_{1/2}$ and the reverse of $\Gamma_1 \rightsquigarrow \Gamma_{1/2}$.

3 Longitudinal Shelling

A **shelling** of a spherical triangulation Γ is an ordering $f_1, f_2, \ldots, f_{2n-4}$ of the faces of Γ such that the union of any prefix $U_k = \bigcup_{i \le k} f_i$ is either empty ($k = 0$), the entire sphere ($k = 2n - 4$), or a topological disk. We call a shelling **longitudinal** if every disk U_k is θ-monotone. Bruggesser and Mani's line shelling [9] implies that every *coherent* triangulation is longitudinally shellable.

3.1 Characterization

We characterize the shellability of Γ in terms of three directed graphs, which are illustrated in Figure 3.1.

[1] Rotations should not be confused with *rotation systems*, which encodes the cyclic order of edges around each vertex in a planar or spherical embedding.

[2] If the triangulation Γ_0 is not full, we can rotate it directly to a southern triangulation $\Gamma'_0 = \Gamma''_0$.

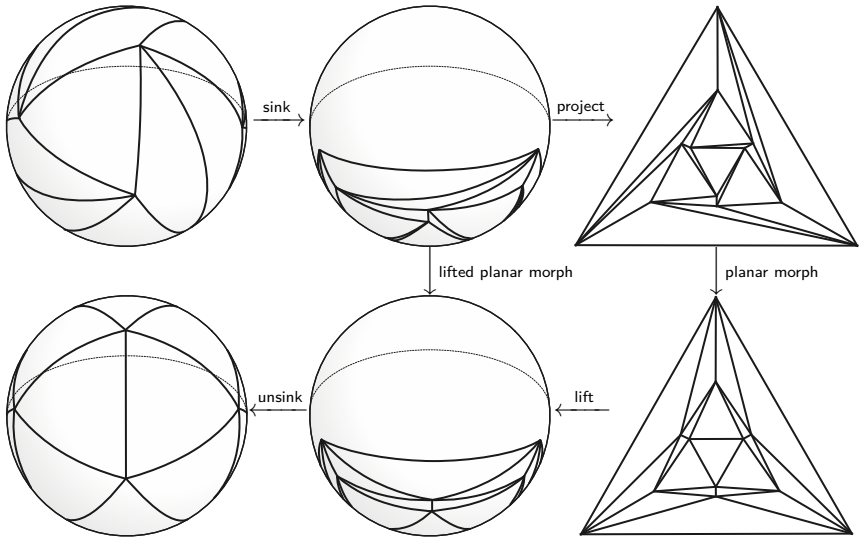


Figure 2.2 Morphing a six-beaked shaddock [19, 38] into a regular icosahedron; compare with Kobourov and Landis [41, Figure 1].

- The directed dual graph $\mathbf{\Gamma^{\downarrow}}$ contains a directed edge from face f_i to face f_j if and only if f_i and f_j share an edge in Γ, and some point in f_i is due north (on the same longitude) of some point in f_j.
- The oriented primal graph $\mathbf{\Gamma^{\rightarrow}}$ is the directed dual of $\Gamma^{\downarrow}$. For every undirected edge ij of Γ, the graph $\Gamma^{\rightarrow}$ contains the directed edge $i{\rightarrow}j$ if and only if vertex j is east of vertex i, or more formally, if the determinant

$$\det \begin{pmatrix} 0 & 0 & 1 \\ x_i & y_i & z_i \\ x_j & y_j & z_j \end{pmatrix} = \det \begin{pmatrix} x_i & y_i \\ x_j & y_j \end{pmatrix}$$

  is positive, where (x_i, y_i, z_i) is the coordinate vector for vertex i.
- Finally, $\mathbf{\Gamma^{\swarrow}}$ is a directed proper subgraph of Γ whose edges are the legs of each down-face, each directed toward its apex.

We defer the proof of the following lemma to the full version [24].

▶ **Lemma 3.1.** *For every generic shortest-path triangulation Γ, the following conditions are equivalent:*

(a) Γ *is longitudinally shellable.*
(b) $\Gamma^{\downarrow}$ *is acyclic.*
(c) $\Gamma^{\rightarrow}$ *is strongly connected.*
(d) $\Gamma^{\rightarrow}$ *contains directed paths from f_{north} to f_{south} and from f_{south} to f_{north}.*
(e) $\Gamma^{\swarrow}$ *is acyclic.*

Each of the conditions (b), (c), (d), and (e) can be tested in $O(n)$ time using textbook graph algorithms. A straightforward extension of Lemma 3.1 to triangulations with vertical edges implies a theorem of Awartani and Henderson [4]: Every triangulation with *longitudinal seam* is longitudinally shellable. See the full version [24] for more details.

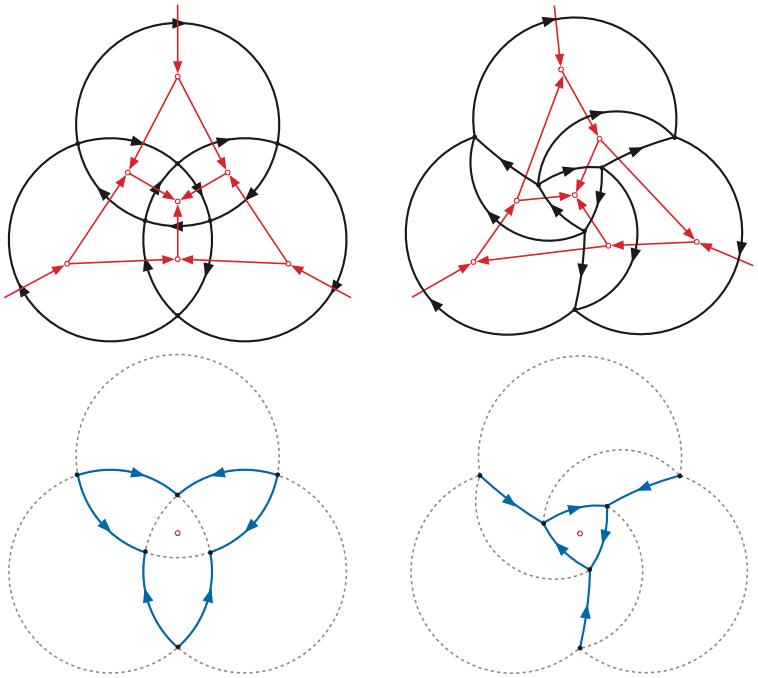


Figure 3.1 Stereographic projections of directed graphs $\Gamma^{\downarrow}$ (red straight edges), $\Gamma^{\rightarrow}$ (black curved edges), and $\Gamma^{\searrow\swarrow}$ (blue, second row) for the Platonic octahedral triangulation (which is longitudinally shellable) and a Schönhardt triangulation (which is not). Compare with Figure 1.1.

3.2 The Awartani–Henderson Embedding

▶ **Theorem 3.2** (Awartani and Henderson [4]). *Every longitudinally shellable triangulation of the sphere is sinkable. Moreover, given any longitudinally shellable triangulation Γ, we can compute a longitudinal morph from Γ to a southern triangulation in $O(n)$ time.*

Proof sketch. Let Γ be a longitudinally shellable triangulation. We construct a southern weak triangulation Γ' that is θ-equivalent to Γ by considering vertices in any topological order of $\Gamma^{\searrow\swarrow}$. The first three vertices lie on the north face and thus can be placed anywhere below the equator on their respective longitudes. For each later vertex i, we make z'_i as large as possible, such that no face induced by vertices 1 through i (except the north face) is inverted. Every down-face in the resulting drawing Γ' is degenerate. The theorem now follows from Corollary 2.3. (We give a more detailed proof, more closely following Awartani and Henderson [4], in the full version [24].) ◀

4 Longitudinally Shellable Rotations

When a given triangulation Γ is not longitudinally shellable, we can still apply our morphing strategy if we can find a rotation of the sphere that transforms Γ into a longitudinally shellable triangulation. In this section, we describe an efficient algorithm that either finds such a rotation or correctly reports that no such rotation exists. Our experimental results (described in the full version [24]) suggest that "in practice", even for triangulations with many long skinny triangles, a significant fraction of rotations of Γ are longitudinally shellable, so it suffices to consider a small number of random rotations. In fact, every one of the thousands of random adversarial triangulations we generated had shellable rotations. Nevertheless, in Section 4.2, we construct a triangulation with no longitudinally shellable rotation.

4.1 Finding a Shelling Direction

Instead of considering different rotations of Γ, it is convenient to keep Γ fixed and consider different locations for the "north pole". We call a unit vector p a **shelling direction** for Γ if any (and therefore every) rotation of the sphere that takes p to the standard north pole $(0, 0, 1)$ also takes Γ to a longitudinally shellable triangulation. We similarly define the directed graphs $\Gamma^{\downarrow}(p)$ and $\Gamma^{\rightarrow}(p)$ and $\Gamma^{\searrow\swarrow}(p)$. For example, for any unit vector $p = (x, y, z)$, the graph $\Gamma^{\rightarrow}(p)$ contains the edge $i{\rightarrow}j$ if and only if

$$\mathrm{vol}(p, i, j) := \det \begin{pmatrix} x & y & z \\ x_i & y_i & z_i \\ x_j & y_j & z_j \end{pmatrix} > 0,$$

and p is a shelling direction for Γ if and only if the graphs $\Gamma^{\downarrow}(p)$ and $\Gamma^{\searrow\swarrow}(p)$ are acyclic.

▶ **Theorem 4.1.** *Given any shortest-path triangulation Γ on the sphere, we can either compute a shelling direction for Γ or report correctly that no such direction exists, in $O(n^{2.5} \log^3 n)$ time.*

Proof. Each edge ij in Γ lies on a unique great circle; let $\mathcal{A}(\Gamma)$ denote the arrangement of all $3n - 6$ such great circles. For any two unit vectors p and q, the graphs $\Gamma^{\rightarrow}(p)$ and $\Gamma^{\rightarrow}(q)$ are identical if and only if p and q lie in the same cell of $\mathcal{A}(\Gamma)$. Thus, for any cell C, we can write $\Gamma^{\rightarrow}(C)$ to denote the graph $\Gamma^{\rightarrow}(p)$ for any $p \in C$. If C and C' are adjacent two-dimensional cells of $\mathcal{A}(\Gamma)$, then $\Gamma^{\rightarrow}(C)$ and $\Gamma^{\rightarrow}(C')$ differ in the direction of exactly one edge.

We can find a shelling direction for Γ in $O(n^3)$ time by constructing the great-circle arrangement $\mathcal{A}(\Gamma)$, and then for each two-dimensional cell C, check whether $\Gamma^{\rightarrow}(C)$ is strongly connected. (This arrangement is centrally symmetric, and the gnomonic projection of either hemisphere to any tangent plane is an arrangement of *lines*. Thus, we can use any classical algorithm to construct line arrangements [16, 22, 23].

We can speed up this naive algorithm using a data structure of Diks and Sankowski for reachability queries in dynamic directed plane graphs [18]. Diks and Sankowski's data structure maintains a directed planar graph with a fixed embedding, supports edge insertions and deletions that do not change the embedding in $O(\sqrt{n} \log^3 n)$ time, and supports queries of the form "Is there a directed path from vertex i to vertex j?" in $O(\sqrt{n} \log^2 n)$ time.

To use this data structure, we traverse the dual graph of the arrangement $\mathcal{A}(\Gamma)$. At each two-dimensional cell C, we perform two reachability queries in $\Gamma^{\rightarrow}(C)$ between the two polar faces. If both reachability queries succeed, then by Lemma 3.1(d), cell c contains a shelling direction. When we move from a cell C to a neighboring cell C', we delete one edge and insert its reversal. Altogether, we perform $O(n^2)$ queries and $O(n^2)$ updates. ◀

4.2 Unshellable From Every Direction

Regard each edge of the undirected dual graph Γ^* as a pair of opposing directed edges or **darts**. For any directed cycle γ^* of darts in Γ^*, the **polar region $P(\gamma^*)$** is the set of all north poles p such that the directed dual graph $\Gamma^{\downarrow}(p)$ contains every dart in γ^*. A unit vector p is a shelling direction for Γ if and only if $p \notin P(\gamma^*)$ for *every* directed dual cycle γ^*.

▶ **Lemma 4.2.** *For every directed cycle γ^* of darts in the dual graph Γ^*, the corresponding polar region $P(\gamma^*)$ is the interior of a convex spherical polygon, that is, the intersection of S^2 with a (possibly empty) open convex polyhedral cone.*

Proof. Let $i{\to}j$ be any dart in Γ, and let f and g be the faces incident to $i{\to}j$ on the left and right, respectively. The directed graph $\Gamma^{\downarrow}(p)$ contains the edge $f{\to}g$ if and only if $\mathrm{vol}(p,i,j) > 0$. The set of all points p satisfying this inequality is an open hemisphere $H(f{\to}g)$ bounded by the great circle through $i{\to}j$. Finally, the polar region $P(\gamma^*)$ of any dual cycle γ^* is the intersection of the hemispheres $H(f{\to}g)$ for all $f{\to}g \in \gamma^*$. ◀

A **rotor** is the subset of faces of Γ dual to any directed cycle γ^* in the dual graph Γ^*. We sketch the construction of a triangulation Γ containing a small number of rotors, whose polar regions cover the entire sphere. We describe our construction in terms of an arbitrary sufficiently small angular parameter $\varepsilon > 0$; in practice, it suffices to set $\varepsilon \approx 0.01 \approx 0.5°$.

First we define a simple *equatorial rotor*, which triangulates a belt of width $O(\varepsilon^2)$ around a great circle using $O(1/\varepsilon)$ isosceles triangles with aspect ratio $O(\varepsilon)$ and with all edges at angle $O(\varepsilon)$ from the great circle. The triangulation edges are consistently oriented so that for any north pole p sufficiently far from the rotor, the rotor defines a cycle in the directed dual graph $\Gamma^{\downarrow}(p)$. The polar region of (one orientation of) the rotor is a convex spherical polygon whose boundary has Hausdorff distance $O(\varepsilon)$ from the rotor itself. In particular, in the limit as ε approaches zero, this polar region approaches an open hemisphere.

Figure 4.1 Local structure of an equatorial rotor.

Our unshellable triangulation contains four modified equatorial rotors, whose central great circles are parallel to the faces of a regular tetrahedron. Let Q denote the arrangement of these four great circles; Q is the central projection of an inscribed semi-regular cuboctahedron; see Figure 4.2. Each pair of great circles meets at an angle of $\arccos(1/3) \approx 70.529°$.

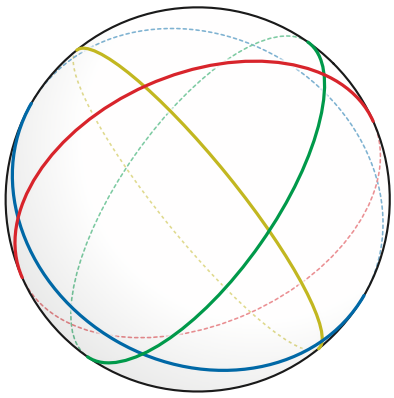
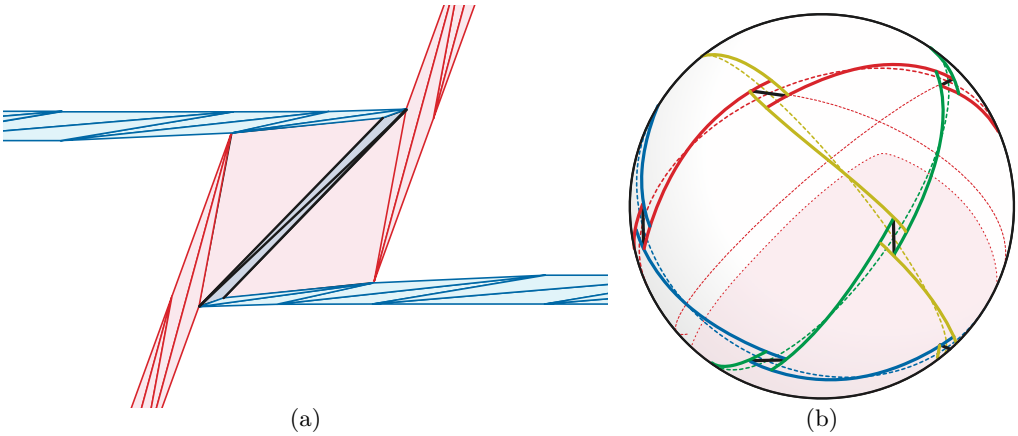


Figure 4.2 Four great circles defining a spherical cuboctahedron.

At each vertex of Q, we modify the two equatorial rotors by introducing a *crossing gadget*, illustrated in Figure 4.3(a). Overall the gadget resembles a rhombus with diameter $O(\varepsilon)$ whose edges are parallel to the great circles defining the rotors. Each of the equatorial rotors is broken and offset by $O(\varepsilon)$ to cover two opposite edges of this rhombus. Then three new edges are added to reconnect both rotors; these edges have distance and angle $O(\varepsilon)$ from the long diagonal of the rhombus and the bisector of the smaller angle between the two great circles. The two fat triangles inside the rhombus are incorporated into one of the two rotors (the vertical red rotor in Figure 4.3); the two thin triangles near the diagonal are incorporated into both. Figure 4.3(b) shows a schematic of our overall construction.



(a) (b)

Figure 4.3 (a) A single crossing gadget. (b) Constructing a triangulation with no shelling direction. Each color of broken lines is an equatorial rotor; each short black line is the diagonal of a crossing gadget. The polar region of one (red) equatorial rotor is shaded.

Now let γ^* be a directed cycle in the dual graph Γ^* through the faces of one equatorial rotor. The corresponding polar region $P(\gamma^*)$ nearly fills a triangular region bounded by three bisector circles, each defined by an antipodal pair of crossing gadgets. This polar region completely covers one triangular face and nearly half of three square faces of Q. The reversal of γ^* defines an antipodally symmetric polar region.

Thus, the union of the polar regions defined by all four equatorial rotors covers the entire sphere except for a small "hole" near the center of each square face of Q, which we can make arbitrarily small by choosing ε appropriately. We cover these holes by adding a small rotor, reminiscent of the projection of a Schönhardt polyhedron, inside each square face of Q; see Figure 4.4. Finally, to complete the triangulation Γ, we arbitrarily triangulate the area between the rotors.

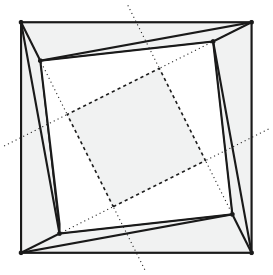


Figure 4.4 A square rotor (solid lines) and its polar region (dashed lines).

▶ **Theorem 4.3.** *There is a shortest-path triangulation of the sphere with no longitudinal shelling direction.*

Proof. By construction, every point $p \in S^2$ lies in the polar region of at least one rotor, and therefore in the polar region of at least one directed cycle in the dual graph Γ^*. Thus, for every $p \in S^2$, the directed dual graph $\Gamma^{\downarrow}(p)$ is not acyclic. The theorem follows immediately from Lemma 3.1. ◀

Nothing in our construction prevents our bad triangulation Γ from being *sinkable*. In particular, just before we add the final rotors to cover the "holes", the partial triangulation is still longitudinally shellable from any direction inside one of those holes. After rotating and sinking this partial triangulation and then projecting to the plane $z = -1$, we can add the final rotors to the resulting planar embedding. Moreover, experimental evidence [24] strongly suggests that every triangulation in which all edges have length less than $\pi/2$ is sinkable; our bad triangulation Γ satisfies this condition.

5 Sinking

Finally, we present an exact characterization of sinkable triangulations. Let Γ be our initial full shortest-path triangulation of the sphere. Recall from Corollary 2.3 that Γ is sinkable if and only if there is a southern weak triangulation Γ' that is θ-equivalent to Γ.

Without loss of generality we can consider only weak triangulations Γ' where each vertex has the same x- and y-coordinates as in Γ. Let (x_i, y_i, z_i) and (x_i, y_i, z'_i) denote the coordinates of vertex i in Γ and Γ', respectively. Then Γ' is θ-equivalent to Γ if and only if $\mathrm{vol}'(i,j,k) \geq 0$ for every non-polar face (i,j,k), where

$$\mathrm{vol}'(i,j,k) := \det \begin{pmatrix} x_i & y_i & z'_i \\ x_j & y_j & z'_j \\ x_k & y_k & z'_k \end{pmatrix}$$

We reiterate that because we have fixed all x- and y-coordinates, the volume determinant $\mathrm{vol}'(i,j,k)$ is a linear function of the vector z'.

▶ **Lemma 5.1.** *A spherical triangulation Γ is sinkable if and only if the following linear program is feasible:*

$$\begin{array}{lrll} \textit{maximize} & \sum_i z'_i & & \\ \textit{subject to} & z'_i = -1 & \textit{for every vertex } i \textit{ of } f_{\textit{north}} & (5.1) \\ & \mathrm{vol}'(i,j,k) \geq 0 & \textit{for every non-polar face } (i,j,k) & \end{array}$$

Proof. Suppose $z' = (z'_1, \ldots, z'_n)$ is the z-coordinate vector of a weak triangulation Γ' that is θ-equivalent to Γ, such that $z'_i < 0$ for all i. We can move the vertices of north face of Γ' to the plane $z = -1$ by applying a linear transformation that preserves the z-axis and all longitudes. The coordinate vector of every other vertex i is a positive linear combination of the north-face coordinate vectors, so $z'_i < 0$ for all i. ◀

Lemma 5.1 immediately implies an algorithm to compute a sinking longitudinal morph, or report correctly that none exists, in weakly polynomial time, provided the input x- and y-coordinates are rational [32]. We obtain a simpler and faster algorithm by identifying the optimal basis of the linear program, subject to a mild technical condition.

First we need a small extension of a result used in many planar morphing algorithms [17, 20, 37, 39, 40]. A polygon is *weakly convex* if its interior is convex and each of its vertices has interior angle *at most* π. Let T be any straight-line plane graph whose outer face is a simple x-monotone polygon and whose inner faces are triangles. A weakly convex polygon W is *compatible* with T if there is a homeomorphism from the boundary of T to the boundary of W that preserves x-coordinates. A weak planar triangulation T' is *x-equivalent* to T if it has the same underlying graph, every nondegenerate face of T' has the same orientation as the corresponding face of T, and corresponding vertices have equal x-coordinates.

▶ **Lemma 5.2.** *Given a triangulation T of an x-monotone polygon in the plane, and a weakly convex polygon W that is compatible with T, there is a weak planar triangulation T' that is x-equivalent to T and whose outer face is W.*

Proof sketch. If W is a *strictly* convex polygon (all interior angles are strictly less than π), the lemma follows immediately from algorithms of Chrobak, Goodrich, Tammassia [17] and Kleist, Klemz, Lubiw, Schlipf, Staals, and Strash [39]; we extend these algorithms to weakly convex W using a limiting argument. See the full version [24] for a complete proof. ◀

We emphasize that our algorithms never *compute* the weak triangulation T'; we only need to prove that it exists.

▶ **Theorem 5.3.** *If $z' = (z'_1, \ldots, z'_n)$ is an optimal solution to linear program* (5.1)*, then z' is also a solution to the following $n \times n$ linear system:*

$$\begin{aligned} z'_i &= -1 && \text{for every vertex } i \text{ of } f_{\mathsf{north}} \\ \mathrm{vol}'(i,j,k) &= 0 && \text{for every down-face } (i,j,k) \end{aligned} \tag{5.2}$$

Proof. Let z' be any feasible solution for linear program; Lemma 5.1 implies that $z'_i < 0$ for all i. Let Γ' be the southern weak triangulation defined by z', and suppose that at least one down-face of Γ' is nondegenerate. We argue that we can construct another southern weak triangulation Γ'' (that is, another feasible solution z'' to (5.1)) by increasing some z-coordinates and leaving all other coordinates fixed, which implies that z' is not an optimal solution to (5.1).

Call a vertex of Γ' *sober* if it is not incident to the north face or to any degenerate face. If any vertex of Γ' is sober, we can construct Γ'' by moving any sober vertex upward slightly and keeping all other vertices fixed. So without loss of generality, we assume that Γ' has no sober vertices. We call a vertex i *upward-free* if i is not a vertex of the north face and $f_{\mathsf{above}}(i)$ is nondegenerate in Γ', *downward-free* if $f_{\mathsf{below}}(i)$ is nondegenerate in Γ', *totally free* if it is both upward- and downward-free, and *trapped* if $f_{\mathsf{above}}(i)$ and $f_{\mathsf{below}}(i)$ are both degenerate.

A *bar* in Γ' is a maximal circular arc that is covered by edges of Γ' and has no totally free vertices in its interior. Each bar is either a single edge or the union of one or more degenerate faces; in the latter case, the corresponding subset of faces in Γ are edge-connected, and their union is a θ-monotone disk. Equivalently, a nontrivial bar is the image in Γ' of a maximal edge-connected subset of faces of Γ that are all degenerate in Γ'. We call the subcomplex of Γ induced by these faces a *pre-bar*. The *endpoints* of a pre-bar are the preimages of the endpoints of its bar. Because no vertex of Γ' is sober, every vertex not incident to the north face is contained in at least one bar.

Every upward-free vertex in Γ' is the image of a vertex on the northern boundary of a pre-bar in Γ. Every downward-free vertex in Γ' is either incident to the north face or the image of a vertex on the southern boundary of a pre-bar in Γ. Each endpoint of a bar can be upward-free, downward-free, or both. Finally, each trapped vertex in Γ' is the image of a vertex in the interior of a pre-bar in Γ, and thus lies on a unique bar in Γ'. See Figure 5.1(a) and (b).

We construct a new weak triangulation Γ'' by defining new coordinates $z''_i \geq z'_i$ for each vertex i. Let $\varepsilon > 0$ be any sufficiently small positive constant. Defining new coordinates for the free vertices of Γ' is straightforward:

- For every vertex i of the north face, we define $z''_i = z'_i$.
- For every upward-free vertex i, we define $z''_i = z'_i/(1+\varepsilon) > z'_i$.
- For every downward-free vertex i that is not also upward free, we define $z''_i = z'_i$.

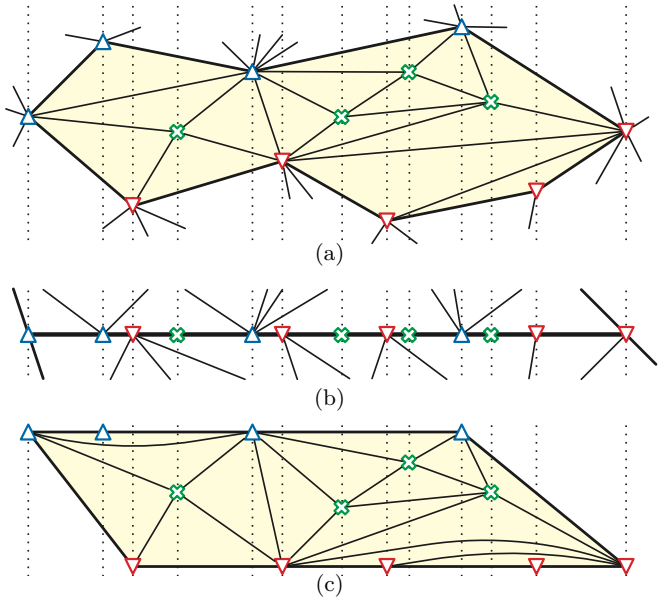


Figure 5.1 Planar projections of (a) a pre-bar in Γ, (b) the corresponding bar in Γ', and (c) the corresponding perturbed bar in Γ'' (with curved edges to show degenerate facets). Triangles indicate upward- and downward-free vertices; X's indicate trapped vertices. Dotted lines indicate longitudes.

Choosing sufficiently small ε already guarantees that any face that is non-degenerate in Γ' is also non-degenerate in Γ''. However, placing the trapped vertices to avoid inverting degenerate faces of Γ' requires more care.

Let β be any nontrivial bar in Γ', and let B be the corresponding pre-bar in Γ. We have already mapped the entire boundary of B to a spherical trapezoid τ in Γ''. Specifically, if β lies on the plane $z = ax + by$, then all upward-free vertices in β lie on the plane $z = (ax + by)/(1 + \varepsilon)$ in Γ''.

Because β projects to a line segment in the plane $z = -1$, it lies inside an open hemisphere bounded by a *vertical* plane through the origin. Without loss of generality, suppose β lies in the hemisphere $y > 0$. Let π_y denote the function $(x, y, z) \mapsto (x/y, z/y)$; we can interpret this function geometrically as the gnomonic projection from S^2 onto the plane $y = 1$. Then $\pi_y(\beta)$ is a non-vertical line segment, $\pi_y(B)$ is a triangulation of an x-monotone polygon, and $\pi_y(\tau)$ is a planar trapezoid whose lower boundary is a subset of $\pi_y(\beta)$ and whose upper boundary lies on a line parallel to $\pi_y(\beta)$. Each longitude ℓ through a trapped vertex i on β defines a *vertical* line $\pi_y(\ell)$ that must contain the point $(x_i/y_i, z_i''/y_i)$. See Figure 5.1.

To summarize, $\pi_y(B)$ is a triangulation of an x-monotone polygon, and $\pi_y(\tau)$ is a weakly convex polygon (specifically, a trapezoid) that is compatible with $\pi_y(B)$. Thus, Lemma 5.2 implies that there is a weak triangulation $\pi_y(T'')$ that is x-equivalent to $\pi_y(B)$ and whose outer face is $\pi_y(\tau)$. Pulling $\pi_y(T'')$ back to the sphere gives us a weak triangulation T'' of the spherical trapezoid τ that is θ-equivalent to B. Each interior vertex i of B maps to a vertex in the closed interior of τ, which implies $z_i' \le z_i'' \le z_i'/(1 + \varepsilon)$.

We assemble the overall weak triangulation Γ'' by applying this construction to every bar in Γ'. (Processing each bar requires projecting to a different vertical tangent plane.) Because $z_i'' \ge z_i'$ for every vertex i, and $z_i'' > z_i'$ for at least one vertex i, the original weak triangulation Γ' cannot be an optimal solution to our linear program (5.1). ◀

▶ **Corollary 5.4.** *Given a shortest-path triangulation* Γ *of the sphere, we can either compute a longitudinal morph from* Γ *to a southern triangulation, or report correctly that no such morph exists, in* $O(n^{\omega/2})$ *time, assuming linear system* (5.2) *is non-singular.*

Proof. The support of linear system (5.2) is precisely the directed planar graph $\Gamma^{\searrow\swarrow}$. Thus, assuming the system is non-singular, it can be solved in $O(n^{\omega/2}) = O(n^{1.1864})$ time (in the real RAM model) via nested dissection and fast matrix multiplication [3,42]. If the solution z' is a feasible point for linear program (5.1), the corresponding drawing Γ' is a weak southern triangulation θ-equivalent to Γ. If the system is non-singular, and the solution z' is not a feasible point for linear program (5.1), then by Theorem 5.3, the linear program is infeasible, so we can correctly report that Γ is not sinkable. ◀

Theorem 5.3 is a strict generalization of the Awartani-Henderson embedding, described in the proof of Theorem 3.2. When Γ is shellable, its support graph $\Gamma^{\searrow\swarrow}$ is acyclic. Thus, we can permute the rows and columns of (5.2) to obtain an upper-triangular system, which we can then solve by back-substitution. Each step of back-substitution assigns one z-coordinate, exactly mirroring one step of the Awartani-Henderson construction.

References

1. Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. In *Proc. 29th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 274–292, 2018. doi:10.1137/1.9781611975031.20.
2. Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M. Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. How to morph planar graph drawings. *SIAM J. Comput.*, 46(2):824–852, 2017. doi:10.1137/16M1069171.
3. Noga Alon and Raphael Yuster. Matrix sparsification and nested dissection over arbitrary fields. *J. ACM*, 60(4):article 25, 18pp., 2013. doi:10.1145/2508028.2505989.
4. Marwan Awartani and David W. Henderson. Spaces of geodesic triangulations of the sphere. *Trans. Amer. Math. Soc.*, 304(2):721–732, 1987. doi:10.2307/2000738.
5. Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. Morphing Schnyder drawings of planar triangulations. *Discrete Comput. Geom.*, 61(1):161–184, 2019. doi:10.1007/s00454-018-0018-9.
6. Therese Biedl, Anna Lubiw, and Jack Spalding-Jamieson. Morphing planar graph drawings via orthogonal box drawings. In *Proc. 32nd Int. Symp. Graph Drawing Network Vis.*, page to appear, 2024. doi:10.48550/arXiv.2409.04074.
7. R. H. Bing and Michael Starbird. Linear isotopies in E^2. *Trans. Amer. Math. Soc.*, 237:205–222, 1978. doi:10.2307/1997619.
8. Gunnar Brinkmann. A simple and elementary proof of Whitney's unique embedding theorem. *Ars Math. Contemp.*, 20(2):195–197, 2021. doi:10.26493/1855-3974.2334.331.
9. Heinz Bruggesser and Peter Mani. Shellable decompositions of cells and spheres. *Math. Scand.*, 29:197–205, 1971. doi:10.7146/math.scand.a-11045.
10. Kevin Buchin, William S. Evans, Fabrizio Frati, Irina Kostitsyna, Maarten Löffler, Tim Ophelders, and Alexander Wolff. Morphing planar graph drawings through 3D. In *Proc. 48th Int. Conf. Current Trends in Theory and Practice Comput. Sci. (SOFSEM)*, volume 13878 of *Lecture Notes Comput. Sci.*, pages 80–95. Springer, 2023. doi:10.1007/978-3-031-23101-8_6.
11. Stewart S. Cairns. Deformations of plane rectilinear complexes. *Amer. Math. Monthly*, 51(5):247–252, 1944. doi:10.2307/2304300.
12. Stewart S. Cairns. Isotopic deformations of geodesic complexes on the 2-sphere and on the plane. *Ann. Math.*, 45(2):207–217, 1944. doi:10.2307/1969263.

13 Jean Cerf. About the Bloch–Connelly–Henderson theorem on the simplexwise linear homeomorphisms of a convex 2-disk. Preprint, October 2019. doi:10.48550/arXiv.1910.00240.

14 Erin Wolf Chambers, Jeff Erickson, Patrick Lin, and Salman Parsa. How to morph graphs on the torus. In *Proc. 32nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 2759–2778, 2021. doi:10.1137/1.9781611976465.164.

15 Steven Chaplick, Philipp Kindermann, Jonathan Klawitter, Ignaz Rutter, and Alexander Wolff. Morphing rectangular duals. In *Proc. 30th Int. Symp. Graph Drawing Network Vis.*, number 13764 in Lecture Notes Comput. Sci., pages 389–403. Springer, 2023. doi:10.1007/978-3-031-22203-0_28.

16 Bernard Chazelle, Leo J. Guibas, and D. T. Lee. The power of geometric duality. *BIT Numer. Math.*, 25(1):76–90, 1985. doi:10.1007/BF01934990.

17 Marek Chrobak, Michael T. Goodrich, and Roberto Tamassia. Convex drawings of graphs in two and three dimensions (preliminary version). In *Proc. 12th Ann. Symp. Comput. Geom.*, pages 319–328, 1996. doi:10.1145/237218.237401.

18 Krzysztof Diks and Piotr Sankowski. Dynamic plane transitive closure. In *Proc. 15th Ann. Europ. Symp. Algorithms*, number 4698 in Lecture Notes Comput. Sci., pages 594–604. Springer, 2007. doi:10.1007/978-3-540-75520-3_53.

19 Adrien Douady. Le shaddock à six becs. *Bull. APMEP*, 281:699–701, 1971. URL: https://www.apmep.fr/IMG/pdf/AAA71038.pdf.

20 Peter Eades, Qingwen Peng, Xuemin Lin, and Hiroshi Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006. doi:10.1007/s00453-004-1144-8.

21 Victor Eberhard. *Zur Morphologie der Polyeder*. Teubner, 1891.

22 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.

23 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.

24 Jeff Erickson and Christian Howard. Shelling and sinking graphs on the sphere. Preprint, April 2025. doi:10.48550/arXiv.2504.05098.

25 Jeff Erickson and Patrick Lin. Planar and toroidal morphs made easier. In *Proc. 29th Int. Symp. Graph Drawing Network Vis.*, number 12868 in Lecture Notes Comput. Sci., pages 123–137, 2021. doi:10.1007/978-3-030-92931-2_9.

26 Cesim Erten, Stephen G. Kobourov, and Chandan Pitta. Intersection-free morphing of planar graphs. In Giuseppe Liotta, editor, *Proc. 11th Symp. Graph Drawing*, volume 2912 of *Lecture Notes Comput. Sci.*, pages 320–331. Springer, 2003. doi:10.1007/978-3-540-24595-7_30.

27 Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Comp. Aided Geom. Design*, 14(3):231–250, 1997. doi:10.1016/S0167-8396(96)00031-3.

28 Michael S. Floater. Parametric tilings and scattered data approximation. *Int. J. Shape Modeling*, 4(3–4):165–182, 1998. doi:10.1142/S021865439800012X.

29 Michael S. Floater and Craig Gotsman. How to morph tilings injectively. *J. Comput. Appl. Math.*, 101(1–2):117–129, 1999. doi:10.1016/S0377-0427(98)00202-7.

30 Radoslav Fulek and Jan Kyncl. Hanani-Tutte for approximating maps of graphs. In *Proc. 34th Int. Symp. Comput. Geom.*, number 99 in Leibniz Int. Proc. Informatics, pages 39:1–39:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.39.

31 Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001. doi:10.1016/S0097-8493(00)00108-4.

32 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Number 2 in Algorithms and Combinatorics. Springer-Verlag, 2nd edition, 1993. doi:10.1007/978-3-642-78240-4.

33 Branko Grünbaum. Graphs of polyhedra; polyhedra as graphs. *Discrete Math.*, 307(3–5):445–463, 2007. doi:10.1016/j.disc.2005.09.037.

34 Chung-Wu Ho. On certain homotopy properties of some spaces of linear and piecewise linear homeomorphisms. I. *Trans. Amer. Math. Soc.*, 181:213–233, 1973. doi:10.2307/1996630.

35 Chung-Wu Ho. Deforming P.L. homeomorphisms on a convex polygonal disk. *Pacific J. Math.*, 55(2):427–439, 1974. doi:10.2140/pjm.1974.55.427.

36 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of hierarchical planar graphs and clustered planar graphs. *J. Discrete Algorithms*, 8(3):282–295, 2010. doi:10.1016/j.jda.2009.05.003.

37 John E. Hopcroft and Peter J. Kahn. A paradigm for robust geometric algorithms. *Algorithmica*, 7(1–6):339–380, 1992. doi:10.1007/BF01758769.

38 Børge Jessen. Orthogonal icosahedra. *Nordisk Matematisk Tidskrift*, 15(2/3):90–96, 1967. URL: https://www.jstor.org/stable/24524998.

39 Linda Kleist, Boris Klemz, Anna Lubiw, Lena Schlipf, Frank Staals, and Darren Strash. Convexity-increasing morphs of planar graphs. *Comput. Geom. Theory Appl.*, 84:69–88, 2019. doi:10.1016/j.comgeo.2019.07.007.

40 Boris Klemz. Convex drawings of hierarchical graphs in linear time, with applications to planar graph morphing. In *Proc. 29th Ann. Europ. Symp. Algorithms*, number 204 in Leibniz Int. Proc. Informatics, pages 57:1–57:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.57.

41 Stephen G. Kobourov and Matthew Landis. Morphing planar graphs in spherical space. *J. Graph Algorithms Appl.*, 12(1):113–127, 2006. doi:10.7155/jgaa.00162.

42 Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979. doi:10.1137/0716027.

43 Anna Lubiw and Mark Petrick. Morphing planar graph drawings with bent edges. *J. Graph Algorithms Appl.*, 15(2):205–227, 2011. doi:10.7155/jgaa.00223.

44 Yanwen Luo, Tianqi Wu, and Xiaoping Zhu. The deformation space of geodesic triangulations of flat tori. Preprint, July 2021. doi:10.48550/arXiv.2107.05159.

45 Yanwen Luo, Tianqi Wu, and Xiaoping Zhu. The deformation space of geodesic triangulations and generalized Tutte's embedding theorem. *Geom. Topol.*, 7:3361–3385, 2023. doi:10.2140/gt.2023.27.3361.

46 Yanwen Luo, Tianqi Wu, and Xiaoping Zhu. Spaces of geodesic triangulations are cells. Preprint, August 2024. doi:10.48550/arXiv.2204.10833.

47 Jürgen Richter-Gebert. *Realization spaces of polytopes*. Number 1643 in Lecture Notes Math. Springer, 1996. doi:10.1007/BFb0093761.

48 Erich Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.*, 98(1):309–312, 1928. doi:10.1007/BF01451597.

49 Ernst Steinitz. Polyeder und Raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, III.AB(12):1–139, 1916. URL: https://gdz.sub.uni-goettingen.de/id/PPN360609767.

50 Ernst Steinitz and Hans Rademacher. *Vorlesungen über die Theorie der Polyeder: unter Einschluß der Elemente der Topologie*. Number 41 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1934. Reprinted 1976. doi:10.1007/978-3-642-65609-5.

51 Jorge Stolfi. Oriented projective geometry. In *Proc. 3rd Ann. Symp. Comput. Geom.*, pages 76–85, 1987. doi:10.1145/41958.41966.

52 Jorge Stolfi. *Primitives for Computational Geometry*. Ph.D. thesis, Dept. Comput. Sci., Stanford, May 1988. Reprinted as Technical Report SRC-RR-36, DEC / HP Labs Systems Research Center, January 1989. URL: https://shiftleft.com/mirrors/www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-36.pdf.

53 Jorge Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, New York, NY, 1991. doi:10.1016/C2013-0-11551-6.

54 Fred Supnick. On the perspective deformation of polyhedra. *Ann. Math.*, 49(3):714–730, 1948. doi:10.2307/1969055.

55 Fred Supnick. On the perspective deformation of polyhedra. II. Solution of the convexity problem. *Math. Ann.*, 53(2):551–555, 1951. doi:10.2307/1969572.

56 Vitaly Surazhsky and Craig Gotsman. Controllable morphing of compatible planar triangulations. *ACM Trans. Graphics*, 20(4):203–231, 2001. doi:10.1145/502783.502784.

57 Vitaly Surazhsky and Craig Gotsman. Morphing stick figures using optimal compatible triangulations. In *Proc. 9th Pacific Conf. Comput. Graphics Appl.*, pages 40–49, 2001. doi:10.1109/PCCGA.2001.962856.

58 Vitaly Surazhsky and Craig Gotsman. Intrinsic morphing of compatible triangulations. *Int. J. Shape Modeling*, 9(2):191–201, 2003. doi:10.1142/S0218654303000115.

59 Carsten Thomassen. Deformations of plane graphs. *J. Comb. Theory Ser. B*, 34(3):244–257, 1983. doi:10.1016/0095-8956(83)90038-2.

60 William T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13(3):743–768, 1963. doi:10.1112/plms/s3-13.1.743.

61 Arthur van Goethem and Kevin Verbeek. Optimal morphs of planar orthogonal drawings. In *Proc. 34th Int. Symp. Comput. Geom.*, number 99 in Leibniz Int. Proc. Informatics, pages 42:1–42:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.42.

62 Hassler Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54(1):150–168, 1932. doi:10.2307/2371086.

Exact Algorithms for Minimum Dilation Triangulation

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany

Michael Perk
Department of Computer Science, TU Braunschweig, Germany

Abstract

We provide a spectrum of new theoretical insights and practical results for finding a Minimum Dilation Triangulation (MDT), a natural geometric optimization problem of considerable previous attention: Given a set P of n points in the plane, find a triangulation T, such that a shortest Euclidean path in T between any pair of points increases by the smallest possible factor compared to their straight-line distance. No polynomial-time algorithm is known for the problem; moreover, evaluating the objective function involves computing the sum of (possibly many) square roots. On the other hand, the problem is not known to be NP-hard.

(1) We provide practically robust methods and implementations for computing an MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points, so we extend the range of optimally solvable instances by a factor of 150.

(2) We develop scalable techniques for accurately evaluating many shortest-path queries that arise as large-scale sums of square roots, allowing us to certify exact optimal solutions, with previous work relying on (possibly inaccurate) floating-point computations.

(3) We resolve an open problem by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon (and thus for arbitrary point sets), improving the previous worst-case lower bound of 1.4308 and greatly reducing the remaining gap to the upper bound of 1.4482 from the literature. In the process, we provide optimal solutions for regular n-gons up to $n = 100$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Discrete optimization; General and reference → Experimentation; Mathematics of computing → Interval arithmetic; Mathematics of computing → Arbitrary-precision arithmetic

Keywords and phrases dilation, minimum dilation triangulation, exact algorithms, algorithm engineering, experimental evaluation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.48

Related Version *Full Version*: https://arxiv.org/abs/2502.18189

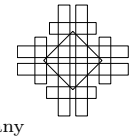
Supplementary Material *Software (Source Code)*: https://doi.org/10.5281/zenodo.14266122
Dataset (Experiment Data): https://doi.org/10.5281/zenodo.14266122

Funding The work presented in this paper was largely funded by the DFG grant "Computational Geometry: Solving Hard Optimization Problems" (CG:SHOP), grant FE 407/21-1.

1 Introduction

Triangulating a set of points is one of the classical problems in computational geometry. On the practical side, it has natural applications in wireless sensor networks [47, 49], mesh generation [4], computer vision [45], geographic information systems [46] and many other areas [15]. On the theoretical side, finding a triangulation that is optimal with respect to

© Sándor P. Fekete, Phillip Keldenich, and Michael Perk;
licensed under Creative Commons License CC-BY 4.0



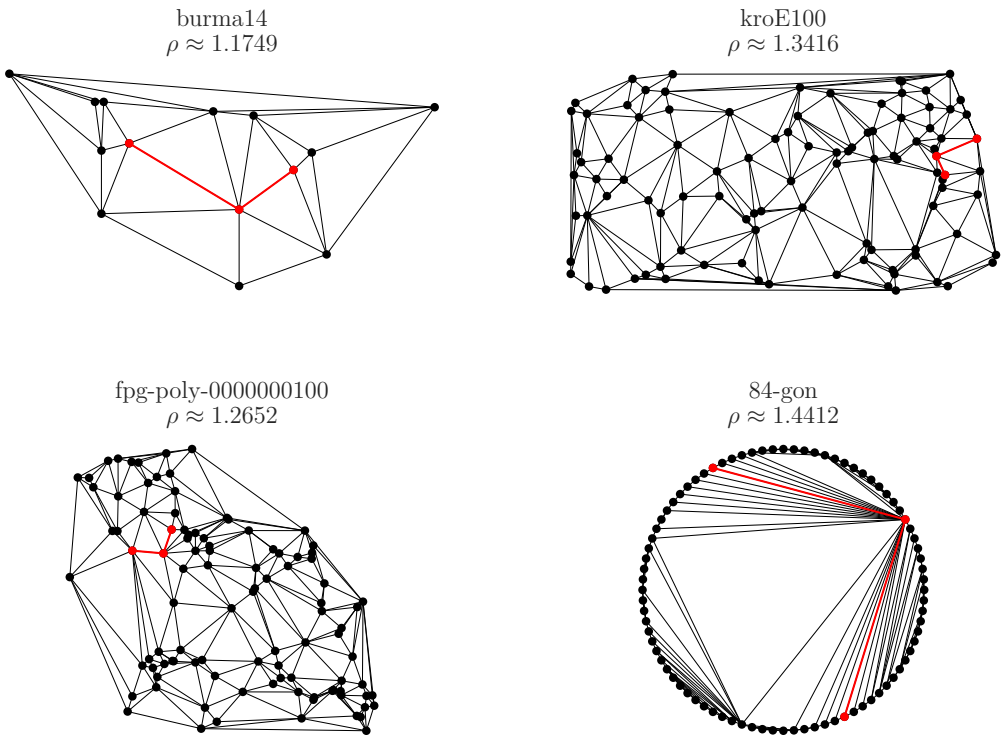


Figure 1 MDT solutions for four instances. The red edges indicate a dilation-defining path.

some objective function has also received considerable attention: The Delaunay triangulation maximizes the minimum angle and minimizes the maximum circumcircle of all triangles. Minimizing the maximum edge length is possible in quadratic time [21]. On the other hand, maximizing the minimum edge length is NP-complete [27]. Famously, Mulzer and Rote [36] showed that computing the Minimum Weight Triangulation (MWT) is NP-hard.

In this paper, we provide new results and insights for another natural objective that considers triangulations as sparse structures with relative low cost for ensuing detours: The *dilation* of a triangulation T of a point set P is the worst-case ratio (among all $s, t \in P$) between the shortest s-t-path in T and the Euclidean distance between s and t. The Minimum Dilation Triangulation (MDT) problem asks for a triangulation that minimizes the dilation ρ, see Figure 1 for examples. This problem is closely related to the concept of a Euclidean t-spanner: a subgraph with dilation (also called *spanning ratio* or *stretch factor* [37]) at most t. Spanners have application in areas as robotics, network design [1, 26], sensor networks [10, 25] and design of parallel machines [3] and have been studied extensively [11]. Computing the MDT amounts to computing a *plane* spanner with smallest spanning ratio, as every plane spanner can be extended into a triangulation. For many types of triangulations, such as the Delaunay triangulation or the MWT, both lower and upper bounds on the worst-case dilation are known. Moreover, a constant upper bound on the worst-case dilation of a triangulation also implies a constant-factor approximation of the MDT. Lower and upper bounds on the worst-case dilation of the MDT have been studied, both for general point sets and for special point sets such as regular polygons or points on a circle; see Section 1.2 for further details.

Despite this importance and attention, actually computing a Minimum Dilation Triangulation is a challenging problem. Its computational complexity is still unresolved, signaling that there may not be a simple and elegant algorithmic solution that scales well. Moreover, actually

computing accurate dilations involves computing shortest paths under Euclidean distances between $\Theta(n^2)$ pairs of points requires dealing with another famous challenge [39, 18, 6, 23], as it corresponds to evaluating and comparing numerous sums of many square roots.

1.1 Our contributions

We provide various new contributions, both to theory and practice.

1. We present practically robust methods and implementations for computing an optimal MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points (involving one computational routine of complexity $\Theta(n^4)$ instead of our improved complexity of $O(n^2 \log n)$), so we extend the range of practically solvable instances by a factor of 150.
2. We develop scalable techniques for accurately evaluating many shortest-path queries that arise as large-scale sums of square roots, allowing us to certify exact optimal solutions. This differs from previous work, which relied on floating-point computations, without regard for errors resulting from numerical issues.
3. We resolve an open problem from [20] by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon (and thus for arbitrary point sets). This improves the previous worst-case lower bound of 1.4308 from the regular 23-gon and greatly reduces the remaining gap to the upper bound of 1.4482 from [43]. In the process, we provide optimal solutions for regular n-gons up to $n = 100$.

1.2 Related work

The complexity of finding the MDT is unknown [24]. [28] prove that finding the minimum dilation graph with a limited number of edges is NP-hard. [12] show that finding a spanning tree of given dilation is also NP-hard. Kozma [33] proves NP-hardness for minimizing the expected distance between random points in a triangulation, with edge weights instead of Euclidean distances. For surveys, see Eppstein [24] until 2000 and [38] for more recent results.

On the practical side, the authors of [44] study the problem of finding sparse low dilation graphs for large point sets in the plane. The authors of [9] present an approximation algorithm for the related problem of improving a given graph with a budget of k edges such that the dilation is minimized. Regarding the MDT, all practical approaches in the literature are based on fixed-precision arithmetic. Klein [31] used an enumeration algorithm to find an optimal MDT for up to 10 points. The authors of [19] present heuristics for the MDT and evaluate their performance on instances with up to 200 points. Instances with up to 70 points were solved by the authors of [8] using integer linear programming techniques. In their approach, the edge elimination strategy from Knauer and Mulzer [32] was used to eliminate edges from the complete graph. Recently, Sattari and Izadi [41] presented an exact algorithm based on branch and bound that was evaluated on instances with up to 200 points.

The MDT is closely related to finding a plane t-spanner; see [34] for an overview. Chew [13] first proved an upper bound of $\sqrt{10}$ on plane t-spanners in the L_1-metric, which he later improved [14] to 2 for the triangular-distance Delaunay graph in the plane. [5] proved that any convex point set admits a plane 1.88-spanner. For a centrally symmetric convex point set containing n points, Sattari and Izadi [42] give an upper bound of $n/2 \sin(\pi/n)$. The best known upper bound on the dilation of arbitrary point sets is by Xia [48], who established and upper bound of 1.998 for the dilation of the Delaunay triangulation.

Mulzer [35] studied the MDT for the set of vertices of a regular n-gon and proved an upper bound of 1.48586. Amarnadh and Mitra [2] improved this bound to 1.48454 for any point set that lies on the boundary of a circle. Sattari and Izadi [43] again improved the bound to 1.4482. Dumitrescu and Ghosh [20] show that any triangulation of a regular 23-gon has dilation at least 1.4308, improving upon the bound of 1.4161 by Mulzer [35] and answering a question posed by Bose and Smit [7] as well as Kanj [30]. Dumitrescu and Ghosh [20] also computed dilations of regular 22-gon and regular 24-gon.

2 Preliminaries

Let $P \subset \mathbb{R}^2$ be a set of points in the plane. We denote the Euclidean distance between two points $u, v \in P$ by $d(u, v)$. For a connected geometric graph $G = (P, E)$ with $E \subseteq \binom{P}{2}$, we denote the Euclidean shortest path between two points $u, v \in P$ by $\pi_G(u, v)$ and its length by $|\pi_G(u, v)|$, omitting G if it is clear from context. The dilation $\rho_G(u, v)$ between two points u, v in G is the ratio $\rho_G(u, v) := \frac{|\pi_G(u,v)|}{d(u,v)}$ between the shortest path length and the Euclidean distance. The dilation $\rho(G)$ of the graph G is defined as the maximum dilation between any two points in P, i.e., $\rho(G) := \max\{\rho_G(u, v) \mid u, v \in P, u \neq v\}$.

In the remainder of this work, the graph G we consider is a triangulation, i.e., a maximal crossing-free graph on P. Two edges $e_1 = (p_1, q_1), e_2 = (p_2, q_2)$ are said to *cross* or *intersect* iff the line segments they induce intersect in their interior. Given a point set P, the Minimum Dilation Triangulation problem (MDT) asks to find a triangulation T of P minimizing $\rho(T)$.

3 Candidate edge enumeration

Here we describe a novel and practically efficient scheme for enumerating a set of edges that induces a supergraph of the MDT. We start with the underlying theoretical ideas for this supergraph, followed by an algorithm for computing a supergraph of any triangulation with dilation *strictly less* than a given bound ρ, which we exploit to enumerate a supergraph of the MDT with a (usually) small number of edges. This is further adapted to reductions of the dilation bound ρ. We also discuss the computation of lower bounds on the dilation of the MDT. Some additional implementation details can be found in the full version.

3.1 Theoretical background

Our supergraph is based on the well-known *ellipse property* (used in [8, 28, 32]) that all edges of a triangulation T with dilation below ρ must satisfy.

▶ **Definition 1.** *A pair of points s, t has the* ellipse property *with respect to a point set P and a dilation bound ρ if, for any pair of points $\ell, r \in P \setminus \{s, t\}$ such that ℓr and st intersect,* $\min\{d(\ell, s) + d(s, r), d(\ell, t) + d(t, r)\} < \rho d(\ell, r)$.

Recall that the set of points that have the same sum of distances κ to two points ℓ, r with $\kappa \geq d(\ell, r)$ is an *ellipse* with ℓ, r as its focal points (or *foci*); thus, all paths between ℓ and r with length less than $\kappa = \rho \cdot d(\ell, r)$ must lie strictly inside this ellipse.

If a pair of points s, t does not have the ellipse property, then there is a pair of points ℓ, r such that st cuts through all paths between ℓ and r that could have dilation less than ρ; see Figure 2. We call such a pair of points ℓ, r an *exclusion certificate* for s, t.

▶ **Observation 2.** *Let $\rho \geq 1$ be some dilation bound and let T be a triangulation containing the edge st for points s, t that do not have the ellipse property w.r.t. P and ρ. Then, $\rho(T) \geq \rho$.*

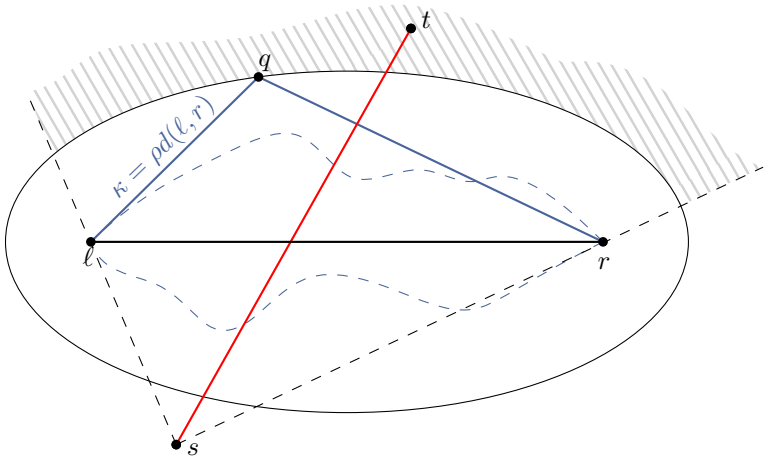


Figure 2 Any path connecting ℓ and r with length below $\kappa = \rho d(\ell, r)$ must lie within the ellipse (dashed black lines). The edge st does not have the *ellipse property* as neither s nor t lie inside the ellipse; therefore, inserting st makes connecting ℓ and r by a sufficiently short path impossible.

3.2 High-level description

Preliminary experiments showed that a brute-force check of each of the $\Theta(n^4)$ pairs of potential edges is impractical for large n, dominating the overall runtime time even for early versions of our solution approach.

We therefore developed a more efficient scheme to enumerate a superset of the edges that satisfy the ellipse property. This scheme performs a *filtered incremental nearest-neighbor search* from each point $p \in P$ to identify all candidate edges of the form pt, i.e., looking for all possible *neighbors* t of p. This search is efficiently implemented on a quadtree containing all points. While enumerating candidates, we construct so-called *dead sectors*, i.e., regions of the plane that cannot contain possible neighbors of p. We exclude all points that lie in dead sectors; we also use dead sectors to prune entire nodes of the quadtree and to terminate the search early if it has become clear that all remaining points must be in a dead sector. This usually avoids considering most points as potential neighbors of p individually. The algorithm has a runtime of $O(nk \log n)$, where k is the average number of points and quadtree vertices considered individually from each point. At worst, this can be $O(n^2 \log n)$; in practice, k is often much lower than n. A related, slightly less complex enumeration algorithm applied to minimum-weight triangulations by Haas [29] scales to 10^8 points. In the following, we describe the components of the enumeration scheme in more detail.

3.3 Dead sectors

We begin by giving a definition of the dead sectors we use.

▶ **Definition 3.** *Given a dilation ρ, a source point p and two points ℓ, r, the* dead sector $\mathcal{DS}_\rho(p, \ell, r) \subset \mathbb{R}^2$ *is the region of all points t such that pt intersects ℓr and neither p nor t lie in the ellipse with foci ℓ, r and focal distance sum $\kappa = \rho d(\ell, r)$.*

Depending on p, ρ, ℓ and r, $\mathcal{DS}_\rho(p, \ell, r)$ is either empty (if p is in the ellipse) or it is bounded by two rays and an elliptic arc; see Figure 3. In that case, it can also be interpreted as an elliptic arc and an interval of polar angles around p.

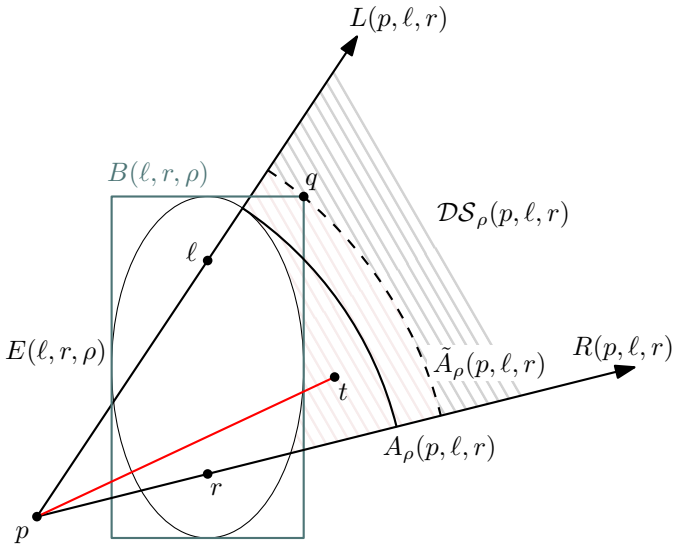


Figure 3 A dead sector $\mathcal{DS}_\rho(p,\ell,r)$, shaded in gray and red with the ellipse $E(\ell,r,\rho)$. We approximate the ellipse by a disk centered at p with radius $\tilde{A}_\rho(p,\ell,r)$ (thereby ignoring the red area), which is the distance from p to the farthest point q of the rectangle $B(\ell,r,\rho)$ in $\mathcal{DS}_\rho(p,\ell,r)$.

During our enumeration we construct many dead sectors, the union of which can become quite complex, making it cumbersome and inefficient to work with directly. We instead chose to simplify the shape of our dead sectors, giving up a small fraction of excluded area in exchange for a simple and efficient representation. We replace the elliptic arc by a single disk centered on p, whose radius is at least the maximum distance from p to any point on the elliptic arc. We can hence represent each non-empty simplified dead sector by a polar angle interval around p and a single radius called *activation distance* $A_\rho(p,\ell,r)$; see Figure 3.

We initially attempted to compute fairly precise upper bounds on the activation distance; however, due to computational and numerical issues described in the full version, we decided to use a more robust and efficient approach. Instead of using the elliptic arc, we compute an upper bound $\tilde{A}_\rho(p,\ell,r)$ using a minimal rectangle $B(\ell,r,\rho)$ containing the ellipse with sides are parallel and perpendicular to ℓr. We only need to check the extreme points of $B(\ell,r,\rho)$ and its intersections with the rays $L(p,\ell,r)$ and $R(p,\ell,r)$ to compute an upper bound $\tilde{A}_\rho(p,\ell,r)$; see Figure 3.

In the worst case, these simplifications may lead to additional candidate edges. While this cannot make the resulting graph exclude any edges that satisfy the ellipse property, it is still undesirable; we use additional checks for further reductions later on.

3.4 Quadtree

We use a quadtree containing all points in P for the filtered incremental search. The points are stored in a contiguous array A_P outside the tree. Each quadtree node v is associated with a contiguous subrange of A_P represented by two pointers. This subrange contains the points in the subtree $\mathcal{T}_v$ rooted at v. Each node also has a bounding box that contains all points in $\mathcal{T}_v$. Each interior node has precisely four children; each leaf node contains at most a small constant number of points. To allow the contiguity of the subranges, points are reordered during tree construction. This has the added benefit of spatially sorting the points, improving the probability that geometrically close points are near each other in memory [29].

3.5 Enumeration process

One can think of the filtered incremental search from p as a process of continuously growing a disk centered at p starting with a radius of 0. As in the sweep-line paradigm, one encounters different types of events at discrete disk radii. We primarily encounter events when the disk first touches a point of P or the bounding box of a quadtree node.

Observe that, during a search from a point p, the dead sectors have two states: either their activation distance is not yet reached, in which case they are *inactive* and do not exclude any points, or they are *active* and exclude all points in a certain polar angle interval around p. We therefore also introduce events when the disk radius reaches the activation distance of a dead sector. This enables efficient management of active dead sectors as a set of polar angle intervals around p; see the full version for a discussion of the ensuing numerical issues.

When we first encounter a point t, we have to determine whether t is in any dead sector by checking the active dead sector data structure. If it is not, we have to report it as potential neighbor of p, adding it to a set of points sorted by polar angle around p. Furthermore, to construct new dead sectors, we combine t with $O(1)$ other points of P; which points we use is decided by a heuristic discussed in the full version.

When we encounter a quadtree node v, we have to determine whether v's bounding box is fully contained in the union of all dead sectors and can thus be pruned; we thus again check the active dead sectors. Otherwise, we have to take v's children, or the points it contains if it is a leaf, into account; they are then considered as future events.

3.6 Initialization and postprocessing

We initially compute the Delaunay triangulation of the point set P and compute its dilation. We also optionally attempt to improve the dilation of the triangulation by a simple improvement heuristic. The heuristic is based on computing constrained Delaunay triangulations, greedily adding shortcut edges as constraints to reduce the length of the path currently defining the dilation. We then use the resulting dilation ρ as bound for the enumeration process outlined in the previous sections, enumerating only edges that could locally be in a triangulation with dilation strictly below ρ.

After the initial enumeration process is complete, we are left with a set of *possible* edges and can safely ignore all other edges. We postprocess these as follows. For each possible edge pq, we compute the set of possible edges intersecting it. We need this information later on to model the problem of finding a triangulation on the set of possible edges. For each pair st, ℓr of intersecting edges, we explicitly check whether either pair is an exclusion certificate for the other. In many cases, this postprocessing gets us very close to the edge set that would be obtained by the trivial $\Theta(n^4)$ edge candidate enumeration algorithm; see the experiment section for details. We mark each edge that does not have intersecting possible edges as *certain*; certain edges must be part of any triangulation with dilation less than ρ.

3.7 Dilation thresholds

We also compute a *dilation threshold* $\vartheta(st)$ for each possible edge st. Let $I(st)$ be the set of possible edges intersecting st. For each $\ell r \in I(st)$, we can compute a lower bound on the dilation of the shortest path connecting ℓ to r, should st be present, as

$$\rho_{st}(\ell r) = \min\{d(\ell, s) + d(s, r), d(\ell, t) + d(t, r)\}/d(\ell, r);$$

see Figure 4. The dilation threshold of st is then $\vartheta(st) = \max_{\ell r \in I(st)} \rho_{st}(\ell r)$.

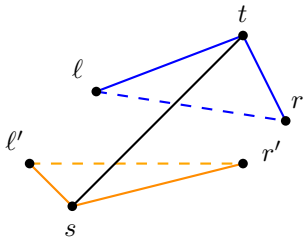


Figure 4 If st (black) is present, the pairs ℓ, r (resp. ℓ', r') can only be connected by paths that have at least the length of the solid blue (resp. orange) path. Hence, the ratio between the solid and dashed blue (resp. orange) paths is a lower bound on the dilation of any triangulation containing st. The maximum of such lower bounds for st is the dilation threshold $\vartheta(st)$ of st.

▶ **Observation 4.** *If an edge st has dilation threshold $\vartheta(st)$, it is not in any triangulation with dilation $\rho < \vartheta(st)$.*

This allows us to quickly exclude further edges if we lower the dilation bound ρ we aim for, without reenumerating edges or recomputing intersections.

3.8 Lower bounds

We use the dilation thresholds to bound the minimum dilation. For each edge st, either st or some edge crossing it must be in any triangulation. Thus, the minimum of all dilation thresholds among $\{st\} \cup I(st)$ is a lower bound on the minimum dilation. Combining all edges, we obtain the following lower bound:

$$\rho(T) \geq \max_{st} \min\{\vartheta(pq) \mid pq \in \{st\} \cup I(st)\}.$$

We use interval arithmetic to compute a safe lower bound on this value in $O(1)$ time per intersection between two edges resulting from our enumeration.

Furthermore, by computing the points on the convex hull, we also know the number of edges g of any triangulation. This also gives us a lower bound on the minimum dilation by considering the g lowest dilation thresholds. We also use Kruskal's algorithm to compute the lowest dilation threshold that admits a connected graph.

4 Exact algorithms

Now we present two exact algorithms: IncMDT is an incremental method that uses a SAT solver for iterative improvement, until it can prove that no better solution exists. BinMDT is based on a binary search for the optimal dilation ρ; once the lower and upper bound are reasonably close, the approach falls back to IncMDT to reach a provably optimal solution.

4.1 Triangulation supergraph

Both algorithms rely on the MDT supergraph mentioned in Section 3. As part of this computation, we also obtain an initial triangulation and its dilation, as well the intersecting possible edges $I(st)$ for each possible edge st. In both algorithms, we may gradually discover triangulations with lower dilation; these are used to exclude additional edges using the precomputed dilation thresholds $\vartheta(e)$. To keep track of the status of each edge, we insert all points and possible edges into a graph data structure we call *triangulation supergraph*. In

this structure, we mark each edge as *possible*, *impossible* or *certain*. Initially, all enumerated edges are *possible*. If, at any point, all edges intersecting an edge e become *impossible*, e becomes *certain*. If an *impossible* edge becomes *certain* or vice versa, the graph does not contain a triangulation any longer. If this happens, we say we encounter an *edge conflict*.

4.2 SAT formulation

Given a triangulation supergraph $G = (P, E)$, we model the problem of finding a triangulation on *possible* and *certain* edges using the following simple SAT formulation. Let $E_p \subseteq E$ be the set of non-*impossible* edges when the SAT formulation is constructed. For each edge $e \in E_p$, we have a variable x_e. We use the following clauses in our formulation.

$$\neg x_{e_1} \vee \neg x_{e_2} \qquad \forall e_1, e_2 \in E_p : e_2 \in I(e_1) \tag{1}$$

$$x_e \vee \bigvee_{e_j \in I(e)} x_{e_j} \qquad \forall e \in E_p \tag{2}$$

Clauses (1) ensure crossing-freeness and clauses (2) ensure maximality. When an edge e becomes certain, we add the clause x_e; similarly, when an edge becomes impossible, we add the clause $\neg x_e$. Both algorithms are based on this simple formulation; in the following, we describe how they use and modify it to find an MDT.

4.3 Clause generation

The following subproblem, which we call *dilation path separation*, arises in both our algorithms: Given a dilation bound ρ, a triangulation supergraph $G = (P, E)$ excluding only edges that cannot be in any triangulation with dilation less than ρ, a current triangulation T and a pair of points $s, t \in P$ such that $|\pi_T(s, t)| \geq \rho \cdot d(s, t)$, find a clause C that is (a) violated by T and (b) satisfied by any triangulation T' with $\rho(T') < \rho$.

▶ **Lemma 5.** *Assuming a polynomial-time oracle for comparing sums of square roots, there is a polynomial-time algorithm that solves the dilation path separation problem.*

Proof. Let Π be the set of all s-t-paths π in G with $|\pi| < \rho \cdot d(s, t)$. We begin by observing that, along every path $\pi \in \Pi$, there is an edge $e \in E$ that is not in T; otherwise, we get a contradiction to $|\pi_T(s, t)| \geq \rho \cdot d(s, t)$. Let $E' \subseteq E \setminus T$ be a set of edges such that for each $\pi \in \Pi$, there is an edge $e \in E'$ on π. Then, $C = \bigvee_{e \in E'} x_e$ is a clause that satisfies the requirements; note that if Π is empty, the empty clause can be returned.

T contains no edge from E', so C is violated by T. Furthermore, if a triangulation T' with $\rho(T') < \rho$ does not contain any of the edges in E', it contains none of the paths in Π. Therefore, $\pi_{T'}(s, t)$ uses an edge that is not in E, which has been excluded from all triangulations with dilation less than ρ; a contradiction. E' can be computed by repeatedly computing shortest s-t-paths π; as long as $\pi < \rho d(s, t)$, we find an edge $e \notin T$ on π, add e to E' and forbid it in future paths. The number of edges bounds the number of iterations of this process; using the comparison oracle, we can efficiently perform each iteration. ◀

For a description of how we compute E' in practice, see the full version.

4.4 Incremental algorithm

Based on the SAT formulation and the algorithm for the dilation path separation problem, IncMDT is simple. Given an initial triangulation T with dilation ρ, we enumerate the set of candidate edges and construct a triangulation supergraph G with bound ρ. We construct the

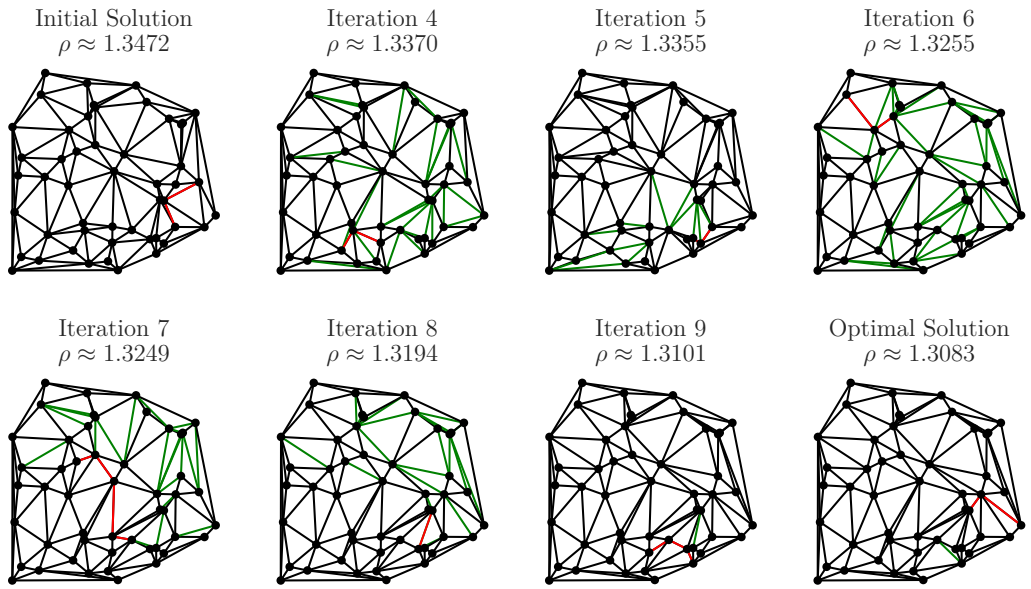


Figure 5 Progress of the incremental algorithm on an instance with $n = 50$ points. Green edges indicate changes in the triangulation, red edges indicate a dilation-defining path.

initial SAT formula M and solve it; if it is unsatisfiable, the initial triangulation is optimal. Otherwise, we repeat the following until the model becomes unsatisfiable or we encounter an edge conflict, keeping track of the best triangulation found, see Figure 5.

We extract the new triangulation T' from the SAT solver and compute the dilation ρ' and a pair s, t of points realizing ρ'. If ρ' is better than the best previously found dilation ρ, we update ρ and mark all edges e with $\vartheta(e) \geq \rho'$ as *impossible*. We then set $T = T'$ and solve the dilation path separation problem for ρ, G, T, s and t. We add the resulting clause to M and let the SAT solver find a new solution.

4.5 Binary search

Preliminary experiments with IncMDT showed that we spend almost all runtime for computing dilations, even for instances for which we could rely exclusively on interval arithmetic, requiring no exact computations. For many instances, most iterations of IncMDT resulted in tiny improvements of the dilation. To reduce the number of iterations (and thus, dilations computed), we considered the binary search-based algorithm BinMDT.

4.5.1 High-level idea

At any point in time, aside from the dilation ρ_{ub} of the best known triangulation, BinMDT maintains a lower bound ρ_{lb} on the dilation, initialized as described in Section 3.

As long as $\rho_{\text{ub}} - \rho_{\text{lb}} \geq \sigma$ for a small threshold value σ, BinMDT performs a binary search. It computes a new dilation bound $\rho = \frac{1}{2}(\rho_{\text{lb}} + \rho_{\text{ub}})$. It then uses the SAT model in a similar way as IncMDT to determine whether a triangulation T with $\rho(T) < \rho$ exists. If it does, it updates $\rho_{\text{ub}} = \rho(T)$; otherwise, it updates $\rho_{\text{lb}} = \rho$.

Once $\rho_{\text{ub}} - \rho_{\text{lb}}$ falls below σ, BinMDT falls back to a slightly modified version of IncMDT to find the MDT, starting from the best known triangulation with dilation ρ_{ub}.

In the following, we describe and motivate the differences between how IncMDT and BinMDT use the SAT formulation; for more details, see the full version.

4.5.2 Dilation sampling

To further reduce the time spent on computing dilations, observe the following. When a node v of some graph G is expanded in Dijkstra's algorithm from source s, we know the shortest path from s to v and thus the dilation $\rho_G(s, v)$. Because $\rho_G(s, v) \leq \rho(G)$, we can compute a lower bound on the dilation much faster than the precise value by only performing a constant number of node expansions from each point $p \in P$. We call this *sampling* of the dilation. Given a bound ρ on the dilation, we can sample a triangulation T for violations, i.e., pairs s, t of points with $\rho_T(s, t) \geq \rho$. We observed that a dilation-defining path usually consisted of few edges; thus, we have a good chance of finding it by sampling.

If it is likely that a new-found triangulation T' violates a given bound ρ, we can thus expect to save time by sampling for violations instead of computing the dilation exactly. Sampling also allows us to use multiple violations to construct multiple clauses in each iteration, potentially further reducing the number of iterations. BinMDT uses sampling after each SAT call with a small constant limit on the number of violations. If violations are found, no full dilation computation is required and violations are used to construct clauses. Only if no violations are found, we compute the exact dilation; ideally, this only happens once for each upper bound reduction in the binary search, namely once we find a triangulation satisfying the current bound. We also sample in the final improvement phase of BinMDT. For an experimental overview on the number of times sampling was sufficient in comparison to the number of times the dilation had to be computed exactly, see the full version.

5 Empirical evaluation

Now we present experiments to evaluate our algorithms. We used Python 3.12, with a core module written in C++20 for all computationally heavy tasks; the code was compiled with GCC 13.2.0 in release mode. We use CGAL 5.6.1 for geometric primitives and exact number types, Boost 1.83 for utility functions and pybind11 2.12 for Python bindings and use the incremental SAT solver CaDiCaL 1.9.5 via the PySAT interface for solving the SAT models. All experiments were performed on Linux workstations equipped with AMD Ryzen 9 7900 CPUs with 12 cores/24 threads and 96 GiB of DDR5-5600 RAM running Ubuntu 24.04.1.

5.1 Research questions

Our experimental evaluation aims to answer the following questions.

Q1 How does the edge enumeration algorithm from Section 3 compare to the brute force enumeration in terms of runtime and the number of edges eliminated? Can we solely rely on this algorithm or should we reconsider using the brute force enumeration?

Q2 What is the quality of the initial solutions computed by the Delaunay triangulation and the constrained Delaunay triangulations from Section 3 compared to the optimal solution?

Q3 How do our approaches compare to existing algorithms for the MDT w.r.t. runtime and solution quality? Can we solve instances with thousands of points to provable optimality?

Q4 How do BinMDT and IncMDT compare? Which should be used for large instances?

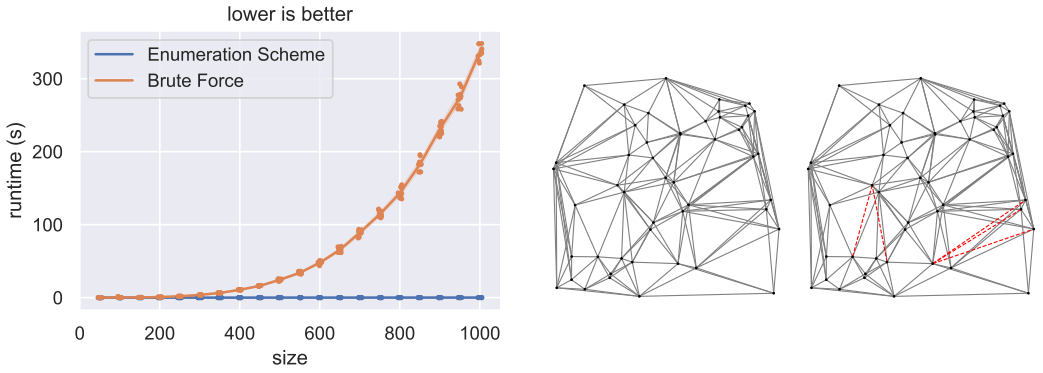


Figure 6 Comparison of our approach from Section 3 and the $\Theta(n^4)$ brute force elimination on the *random* instance set. **(Left)** The $\Theta(n^4)$ elimination is infeasible for larger instances. **(Right)** In all cases, only a small number of (red) edges were not eliminated by our approach.

5.2 Experiment design

To answer our questions, we collected and generated a large set of 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-small We include instances from the work of [8]. The 210 instances have fixed sizes $n \in \{10, 20, \ldots, 70\}$ and were generated by placing uniformly random points inside a 10×10 square. For each size a total of 30 instances were generated.

random We include two sets of randomly generated instances. These encompass a set of instances with points with float coordinates chosen uniformly between 0 and 10^3, ranging from 50 to 10,000 points. This resulted in a total of 800 instances.

public We include instances from all well-known publicly available benchmark instance sets we could locate. These include point sets previously used in the CG:SHOP challenges [17, 16], TSPLIB instances [40], instances from a VLSI dataset[1] and point sets from the Salzburg Database of Polygonal Inputs [22]. In total, we collected 486 instances with up to 10,000 points and an additional 38 with up to 30,000 points.

5.3 Q1: Edge enumeration

We first compare the edge enumeration algorithm from Section 3 to the $\Theta(n^4)$ brute force enumeration of all possible edges on the *random* instance set. Both preprocessing options include finding all pairwise intersections between the enumerated segments. Due to the $\Theta(n^4)$ runtime, we only consider instances with up to 1000 points; see Figure 6.

The $\Theta(n^4)$ algorithm precisely identifies all edges that have the ellipse property; it can hence only eliminate more edges than the edge enumeration algorithm from Section 3. However, for our test instances, the number of edges that can be eliminated by our approach is almost identical to the $\Theta(n^4)$ algorithm; see Figure 6 for an example. The runtime makes the $\Theta(n^4)$ algorithm infeasible for larger instances; it takes more than 250 s for instances with only 1000 points, compared to < 1 s for our more efficient approach.

[1] `https://www.math.uwaterloo.ca/tsp/vlsi/index.html`

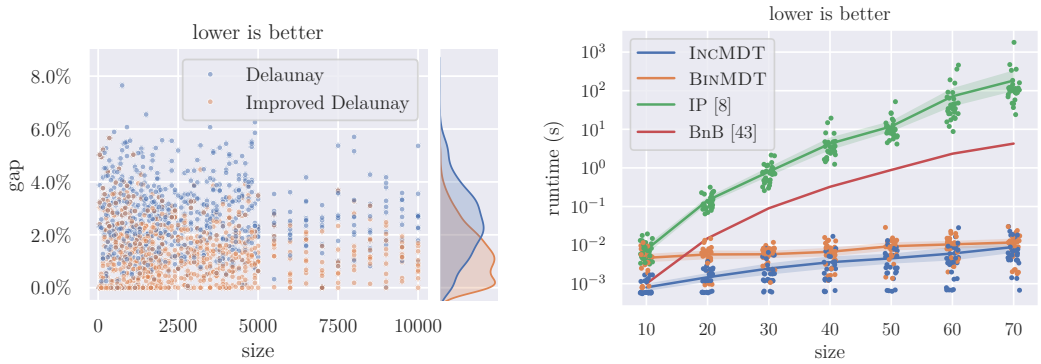


Figure 7 (Left) Initial solution comparison on the *random* set with up to 10,000 points. Delaunay triangulations can be improved by shortcut edges, but are often close to the optimal solution. **(Right)** Runtime comparison with the approaches from [8] and [41] on the *random-small* set.

5.4 Q2: Initial solutions

Better initial solutions can reduce the number of candidate edges further than bad ones and thus affect the runtime of the overall algorithm. Delaunay triangulations are a natural choice for the initial solution, but we suspect that they can be improved by adding shortcut edges. Figure 7 shows that the relative dilation gap between the Delaunay triangulation and the optimal solution for the *random* instance set. It can be seen that the introduction of shortcut edges can indeed reduce the gap to the MDT to around 1.5 %.

5.5 Q3: Comparison to state-of-the-art

We compare our approaches to two exact state-of-the-art algorithms for the MDT. Note that both of these use floating-point arithmetic and are not guaranteed to find the optimal solution. However, we can confirm that all previous solutions are within a small relative error of our optimal solution. The first approach is an integer programming (IP) approach by [8] that used the commercial software CPLEX to solve the MDT. The second comparison is with the most recent exact algorithm (BnB) from Sattari and Izadi [41]. For both approaches the source code is no longer available, so we cannot compare our results on the same hardware. Also for BnB [41] the instance data and results are no longer available. For both approaches we therefore use the data the authors published in their papers. Note that the drastic runtime difference that we see in our experiments cannot be attributed to improved hardware alone.

For *random-small*, both IncMDT and BinMDT outperform the IP and BnB approach by a large margin (up to four orders of magnitude), see Figure 7. All instances were solvable in less than 0.1 s. Additionally, Sattari and Izadi [41] provided results for TSPLIB [40] instances (part of our *public* instance set) with up to 200 points. Our approach is significantly faster than theirs, solving each of these instances to provable optimality in less than 1 s instead of up to 1248 s; a table with all instances and runtimes can be found in the full version.

5.6 Q4: Algorithm comparison

We now compare IncMDT to BinMDT. Recall that BinMDT aims to reduce the time spent on dilation computations by reducing the number of dilation computations and merely sampling the dilation whenever that suffices. The first experiment was conducted on the

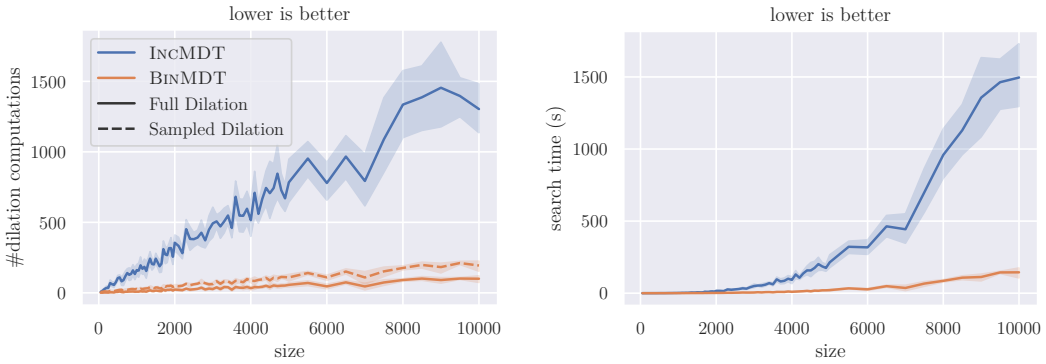


Figure 8 Experiments on the *random* benchmark set. **(Left)** The number of dilation computations of BinMDT is significantly reduced compared to the incremental algorithm. **(Right)** BinMDT is significantly faster than the incremental algorithm.

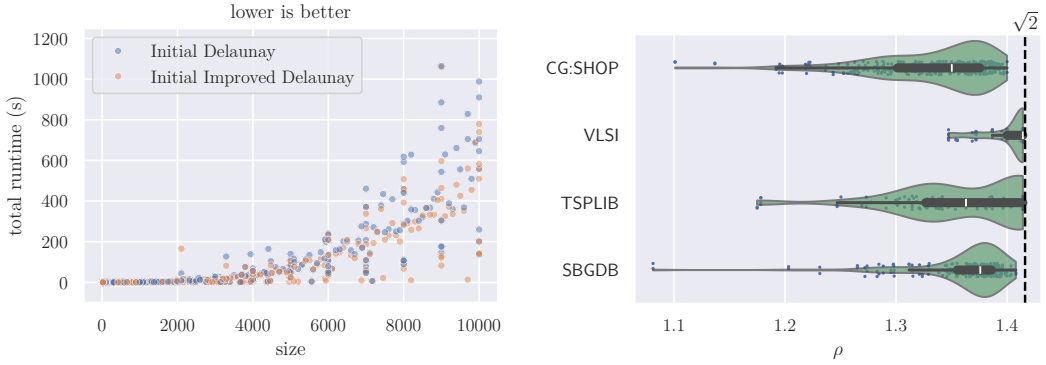


Figure 9 Experiments on the *public* benchmark set. **(Left)** Using the improved Delaunay triangulation as an initial solution significantly improved the performance. **(Right)** The dilation of the MDTs is at most $\sqrt{2}$ for all instances.

random instances with up to 10,000 points; this experiment confirms that BinMDT indeed achieves a significantly lower runtime, requires fewer dilation computations, and that sampling is sufficient in most cases, see Figure 8.

After we confirmed that BinMDT is superior, we conducted an additional experiment with two different initial solution strategies on the *public* instances up to 10,000 points, see Figure 9. The first strategy uses the Delaunay triangulation as an initial solution; the second strategy uses the improved Delaunay triangulation with shortcut edges. The improved Delaunay triangulation significantly reduces the runtime of BinMDT for almost all instances. Detailed results for all *public* instances, as well as an additional experiment on the *public* instance set showing that BinMDT can solve instances with up to 30,000 points in less than 17 h to provable optimality, can be found in the full version.

6 Improved bounds for regular n-gons

Vertex sets of regular n-gons are particularly challenging, as there are no points in the interior to serve as Steiner points for shortcuts, making local heuristics less successful. They are also natural candidates for worst-case dilation, as each diagonal constitutes an obstacle for the

separated points. Thus, they have received considerable attention [35, 20, 43], with a lower bound of 1.4308 (see [20]) and an upper bound of 1.4482 (see [43]) on their worst-case dilation. Improving this gap is an open question posed by [20, Problem 1], originating from [7, 30]. With the help of our exact algorithm (see the full version), we were able to compute bounds for $n \in \{4, 5, \ldots, 100\}$, answering the problem by Dumitrescu and Ghosh.

Our implementation currently uses floating-point precision to represent the coordinates of the input points. Thus, we cannot exactly represent the necessarily irrational points of any regular n-gon ($n \geq 5$) in our solver. However, we can compute the MDT of a rational point set that is a good approximation of a regular n-gon. We can then bound the error and thus obtain a rigorous lower bound on the dilation of the regular n-gon.

▶ **Theorem 6.** *Let P be a set of $n = 84$ points placed at the vertices of a regular n-gon. Then the dilation of the MDT of P is $\rho \geq 1.44116645381$.*

Proof. Let P be the point set of a regular 84-gon and Q be a point set that contains floating-point approximations of the points of P. There is a bijection $f : P \to Q$ with inverse f^{-1} that maps each point in P to the closest point in Q. For a given triangulation or path of P, we write $f(\cdot)$ to denote the triangulation or path where the points are transformed by pointwise application of f. Neither P nor Q contain collinear points and all points are in convex position, therefore T is a triangulation of P iff $f(T)$ is a triangulation of Q.

Let $\epsilon \geq \max_{p_i,p_j \in P} |d(p_i, p_j) - d(f(p_i), f(p_j))|$ be a bound on the maximum absolute error on distances and let δ be a chosen such that

$$\forall p_i, p_j \in P : (1-\delta)d(p_i, p_j) \leq d(f(p_i), f(p_j)) \leq (1+\delta)d(p_i, p_j).$$

Let T be the MDT of P with dilation ρ. Let $f(p_i), f(p_j)$ be a dilation-defining pair in $f(T)$ with path $\pi \subset Q$. Because T has dilation ρ, we have (i) $|f^{-1}(\pi)|/d(p_i,p_j) \leq \rho$. As π has at most $n-1$ edges, we have (ii) $|\pi| - (n-1)\epsilon \leq |f^{-1}(\pi)| \leq |\pi| + (n-1)\epsilon$ and thus the following upper bound on the MDT of Q.

$$\begin{aligned}\frac{|\pi|}{d(f(p_i), f(p_j))} &= \frac{|\pi| - (n-1)\epsilon}{d(f(p_i), f(p_j))} + \frac{(n-1)\epsilon}{d(f(p_i), f(p_j))} \overset{(ii)}{\leq} \frac{|f^{-1}(\pi)|}{(1-\delta)d(p_i,p_j)} + \frac{((n-1)\epsilon)}{d(f(p_i), f(p_j))} \\ &\overset{(i)}{\leq} \frac{1}{1-\delta}\rho + \frac{(n-1)\epsilon}{d(f(p_i), f(p_j))} \leq \frac{1}{1-\delta}\rho + \frac{(n-1)\epsilon}{\min_{q_i,q_j \in Q} d(q_i, q_j)}.\end{aligned}$$

Using exact calculations done with *sympy*, we computed $\epsilon \leq 2.730751 \cdot 10^{-16}$ and $\delta \leq 6.458762 \cdot 10^{-16}$, which, combined with the lower bound on the dilation of Q from our solver gives us $\rho \geq 1.44116645381$. ◀

7 Conclusion

We have presented exact algorithms for minimum dilation triangulations, greatly outperforming previous methods from the literature. This has also yielded insights into the intricate structure of optimal solutions for regular n-gons, together with new lower bounds on the worst-case dilation of triangulations. This demonstrates the value of computational tools for gaining analytic insights that seem out of reach with purely manual analysis.

On the theoretical side, one tantalizing open question remains the complexity of the MDT; here the intricate structure of solutions for regular n-gons may show the way for hardness constructions even for convex arrangements. An equally fascinating problem is the worst-case dilation of triangulations; here we conjecture that the asymptotic dilation for regular n-gons corresponds to the worst-case dilation for general point sets, implying a value between 1.44116 and 1.4482, with a better lower bound achievable via even larger n-gons.

On the practical side, the runtime of our algorithms is dominated by repeatedly computing the dilation of a triangulation. With sums of square roots handled as described, this does not appear to be a difficult problem per se, so appropriately tailored algorithms or data structures for carrying out huge numbers of dilation queries would be extremely helpful.

References

1 Khaled M. Alzoubi, Xiang-Yang Li, Yu Wang, Peng-Jun Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Trans. Parallel Distributed Syst.*, 14(4):408–421, 2003. doi:10.1109/TPDS.2003.1195412.

2 Narayanasetty Amarnadh and Pinaki Mitra. Upper bound on dilation of triangulations of cyclic polygons. In *Computational Science and Its Applications - ICCSA 2006*, volume 3980 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2006. doi:10.1007/11751540_1.

3 Boris Aronov, Mark de Berg, Otfried Cheong, Joachim Gudmundsson, Herman J. Haverkort, Michiel H. M. Smid, and Antoine Vigneron. Sparse geometric graphs with small dilation. *Comput. Geom.*, 40(3):207–219, 2008. doi:10.1016/J.COMGEO.2007.07.004.

4 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.

5 Ahmad Biniaz, Mahdi Amani, Anil Maheshwari, Michiel H. M. Smid, Prosenjit Bose, and Jean-Lou De Carufel. A plane 1.88-spanner for points in convex position. *J. Comput. Geom.*, 7(1):520–539, 2016. doi:10.20382/JOCG.V7I1A21.

6 Johannes Blömer. Computing sums of radicals in polynomial time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 670–677, 1991. doi:10.1109/SFCS.1991.185434.

7 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.

8 Aléx F. Brandt, Miguel M. Gaiowski, Cid C. de Souza, and Pedro J. de Rezende. Minimum dilation triangulation: Reaching optimality efficiently. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014. URL: http://www.cccg.ca/proceedings/2014/papers/paper09.pdf.

9 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, and Sampson Wong. Bicriteria approximation for minimum dilation graph augmentation. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024*, volume 308 of *LIPIcs*, pages 36:1–36:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ESA.2024.36.

10 Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM J. Discret. Math.*, 8(3):359–387, 1995. doi:10.1137/S0895480192237403.

11 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.

12 Otfried Cheong, Herman J. Haverkort, and Mira Lee. Computing a minimum-dilation spanning tree is NP-hard. *Comput. Geom.*, 41(3):188–205, 2008. doi:10.1016/J.COMGEO.2007.12.001.

13 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry*, pages 169–177. ACM, 1986. doi:10.1145/10515.10534.

14 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.

15 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

16 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*, 2020. arXiv:2004.04207.

17 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.

18 Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The open problems project: Problem #33, 2001. URL: http://topp.openproblem.net/.

19 Maria Gisela Dorzán, Mario Guillermo Leguizamón, Efrén Mezura-Montes, and Gregorio Hernández-Peñalver. Approximated algorithms for the minimum dilation triangulation problem. *J. Heuristics*, 20(2):189–209, 2014. doi:10.1007/S10732-014-9237-2.

20 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.

21 Herbert Edelsbrunner and Tiow Seng Tan. A quadratic time algorithm for the minimax length triangulation. *SIAM J. Comput.*, 22(3):527–551, 1993. doi:10.1137/0222036.

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 Friedrich Eisenbrand, Matthieu Haeberle, and Neta Singer. An improved bound on sums of square roots via the subspace theorem. In *Symposium on Computational Geometry (SoCG)*, 2024.

24 David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50010-3.

25 Kai-Wei Fan, Sha Liu, and Prasun Sinha. Scalable data aggregation for dynamic events in sensor networks. In Andrew T. Campbell, Philippe Bonnet, and John S. Heidemann, editors, *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys 2006*, pages 181–194. ACM, 2006. doi:10.1145/1182807.1182826.

26 Arthur M. Farley, Andrzej Proskurowski, Daniel Zappala, and Kurt J. Windisch. Spanners and message distribution in networks. *Discret. Appl. Math.*, 137(2):159–171, 2004. doi:10.1016/S0166-218X(03)00259-2.

27 Sándor P. Fekete, Winfried Hellmann, Michael Hemmer, Arne Schmidt, and Julian Troegel. Computing maxmin edge length triangulations. *J. Comput. Geom.*, 9(1):1–26, 2018. doi:10.20382/JOCG.V9I1A1.

28 Panos Giannopoulos, Rolf Klein, Christian Knauer, Martin Kutz, and Dániel Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comput. Geom. Appl.*, 20(2):147–173, 2010. doi:10.1142/S0218195910003244.

29 Andreas Haas. Solving large-scale minimum-weight triangulation instances to provable optimality. In *34th International Symposium on Computational Geometry, SoCG 2018*, volume 99 of *LIPIcs*, pages 44:1–44:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.SOCG.2018.44.

30 Iyad Kanj. Geometric spanners: Recent results and open directions. In *Third International Conference on Communications and Information Technology, ICCIT 2013*, pages 78–82. IEEE, 2013. doi:10.1109/ICCITECHNOLOGY.2013.6579526.

31 Alexander Klein. Effiziente Berechnung einer dilationsminimalen Triangulierung. Master's thesis, Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn, 2006.

32 Christian Knauer and Wolfgang Mulzer. An exclusion region for minimum dilation triangulations. In *(Informal) Proceedings of the 21st European Workshop on Computational Geometry (EuroCG 2005)*, pages 33–36. Technische Universiteit Eindhoven, 2005. URL: http://www.win.tue.nl/EWCG2005/Proceedings/9.pdf.

33 László Kozma. Minimum average distance triangulations. In Leah Epstein and Paolo Ferragina, editors, *20th Annual European Symposium on Algorithms (ESA 2012)*, volume 7501 of *Lecture Notes in Computer Science*, pages 695–706. Springer, 2012. doi:10.1007/978-3-642-33090-2_60.

34 Joseph SB Mitchell and Wolfgang Mulzer. Proximity algorithms. In *Handbook of Discrete and Computational Geometry*, pages 849–874. CRC Press, Inc., 2017.

35 Wolfgang Mulzer. Minimum dilation triangulations for the regular n-gon. *Master's Thesis. Freie Universität Berlin, Germany.*, 2004.

36 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, 2008. doi:10.1145/1346330.1346336.

37 Giri Narasimhan and Michiel H. M. Smid. Approximating the stretch factor of euclidean graphs. *SIAM J. Comput.*, 30(3):978–989, 2000. doi:10.1137/S0097539799361671.

38 Giri Narasimhan and Michiel H. M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

39 Joseph O'Rourke. Advanced problem 6369. *Amer. Math. Monthly*, 88(10):769, 1981.

40 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991. doi:10.1287/IJOC.3.4.376.

41 Sattar Sattari and Mohammad Izadi. An exact algorithm for the minimum dilation triangulation problem. *J. Glob. Optim.*, 69(2):343–367, 2017. doi:10.1007/S10898-017-0517-X.

42 Sattar Sattari and Mohammad Izadi. Upper bounds for minimum dilation triangulation in two special cases. *Inf. Process. Lett.*, 133:33–38, 2018. doi:10.1016/J.IPL.2018.01.001.

43 Sattar Sattari and Mohammad Izadi. An improved upper bound on dilation of regular polygons. *Comput. Geom.*, 80:53–68, 2019. doi:10.1016/J.COMGEO.2019.01.009.

44 FNU Shariful, Justin Weathers, Anirban Ghosh, and Giri Narasimhan. Engineering an algorithm for constructing low-stretch geometric graphs with near-greedy average-degrees. *CoRR*, 2023. doi:10.48550/arXiv.2305.11312.

45 Israel Vite Silva, Nareli Cruz Cortés, Gregorio Toscano Pulido, and Luis Gerardo de la Fraga. Optimal triangulation in 3D computer vision using a multi-objective evolutionary algorithm. In *Applications of Evolutinary Computing, EvoWorkshops 2007*, volume 4448 of *Lecture Notes in Computer Science*, pages 330–339. Springer, 2007. doi:10.1007/978-3-540-71805-5_36.

46 Victor J. D. Tsai. Delaunay triangulations in TIN creation: An overview and a linear-time algorithm. *Int. J. Geogr. Inf. Sci.*, 7(6):501–524, 1993. doi:10.1080/02693799308901979.

47 Chun-Hsien Wu, Kuo-Chuan Lee, and Yeh-Ching Chung. A delaunay triangulation based method for wireless sensor network deployment. *Comput. Commun.*, 30(14-15):2744–2752, 2007. doi:10.1016/J.COMCOM.2007.05.017.

48 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.

49 Hongyu Zhou, Hongyi Wu, Su Xia, Miao Jin, and Ning Ding. A distributed triangulation algorithm for wireless sensor networks on 2d and 3d surface. In *30th IEEE International Conference on Computer Communications (INFOCOM 2011)*, pages 1053–1061. IEEE, 2011. doi:10.1109/INFCOM.2011.5934879.

On Sparse Covers of Minor Free Graphs, Low Dimensional Metric Embeddings, and Other Applications

Arnold Filtser
Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Abstract

Given a metric space (X, d_X), a (β, s, Δ)-sparse cover is a collection of clusters $\mathcal{C} \subseteq P(X)$ with diameter at most Δ, such that for every point $x \in X$, the ball $B_X(x, \frac{\Delta}{\beta})$ is fully contained in some cluster $C \in \mathcal{C}$, and x belongs to at most s clusters in $\mathcal{C}$. Our main contribution is to show that the shortest path metric of every K_r-minor free graphs admits $(O(r), O(r^2), \Delta)$-sparse cover, and for every $\epsilon > 0$, $(4+\epsilon, O(\frac{1}{\epsilon})^r, \Delta)$-sparse cover (for arbitrary $\Delta > 0$). We then use this sparse cover to show that every K_r-minor free graph embeds into $\ell_\infty^{\tilde{O}(\frac{1}{\epsilon})^{r+1} \cdot \log n}$ with distortion $3+\epsilon$ (resp. into $\ell_\infty^{\tilde{O}(r^2) \cdot \log n}$ with distortion $O(r)$). Further, among other applications, this sparse cover immediately implies an algorithm for the oblivious buy-at-bulk problem in fixed minor free graphs with the tight approximation factor $O(\log n)$ (previously nothing beyond general graphs was known).

2012 ACM Subject Classification Theory of computation → Random projections and metric embeddings; Theory of computation → Computational geometry; Theory of computation → Graph algorithms analysis

Keywords and phrases Sparse cover, minor free graphs, metric embeddings, ℓ_∞, oblivious buy-at-bulk

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.49

Related Version The reader is strongly encouraged to read the full version of the paper.
Full Version: `https://arxiv.org/abs/2401.14060` [43]

Funding This research was supported by the Israel Science Foundation (grant No. 1042/22).

1 Introduction

Given a metric space (X, d_X) a (β, s, Δ)-sparse cover is a collection of clusters $\mathcal{C} \subseteq P(X)$ all with diameter at most Δ, such that every point belongs to at most s clusters, and every ball $B_X(x, \frac{\Delta}{\beta})$ is fully contained in some cluster $C \in \mathcal{C}$. Sparse covers are very useful for algorithmic design, and in particular for divide and concur. Since their introduction by Awerbuch and Peleg [14], sparse covers found numerous applications. A partial list of which includes: compact routing schemes [81, 80, 86, 8, 7, 3, 23], distant-dependent distributed directories [15, 79, 80, 23], network synchronizers [11, 12, 73, 80, 23], distributed deadlock prevention [13], construction of spanners and ultrametric covers [55, 50, 70, 56, 48, 44], metric embeddings [82, 67], universal TSP and Stiner tree constructions [61, 22, 41, 24], and Oblivious buy-at-bulk [84].

We will study sparse covers in weighted graphs, where we distinguish between two different types of diameter. The *weak* diameter of a cluster $A \subseteq V$ in a weighted graph $G = (V, E, w)$ is the maximum pairwise distance $\max_{u,v \in A} d_G(u, v)$ w.r.t. the original shortest path distance, while the *strong* diameter is the maximum pairwise distance $\max_{u,v \in A} d_{G[A]}(u, v)$ in the induced subgraph. We continue with a formal definition:

▶ **Definition 1** (Sparse Cover). *Given a weighted graph $G = (V, E, w)$, a collection of clusters $\mathcal{C} = \{C_1, ..., C_t\}$ is called a weak/strong (β, s, Δ) sparse cover if the following conditions hold.*
1. *Bounded diameter: The weak/strong diameter of every cluster $C_i \in \mathcal{C}$ is bounded by Δ.*

© Arnold Filtser;
licensed under Creative Commons License CC-BY 4.0

2. *Padding: For each $v \in V$, there exists a cluster $C_i \in \mathcal{C}$ such that $B_G(v, \frac{\Delta}{\beta}) \subseteq C_i$.*
3. *Sparsity: For each $v \in V$, there are at most s clusters in $\mathcal{C}$ containing v.*
If the clusters $\mathcal{C}$ can be partitioned into s partitions $\mathcal{P}_1, \ldots, \mathcal{P}_s$ s.t. $\mathcal{C} = \cup_{i=1}^{s} \mathcal{P}_i$, then $\{\mathcal{P}_1, \ldots, \mathcal{P}_s\}$ is called a weak/strong (β, s, Δ) sparse partition cover. We say that a graph G admits a weak/strong (β, s) sparse (partition) cover scheme, if for every parameter $\Delta > 0$ it admits a weak/strong (β, s, Δ) sparse (partition) cover that can be constructed in expected polynomial time. Sparse partition cover scheme is abbreviated SPCS .

The notion of sparse cover scheme is the more common in the literature. However, SPCS provides additional structure that is crucial for different applications. [1] Obtaining strong diameter guarantee is also considerably more challenging, however it is frequently required in "subgraph based" applications. [2]

Sparse covers were introduced by Awerbuch and Peleg [14] who showed that for every $k \in \mathbb{N}$, every n vertex graph admits a strong $(4k-2, 2k \cdot n^{\frac{1}{k}})$-SPCS . Klein, Plotkin, and Rao [65] constructed a celebrated weak padded decomposition [3] for K_r-minor free graphs with padding parameter $O(r^3)$ (later improved to $O(r^2)$ [36]). It is folklore that padded decomposition with padding parameter ρ implies a $(\rho, O(\log n))$-SPCS (by taking the union of $O(\log n)$ independent partitions). Thus [65, 36] implied a weak $(O(r^2), O(\log n))$-SPCS for K_r-minor free graphs. Krauthgamer et al. [67] observed that one can use [65, 36] padded decomposition to construct a weak $(O(r^2), 2^r)$-SPCS for K_r-minor free graphs (see [41] for an explicit proof). This was the first sparse cover where all the parameters are independent from the cardinality of the vertex set n. Busch et al. [23] used shortest path separators to obtain a strong $(8, f(r) \cdot \log n)$-SPCS . [4] Abraham et al. [3] constructed a strong sparse cover. Specifically, they made some adaptations to [65] to obtain a strong $(O(r^2), 2^{O(r)} \cdot r!)$-sparse cover scheme (not a SPCS). The final major piece in our story is a weak padded decomposition for K_r-minor free graphs with padding parameter $O(r)$ by Abraham et al. [6]. Which was later improved to strong padded decomposition with the same parameter by Filtser [38]. In particular, this implies a strong $(O(r), O(\log n))$-sparse cover scheme. Interestingly, it was unknown how to turn this padded decomposition into a sparse cover with sparsity independent of n. Indeed, Filtser explicitly asked whether it is possible to construct $(O(r), g(r))$-sparse cover scheme [38] . The main result of this paper is an affirmative answer to this question. A decade after the publication of [2] we finally managed to turn this padded decomposition into a sparse cover.

▶ **Theorem 2** (Cover for Minor Free Graphs). *Every K_r-minor free graph admits the following:*
- *Strong $\big(O(r), O(r^2)\big)$-SPCS .*
- *For $\epsilon \in (0, \frac{1}{2})$, strong $\big(4+\epsilon, O(\frac{1}{\epsilon})^r\big)$-SPCS .*

The first bullet in Theorem 2 improves over the previous SPCS state of the art [36, 67, 41] in a threefold manner: (1) the padding is improved quadratically from $O(r^2)$ to $O(r)$, (2)

---

[1] In this paper we require the partition property of SPCS for metric embeddings into ℓ_∞, and for the oblivious buy-at-bulk. This property is also crucial in the construction of ultrametric covers [48, 42, 44].
[2] In this paper we use the strong diameter guarantee for routing, buy-at-bulk, and path reporting distance oracle.
[3] Roughly, padded decomposition with padding parameter ρ is a random partition into clusters of diameter at most Δ such that every ball of radius $\frac{\Delta}{\rho}$ is fully contained in a single cluster with probability at least $\frac{1}{2}$. See [38].
[4] $f(r)$ is the enormous constant hiding in the Robertson Seymour [83] structure theorem. Johnson [62] estimated $f(r) \geq 2 \Uparrow (2 \Uparrow (2 \Uparrow \frac{r}{2})) + 3)$ where $2 \Uparrow t$ is the exponential tower function ($2 \Uparrow 0 = 1$ and $2 \Uparrow t = 2^{2 \Uparrow (t-1)}$) .

Table 1 Summary of new and previous work on sparse covers.

Family	Padding	Sparsity	SPCS ?	Diameter	Ref
General	$4k-2$	$2k \cdot n^{\frac{1}{k}}$	yes	strong	[14]
Planar	32	18	yes (implicit)	strong	[23]
K_r-minor free	$O(r^3)$	$O(\log n)$	yes	weak	[65]
	$O(r^2)$	$O(\log n)$	yes	weak	[36]
	$O(r)$	$O(\log n)$	yes	weak	[6]
	$O(r)$	$O(\log n)$	yes	strong	[38]
	8	$f(r) \cdot \log n$ (4)	yes (implicit)	strong	[23]
	$O(r^2)$	2^r	yes	weak	[36, 67]
	$O(r^2)$	$2^{O(r)} \cdot r!$	no	strong	[3]
	$O(r)$	$O(r^2)$	yes	strong	Theorem 2
	$4+\epsilon$	$O(\frac{1}{\epsilon})^r$	yes	strong	

the sparsity is improved exponentially from 2^r to $O(r^2)$, (3) the diameter guarantee is now strong. Alternately, the second bullet improves the padding parameter from $O(r^2)$ to $O(1)$ while keeping a similar sparsity. See Table 1 for a comparison of ours and previous work.

1.1 Low dimensional metric embeddings into ℓ_∞

Metric embedding is a map between two metric spaces that approximately preserves pairwise distances. Given a (finite) metric space (X, d_X), a map $\phi : V \to \mathbb{R}^k$, and a norm $\|\cdot\|$, the *contraction* and *expansion* of the map ϕ are the smallest $\xi, \rho \geq 1$, respectively, such that for every pair $x, y \in X$,

$$\frac{1}{\xi} \cdot d(x,y) \leq \|\phi(x) - \phi(y)\| \leq \rho \cdot d(x,y) \quad .$$

The *distortion* of the map is then $\xi \cdot \rho$. The expansion ρ is also called the Lipschitz constant of the embedding ϕ. Metric embeddings into norm spaces were thoroughly studied [21, 72, 74, 4], and have a plethora of applications.

There is a special interest for metric embeddings into ℓ_∞. From an algorithmic viewpoint, there is a significant advantage in the additional structure the norm space is providing. One example being nearest neighbor search (NNS) [58, 19], where ℓ_∞ enjoys succinct data structures. NNS for many other spaces works by first embedding the space into ℓ_∞, and then using the ℓ_∞ NNS data structure to answer queries in the original metric space (see e.g. [37, 59]). From a geometric view point, ℓ_∞ is a special norm as it is a universal host metric space. That is, every finite metric spaces embeds isometrically (that is with distortion 1) into ℓ_∞ (the so called Fréchet embedding). However, there are n-point metric spaces of which every isometric embedding into ℓ_∞ requires $\Omega(n)$ dimensions [72]. It is desirable to have low dimensional metric embeddings, as these are much more useful for algorithmic design. Matoušek [74] showed that for every integer $t \geq 2$, every metric space embeds with distortion $2t-1$ into ℓ_∞ of dimension $O(n^{1/t} \cdot t \cdot \log n)$ (which is almost tight assuming the Erdős girth conjecture [34]). For distortion $O(\log n)$, Abraham et al. [4] later improved the dimension to $O(\log n)$.

For more restrictive metric spaces better results are known. Linial et al. [72] showed that every n-point tree metric embeds isometrically into $\ell_\infty^{O(\log n)}$, while Neiman [75] showed that every metric space with doubling dimension d embeds into $\ell_\infty^{\epsilon^{-O(d)} \cdot \log n}$ with distortion $1+\epsilon$. Krauthgamer et al. [67] showed that every n-point K_r-minor free graph G embeds

Table 2 Summary of new and previous work on metric embeddings into ℓ_∞.

Family	Distortion	Dimension	Ref
General Metric	1	$n-1$	Fréchet
	$2k-1$	$O(k \cdot n^{1/k} \cdot \log n)$	[74]
	$O(\log n)$	$O(\log n)$	[4]
Tree	1	$\Theta(\log n)$	[72]
K_r-Minor Free	$O(r^2)$	$\tilde{O}(3^r) \cdot \log n$	[67]
	$O(r)$	$\tilde{O}(r^2) \cdot \log n$	Corollary 3
	$3+\epsilon$	$\tilde{O}(\frac{1}{\epsilon})^{r+1} \cdot \log n$	Theorem 4

into $\ell_\infty^{\tilde{O}(3^r \cdot \log n)}$ with distortion $O(r^2)$. Their embedding follows from a SPCS based on [65, 36]. However, their construction uses additional properties of that cover, and we cannot simply plug in our SPCS to get an improved embedding. We show that under very general conditions,[5] one can use SPCS in a black box manner to obtain a metric embedding into ℓ_∞ (see full version [43]). As a corollary, we improve both the distortion and dimension compared to [67]. We than slightly tailor the embedding for minor free graphs to push the distortion down all the way to $3+\epsilon$. See Table 2 for a comparison of new and old results.

▶ **Corollary 3.** *Every n vertex K_r-minor free graph embeds into $\ell_\infty^{O(r^2 \cdot \log r \cdot \log n)}$ with distortion $O(r)$.*

▶ **Theorem 4.** *For every $\epsilon \in (0, \frac{1}{2})$, every n vertex K_r-minor free graph G can be embedded into $\ell_\infty^{O(\frac{1}{\epsilon})^{r+1} \cdot \log \frac{1}{\epsilon} \cdot \log \frac{n}{\epsilon}}$ with distortion $3+\epsilon$.*

1.2 Oblivious Buy-at-Bulk

Given a weighted graph $G = (V, E, w)$ and a *canonical fusion function* $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ (see full version [43] for a definition), in the oblivious buy-at-bulk problem, the goal is to pick a route P_i for every possible demand pair $\delta_i = (s_i, t_i) \in \binom{V}{2}$. Then, given a specific set of demands $A = \{\delta_1, \ldots, \delta_k\}$, the cost of our oblivious solution $\mathcal{P} = \{P_1, \ldots, P_k\}$ is $\text{cost}(\mathcal{P}) = \sum_{e \in E} f(\varphi_e) \cdot w(e)$, where φ_e is the number of paths in $\mathcal{P}$ using e. The solution is said to have approximation ratio ρ if for every subset of demands, the induced cost of the oblivious solution is at most ρ times the optimal solution. Due to the concavity of the canonical fusion function f, it is advantageous for the chosen paths to intersect as much as possible. The best known approximation for general graphs is $O(\log^2 n)$ [53], while for planar graphs $O(\log n)$ approximation is known [84], which is tight [57]. Srinivasagopalan et al. [84] left it as an explicit open problem to "obtain efficient solutions to other related network topologies, such as minor-free graphs." More than a decade later, compared with general graphs, nothing better for K_r-minor free graphs is known. Using our sparse covers on top of a black box reduction from [84], we obtain the following tight result ([57]):

▶ **Corollary 5.** *For every n-vertex weighted K_r-minor free graph $G = (V, E, w)$ admits an efficiently commutable solution to the oblivious buy-at-bulk problem with approximation ratio $O(r^6 \cdot \log n)$. Furthermore, the solution is also oblivious to the concave faction f.*

[5] In fact the reduction hold in general with no condition, while the dimension slightly increases. See full version [43].

1.3 Further Applications

Sparse partition and universal TSP / Steiner tree

Given a weighted graph $G = (V, E, w)$, a (α, τ, Δ)-sparse partition is a partition $\mathcal{C}$ of V into clusters with weak diameter at most Δ, such that every ball of radius $\frac{\Delta}{\alpha}$ intersects at most τ clusters from $\mathcal{C}$. G admits a (α, τ)-sparse partition scheme if it admits (α, τ, Δ)-sparse partition for every $\Delta > 0$. Using our Theorem 2, we construct $\big(O(r), O(r^2)\big)$-sparse partition scheme for K_r minor free graphs, improving over the previous state of the art of $\big(O(r^2), 2^r\big)$-sparse partition scheme [61, 41]. For further details, see full version [43].

In the universal TSP problem, we are given a metric space (X, d_X) and the goal is to choose a single permutation π (a universal TSP) of X, such that given a subset $S \subseteq X$, we visit the points in S w.r.t. the order in π. This is the induced TSP tour by π. The permutation π has stretch ρ, if the length of the induced tour for every subset S is at most ρ times larger than the optimal tour for S. There is a general reduction from sparse partition scheme to universal TSP. Using this reduction and our sparse partitions, given a shortest path metric of an n point K_r minor free graph, we construct universal TSP with stretch $O(r^4) \cdot \log n$, exponentially improving the dependence on r compared with previous results $(O(1)^r \cdot \log n)$. The same phenomena occurs also for the universal Steiner tree problem. See full version [43] for further details.

Name Independent Routing

Here we are given an unweighted graph $G = (V, E)$ where the names (and ports) of all nodes are fixed. The goal is to design a compact routing scheme that will allow sending packages in the network, where routing decisions are made using small local routing tables, and the resulting routing paths are approximate shortest paths. This regime is considered more challenging and practical from the regime where nodes names and ports could be chosen by the routing scheme designer. Given a hereditary graph family that has α-orientation (see full version [43] for a definition), where each graph possess strong (τ, β)-sparse cover scheme, Abraham et al. [3] constructed name independent compact routing scheme with stretch $O(\beta)$, $O(\log n + \log \tau)$-bit headers and tables of $O(\frac{\log^3 n}{\log \log n} + \alpha \cdot \log n) \cdot \tau \cdot \log D$ bits. We use our strong sparse covers (Theorem 2) to construct name independent compact routing scheme significantly improving over previous work. See full version [43] for a comparison.

Path reporting distance oracle

A path reporting distance oracle (PRDO) for a weighted graph $G = (V, E, w)$ is a succinct data structure that given a query $\{x, y\}$, efficiently returns an approximate x-y shortest path P. We say that a PRDO has stretch k and query time t, if for every query (x, y), the oracle returns a path P of weight at most $k \cdot d_G(x, y)$ in $O(|P|) + t$ time. PRDO were first studied for general graphs. Later, Elkin et al. [33] constructed a PRDO for K_r-minor free graphs based on strong sparse covers. We plug in our strong sparse cover from Theorem 2 to obtain improvements in both space and stretch, see full version [43] for details.

1.4 Related Work

We provided background on sparse covers in the introduction. We refer to the cited papers for additional background on sparse partitions, UTSP, UST, routing and distance oracles. Here we provide additional background on metric embeddings in order to put our results (Corollary 3, Theorem 4) in a wider context. We begin with metric embeddings into ℓ_p spaces.

Every n point metric space embeds into $\ell_2^{O(\log n)}$ with distortion $O(\log n)$ [21], which is also tight [72]. Planar graphs, and more generally fixed minor free graphs, embed into $\ell_2^{O(\log n)}$ with distortion $O(\sqrt{\log n})$ [82, 5], which is also tight [76]. The big open question here is regarding the embedding of such graphs into ℓ_1. The upper bounds are the same as for ℓ_2, while the only lower bound is 2 [71]. A long standing conjecture by Gupta et al. [54] states that every graph family excluding a fixed minor, and in particular planar graphs, can be embedded into ℓ_1 with constant distortion. Some partial progress for planar graph was made in the cases where we care only about vertices laying on a small number of faces [68, 40], or only about vertex pairs laying on the same face [78, 69].

Refined notion of distortion in metric embeddings were studied, such as scaling distortion [4, 18], and terminal/prioritized distortion [30, 31]. In particular, there been study of prioritized low dimensional embeddings into ℓ_∞ [45, 32]. Online metric embeddings into normed spaces were also studied [60, 77, 20].

A different venue of research is metric embeddings into trees, or more generally low treewidth graphs. Every n-point metric space stochastically embeds into trees with expected distortion $O(\log n)$ [16, 35]. This result is tight even when the metric is the shortest path metric of a planar graph [9]. Every planar graph with diameter Δ can be (deterministically) embedded into a graph with treewidth $\tilde{O}(\epsilon^{-3})$ with additive distortion $\epsilon \cdot \Delta$ [51, 49, 25]. Every K_r-minor free graphs stochastically embeds into graphs with treewidth $f(r) \cdot O(\frac{\log\log n}{\epsilon})^2$ with expected additive distortion $\epsilon \cdot \Delta$ [28, 49]. Clan embeddings and Ramsey-type embeddings of K_r-minor free graphs were also studied [48]. Finally, recently it was shown that every K_r-minor free graphs stochastically embeds into graphs with treewidth $f(r) \cdot \tilde{O}(\frac{1}{\epsilon}) \cdot \mathrm{poly}(\log(n \cdot \Phi))$ with multiplicative expected distortion $1 + \epsilon$ [29] (here Φ is the aspect ratio).

2 Preliminaries

$\tilde{O}$ notation hides poly-logarithmic factors, that is $\tilde{O}(g) = O(g) \cdot \mathrm{polylog}(g)$. All logarithms are at base 2 (unless specified otherwise), ln stand for the natural logarithm. Given a set A, $\binom{A}{2} = \{\{x, y\} \mid x, y \in A, x \neq y\}$ denotes all the subsets of size 2. For a number $k \in \mathbb{N}$, $[k] = \{1, 2, \ldots, k\}$.

We consider connected undirected graphs $G = (V, E, w)$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$. We say that vertices v, u are neighbors if $\{v, u\} \in E$. Let d_G denote the shortest path metric in G. $B_G(v, r) = \{u \in V \mid d_G(v, u) \leq r\}$ is the closed ball of radius r around v. For a vertex $v \in V$ and a subset $A \subseteq V$, let $d_G(x, A) := \min_{a \in A} d_G(x, a)$, where $d_G(x, \emptyset) = \infty$. For a subset of vertices $A \subseteq V$, $G[A]$ denotes the induced graph on A, and $G \setminus A := G[V \setminus A]$. The *diameter* of a graph G is $\mathrm{diam}(G) = \max_{v,u \in V} d_G(v, u)$, i.e. the maximal distance between a pair of vertices. Given a subset $A \subseteq V$, the *weak*-diameter of A is $\mathrm{diam}_G(A) = \max_{v,u \in A} d_G(v, u)$, i.e. the maximal distance between a pair of vertices in A, w.r.t. to d_G. The *strong*-diameter of A is $\mathrm{diam}(G[A])$, the diameter of the graph induced by A.

The *aspect ratio* of a metric space (X, d_X) is usually defined as the ratio between the maximum and minimum distances. However, here we mainly work with the shortest path distance in graphs, where we allow 0-weights. Formally, here the shortest path metric is actually a pseudometric. Accordingly, we will define aspect ratio of a graph $G = (V, E, w)$ as the ratio between the maximum distance to the minimum non zero distance: $\Phi(G) = \frac{\max_{u,v \in V} d_G(u,v)}{\min_{u,v \in V \text{ s.t. } d_G(u,v) > 0} d_G(u,v)}$.

A graph H is a *minor* of a graph G if we can obtain H from G by edge deletions/-contractions, and isolated vertex deletions. A graph family $\mathcal{G}$ is *H-minor-free* if no graph $G \in \mathcal{G}$ has H as a minor. Some examples of minor free graphs are planar graphs (K_5 and $K_{3,3}$ minor-free), outer-planar graphs (K_4 and $K_{3,2}$ minor-free), series-parallel graphs (K_4 minor-free) and trees (K_3 minor-free).

The ℓ_∞-norm of a vector $x = (x_1, \ldots, x_k) \in \mathbb{R}^k$ is $\|x\|_\infty := \max_{i\in[k]} |x_i|$. An embedding from a metric space (X, d_X) into ℓ_∞ is a function $f : X \to \mathbb{R}^k$. The embedding f has distortion $c \cdot t$ if for every $x, y \in X$, $\frac{1}{c} \cdot d_X(x, y) \leq \|f(x) - f(y)\|_\infty \leq t \cdot d_X(x, y)$. t is the *expansion* (also known as a *Lipschitz* constant) of f, while c is that *contraction* of f. An embedding with distortion 1 (where $c = t = 1$) is called *isometric*. Embedding $f : X \to \ell_\infty^k$ can be viewed as a collection of embeddings $\{f_i\}_{i=1}^k$ into the line $\mathbb{R}$, where $f_i(x)$ equals to the i'th coordinate of $f(x)$. We will also denote $(f(x))_i = f_i(x)$. Using this notation, f has expansion t if for every $x, y \in X$ and $i \in [k]$, $|f_i(x) - f_i(y)| \leq t \cdot d_X(x, y)$. Similarly, f has contraction c if for every $x, y \in X$ there is some $i \in [k]$ such that $|f_i(x) - f_i(y)| \geq \frac{1}{c} \cdot d_X(x, y)$. In this case, we will say that the pair x, y is *satisfied* by the coordinate i.

3 Technical Overview

Cop Decomposition

Abraham et al. [6] constructed a padded decomposition for K_r-minor free graphs based on the cops-and-robbers game [10]. Fix the scale parameter $\Delta > 0$. The process works as follows: pick arbitrary x_1, and let the ball $B_G(x_1, r_1)$ be the first cluster η_1, where $r_1 \in [0, \Delta]$ is sampled using truncated exponential distribution. To construct the second cluster, pick an arbitrary connected component C_2 of $G \setminus \eta_1$, and arbitrary $x_2 \in C_2$. Let T_2 be a shortest path from x_2 to some vertex y with a neighbor in η_1. Then the second cluster $\eta_2 = B_{G[C_2]}(T_2, r_2)$ is a ball around T_2 in the graph induced by the connected component, where the radius $r_2 \in [0, \Delta]$ is sampled using truncated exponential distribution. In general, suppose that we already constructed clusters $\eta_1, \ldots, \eta_{k-1}$. Let C_k be an arbitrary connected component of $G \setminus \cup_{i<k}\eta_i$, and $x_k \in C_k$ arbitrary vertex. Let $\mathcal{K}_{C_k} \subseteq \{\eta_1, \ldots, \eta_{k-1}\}$ be all the previously created clusters η_i, such that there is an edge from η_i to C_k. Let T_k be a shortest path tree in C_k, with x_k as a root, and such that for every $\eta_i \in \mathcal{K}_{C_k}$, there is an edge from a vertex in T_k to η_i. In particular, T_k will have at most $|\mathcal{K}_{C_k}|$ leaves. The k'th cluster $\eta_k = B_{G[C_k]}(T_k, r_k)$ is a ball around T_k in the induced graph $G[C_k]$, where $r_k \in [0, \Delta]$ is sampled using truncated exponential distribution. See Figure 1 for illustration.

One can run the cop decomposition on general graphs without getting any interesting structure. What makes it particularly interesting for K_r-minor free graphs is the fact that the size of the set $\mathcal{K}_{C_k}$ of neighboring previously created clusters is always bounded by $r - 2$. Indeed, one can argue that if $|\mathcal{K}_{C_k}| \geq r - 1$, than one can contract all the internal edges inside each cluster in $\mathcal{K}_{C_k}$, and the connected component C_k, and obtain K_r as a minor. Thus T_k is a shortest path tree (w.r.t. $G[C_k]$) with at most $r - 2$ leaves, and the cluster C_k is a ball of radius at most Δ around this tree. We will call each such cluster a *supernode*, and the shortest path tree T_k it's *skeleton*. In addition, we construct a tree $\mathcal{T}$ over the supernodes. Here each supernode η_k will be the child in $\mathcal{T}$ of the last created supernode $\eta_i \in \mathcal{K}_{C_k}$. Note that the only outgoing edges from η_k are either to its descendants or ancestors in $\mathcal{T}$. Furthermore, η_k have at most $r - 2$ "neighbor" ancestors. We will call the connected component C_k where we constructed the supernode η_k (with skeleton T_k) the *domain* of η_k, denoted $\mathrm{dom}(\eta_k)$. Note that the vertices in $\mathrm{dom}(\eta_k)$ will either belong to η_k, or to the descendants of η_k w.r.t. $\mathcal{T}$. See Figure 1 for illustration.

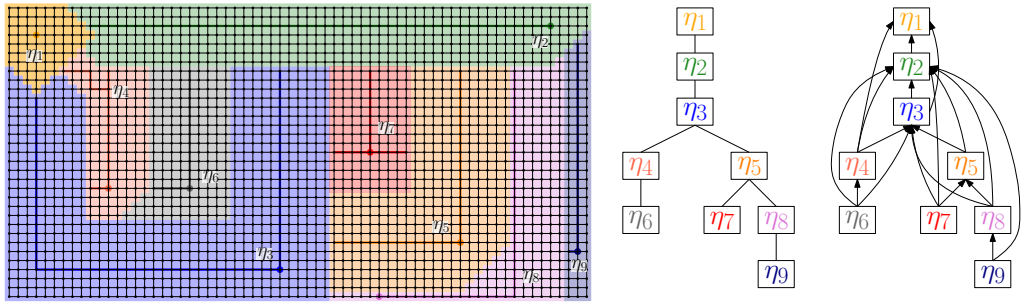


Figure 1 (a) Illustration of an (Δ, γ, w)-buffered cop decomposition of the unweighted grid graph together with (b) - the associated tree $\mathcal{T}$. There are 9 different supernodes $\eta_1, \ldots, \eta_9$, all colored with different colors. Each supernode η contains a shortest path tree T_η (the bold lines) with at most 3 leaves, where all the vertices $x \in \eta$ in the super node are at distance at most $\Delta = 6$ from T_η. The domain of each supernode consist of all the supernodes in its subtree. For example $\mathrm{dom}(\eta_5) = G[\eta_5 \cup \eta_7 \cup \eta_8 \cup \eta_9]$, and $\mathrm{dom}(\eta_3) = G[V \setminus (\eta_1 \cup \eta_2)]$. As η_3 and η_9 are not adjacent, the distance from η_3 to any vertex in η_9 (w.r.t. $\mathrm{dom}(\eta_3)$) is at least γ. The associated digraph $\overrightarrow{G}_{\mathcal{C}}$ is illustrated in (c).

Due to the truncated exponential distribution, Abraham et al. [6] showed that a small ball $B_G(x, \gamma\Delta)$ is likely to be fully contained in a single supernode. [6] Unfortunately, the supernodes $\{\eta_i\}_{i \geq 1}$ do not have a bounded diameter. Nevertheless, following [38], using the skeleton T_k, one can partition each supernode η_k into clusters of diameter $O(\Delta)$, while cutting each small ball only with a small probability. Combining these two processes together, one obtains a strong $(O(r), \Omega(\frac{1}{r}), O(\Delta))$-padded decomposition. However, it is unclear if it is possible to use the cop decomposition to create a sparse cover. Indeed, the "dependence tree" $\mathcal{T}$ does not have a bounded depth, and the entire process looks very chaotic. Indeed, every small change in the sampling of the radii leads to a completely different outcome. In contrast, the [65] (as well as [3]) clustering process had only a depth of r, and thus [67] enumerated all the possible choices in the process leading to a sparse cover of exponential sparsity.

Buffered Cop Decomposition

In a recent work, Chang et al. [26] obtained a new "separation"-property in the cop-decomposition. Instead of growing a ball with a random radius around the skeleton T_k, Chang et al. constructed the supernode T_K deterministically. The new separation property is the following, consider a supernode η, and a vertex $v \in \mathrm{dom}(\eta)$ such that v belongs to a supernode η', which is descendant of η, but there is no edge from η to η'. Then $d_{G[\mathrm{dom}(\eta)]}(\eta, v) > \frac{\Delta}{r} = \gamma$. In other words, for every descendant η' of η, either they are neighbors, or every path in $\mathrm{dom}(\eta)$ from η to η' is of length at least γ. [7] Chang et al. called the new partition *Buffered Cop Decomposition*, because there is now a buffer between

[6] Specifically, using a sophisticated argument, based on a potential function, Abraham et al. [6] showed that with probability at least $e^{-O(r)\cdot\gamma}$, the ball $B_G(x, \gamma\Delta)$ is fully contained in a single supernode.

[7] Roughly speaking, [26] begin with the supernode η_k being equal to the skeleton T_k. Then, as the algorithm progresses, each time the buffer property is violated we add the violating vertices to a previously created supernode. One can argue that the depth of this process is bounded by r (once for each neighboring ancestor supernode), and thus the cluster vertices are all within Δ distance from the skeleton.

non-neighboring clusters. They used the new buffered cop decomposition to construct a *shortcut-partition* (which is a generalization of scattering partition [41]). Roughly, a shortcut-partition is a partition of the vertices into clusters of diameter at most $\epsilon \cdot D$, such that for every pair of vertices u, v at distance at most D, there is an approximate shortest path going through at most $O_r(\frac{1}{\epsilon})$ clusters. Chang et al. used their shortcut-partitions to construct tree covers [17, 25], distance oracles [85, 66, 1], to solve the Steiner point removal problem [39, 41, 64, 27, 46], and to construct additive embedding of apex minor free graphs into low treewidth graphs [51, 28, 47, 49, 25].

Sparse Covers

The starting point of this paper is the buffered cop decomposition of [26]. We begin by observing some additional properties. Let $\overrightarrow{G}_{\mathcal{C}}$ be a digraph where the supernodes are the vertices, and there is a directed edge from a supernode η to its ancestor η' (w.r.t. $\mathcal{T}$) iff there is an edge between vertices in η and η'. $\overrightarrow{G}_{\mathcal{C}}$ is a DAG (directed acyclic graph) with maximum out-degree $r-2$, and it has additional crucial property: if v has outgoing edges towards u, z than there has to be an edge between u and z (in one way or another). We call a graph with this property a *transitive DAG*. In transitive DAG's, neighboring vertices tend to share many of their neighbors. Denote by $B_{\overrightarrow{G}_{\mathcal{C}}}(\eta, q)$ the set of supernodes towards which there is a directed path from η of length at most q. We use the transitive DAG property to show that the size of $B_{\overrightarrow{G}_{\mathcal{C}}}(\eta, q)$ is bounded by $\binom{r+q}{r}$ [8]. Note that crucially, for $q \geq r$, $\binom{r+q}{r} \approx O(q)^r$ the growth rate is sub-exponential in q. Next, we generalize the buffer property to argue that for every ancestor supernode η' of η such that $\eta' \notin B_{\overrightarrow{G}_{\mathcal{C}}}(\eta, 2q+1)$ it holds that the distance from every vertex $v \in \eta$ to η' in $\mathrm{dom}(\eta')$ is at least $(q+1) \cdot \frac{\Delta}{r}$. Combining these two properties together, it follows that for every vertex v there are at most $\binom{r+q}{r}$ ancestor supernodes at distance $q \cdot \frac{\Delta}{r}$.

Similar to the process of padded decomposition [6, 38], our sparse cover is constructed in two steps. First we cover the vertices using *enlarged* supernodes, and then we separately cover each enlarged supernode. Fix $q \geq 0$, and for every supernode η, let $\hat{\eta} = B_{G[\mathrm{dom}(\eta)]}(\eta, q \cdot \frac{\Delta}{r})$ be all the vertices at distance at most $q \cdot \frac{\Delta}{r}$ from η in $\mathrm{dom}(\eta)$. In particular, a vertex $v \in \eta$ can join only to the enlarged supernodes which are ancestors of η at distance at most $q \cdot \frac{\Delta}{r}$. Consider the ball $B = B_G(v, \frac{q}{2} \cdot \frac{\Delta}{r})$, and let η be the first supernode containing some vertex from B. By the minimality of η, and the triangle inequality, it will follow that the ball $B \subseteq \hat{\eta}$ is contained in the enlarged supernode. From the other hand, due to the properties discussed above, each vertex v will belong to at most $\binom{r+q}{r}$ enlarged supernodes. Thus we get both the sparsity and the padding properties we wanted. The only missing property at this point is the bounded diameter. Next we cover each enlarged supernode $\hat{\eta}$ using the skeleton T_η. T_η consist of at most r shortest path. We go over these shortest paths, and choose a Δ-net N. Specifically, a set such that every two net points are at distance at least Δ, and every point has a net point at distance at most Δ. Fix $R = 2\Delta + q \cdot \frac{\Delta}{r}$. Due to the properties of shortest paths, every point has at most $O(r+q)$ net points at distance $2R$. Now taking all the balls of radius R around net points provides us the desired sparse cover for the enlarged supernode. Taking the union of all the covers for all the enlarged supernodes we obtain the sparse cover for the graph. Fixing $q = 1$ we obtain padding $O(r)$ and sparsity $O(r^2)$, while by taking $q = \Theta(\frac{r}{\epsilon})$ we obtain padding $4+\epsilon$ and sparsity $O(\frac{1}{\epsilon})^r$.

[8] It was brought to our attention that this fact actually follows from Theorem 4.2. in [52].

Metric embedding into ℓ_∞

Our metric embedding into ℓ_∞ is based on our SPCS . The first to construct a sparse cover based metric embedding was Rao [82], who used the [65] padded decomposition to embed K_r minor free graphs into ℓ_2 with distortion $O(r^3 \cdot \sqrt{\log n})$. Given a partition $\mathcal{P}$, for a vertex v belonging to cluster $v \in C_v \in \mathcal{P}$, let $\partial_{\mathcal{P}}(v) = d_X(v, V \setminus C_v)$ be the distance between v to the boundary of the cluster C_v. Note that if v is padded $B_G(v, \frac{\Delta}{\beta}) \subseteq C_v$, then $\partial_{\mathcal{P}}(v) > \frac{\Delta}{\beta}$. For each cluster $C \in \mathcal{P}$, sample $\alpha_C \in \{\pm 1\}$ u.a.r. . For every vertex v, send v to $\alpha_{C_v} \cdot \partial_{\mathcal{P}}(v)$. By the triangle inequality, it follows that the expansion is at most 2. But for which vertex pairs can we guarantee small contraction?

Consider a pair u, v at distance $d_G(v, u) \in (\Delta, 2\Delta]$, and suppose that $\mathcal{P}$ has diameter at most Δ, and v is padded in $\mathcal{P}$. Then u and v must belong to different clusters C_u, C_v. If it so happened that $\alpha_{C_u} \neq \alpha_{C_v}$, then $|\alpha_{C_v} \cdot \partial_{\mathcal{P}}(v) - \alpha_{C_u} \cdot \partial_{\mathcal{P}}(u)| = \partial_{\mathcal{P}}(v) + \partial_{\mathcal{P}}(u) \geq \frac{\Delta}{\beta} \geq \frac{d_G(u,v)}{2\beta}$, and we obtain a bound on the contraction! Rao took $O(\log n)$ independent samples of the coefficients $\{\alpha_C\}_{C \in \mathcal{P}}$, and gets the contraction guarantee in a constant fraction of the samples. Next, Rao also took $O(\log n)$ independent partitions to get that v is padded in a constant fraction of them. Finally, Rao sampled partitions for all possible distance scales, concatenated the resulting embeddings of them all, and obtained the desired $O(r^3 \cdot \sqrt{\log n})$ distortion. Assuming all the distances are in $[1, \text{poly}(n)]$, there are $O(\log n)$ different distances scales, and the resulting dimension is $O(\log^3 n)$: one log for the number of scales, one log to sample many partitions for each scale, and one log to sample the coefficients $\{\alpha_C\}_{C \in \mathcal{P}}$. Nevertheless, in ℓ_2, using dimension reduction [63], one can reduce the dimension to $O(\log n)$ without significantly increasing the distortion.

The focus of our paper is embeddings into ℓ_∞. One can repeat Rao's embedding exactly as is, and get embedding into $\ell_\infty^{O(\log^3 n)}$ with distortion $O(r^3)$ (or $O(r)$ using the improved padding parameter from [6]). Unfortunately, there is no general dimension reduction in ℓ_∞ (ala [63]). Nevertheless, the dimension still can be dramatically improved. First, observe that when embedding into ℓ_∞, we don't need to succeed on a constant fraction of the partitions (or the α coefficients), it is enough to be successful only once! Thus, instead taking $O(\log n)$ independent samples from a padded decomposition, one can use a SPCS . Indeed, Krauthgamer et al. [67] constructed an $(O(r^2), 3^r)$-SPCS for K_r minor free graphs. Using this SPCS immediately leads to an embedding into $\ell_\infty^{\tilde{O}(3^r) \cdot \log^2 n}$ with distortion $O(r^2)$. To remove additional $\log n$ factor, [67] used an additional property of the [65] based SPCS that does not holds in general: it is possible to create 3^r partitions such that for every pair u, v, there will be a single partition where both u and v will be padded **simultaneously**. [67] heavily relied on this property, while our SPCS (and actually all the others as well) lacking it. Hence we cannot apply [67] as is.

Our solution follow similar lines to [67], but avoids using the additional special structure of [65]. Consider a graph G with a (β, τ)-SPCS . First, for every scale $\Delta_i = \rho^i$, for $\rho = O(\frac{\beta}{\epsilon})$, create τ partitions $\mathcal{P}_i^1, \mathcal{P}_i^2, \ldots, \mathcal{P}_i^\tau$, all with diameter Δ_i, and such that every vertex is β-padded in one of them (that is $\forall v,\ B_G(v, \frac{\Delta}{\beta})$ is contained in some cluster). Next, create, laminar partitions that closely resemble the original partitions. Specifically, we create $\{\tilde{\mathcal{P}}_i^1\}_{i \in \mathbb{Z}}, \{\tilde{\mathcal{P}}_i^2\}_{i \in \mathbb{Z}}, \ldots, \{\tilde{\mathcal{P}}_i^\tau\}_{i \in \mathbb{Z}}$, where for every i, j, $\mathcal{P}_i^j$ refines $\mathcal{P}_{i+1}^j$, $\mathcal{P}_i^j$ has radius at most $(1+\epsilon) \cdot \Delta_i$, and for every $v \in V$, and $i \in \mathbb{Z}$, $B_G(v, \frac{\Delta}{(1+\epsilon) \cdot \beta})$ is fully contained in a cluster of one of $\tilde{\mathcal{P}}_i^1, \ldots, \tilde{\mathcal{P}}_i^\tau$. In other words, for every $v \in V$, and $i \in \mathbb{Z}$, $\max_{j \in [\tau]} \partial_{\mathcal{P}_i^j}(v) > \frac{\Delta_i}{(1+\epsilon) \cdot \beta}$. Similar laminar partitions were also created in [67].

Next, our goal is to embed w.r.t. each laminar partition $\{\tilde{\mathcal{P}}_i^j\}_{i\in\mathbb{Z}}$ independently, such that for every pair u,v which is separated in partition $\tilde{\mathcal{P}}_i^j$ it will hold that $\|f(u)-f(v)\|_\infty \geq \partial_{\tilde{\mathcal{P}}_i^j}(v)+\partial_{\tilde{\mathcal{P}}_i^j}(u)$. [9] Note that given such embedding for all the laminar partitions, we can concatenate them all to obtain the desired distortion. Indeed, constant expansion follows by triangle inequality, while for every u,v, $\|f(u)-f(v)\|_\infty \geq \max_{i,j}\left(\partial_{\tilde{\mathcal{P}}_i^j}(v)+\partial_{\tilde{\mathcal{P}}_i^j}(u)\right) = \Omega(\frac{d_G(u,v)}{\beta})$. To create the embedding w.r.t. to the laminar partition $\{\tilde{\mathcal{P}}_i^j\}_{i\in\mathbb{Z}}$, let $i_0\in\mathbb{Z}$ be a the maximum index such that $\tilde{\mathcal{P}}_{i_0}^j$ is partitioned into singletons, and let $k\in\mathbb{N}$ be the maximum such that $\tilde{\mathcal{P}}_{i_0+k}^j$ is not the trivial partition into a single cluster $\{V\}$. Clearly it is enough to embed only w.r.t. the laminar partition $\{\tilde{\mathcal{P}}_i^j\}_{i=i_0}^{i_0+k}$. Let $\tilde{\mathcal{P}}_{i_0+k}^j = \{A_1,\ldots,A_m\}$ be the clusters in the top partition in our hierarchy. We create a prefix free code $\alpha:\tilde{\mathcal{P}}_{i_0+k}^j\to\{\pm1\}^*$. Specifically, each cluster A_q is assigned a string of ±1 of length at most $2\cdot\left\lceil\log\frac{|V|}{|A_q|}\right\rceil$. The strings of different clusters might be of different length. However, for every $A_q, A_{q'}$ there is an index s such that $\alpha_s(A_q),\alpha_s(A_{q'})$ exist and differ. Then the embedding of each vertex $v\in A_q$ defined by concatenating the coordinates $\left(\alpha_1(A_q)\cdot\partial_{\tilde{\mathcal{P}}^j_{i_0+k}}(v),\alpha_2(A_q)\cdot\partial_{\tilde{\mathcal{P}}^j_{i_0+k}}(v),\ldots\right)$ with an embedding created inductively for A_q w.r.t. the induced partition by $\{\tilde{\mathcal{P}}_i^j\}_{i=i_0}^{i_0+k-1}$. To bound the contraction, consider a pair u,v and suppose that they are first separated in level $k'\in[i_0,i_0+k]$. Then the embeddings of u and v are "aligned" in scales $[k'+1,i_0+k]$, while in scale k' they belong to respective clusters $A_v,A_u\in\tilde{\mathcal{P}}_{k'}^j$. The respective codes $\alpha(A_v),\alpha(A_u)$ will differ in some coordinate s and thus $\|f(u)-f(v)\|_\infty \geq \left|\alpha_s(A_u)\cdot\partial_{\tilde{\mathcal{P}}^j_{k'}}(u)-\alpha_s(A_v)\cdot\partial_{\tilde{\mathcal{P}}^j_{k'}}(v)\right| = \partial_{\tilde{\mathcal{P}}^j_{k'}}(u)+\partial_{\tilde{\mathcal{P}}^j_{k'}}(v)$. To conclude a bound on the contraction it remains to observe that for every partition $\tilde{\mathcal{P}}_l^j$ that refines $\tilde{\mathcal{P}}_{k'}^j$ it holds that $\partial_{\tilde{\mathcal{P}}^j_{k'}}(v)+\partial_{\tilde{\mathcal{P}}^j_{k'}}(u)\geq\partial_{\tilde{\mathcal{P}}^j_l}(v)+\partial_{\tilde{\mathcal{P}}^j_l}(u)$.

The overall number of coordinates used for the embedding of the laminar partition $\{\tilde{\mathcal{P}}_i^j\}_{i=i_0}^{i_0+k}$ is $2\cdot\lceil\log n\rceil+2(k+1)$, where k have to be bounded by a logarithm of the aspect ratio $O(\log\Phi)$. If we started from a (β,τ)-SPCS , there are τ laminar partitions and thus the overall dimension is $O(\tau\cdot\log(n\cdot\Phi))$. We remove the dependence on the aspect ratio using fairly standard techniques. Specifically, by observing that for far enough scales, we can use the same coordinate. For this to hold, when treating scale ρ^i, we need to "contract" all vertex pairs at distance at most $\frac{\rho^i}{n^2}$ (as otherwise a pair can accumulate error in unbounded number of scales). Hence we can remove the dependence on the aspect ratio only if the contracted graph still admits a SPCS .

References

1 I. Abraham and C. Gavoille. Object location using path separators. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 188–197, 2006. Full version: `https://www.cse.huji.ac.il/~ittaia/papers/AG-TR.pdf`. `doi:10.1145/1146381.1146411`.

2 I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 79–88, 2014. `doi:10.1145/2591796.2591849`.

[9] For comparison, [67] used the simultaneous padding property of [65] and only guaranteed $\|f(u)-f(v)\|_\infty \geq \min\left\{\partial_{\tilde{\mathcal{P}}_i^j}(v),\partial_{\tilde{\mathcal{P}}_i^j}(u)\right\}$.

3 I. Abraham, C. Gavoille, D. Malkhi, and U. Wieder. Strong-diameter decompositions of minor free graphs. *Theory of Computing Systems*, 47(4):837–855, 2010. doi:10.1007/s00224-010-9283-6.

4 Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. *Advances in Mathematics*, 228(6):3026–3126, 2011. doi:10.1016/j.aim.2011.08.003.

5 Ittai Abraham, Arnold Filtser, Anupam Gupta, and Ofer Neiman. Metric embedding via shortest path decompositions. *SIAM J. Comput.*, 51(2):290–314, 2022. a priliminary version apperared in the proceedings of STOC 18. doi:10.1137/19m1296021.

6 Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. *SIAM J. Comput.*, 48(3):1120–1145, 2019. preliminary version published in STOC 2014. doi:10.1137/17M1112406.

7 Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Compact routing for graphs excluding a fixed minor. In Pierre Fraigniaud, editor, *Distributed Computing, 19th International Conference, DISC 2005, Cracow, Poland, September 26-29, 2005, Proceedings*, volume 3724 of *Lecture Notes in Computer Science*, pages 442–456. Springer, 2005. doi:10.1007/11561927_32.

8 Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algorithms*, 4(3):37:1–37:12, 2008. doi:10.1145/1367064.1367077.

9 Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995. preliminary version published in On-Line Algorithms 1991. doi:10.1137/S0097539792224474.

10 Thomas Andreae. On a pursuit game played on graphs for which a minor is excluded. *J. Comb. Theory, Ser. B*, 41(1):37–47, 1986. doi:10.1016/0095-8956(86)90026-2.

11 Hagit Attiya and Jennifer L. Welch. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.)*. Wiley series on parallel and distributed computing. Wiley, 2004.

12 B. Awerbuch and D. Peleg. Network synchronization with polylogarithmic overhead. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 514–522, 1990.

13 Baruch Awerbuch, Shay Kutten, and David Peleg. On buffer-economical store-and-forward deadlock prevention. *IEEE Trans. Commun.*, 42(11):2934–2937, 1994. doi:10.1109/26.328973.

14 Baruch Awerbuch and David Peleg. Sparse partitions. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513, 1990. doi:10.1109/FSCS.1990.89571.

15 Baruch Awerbuch and David Peleg. Concurrent online tracking of mobile users. In Lyman Chapin, editor, *Proceedings of the Conference on Communications Architecture & Protocols, SIGCOMM 1991, Zürich, Switzerland, September 3-6, 1991*, pages 221–233. ACM, 1991. doi:10.1145/115992.116013.

16 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.

17 Yair Bartal, Nova Fandina, and Ofer Neiman. Covering metric spaces by few trees. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 20:1–20:16, 2019. doi:10.4230/LIPIcs.ICALP.2019.20.

18 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.

19 Yair Bartal and Lee-Ad Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces ($2 < p < \infty$). *Theor. Comput. Sci.*, 757:27–35, 2019. doi:10.1016/J.TCS.2018.07.011.

20 Sujoy Bhore, Arnold Filtser, and Csaba D. Toth. Online duet between metric embeddings and minimum-weight perfect matchings. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4564–4579, 2024. doi:10.1137/1.9781611977912.162.

21 J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. doi:10.1007/BF02776078.

22 Costas Busch, Chinmoy Dutta, Jaikumar Radhakrishnan, Rajmohan Rajaraman, and Srinivasagopalan Srivathsan. Split and join: Strong partitions and universal steiner trees for graphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 81–90. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.45.

23 Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Sparse covers for planar graphs and graphs that exclude a fixed minor. *Algorithmica*, 69(3):658–684, 2014. doi:10.1007/S00453-013-9757-4.

24 Ostas 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 *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 60–76, 2023. doi:10.1109/FOCS57990.2023.00012.

25 Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): O(1) trees suffice. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2231–2261, 2023. doi:10.1109/FOCS57990.2023.00139.

26 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, 2024. doi:10.1137/1.9781611977912.191.

27 Yun Kuen Cheung. Steiner point removal - distant terminals don't (really) bother. 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 1353–1360. SIAM, 2018. doi:10.1137/1.9781611975031.89.

28 Vincent Cohen-Addad, Arnold Filtser, Philip N. Klein, and Hung Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. *CoRR*, abs/2009.05039, 2020. To appear in FOCS 2020,https://arxiv.org/abs/2009.05039. arXiv:2009.05039.

29 Vincent Cohen-Addad, Hung Le, Marcin Pilipczuk, and Michał Pilipczuk. Planar and minor-free metrics embed into metrics of polylogarithmic treewidth with expected multiplicative distortion arbitrarily close to 1*. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2262–2277, 2023. doi:10.1109/FOCS57990.2023.00140.

30 M. Elkin, A. Filtser, and O. Neiman. Terminal embeddings. *Theoretical Computer Science*, 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.

31 Michael Elkin, Arnold Filtser, and Ofer Neiman. Prioritized metric structures and embedding. *SIAM J. Comput.*, 47(3):829–858, 2018. doi:10.1137/17M1118749.

32 Michael Elkin and Ofer Neiman. Lossless prioritized embeddings. *SIAM J. Discret. Math.*, 36(3):1529–1550, 2022. doi:10.1137/21M1436221.

33 Michael Elkin, Ofer Neiman, and Christian Wulff-Nilsen. Space-efficient path-reporting approximate distance oracles. *Theor. Comput. Sci.*, 651:1–10, 2016. doi:10.1016/j.tcs.2016.07.038.

34 P. Erdős. Extremal problems in graph theory. In *in "Theory of Graphs and Its Applications," Proc. Sympos. Smolenice*, pages 29–36, 1964.

35 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, November 2004. preliminary version published in STOC 2003. doi:10.1016/j.jcss.2004.04.011.

36 Jittat Fakcharoenphol and Kunal Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *RANDOM-APPROX*, pages 36–46, 2003. doi:10.1007/978-3-540-45198-3_4.

37 Martin Farach-Colton and Piotr Indyk. Approximate nearest neighbor algorithms for hausdorff metrics via embeddings. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 171–180. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814589.

38 Arnold Filtser. On strong diameter padded decompositions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, pages 6:1–6:21, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.6.

39 Arnold Filtser. Steiner point removal with distortion $O(logk)$ using the relaxed-voronoi algorithm. *SIAM J. Comput.*, 48(2):249–278, 2019. doi:10.1137/18M1184400.

40 Arnold Filtser. A face cover perspective to ℓ_1 embeddings of planar graphs. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1945–1954. SIAM, 2020. doi:10.1137/1.9781611975994.120.

41 Arnold Filtser. Scattering and sparse partitions, and their applications. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 47:1–47:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.47.

42 Arnold Filtser. Labeled nearest neighbor search and metric spanners via locality sensitive orderings. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 33:1–33:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SoCG.2023.33.

43 Arnold Filtser. On sparse covers of minor free graphs, low dimensional metric embeddings, and other applications. *CoRR*, abs/2401.14060, 2024. doi:10.48550/arXiv.2401.14060.

44 Arnold Filtser, Yuval Gitlitz, and Ofer Neiman. Light, reliable spanners. *CoRR*, abs/2307.16612, 2023. doi:10.48550/arXiv.2307.16612.

45 Arnold Filtser, Lee-Ad Gottlieb, and Robert Krauthgamer. Labelings vs. embeddings: On distributed and prioritized representations of distances. *Discret. Comput. Geom.*, 2023. doi:10.1007/s00454-023-00565-2.

46 Arnold Filtser, Robert Krauthgamer, and Ohad Trabelsi. Relaxed voronoi: A simple framework for terminal-clustering problems. 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 10:1–10:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASICS.SOSA.2019.10.

47 Arnold Filtser and Hung Le. Clan embeddings into trees, and low treewidth graphs. 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 342–355. ACM, 2021. doi:10.1145/3406325.3451043.

48 Arnold Filtser and Hung Le. Locality-sensitive orderings and applications to reliable spanners. 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 1066–1079. ACM, 2022. doi:10.1145/3519935.3520042.

49 Arnold Filtser and Hung Le. Low treewidth embeddings of planar and minor-free metrics. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1081–1092. IEEE, 2022. doi:10.1109/FOCS54457.2022.00105.

50 Arnold Filtser and Ofer Neiman. Light spanners for high dimensional norms via stochastic decompositions. *Algorithmica*, 2022. doi:10.1007/s00453-022-00994-0.

51 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.

52 Martin Grohe, Stephan Kreutzer, Roman Rabinovich, Sebastian Siebertz, and Konstantinos S. Stavropoulos. Coloring and covering nowhere dense graphs. *SIAM J. Discret. Math.*, 32(4):2467–2481, 2018. doi:10.1137/18M1168753.

53 Anupam Gupta, Mohammad Taghi Hajiaghayi, and Harald Räcke. Oblivious network design. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 970–979. ACM Press, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109665.

54 Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Cuts, trees and l_1-embeddings of graphs. *Comb.*, 24(2):233–269, 2004. doi:10.1007/S00493-004-0015-X.

55 Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *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, 2013. doi:10.1137/1.9781611973105.57.

56 Sariel Har-Peled, Manor Mendel, and Dániel Oláh. Reliable spanners for metric spaces. *ACM Trans. Algorithms*, 19(1):7:1–7:27, 2023. doi:10.1145/3563356.

57 Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discret. Math.*, 4(3):369–384, 1991. doi:10.1137/0404033.

58 Piotr Indyk. On approximate nearest neighbors in non-euclidean spaces. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 148–155. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743438.

59 Piotr Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the 8th Symposium on Computational Geometry*, pages 102–106, Barcelona, Spain, June 2002. ACM Press. doi:10.1145/513400.513414.

60 Piotr Indyk, Avner Magen, Anastasios Sidiropoulos, and Anastasios Zouzias. Online embeddings. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 246–259. Springer, 2010. doi:10.1007/978-3-642-15369-3_19.

61 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for tsp, steiner tree, and set cover. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 386–395. ACM, 2005. doi:10.1145/1060590.1060649.

62 D. S. Johnson. The NP-completeness column: An ongoing guide (column 19). *Journal of Algorithms*, 8(3):438–448, 1987.

63 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. see here.

64 Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting corners cheaply, or how to remove steiner points. *SIAM J. Comput.*, 44(4):975–995, 2015. doi:10.1137/140951382.

65 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, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167261.

66 Philip N. Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 820–827, 2002. see here. URL: http://dl.acm.org/citation.cfm?id=545381.545488.

67 Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: a new embedding method for finite metrics. *Geometric and Functional Analysis*, 15(4):839–858, 2005. preliminary version published in FOCS 2004. doi:10.1007/s00039-005-0527-6.

68 Robert Krauthgamer, James R. Lee, and Havana Rika. Flow-cut gaps and face covers in planar graphs. 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 525–534. SIAM, 2019. doi:10.1137/1.9781611975482.33.

69 Nikhil Kumar. An approximate generalization of the okamura-seymour theorem. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1093–1101. IEEE, 2022. doi:10.1109/FOCS54457.2022.00106.

70 Hung Le and Shay Solomon. A unified framework for light spanners. In Barna Saha and Rocco A. Servedio, editors, *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.

71 James R. Lee and Prasad Raghavendra. Coarse differentiation and multi-flows in planar graphs. *Discret. Comput. Geom.*, 43(2):346–362, 2010. doi:10.1007/S00454-009-9172-4.

72 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Comb.*, 15(2):215–245, 1995. preliminary version published in FOCS 1994. doi:10.1007/BF01200757.

73 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

74 Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996. doi:10.1007/BF02761110.

75 Ofer Neiman. Low dimensional embeddings of doubling metrics. *Theory Comput. Syst.*, 58(1):133–152, 2016. doi:10.1007/S00224-014-9567-3.

76 Ilan Newman and Yuri Rabinovich. A lower bound on the distortion of embedding planar metrics into euclidean space. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 94–96. ACM Press, 2002. doi:10.1145/513400.513412.

77 Ilan Newman and Yuri Rabinovich. Online embedding of metrics. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, June 22-24, 2020, Tórshavn, Faroe Islands*, volume 162 of *LIPIcs*, pages 32:1–32:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.SWAT.2020.32.

78 Haruko Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981. doi:10.1016/S0095-8956(81)80012-3.

79 David Peleg. Distance-dependent distributed directories. *Inf. Comput.*, 103(2):270–298, 1993. doi:10.1006/INCO.1993.1020.

80 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, PA, 2000. doi:10.1137/1.9780898719772.

81 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989. doi:10.1145/65950.65953.

82 Satish Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry, Miami Beach, Florida, USA, June 13-16, 1999*, pages 300–306, 1999. doi:10.1145/304893.304983.

83 N. Robertson and P. D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *Journal of Combinatoral Theory Series B*, 89(1):43–76, 2003. doi:10.1016/S0095-8956(03)00042-X.

84 Srinivasagopalan Srivathsan, Costas Busch, and S. Sitharama Iyengar. Oblivious buy-at-bulk in planar graphs. In Naoki Katoh and Amit Kumar, editors, *WALCOM: Algorithms and Computation - 5th International Workshop, WALCOM 2011, New Delhi, India, February 18-20, 2011. Proceedings*, volume 6552 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2011. doi:10.1007/978-3-642-19094-0_6.

85 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.

86 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10, 2001. doi:10.1145/378580.378581.

Immersions and Albertson's Conjecture

Jacob Fox
Department of Mathematics, Stanford University, CA, USA

János Pach
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Andrew Suk
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Abstract

A graph is said to contain K_k (a clique of size k) as a *weak immersion* if it has k vertices, pairwise connected by edge-disjoint paths. In 1989, Lescure and Meyniel made the following conjecture related to Hadwiger's conjecture: Every graph of chromatic number k contains K_k as a weak immersion. We prove this conjecture for graphs with at most $1.4(k-1)$ vertices. As an application, we make some progress on Albertson's conjecture on crossing numbers of graphs, according to which every graph G with chromatic number k satisfies $\mathrm{cr}(G) \geq \mathrm{cr}(K_k)$. In particular, we show that the conjecture is true for all graphs of chromatic number k, provided that they have at most $1.4(k-1)$ vertices and k is sufficiently large.

2012 ACM Subject Classification Mathematics of computing → Combinatorics

Keywords and phrases Immersions, crossing numbers, chromatic number

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.50

Funding *Jacob Fox*: supported by NSF Award DMS-2154129.
János Pach: Rényi Institute, Budapest. Supported by NKFIH grant K-176529 and ERC Advanced Grant "GeoScape."
Andrew Suk: Supported by NSF awards DMS-1952786 and DMS-2246847.

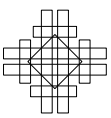
1 Introduction

There are several famous problems in graph theory which state that over all graphs of a given chromatic number, some graph parameter is minimized by a complete graph. Obviously, the chromatic number of a graph is at least its clique number. The converse is false, but partial converses have been of central interest in graph theory. Hadwiger's conjecture states that every graph of chromatic number k contains a K_k-minor. Wagner proved in 1937 that the case $k = 5$ is equivalent to the four color theorem. Hadwiger's conjecture was verified for $k \leq 6$ by Robertson, Seymour, and Thomas [22], and is open for $k \geq 7$. In the 1980's, Kostochka [17] and Thomason [24] proved that every graph of chromatic number k contains a K_t-minor, where $t = \Omega(k/\sqrt{\log k})$, which was improved to $t = \Omega(k/(\log k)^{1/4+\epsilon})$ by Norin, Postle, and Song [20], and very recently, to $t = \Omega(k/\log\log k)$ by Delcourt and Postle [9].

In 1961, Hajós conjectured the following strengthening of Hadwiger's conjecture: Every graph of chromatic number k contains a *subdivision* of the complete graph K_k, i.e., it has k so-called "branch vertices" connected by $\binom{k}{2}$ internally *vertex-disjoint* paths. Hajós' conjecture is true for $k \leq 4$, but for $k \geq 7$ it was disproved by Catlin [8]. In fact, Erdős and Fajtlowicz [13] showed that almost all graphs are counterexamples (see also [14]). The conjecture remains open for $k = 5, 6$.

Lescure and Meyniel [19] suggested a conjecture weaker than Hajós', which may still be true for every k. Instead of requiring that G contains a subdivision of K_k, they wanted to prove the existence of k branch vertices connected by $\binom{k}{2}$ *edge-disjoint* paths. Moreover,

© Jacob Fox, János Pach, and Andrew Suk;
licensed under Creative Commons License CC-BY 4.0



these paths may pass through some branch vertices other than their endpoints. They called such a subgraph of G a *weak immersion* of K_k.

More precisely, a graph G contains H as a weak immersion if there is a mapping ϕ from $V(H) \cup E(H)$, which maps each vertex of H to a vertex in G and each edge of H to a path in G such that

1. $\phi(u) \neq \phi(v)$, for distinct vertices $u, v \in V(H)$;
2. for distinct edges $e, f \in E(H)$, the paths $\phi(e)$ and $\phi(f)$ are edge-disjoint; and
3. for each edge $e = uv \in E(H)$, $\phi(e)$ is a path in G with endpoints $\phi(u)$ and $\phi(v)$.

If the following condition is also satisfied, we say that G contains H as a *strong immersion*.

4. For each edge $e \in E(H)$, the path $\phi(e)$ intersects the set of branch vertices, $\phi(V(H))$, only at its endpoints.

In 1989, Lescure and Meyniel conjectured the following.

▶ **Conjecture 1** ([19]). *Every graph with chromatic number k contains a weak immersion of the complete graph K_k.*

For $k \geq 3$, the Lescure-Meyniel conjecture is an immediate corollary of Hajós' conjecture. DeVos, Kawarabayashi, Mohar, and Okamura [11] verified Conjecture 1 for $4 \leq k \leq 6$. Conjecture 1 remains open for $k \geq 7$. According to a result of Gauthier, Le, and Wollan [16], every graph with chromatic number k contains a weak immersion of K_t, where $t = (k-4)/3.54$ (see also [10, 12] for earlier bounds).

Our first result shows that the Lescure-Meyniel conjecture is true for graphs whose number of vertices is not much larger than its clique number.

▶ **Theorem 2.** *If G is a graph with chromatic number k and at most $1.4(k-1)$ vertices, then G contains K_k as a weak immersion.*

As an application, we use Theorem 2 to obtain new bounds on an old conjecture of Albertson.

The *crossing number* of a graph G, $\mathrm{cr}(G)$, is the smallest number of edge crossings in any drawing of G in the plane. In 2007, Albertson conjectured the following.

▶ **Conjecture 3.** *Every graph G with chromatic number k satisfies $\mathrm{cr}(G) \geq \mathrm{cr}(K_k)$.*

Clearly, Albertson's conjecture is weaker than Hajós' conjecture. Moreover, Conjecture 3 vacuously holds for $k \leq 4$, since $\mathrm{cr}(K_4) = 0$ and, for $k = 5$, Conjecture 3 is equivalent to the four color theorem. After a sequence of results [3, 6, 1], it is now known that Albertson's conjecture holds for $k \leq 18$, but it is open for $k \geq 19$.

A graph G is *k-critical* if $\chi(G) = k$, and every proper subgraph of G has chromatic number less than k. A 1-critical graph is just a graph consisting of a single vertex. As $\mathrm{cr}(G) \geq \mathrm{cr}(H)$ holds for all subgraphs $H \subset G$, it suffices to prove Albertson's conjecture for k-critical graphs. In [6], Barát and Tóth verified Conjecture 3 for all k-critical graphs on at most $k + 4$ vertices, and Ackerman [1] proved the conjecture for all k-critical graphs with at least $3.03k$ vertices. Our next result is the following.

▶ **Theorem 4.** *There is a constant $\delta > 0$ such that the following holds. If k is sufficiently large, then every graph G of chromatic number k on $n \leq (1.4+\delta)k$ vertices satisfies $\mathrm{cr}(G) \geq \mathrm{cr}(K_k)$.*

In the last section, we discuss some concluding remarks on how to improve on the constant factor 1.4 in Theorem 2, which in turn implies a larger range for which Theorem 4 holds.

2 Weak immersion

In this section, we prove Theorem 2. First, let us recall the following lemma. A classic result due to Gallai states that if G is a k-critical graph on n vertices, where $n \leq 2k-2$, then the complement of G is disconnected. This implies the following.

▶ **Lemma 5** ([15]). *Let k, n be positive integers with $n \leq 2k-2$. If G is a k-critical graph on n vertices, then there is a vertex partition*

$$V(G) = V_1 \cup V_2 \cup \cdots \cup V_t,$$

where $t \geq 2$, such that V_i is complete to V_j, for $i \neq j$, $|V_i| = n_i$, and the induced subgraph $G[V_i]$ is k_i-critical with $n_i \geq 2k_i - 1$.

The *chromatic index* $\chi'(H)$ of a multigraph H without loops is the minimum number of colors needed to properly color the edges of H, i.e., to color them in such a way that no two edges that share a vertex receive the same color. A classical theorem of Shannon [23] states that every multigraph H with maximum degree Δ satisfies $\chi'(H) \leq 3\Delta/2$.

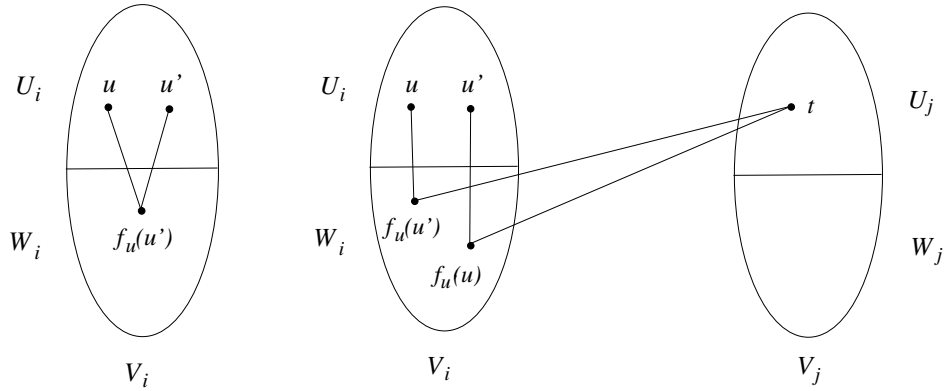
Proof of Theorem 2. We may assume that $k \geq 7$, since otherwise we obtain a weak immersion of K_k by [16]. By possibly deleting vertices and edges, we may assume without loss of generality that G is k-critical. Let $n = |V(G)|$ so $n \leq 1.4(k-1) \leq 2k-2$. By Lemma 5, there is a vertex partition $V(G) = V_1 \cup \cdots \cup V_t$, where $t \geq 2$, such that V_i is complete to V_j for each $i \neq j$, and the induced subgraph $G[V_i]$ is k_i-critical for each i with $n_i \geq 2k_i - 1$ vertices. Hence, $n = \sum_{i=1}^{t} n_i$ and $k = \sum_{i=1}^{t} k_i$. For each i, arbitrarily partition $V_i = U_i \cup W_i$ with $|U_i| = k_i$, so $|W_i| = n_i - k_i$. Let $U = \bigcup_i U_i$.

In what follows, we will construct a weak immersion of K_k with U being the set of branch vertices. Moreover, we will use all edges in U as paths of length one in the weak immersion. By the Gallai decomposition, each nonadjacent pair of vertices in U has both of its vertices in U_i for some i. For each i and vertex $u \in U_i$, let f_u be a one-to-one function from the set of non-neighbors of u in $U_i \setminus \{u\}$ to the set of neighbors of u in W_i. Such a function f_u exists as the degree of u in $G[V_i]$ is at least $k_i - 1$, as $G[V_i]$ is k_i-critical. If a nonadjacent pair (u, u') of vertices in U_i satisfies $f_u(u') = f_{u'}(u)$, then we connect u and u' in the weak immersion by the path of length two with middle vertex $f_u(u')$. Moreover, these paths will be edge-disjoint as f_u is one-to-one. See Figure 1a.

So far we constructed edge-disjoint paths (which are of length one or two) connecting some pairs of branch vertices. We next describe how we connect the remaining pairs of vertices in U by paths, which will each be of length four. For a pair (u, u') of nonadjacent vertices in U_i with $f_u(u') \neq f_{u'}(u)$, we will pick a vertex $t \in U \setminus U_i$ and the path of length four connecting u to u' will have the vertices in order as $u, f_u(u'), t, f_{u'}(u), u'$. We next describe how to pick the vertex $t = t(u, u')$ for each such pair u, u' of nonadjacent vertices in the same U_i with $f_u(u') \neq f_{u'}(u)$. See Figure 1b.

Make an auxiliary multigraph H_i on W_i as follows. For each nonadjacent pair (u, u') with $u, u' \in U_i$ and $f_u(u') \neq f_{u'}(u)$, we add an edge between $f_u(u')$ and $f_{u'}(u)$ in H_i. Clearly, H_i does not contain loops as we require $f_u(u') \neq f_{u'}(u)$. Since f_u is one-to-one, the maximum degree in H_i is at most $|U_i| = k_i$. Being able to pick the desired vertex $t = t(u, u') \in U \setminus U_i$ for each nonadjacent pair $u, u' \in U_i$ with $f_u(u') \neq f_{u'}(u)$, in order to obtain the desired paths of length four for the immersion, is equivalent to being able to properly color the edges of H_i with color set $U \setminus U_i$. As $n_i \geq 2k_i - 1$, we have

$$k_i \leq n_i - k_i + 1 = |W_i| + 1 \leq n - k + 1 \leq 0.4(k-1),$$



(a) $f_u(u') = f_{u'}(u)$ **(b)** $f_u(u') \neq f_{u'}(u)$

Figure 1 Constructing a path from u to u'.

where the last inequality follows from the fact that $n \leq 1.4(k-1)$. This implies that

$$|U \setminus U_i| = k - k_i \geq 3k/5.$$

On the other hand, by Shannon's theorem, we have $\chi'(H_i) \leq 3k_i/2 \leq 3(k-1)/5$. By combining the last two inequalities above, we have $\chi'(H_i) \leq |U \setminus U_i|$, and therefore, we are able to find such a proper edge-coloring, completing the proof. ◀

3 Albertson's conjecture

In this section, we prove Theorem 4.

We recall an old conjecture of Hill, according to which the crossing number of the complete graph on k vertices satisfies $cr(K_k) = H(k)$, where

$$H(k) := \frac{1}{4}\left\lfloor \frac{k}{2} \right\rfloor \left\lfloor \frac{k-1}{2} \right\rfloor \left\lfloor \frac{k-2}{2} \right\rfloor \left\lfloor \frac{k-3}{2} \right\rfloor.$$

It is known that $cr(K_k) \leq H(k)$ by a particular drawing of K_k. In the other direction, Balogh, Lidický, and Salazar [5] proved that $cr(K_k)$ is at least $0.9855H(k)$, for large enough k. In particular, we have the following lemma.

▶ **Lemma 6** ([5]). *If k is sufficiently large, then* $\mathrm{cr}(K_k) > k^4/65$.

A related old conjecture of Zarankiewicz, is that $\mathrm{cr}(K_{a,b}) = \left\lfloor \frac{a}{2} \right\rfloor \left\lfloor \frac{a-1}{2} \right\rfloor \left\lfloor \frac{b}{2} \right\rfloor \left\lfloor \frac{b-1}{2} \right\rfloor$. Towards this conjecture, Balogh et al. [4] recently proved the following result.

▶ **Lemma 7** ([4]). *If a, b are sufficiently large, then* $\mathrm{cr}(K_{a,b}) > .9118a^2b^2/16$.

Before turning to the proof of Theorem 4, as a warm-up, we establish the following useful asymptotic result.

▶ **Lemma 8.** *Let G be a graph on n vertices with $\chi(G) = k$ such that $n \leq 1.4(k-1)$. Then*

$$\mathrm{cr}(G) > \mathrm{cr}(K_k) - k^3/3.$$

In particular, $\mathrm{cr}(G) \geq (1 - o(1))\mathrm{cr}(K_k)$, *as* $k \to \infty$.

Proof. Let G be drawn in the plane with $\mathrm{cr}(G)$ crossings. For a vertex v of G, let $d(v)$ denote the degree of v in G. By Theorem 2, G contains K_k as a weak immersion. Let $v_1, \dots, v_k$ be the branch vertices of the immersion, and let P_{ij} be the path in G used in the weak immersion with endpoints v_i and v_j.

Consider a drawing of K_k in the plane, with vertices $v_1, \dots, v_k$, where the edge between v_i and v_j is drawn along the path P_{ij} such that it goes around every branch vertex that is an internal vertex of P_{ij}. By going either clockwise or counterclockwise around a branch vertex v, we can achieve that in the neighborhood of v, the drawing of the edge between v_i and v_j participates in at most $(d(v)-2)/2$ crossings. Apart from small neighborhoods of the branch vertices along the path P_{ij}, the drawing of the edge connecting v_i to v_j coincides with the drawing of P_{ij}. In particular, the drawing of the edge between v_i and v_j passes through every non-branch vertex that is an internal vertex of P_{ij}.

There are two types of crossings in this drawing of K_k. All crossings that are already crossings in the original drawing of G are of *type 1*, so there are at most $\mathrm{cr}(G)$ of them. The remaining crossings are of *type 2*. They occur in small neighborhoods of vertices of G. The latter crossings fall into two categories depending on whether they occur in a small neighborhood of a non-branch vertex or a branch vertex.

For non-branch vertices v, the number of crossings in the drawing of K_k at v is at most $\binom{f(v)}{2}$, where $f(v)$ denotes the number of paths P_{ij} in the K_k immersion, in which v is an internal vertex of the path P_{ij}. Note that $f(v) \le d(v)/2$, as v is an internal vertex of at most $d(v)/2$ edge-disjoint paths. Obviously, $d(v) \le n-1$ and there are at most $n-k$ non-branch vertices. Therefore, at non-branch vertices, the total number of crossings of type 2 is at most

$$(n-k)\binom{(n-1)/2}{2} \le (n-k)n^2/8.$$

For each of the k branch vertices v_i, there are $k-1$ paths P_{ij} ending at v_i. Thus, v_i is an internal vertex of at most $(d(v_i)-(k-1))/2 \le (n-k)/2$ paths $P_{\ell j}$. In the drawing of K_k, the edge from v_ℓ to v_j participates in at most $(d(v_i)-2)/2 < n/2$ crossings in the neighborhood of v_i, by going either clockwise or counterclockwise around v_i. Thus, in the neighborhoods of branch vertices, altogether there are at most

$$k\frac{n-k}{2}\frac{n}{2}$$

crossings of type 2.

Adding up the above two bounds and using our assumption that $n \le 1.4(k-1)$, we conclude that the total number of crossings in the drawing of K_k, which occur in small neighborhoods of the vertices of G is at most $21k^3/64 < k^3/3$. Thus, we have produced a drawing of K_k with fewer than $\mathrm{cr}(G) + k^3/3$ crossings. Consequently, we have

$$\mathrm{cr}(G) > \mathrm{cr}(K_k) - k^3/3 = (1-o(1))\mathrm{cr}(K_k),$$

as desired. ◀

The well-known crossing lemma discovered by Ajtai, Chvátal, Newborn, Szemerédi [2] and independently, by Leighton [18], states that every graph G with n vertices and $m \ge 4n$ edges satisfies $\mathrm{cr}(G) \ge cm^3/n^2$, where $c > 0$ is an absolute constant. The constant has been improved by several authors. The currently best constant is due to Büngener and Kaufmann.

▶ **Lemma 9** ([7]). *Let G be a graph on n vertices with m edges. If $m \ge 6.95n$, then* $\mathrm{cr}(G) \ge \frac{1}{27.48}\frac{m^3}{n^2}$.

We also need the following two simple lemmas.

▶ **Lemma 10** ([21]). *Let G be a graph with m edges, and let x and y be two nonadjacent vertices. Let $G+xy$ denote the graph obtained by adding the edge (x,y). Then we have*

$$\mathrm{cr}(G+xy) \le \mathrm{cr}(G) + m.$$

▶ **Lemma 11.** *Let G be a graph on n vertices with m edges and $1 \le a \le n$ be an integer. Then G has an induced subgraph on a vertices with at least $m\binom{a}{2}/\binom{n}{2}$ edges.*

Proof. If we take a uniform random subset A of a vertices, the expected number of edges in A is $m\binom{a}{2}/\binom{n}{2}$, and hence, there is an induced subgraph with at least that many edges. ◀

We are now ready to prove Theorem 4. We first prove a variant for k-critical graphs with at most $1.4(k-1)$ vertices.

▶ **Theorem 12.** *There is a constant $c>0$ such that for k sufficiently large, every k-critical graph G on $n \le 1.4(k-1)$ vertices satisfies $\mathrm{cr}(G) \ge \mathrm{cr}(K_k) + c(n-k)k^3$.*

Proof. The proof is by induction on $\ell := n-k$. The base case $\ell = n-k = 0$ is trivial, as in this case $G = K_k$. Let ℓ be a positive integer and suppose we have established the desired result for smaller nonnegative integer values of ℓ. Let k be a large constant that will be specified later, and let G be a k-critical graph on $n \le 1.4(k-1)$ vertices with $n-k=\ell$. By Lemma 5 (Gallai's theorem), G has a vertex partition

$$\mathcal{P} : V(G) = V_1 \cup \ldots \cup V_t$$

into $t \ge 2$ nonempty parts such that each $G[V_i]$ is k_i-critical with n_i vertices with $n_i \ge 2k_i - 1$, and every pair of vertices in different parts are adjacent. In particular, we have $\sum_{i=1}^t k_i = k$ and $\sum_{i=1}^t n_i = n$. Set $\epsilon = 1/8$, $\epsilon' = 2^{-8}$, and $c = 2^{-44}$, say. We distinguish three cases.

Case 1. There is a part V_i with $1 < n_i \le \epsilon k$.

Add missing edges to V_i, one at a time, until V_i is complete. Each time we add an edge, we upper bound the increase of the crossing number by applying Lemma 10. As $G[V_i]$ is k_i-critical, each vertex in $G[V_i]$ has degree at least $k_i - 1$. Hence, the number of nonadjacent pairs in $G[V_i]$ is at most $n_i(n_i-k_i)/2$. In total, by making $G[V_i]$ complete, we increase the crossing number by at most $n^2 n_i(n_i-k_i)/4$. The resulting graph G', obtained by completing part V_i, has n vertices and is k'-critical with $k' = k + n_i - k_i$. Thus,

$$\begin{aligned} \mathrm{cr}(G') &\le \mathrm{cr}(G) + n^2 n_i(n_i-k_i)/4 = \mathrm{cr}(G) + n^2 n_i(k'-k)/4 \\ &\le \mathrm{cr}(G) + \left(\frac{4}{65} - c\right) k^3(k'-k), \end{aligned} \tag{1}$$

where in the last inequality we used that $n \le 1.4(k-1)$, $n_i \le k/8$, and $c = 2^{-44}$.

Applying the induction hypothesis to G', we obtain

$$\mathrm{cr}(G') \ge \mathrm{cr}(K_{k'}) + c(n-k')k'^3. \tag{2}$$

Note that by averaging, we have

$$\begin{aligned} \mathrm{cr}(K_{k'}) &\ge \mathrm{cr}(K_k)\binom{k'}{4}/\binom{k}{4} \ge (k'/k)^4 \mathrm{cr}(K_k) \ge \left(1 + 4\left(\frac{k'}{k} - 1\right)\right)\mathrm{cr}(K_k) \\ &= \mathrm{cr}(K_k) + 4\mathrm{cr}(K_k)(k'-k)/k \ge \mathrm{cr}(K_k) + \frac{4}{65}k^3(k'-k), \end{aligned} \tag{3}$$

where in the last inequality we used Lemma 6.

Putting (1)–(3) together, we obtain

$$\begin{aligned}\mathrm{cr}(G) &\geq \mathrm{cr}(K_{k'}) + c(n-k')k'^3 - \left(\frac{4}{65} - c\right)k^3(k'-k)\\ &\geq \mathrm{cr}(K_k) + ck^3(k'-k) + c(n-k')k'^3\\ &\geq \mathrm{cr}(K_k) + ck^3(n-k).\end{aligned}$$

This completes the proof in this case.

Case 2. There is a part V_i with $k_i \geq \epsilon' k$.

Applying Theorem 2 to G, we obtain K_k as a weak immersion in G. By the proof of Lemma 8, the number of crossings between the edges used in this weak immersion is at least

$$\mathrm{cr}(K_k) - k^3/3. \tag{4}$$

Furthermore, the proof of Theorem 2 shows that no matter how we partition part V_i into $V_i = U_i \cup W_i$ with $|U_i| = k_i$, the edges in $G[W_i]$ are not used in the weak immersion. Note that $G[V_i]$ has minimum degree at least $k_i - 1$, and hence, has at least $n_i(k_i-1)/2$ edges. By Lemma 11, we can pick this partition $V_i = W_i \cup U_i$ so that the number of edges in $G[W_i]$ is at least

$$m_i := \frac{1}{2}n_i(k_i-1)\binom{n_i-k_i}{2}/\binom{n_i}{2} = \frac{1}{2}(k_i-1)(n_i-k_i)(n_i-k_i-1)/(n_i-1).$$

As all three numbers k, $k_i \geq \epsilon' k$, and $n_i - k_i \geq k_i - 1$ are sufficiently large, we have $m_i \geq 6.95(n_i - k_i)$. Thus, we can apply Lemma 9 to obtain that

$$cr(G[W_i]) \geq \frac{1}{27.48}\frac{m_i^3}{(n_i-k_i)^2} \geq 2^{-11}k_i^3(n_i-k_i).$$

We obtain that

$$\begin{aligned}\mathrm{cr}(G) &\geq \mathrm{cr}(K_k) - k^3/3 + \mathrm{cr}(G[W_i])\\ &\geq \mathrm{cr}(K_k) - k^3/3 + 2^{-11}k_i^3(n_i-k_i)\\ &\geq \mathrm{cr}(K_k) + c(n-k)k^3.\end{aligned}$$

In the last inequality, we used that k is sufficiently large, $k_i \geq \epsilon' k$, $n_i - k_i \geq k_i - 1$, $n \leq 1.4(k-1)$, and $c \leq 2^{-44}$. This completes the proof in Case 2.

Case 3. Each part V_i is either a singleton or satisfies $n_i > \epsilon k$ and $k_i < \epsilon' k$.

In this case, as $\ell = n - k > 0$, we must have at least one part that is not a singleton. Recall that $n - k < 0.4k$ and $n - k = \sum_i (n_i - k_i)$. For every i for which V_i is not a singleton, we have $n_i - k_i > \epsilon k - \epsilon' k = (\epsilon - \epsilon')k$. Thus, the number of parts that are not singletons is smaller than $0.4k/(\epsilon - \epsilon')k < 4$, which implies that there are at most *three* non-singleton parts.

Let A be the union of the singleton parts and $B = V(G) \setminus A \neq \emptyset$. Since B is the union of non-singleton parts V_i, each of which is larger than ϵk, we have $|B| > \epsilon k$. The chromatic number of G is k. The chromatic number of $G[B]$, the subgraph of G induced by B, is smaller than $3\epsilon' k$. Using that $A \cup B$ is a vertex partition of G, we obtain that the chromatic number of $G[A]$ is larger than $k - 3\epsilon' k$. As $G[A]$ is a clique, we have $|A| > k - 3\epsilon' k = (1 - 3\epsilon')k$.

It follows by averaging over all cliques of size k in K_{k+1}, just like in (3), that $\mathrm{cr}(K_k)/\binom{k}{4}$ is a monotonically increasing function. Since it is bounded from above, it must converge. As $|A| > (1-3\epsilon')k$, we obtain that

$$\mathrm{cr}(K_{|A|}) \geq (1-12\epsilon')\mathrm{cr}(K_k),$$

provided that k is sufficiently large.

Notice that the clique $G[A]$ and the complete bipartite graph between A and B are disjoint subgraphs of G. Therefore, we get

$$\begin{aligned}\mathrm{cr}(G) &\geq \mathrm{cr}(K_{|A|}) + \mathrm{cr}(K_{|A|,|B|}) \geq (1-12\epsilon')\mathrm{cr}(K_k) + .9118|A|^2|B|^2/16\\ &\geq \mathrm{cr}(K_k) + \left(.9118(1-6\epsilon')\epsilon^2/16 - 12\epsilon'/64\right)k^4\\ &\geq \mathrm{cr}(K_k) + 2^{-7}k^4 > \mathrm{cr}(K_k) + c(n-k)k^3.\end{aligned}$$

Here the second inequality follows by substituting in the bound from Lemma 7 on the crossing number of complete bipartite graphs and using the bound $\mathrm{cr}(K_k) \leq k^4/64$. The last inequality holds with $c = 2^{-44}$, say, because $n - k < .4k$. ◀

Proof of Theorem 4. It suffices to prove the statement for k-critical graphs as every graph of chromatic number k has a k-critical subgraph. So let G be a k-critical graph with chromatic number k with $n \leq (1.4+\delta)k$ vertices. If $n \leq 1.4(k-1)$, then the statement already follows from Theorem 12. So we may assume $1.4(k-1) \leq n \leq (1.4+\delta)k$.

Consider a proper k-coloring of G. Let $k' = (1-3\delta)k$. Let A be the union of the $3\delta k$ largest color classes of this proper k-coloring. Either each of these color classes has size at least two, or the remaining color classes forms a clique on k' vertices.

In the first case, $|A| \geq 6\delta k$, and the remaining induced subgraph after deleting A has chromatic number k'. Let B be the vertex set of a k'-critical subgraph of the induced subgraph on $V(G) \setminus A$. Observe that

$$|B| \leq n - 6\delta k \leq (1.4-5\delta)k = 1.4k' - .8\delta k \leq 1.4(k'-1),$$

provided that $\delta k \geq 7/4$, which holds as k is sufficiently large. By Theorem 12, the induced subgraph on B has crossing number at least $\mathrm{cr}(K_{k'}) + c(|B|-k')k'^3$, where $c > 0$ is an absolute constant (We have seen that $c = 2^{-44}$ will do.) If $|B| \geq 1.2k$, this induced subgraph already has crossing number at least

$$\mathrm{cr}(K_{k'}) + c(|B|-k')k'^3 \geq (1-12\delta)\mathrm{cr}(K_k) + .1ck^4 \geq \mathrm{cr}(K_k) - 12\delta k^4/64 + .1ck^4 \geq \mathrm{cr}(K_k),$$

where we used $\delta = 2^{-45}$. Otherwise, as G is k-critical, each vertex of G has degree at least $k-1$, and there at least $(n-|B|)(k-1)/2 \geq k^2/16$ edges not in $G[B]$. Let G_0 denote the subgraph of G consisting of the edges of G not in $G[B]$. Applying the crossing number bound (Lemma 9) to G_0 and using $n < 2k$, we obtain that

$$\mathrm{cr}(G_0) \geq \frac{1}{27.48}\frac{(k^2/16)^3}{(2k)^2} \geq 2^{-19}k^4.$$

Then

$$\mathrm{cr}(G) \geq \mathrm{cr}(G[B]) + \mathrm{cr}(G_0) \geq \mathrm{cr}(K_{k'}) + 2^{-19}k^4 \geq \mathrm{cr}(K_k) - 12\delta k^4/64 + 2^{-19}k^4 \geq \mathrm{cr}(K_k),$$

where we used $\delta = 2^{-45}$.

In the second case, $|A| = n - k'$ and the edges between A and B form a complete bipartite graph in G with $|A| \geq 2k/5$ and $|B| \geq 4k/5$. This complete bipartite graph has crossing number at least $k^4/200$ by Lemma 7. Together with the $\mathrm{cr}(K_{k'})$ crossings between the edges of the clique of size k', we obtain

$$\mathrm{cr}(G) \geq \mathrm{cr}(K_{k'}) + k^4/200 \geq \mathrm{cr}(K_k) - 12\delta k^4/64 + k^4/200 \geq \mathrm{cr}(K_k).$$ ◀

4 Concluding remarks

One can improve the constant factor 1.4 in Theorem 2, when k is sufficiently large. This can be achieved by being more careful how we pick U and the functions f_u (recall in the proof of Theorem 2, these choices were made arbitrarily). Instead, it is better to make sure that the largest degree vertices in $G[V_i]$ are in U_i, while the remaining vertices in U_i are chosen at random. If there is a pair $u, u' \in U$ for which $f_u(u')$ and $f_{u'}(u)$ haven't already been chosen, then we choose $f_u(u')$, $f_{u'}(u)$ to be equal. Once there is no such pair u, u', we make the remaining choices for the functions f_u uniformly at random. Vizing's theorem [25] states that any multigraph H with maximum degree Δ and maximum edge multiplicity μ satisfies $\chi'(H) \leq \Delta + \mu$. A careful analysis of this procedure produces a considerably smaller bound on the maximum degree of H_i than k_i, and also gives that the maximum edge multiplicity of H_i is $o(k_i)$. This can be proved through optimizing expected values and using standard concentration inequalities.

References

1 E. Ackerman. On topological graphs with at most four crossings per edge. *Comput. Geom.*, 85:101574, 2019. doi:10.1016/J.COMGEO.2019.101574.

2 M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi. On topological graphs with at most four crossings per edge. *Annals of Discrete Mathematics*, 12:9–12, 1982.

3 M. O. Albertson, D. W. Cranston, and J. Fox. Crossings, colorings, and cliques. *Electron. J. Combin.*, 16:11 pp., 2009.

4 J. Balogh, B. Lidický, S. Norin, F. Pfender, G. Salazar, and S. Spiro. Crossing numbers of complete bipartite graphs. *Procedia Comput. Sci.*, 223:78–87, 2023. doi:10.1016/J.PROCS.2023.08.216.

5 J. Balogh, B. Lidický, and G. Salazar. Closing in on Hill's conjecture. *SIAM J. Discrete Math.*, 33:1261–1276, 2019. doi:10.1137/17M1158859.

6 J. Barát and G. Tóth. Towards the Albertson conjecture. *Electron. J. Combin.*, 17:1, 2010.

7 A. Büngener and M. Kaufmann. Improving the crossing lemma by characterizing dense 2-planar and 3-planar graphs. *arxiv:2409.01733*, 17:1, 2024.

8 P. Catlin. Hajós' graph-coloring conjecture: variations and counterexamples. *J. Combin. Theory Ser. B*, 26:268–274, 1979. doi:10.1016/0095-8956(79)90062-5.

9 M. Delcourt and L. Postle. Reducing linear Hadwiger's conjecture to coloring small graphs. *arXiv:2108.01633*, 2021.

10 M. DeVos, Z. Dvořák, J. Fox, J. McDonald, B. Mohar, and D. Scheide. Minimum degree condition forcing complete graph immersion. *Combinatorica*, 34:279–298, 2014. doi:10.1007/S00493-014-2806-Z.

11 M. DeVos, K. Kawarabayashi, B. Mohar, and H. Okamura. Immersing small complete graphs. *Ars Math. Contemp.*, 3:139–146, 2010. doi:10.26493/1855-3974.112.B74.

12 Z. Dvořák and L. Yepremyan. Complete graph immersions and minimum degree. *J. Graph Theory*, 88:211–221, 2018. doi:10.1002/JGT.22206.

13 P. Erdős and S. Fajtlowicz. On the conjecture of Hajós. *Combinatorica*, 1:141–143, 1981. doi:10.1007/BF02579269.

14 J. Fox, C. Lee, and B. Sudakov. Chromatic number, clique subdivisions, and the conjectures of Hajós and Erdős–Fajtlowicz. *Combinatorica*, 33:181–197, 2013. doi:10.1007/S00493-013-2853-X.

15 T. Gallai. Kritische Graphen II. *Publ. Math. Inst. Hungar. Acad. Sci.*, 8:373–395, 1963.

16 G. Gauthier, T.-N. Le, and P. Wollan. Forcing clique immersions through chromatic number. *European J. Combin.*, 81:98–118, 2019. doi:10.1016/J.EJC.2019.04.003.

17 A. V. Kostochka. Lower bound of the Hadwiger number of graphs by their average degree. *Combinatorica*, 4:307–316, 1984. doi:10.1007/BF02579141.

18 T. Leighton. Complexity Issues in VLSI. *Foundations of Computing Series, MIT Press, Cambridge, MA*, 1983.

19 F. Lescure and H. Meyniel. On a problem upon configurations contained in graphs with given chromatic number. *Ann. Discrete Math.*, 41:325–331, 1989.

20 S. Norin, L. Postle, and Z. Song. Breaking the degeneracy barrier for coloring graphs with no K_t minor. *Adv. Math.*, 422:109020, 2023.

21 J. Pach and G. Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9:194–207, 2000.

22 N. Robertson, P. Seymour, and R. Thomas. Hadwiger's conjecture for K_6-free graphs. *Combinatorica*, 13:279–361, 1993.

23 C. E. Shannon. A theorem on coloring the lines of a network. *J. Math. Phys.*, 28:148–152, 1949.

24 A. Thomason. An extremal function for contractions of graphs. *Math. Proc. Cambridge Philos. Soc.*, 95:261–265, 1984.

25 V. G. Vizing. The chromatic class of a multigraph. *Kibernetika*, 3:29–39, 1965.

Quantum Combine and Conquer and Its Applications to Sublinear Quantum Convex Hull and Maxima Set Construction

Shion Fukuzawa
University of California, Irvine, CA, USA

Michael T. Goodrich
University of California, Irvine, CA, USA

Sandy Irani
University of California, Irvine, CA, USA

Abstract

We introduce a quantum algorithm design paradigm called **combine and conquer**, which is a quantum version of the "marriage-before-conquest" technique of Kirkpatrick and Seidel. In a quantum combine-and-conquer algorithm, one performs the essential computation of the combine step of a quantum divide-and-conquer algorithm prior to the conquer step while avoiding recursion. This model is better suited for the quantum setting, due to its non-recursive nature. We show the utility of this approach by providing quantum algorithms for 2D maxima set and convex hull problems for sorted point sets running in $\tilde{O}(\sqrt{nh})$ time, w.h.p., where h is the size of the output.

2012 ACM Subject Classification Theory of computation

Keywords and phrases quantum computing, computational geometry, divide and conquer, convex hulls, maxima sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.51

Funding This work was supported in part by NSF Grant 2212129.

1 Introduction

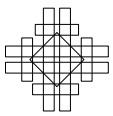
In classical computing, **divide and conquer** is an algorithm design pardigm that is usually described in terms of the following three steps:

1. **Divide**: divide a problem into two or more subproblems.
2. **Conquer**: solve the subproblems, typically using recursion.
3. **Combine**: combine subproblem solutions into a solution to the original problem.

Perhaps because of the wide applicability of algorithm design paradigms, like divide and conquer, there is interest in adapting classical paradigms to the quantum setting. For example, Childs, Kothari, Kovacs-Deak, Sundaram, and Wang [9] describe an adaptation of the divide-and-conquer technique to quantum computing that in some cases results in query complexities better than what is possible classically. Likewise, Allcock, Bao, Belovs, Lee, and Santha [3] study the time complexity of a number of quantum divide-and-conquer algorithms, establishing conditions under which search and minimization problems with classical divide-and-conquer algorithms are amenable to quantum speedups. Akmal and Jin [2] adapt classical divide-and-conquer approaches for string problems to the quantum setting. There is also related work by many others; see, e.g., [4, 6, 8, 14, 25, 35, 37].

A well-known problem that can be solved classically using the divide-and-conquer paradigm is the **convex hull** problem; e.g., see Seidel [34]. In the two-dimensional version of this problem, one is given a set, S, of n points in the plane and asked to output a representation of the smallest convex polygon that contains the points in S. (See Figure 1.)

© Shion Fukuzawa, Michael T. Goodrich, and Sandy Irani;
licensed under Creative Commons License CC-BY 4.0



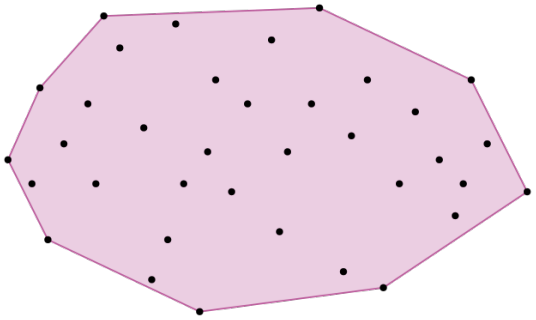


Figure 1 A two-dimensional convex hull.

Asymptotically fast classical convex hull algorithms are **output sensitive**, meaning that their running times depend on both the input size, n, and output size, h, and there are well-known examples that run in $O(n \log h)$ time; e.g., see Kirkpatrick and Seidel [22], Chan [7], and Afshani, Barbay, and Chan [1]. In this paper, we are interested in studying how to efficiently adapt classical output-sensitive divide-and-conquer algorithms, e.g., for convex hull and related problems, to quantum computing models.

Additional Related Work. There is considerable interest in quantum algorithms for solving computational geometry problems. Sadakane, Sugawara, and Tokuyama [32] give a quantum convex hull algorithm that runs in $\tilde{O}(h\sqrt{n})$ time,[1] and this time bound is also achieved by Wang and Zhou [36]. Note that if $h = n$, as can occur, for instance, with points distributed on a circle or parabola, then these algorithms run in $\tilde{O}(n^{3/2})$ time, which is significantly slower than the best classical algorithms. Furthermore, the above quantum algorithms make use of a quantum input data model that assumes that the points are provided in superposition. Many of the procedures require sampling as well, meaning that the algorithms require multiple copies of these input states. We present the algorithm in a standard quantum query model that allows for an even comparison with the existing literature on the subject. In this model, we assume the existence of an oracle that returns the item in the queried index. For completeness, Cortese and Braje [10] showed that there exists a quantum circuit which can take classical data and implement this oracle, with the cost of a logarithmic overhead if the circuit construction time is excluded. Lanzagorta and Uhlmann [23, 24] also describe a quantum convex hull algorithm running in $\tilde{O}(h\sqrt{n})$ time, which, in their case, is a quantum implementation of the well-known Jarvis march convex hull algorithm using Grover search for the inner loop; see, e.g., [11, 16, 29, 30, 33]. They also describe an algorithm that runs in $\tilde{O}(\sqrt{nh})$ time, but this result only applies when h is small and the points are chosen uniformly at random from a bounded convex region. In addition to this quantum prior work, there is a well-developed body of prior work in classical computing models for computing convex hulls where the points are given pre-sorted in lexicographic order; see, e.g., Ghouse and Goodrich [13], Hayashi, Nakano, and Olarlu [18], Berkman, Schieber, and Vishkin [5], Nakano [27], Goodrich [15], and Nakagawa, Man, Ito, and Nakano [26]. The question we seek to answer in this paper is whether it is possible to leverage this assumption in the quantum setting and obtain asymptotically faster output-sensitive algorithms in the model where the input point set is pre-sorted but with no other assumptions about the distribution of points.

[1] We use $\tilde{O}(f(n))$ to denote $O(f(n) \log^c n)$, for some constant $c \geq 0$.

Our Results. In this paper, we introduce **quantum combine and conquer**, which is a novel quantum divide-and-conquer paradigm where we perform the combine step **before** the conquer step and avoid recursion. Our combine-and-conquer paradigm provides a quantum analogue to a classical divide-and-conquer variation that Kirkpatrick and Seidel call "marriage-before-conquest" [20–22]. We show that using this paradigm it is possible to compute the convex hull of a set of points in the plane in $\tilde{O}(\sqrt{nh})$ time given that the input is presorted in lexicographic order. As the sorting problem can be reduced to computing the convex hull, and there is no quantum speed-up for sorting [19], we don't expect quantum algorithms to outperform classical algorithms for this problem in the general case, but this result shows that some reasonable conditions can provide significant speedups. For example, for inputs where h is $O(\log n)$, such as is expected for n points uniformly distributed in a convex polygon with a constant number of sides [17], then our method achieves an $\tilde{O}(n^{1/2})$ running time. Similarly, for inputs where h is $O(n^{1/3})$, such as is expected for n points uniformly distributed in a disk [17, 31], then our method achieves an $\tilde{O}(n^{2/3})$ running time. In general, our algorithms achieve sublinear running times for sorted input sets for h up to almost n. Moreover, our results do not depend on any assumptions about the input points coming from a given distribution, such as uniformly distributed points in a convex region. Indeed, our results hold for adversarial chosen point sets, such as n points on a circle.

To introduce the combine-and-conquer paradigm, we first apply it to constructing the maxima set of a set of lexicographically sorted points in the plane. Our quantum maxima set algorithm runs in $\tilde{O}(\sqrt{nh})$ time, where h is the size of the output. The key idea to our technique is that there is information that can first be computed globally (the combine step), which partitions the computation into the smaller blocks that then use this information to finish the computation locally and (in a conquer step) without recursion.

We then apply our quantum combine-and-conquer paradigm to the convex hull problem. In this case, the combine step is more complex, in that it includes multiple reductions to two-dimensional linear programming. Nonetheless, we show that the combine steps can still be performed before the conquer steps in this case. As a result, we give a quantum convex hull combine-and-conquer algorithm that runs in $\tilde{O}(\sqrt{nh})$ time, where n is the number of (sorted) input points and h is the size of the output. The quantum combine-and-conquer technique provides a novel algorithm design paradigm that may be useful for designing more quantum algorithms that use similar intuition as classical algorithms.

We compare our bounds to previous quantum convex hull algorithms in Table 1. In our algorithm's input model, the data is encoded by a unitary, such that if the input is a list of points, $[p_0, p_1, \ldots, p_{n-1}]$, then we assume access to a unitary U_{in} that maps an index i to the corresponding element in the array p_i: $|i\rangle|0\rangle \xrightarrow{U_{in}} |i\rangle|p_i\rangle$. The existing literature on quantum solutions to this problem assume that the data is given in a uniform superposition state, $(1/\sqrt{n})\sum_i |i\rangle|p_i\rangle$. Access to the unitary U_{in} is a somewhat stronger assumption in that the uniform superposition state can be prepared in unit time given access to U_{in}. On the other hand, the most natural way to prepare such a state is via access to U_{in}, as we do.

2 Preliminaries

We assume basic familiarity with quantum computing; see, e.g., Nielsen and Chuang [28]. A **quantum state** over n qubits is a unit vector of length 2^n with complex entries. The computational basis $\{|j\rangle\}_{j\in[0,\ldots,2^n-1]}$ is a basis over quantum states where $|j\rangle$ represents the unit vector of length n with a 1 in the j-th index and 0 elsewhere. A computational basis state can be used as a classical bit string to simulate any classical algorithm. We can express any quantum state as a weighted sum of these classical basis states,

Table 1 A summary of quantum algorithms to compute the convex hull of a set of n points, where the output has h edges. The algorithms all assume access to a data loading unitary.

	Input point set assumptions	**Runtime**
Sadakane, Sugawara, and Tokuyama [32]	Unsorted, arbitrary point set	$\tilde{O}(h\sqrt{n})$
Wang and Zhou [36]	Unsorted, arbitrary point set	$\tilde{O}(h\sqrt{n})$
Lanzagorta and Uhlmann [23, 24]	Small h, uniformly distributed points	$\tilde{O}(\sqrt{nh})$
This work	Sorted, arbitrary point set	$\tilde{O}(\sqrt{nh})$

$$|\Psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle . \tag{1}$$

If at least two α_j in the above are nonzero, we say the state is in **superposition**. A **measurement** of the state $|\Psi\rangle$ will collapse the state to $|j\rangle$ with probability $|\alpha_j|^2$. The measurement is destructive in the sense that once it is collapsed, there is no way to go back to $|\Psi\rangle$ without running the circuit to prepare that state. The state where all $\alpha_j = \frac{1}{\sqrt{2^n}}$ can be prepared by a circuit of depth 1.

We will assume an input model where the input data $[p_0, p_1, \ldots, p_{n-1}]$ is accessible by a unitary U_{in} that maps an index i to the corresponding element in the array p_i. Since quantum operations must be reversible, it is standard to model the action of this unitary by using an index and data register as $|i\rangle |0\rangle \overset{U_{in}}{\to} |i\rangle |p_i\rangle$. We can use linearity to examine the action of this unitary to a state in superposition

$$U_{in}\left(\sum_{j=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |j\rangle |0\rangle\right) = \sum_{j=0}^{2^n-1} \frac{1}{\sqrt{2^n}} (U_{in} |j\rangle |0\rangle) = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle |p_j\rangle , \tag{2}$$

from which the data point p_j can be retrieved with probability $|\alpha_j|^2$ upon measurement of the index register. Therefore, in this model, we assume that a quantum state representing a distribution of our inputs is preparable in time proportional to the time it takes to prepare the distribution. A transformation from the n-bit zero string $|0^n\rangle$ to a uniform superposition over all n-bit strings can be accomplished by a depth 1 circuit where a Hadamard gate is applied to each qubit in parallel.

Much of the existing literature on this subject assumes access to multiple copies of the state in (2). Our assumption of access to U_{in} is at least as strong of an assumption as having multiple copies of the state. At the same time, it is a standard assumption for states in the form of (2) are generated using access to a unitary like U_{in}. We therefore compare the performance of our algorithm against the existing literature under equal access to U_{in}, and we summarize the performance of each in Table 1 under this model.

Quantum Subroutines. In this section we discuss the two primary quantum subroutines that will be used throughout the paper. The quantum computing components of our algorithms are limited to invocations of these subroutines, and the remainder of the algorithms are done through classical post-processing.

The first subroutine is for preparing superposition states of subsets of points. Our algorithms use the fact that the data $S = [p_0, p_1, \ldots, p_{n-1}]$ is encoded as a sorted list in U_{in}, and we need to prepare h states $|S_0\rangle, |S_1\rangle, \ldots, |S_{h-1}\rangle$ where $|S_j\rangle$ is a uniform superposition of n/h consecutive elements in the array starting at index jn/h and h is a power of 2. Preparation of the $|S_j\rangle$ begins with a register storing an n-bit zero string $|0^n\rangle$, and another register large enough to store a point from the dataset, each of which we will call the index and data register respectively. The first step is to set the first $\log_2 h$ bits of the index register to j. Next, take the remaining $n - \log_2 h$ bits in the index register and prepare a uniform superposition state over the bitstrings of length $n - \log_2 h$. Finally, apply U_{in} on this state and the data register to prepare the superposition of n/h states starting from jn/h,

$$|S_j\rangle = \frac{1}{\sqrt{n/h}} \sum_{k=0}^{n/h-1} |jn/h + k\rangle_{I_j} |p_{jn/h+k}\rangle_{D_j} . \tag{3}$$

We summarize this process in Algorithm 1. Throughout this paper, we use S_j to represent the subset of S storing points in the j-th block, and $|S_j\rangle$ to be the quantum state that encodes this set, as seen in Equation (3).

Algorithm 1 `qPrep`(j, h).

Require: j: The index of the block to prepare; h: the number of blocks to partition into.
Require: Global access to the unitary U_{in} that encodes the sorted data $[d_0, \ldots, d_{n-1}]$
1: Prepare two registers, initialized to $|0^n\rangle_{I_j} |0\rangle_{D_j}$.
2: Set the first $\log_2 h$ bits of I_j to j
3: Apply a Hadamard gate to each remaining qubits in I_j.
4: Apply U_{in} to the pair of registers.
5: **return** $|S_j\rangle$, the state after the previous step as described in equation 3.

The second quantum subroutine we use is a quantum max/min finding algorithm due to Durr and Hoyer [12]. This algorithm takes as input a Boolean function, $f : D \to \{0, 1\}$, a comparator, $\preceq$, to maximize (or minimize) over, the index, j, of the block to perform the search over, and the total number of blocks, h. Our subroutine uses `qPrep`(j) to prepare the superposition state described in Theorem 1. This has the effect of applying the algorithm only to the points in block j, and allows us to search the block in $\tilde{O}(\sqrt{n/h})$ time. Throughout the text, we use the signature, `qMax/qMin(`f`, `$\preceq$`, `j`, `h`)`, when calling this function, and the output is an element of D or null if there are no elements such that $f(d) = 1$. The function f can be passed as, say, a classical circuit which can be converted to a quantum circuit.

▶ **Theorem 1** (Quantum Maximum/Minimum Finding [12]). *Let $D = [d_0, \ldots, d_{n-1}]$ be a list of n elements represented by w bits, and let S be the time required to prepare the state,*

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n} |i\rangle |d_i\rangle , \tag{4}$$

and Q the time it takes to query a boolean function $f : D \to \{0, 1\}$. Let M be the subset of D such that $f(m) = 1$ for all $m \in M$. Also, let $\preceq$ be some ordering of data values in the data register such that comparisons according to this ordering can be performed in $O(\log w)$ time. Then we can find the maximum (or minimum) value of M under the specified ordering in time $S \cdot Q \cdot \tilde{O}(\sqrt{n})$.

3 Quantum Maxima Sets

In this section, we present a quantum algorithm to solve the maxima set problem using the combine-and-conquer paradigm. In the maxima set problem, we are given a set, S, of n points in the plane and are tasked with finding the subset of points in S that are not dominated by any other points in S. We say that a point p **dominates** a point q if $p.x > q.x$ and $p.y > q.y$. Note that the output size, h, can be as small as 1 or as large as n.

We assume that the input set is sorted in lexicographic order (first by x-coordinates and then by y-coodinates in the case of ties), as discussed in the introduction.

Let us begin by reviewing a simple divide-and-conquer algorithm in the classical computing model, which is provided the set, S, of n points as input in sorted order, and returns the maxima set, M (see Preparata and Shamos [30]). The algorithm splits S into left and right sets, S_1 and S_2, and recursively finds the maxima sets of each. Then it prunes away each point in S_1 with smaller y-coordinate than the point, $p_{\max}$, in S_2 with maximum y-coordinate. The resulting maxima set algorithm, which we give in detail in an appendix, takes $\Theta(n \log n)$ time, even when S is sorted [30]. Kirkpatrick and Seidel [21] show that if the combine step of finding $p_{\max}$ is performed before the divide and conquer steps, and used to prune points in recursive subproblems, then the running time can be improved to $O(n \log h)$.

For our output-sensitive quantum algorithm, we also perform the combine step before the conquer step, but our goal is to design a quantum algorithm that runs in $\tilde{O}(\sqrt{nh})$ time without recursion, which requires a more sophisticated approach. We first describe an algorithm where the output size, h, is known in advance and later show how to use this algorithm to solve the case where the output size is not known.

At a high level, the combine-and-conquer approach performs the combine step first before conquering the subproblems. In the maxima set problem, this is achieved in the following way. We begin by dividing our set S into h different blocks using Algorithm 1. The key observation we use is that we can isolate the problem of finding the maxima set to each of the local blocks once we identify representative information from each block. This representative information is used globally, so we describe the process of finding such information as the **combine step**. The combine-and-conquer paradigm works for problems where, once the global information is identified, the blocks do not need to communicate with each other for the remainder of the algorithm. In the case of computing the maxima set, the combine step consists of searching for the set T of the h highest points in each block. Once the set T is found, we classically compute the maxima set M_T of T. Note that M_T cannot contain more than h points, and if a block does not share a point with M_T no points in that block can be in the final output set. Once the points in M_T are identified, we begin the **conquer step** of the algorithm to find the points in each block that appear in the maxima set. Block number j uses information from two points identified in the combine step to restrict the search region. The first is the y coordinate T_j of the tallest point within the j-th block, which immediately removes all points to the left of it inside block j out of the candidate set. The second is the y coordinate R_j of the highest point that appears in any of the blocks to the right of block j. Note that R_j is left-most point from M_T that is to the right of block j. See Figure 2.

Using T_j and R_j, we can define a Boolean function f that takes as input a point p and returns 1 if p is not dominated by either of T_j and R_j. We then apply Theorem 1 to search for the maximum element in lexicographic order subject to this Boolean function. It turns out that this point is part of the final output set, as R_j tells us it is not dominated by any point in blocks to the right of j and T_j tells us it is not dominated by any point in block j. We then update R_j to equal this maximal point we found, and repeat this process in

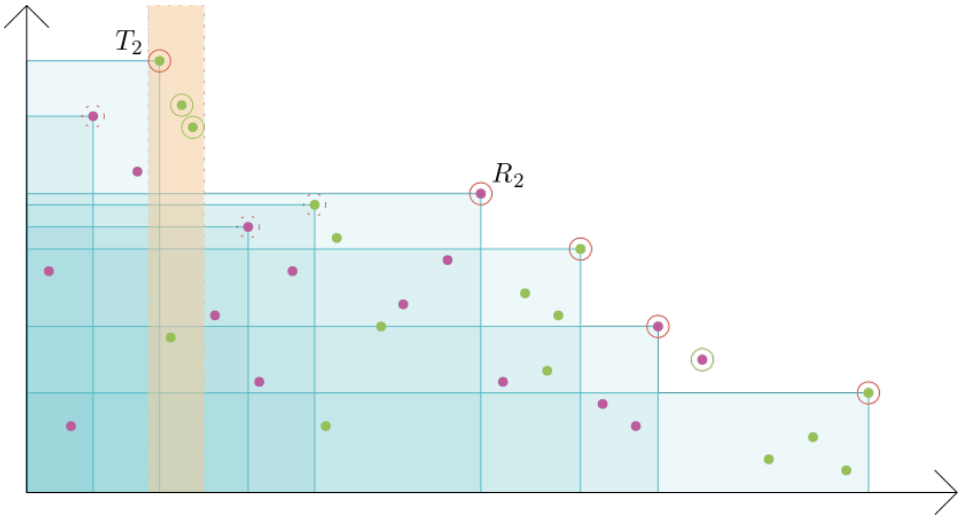


Figure 2 An example instance of maxima set where $n = 32$ and $h = 8$ solved using our quantum combine-and-conquer algorithm. Consecutive blocks of points are colored in alternating purple and green, and the set T of tallest points in each group is circled. The maxima set of T is indicated using solid circles, whereas tallest points that are dominated are circled in dotted lines. To illustrate an example of the conquer step, we focus attention on the group S_2 which is highlighted in orange. In this group, the tallest y coordinate is indicated by T_2, and R_2 is the y coordinate of the tallest point to the right of this group. Thus, the only points that need to be processed are the two points not bound by any blue box. The points that are found in this local check are circled in green.

block j until the Boolean function does not mark any elements in the set. We illustrate the first iteration of this process in Figure 3. Since the output size is h, it suffices to run this process $O(h)$ times in total across all blocks. Finally, we collect the outputs in the blocks that contain a point in M_T, which takes at most $O(h)$ time. See Algorithm 2.

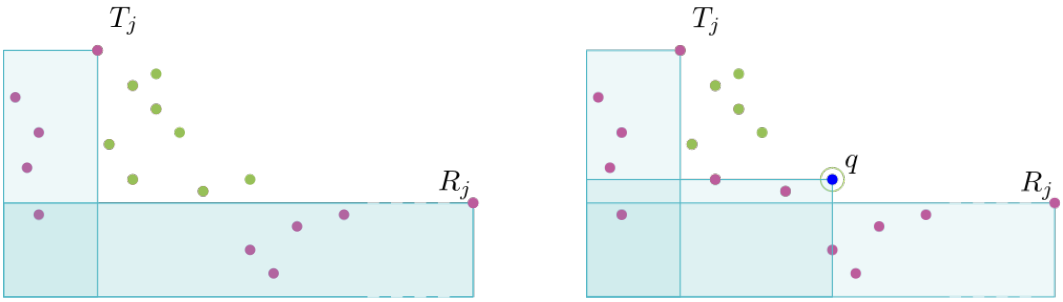


Figure 3 A demonstration of Algorithm 2 where we iteratively search for the maxima point within block j given T_j and R_j. The left shows the state of the first iteration, and all points shown are the contents of block j except for the one labeled R_j. Points are green if they are not dominated by T_j or R_j and thus return 1 to the Boolean function f. Among these points, we search for the lexicographic maximum point q and set this to be the new right boundary for the next iteration. The first q to be discovered in the above instance is marked in blue. This is repeated until there are no remaining green points.

Finally, the case where h is unknown can be handled using an approach where we start with a small estimate of h, then repeat the procedure using $h = 2h$ if we discover more than h points. Thus, the running time forms a geometric sum that is $\tilde{O}(\sqrt{nh})$. See Algorithm 3.

Algorithm 2 `CompleteMaximaSet`$(j,\ h,\ T_j,\ M_j)$.

Require: j, h: The block we are conquering and the total number of blocks.
Require: T_j: The y coordinate of the tallest point in S_j.
Require: R_j: The y coordinate of the first point in M appearing after all points in S_j.
1: Define a Boolean function f that takes as input a point p and returns 1 if p is not dominated by T_j or R_j.
2: **while** `qMax`$(f, \preceq_{Lex}, j, h)$ returns a point q **do**
3: Store q in an output list.
4: Define a new f that takes as input a point p and returns 1 if p is not dominated by T_j or q.

Algorithm 3 `QuantumMaximaSet`(S).

Require: U_{in}: a the unitary that encodes the lexicographically sorted data $[d_0, \ldots, d_{n-1}]$
Require: h: the number of blocks to partition into.
1: $h = 4$.
2: **while** True **do**
3: **for** $j \in [0, 1, \ldots, h-1]$
4: $T_j =$ `qMax`$(f, \preceq_y, j, h)$ where $f(x) = 1$ for all $x \in S$
5: Let $T = [T_0, T_1, \ldots, T_{h-1}]$.
6: $M =$ `ClassicalMaximaSet(`T`)`
7: **for** $j \in [0, 1, \ldots, h-1]$
8: `CompleteMaximaSet(`$j,\ h,\ T_j,\ M_j$`)`
9: **if** The total number of points found exceeds h **then**
10: $h = 2h$
11: **else**
12: Print all the points in the output register of all j blocks and terminate the loop.

▶ **Theorem 2.** *There is a quantum algorithm for finding the maxima set of a presorted set of n points in the plane in $\tilde{O}(\sqrt{nh})$ time, where h is the size of the output.*

Proof. The Grover searches for the tallest points in each block takes $\tilde{O}(h\sqrt{n/h}) = \tilde{O}(\sqrt{nh})$ time. Once this set T is found, the maxima set M of T can be computed classically in $O(h)$ time. The total time incurred by the calls to `CompleteMaximaSet` is dominated by the calls to `qMax`, each of which outputs a point or terminates the call to `CompleteMaximaSet`. Therefore `qMax` is called $O(h)$ times and each call requires $\tilde{O}(\sqrt{n/h})$ time for a total of $\tilde{O}(\sqrt{nh})$ time. Note that the condition that the points we are searching over are between T_j and R_j can be done in constant time. The outer loop repeats at most $O(\log n)$ times, whose contribution is suppressed by $\tilde{O}$. Finally, the time to output the points that were found takes $O(h)$ time and is dominated by the other parts of the algorithm. ◀

4 Quantum Convex Hulls

In this section, we describe a quantum algorithm to construct the convex hull of n points in the plane in $\tilde{O}(\sqrt{nh})$ time, where h is the size of the output. Again, we assume that the input set is sorted in lexicographic order. We follow our combine-and-conquer paradigm, where we use some global properties that can be computed from our subsets to prune off points that do not appear in the output, then use these values to isolate the problem solving within each block. Here, however, the combine step is more complicated that simply finding a point, $p_{\max}$, with maximum y-coordinate.

We begin by reviewing a well-known classical divide-and-conquer algorithm for 2D convex hulls (see, e.g., [11,29,30,33]). The convex hull $CH(S)$ of a set, S, of n points in the plane can be described as a union of the polygonal chains that form the upper hull $UH(S)$ and lower hull $LH(S)$ of the points, where $UH(S)$ (resp., $LH(S)$) is the set of edges of $CH(S)$ with positive (negative) normals. For completeness, we provide a description of a classical divide-and-conquer algorithm for computing $UH(S)$ in appendix D, which divides S into a left set, S_1, and right set, S_2, and recursively finds $UH(S_1)$ and $UH(S_2)$. Then it finds the tanget segment bridge between $UH(S_1)$ and $UH(S_2)$ and prunes away the points under the bridge, resulting in an algorithm running in $O(n \log n)$ time. By a well-known point-line duality, which we also review in an appendix, the bridge edge can be found by a solving a two-dimensional linear program; see, e.g., [11,29,33].

▶ **Lemma 3** (see, e.g., [11, 29, 33]). *Given a vertical line L, and a set S of n points in a plane, one can determine the edge e of $UH(S)$ that intersects L (or that no such edge exists) by finding the highest point of intersection between $O(n)$ halfplanes.*

If we use linear-time 2D linear programming for the bridge-finding step, then the running time for this classical divide-and-conquer algorithm is $O(n \log n)$, even if the points are sorted by x-coordinate. Kirkpatrick and Seidel [22] show that this classical running time can be improved to $O(n \log h)$ by performing the bridge-finding step before the (recursive) conquer steps. Likewise, our quantum convex hull algorithm also performs this combine step before the conquer steps, but avoids recursion.

Before we give our algorithm, let us review another well-known classical algorithm for computing a two-dimensional convex hull – the **Jarvis march** algorithm; see, e.g., [11,30,33]. In this algorithm, we start with a point, p, known to be on the convex hull, such as a point with smallest y-coordinate. Then, we find the next point on the convex hull in a clockwise order, by a simple linear-time maximum-finding step. We iterate this search, winding around the convex hull, until we return to the starting point. Since each iteration takes $O(n)$ time, this algorithm runs in $O(nh)$ time. Lanzagorta and Uhlmann [24] describe a quantum assisted version of the Jarvis march algorithm that runs in $\tilde{O}(h\sqrt{n})$ time, which we review in an appendix.

Let S be a set of n points in the plane sorted lexicographically. For simplicity of expression, let us assume that the points have distinct x-coordinates and we are interested in computing the upper hull, $UH(S)$, of S. As in our maxima set algorithm, let us assume for now that we know the value of h, the number of points on the upper hull of S to simplify the presentation. We divide S into h blocks, $S_0, S_1, \ldots, S_{h-1}$, of size $O(n/h)$ each by vertical lines, $L_1, L_2, \ldots, L_{h-1}$, arranged left-to-right. This is achieved by using Algorithm 1 to partition the input.

At a high level, our quantum combine-and-conquer upper hull algorithm first computes all the bridge edges that are intersected by one of the vertical lines, L_i. This amounts to the combine step of our quantum combine-and-conquer algorithm. Each of these bridge edges belong to the upper hull of S; hence, there are at most h such bridges. Importantly, we show how to compute all these bridges in $\tilde{O}(\sqrt{nh})$ time. Each block of points, S_j, has $O(n/h)$ points, and, of course, there may be remaining upper hull points to compute for each such S_j (between points of tangency for the bridges computed in the combine step). Note that the conquer step for block S_j will return in $O(1)$ time if S_j does not contain any endpoints of a bridge edge or if it contains a single point where two bridge edges meet. See Figure 5 showing the edges that come from the combine step and the edges that come from the conquer step.

To compute the remaining convex hull points, then, we perform a modified quantum Jarvis march for each subset, S_j, which runs in time $\tilde{O}(h_j\sqrt{n/h})$ time, where h_j is the number of additional upper hull points found for S_j. Thus, the total time for this second (conquer) phase of our algorithm is

$$\begin{aligned}\sum_{j=1}^{h} \tilde{O}(h_j\sqrt{n/h}) &\leq \tilde{O}(h\sqrt{n/h}) \\ &= \tilde{O}(\sqrt{nh}).\end{aligned}$$

Given this overview, let us next describe how to perform the different steps of our algorithm, beginning with our quantum algorithm for finding all the bridge edges intersecting the vertical lines, $L_1, L_2, \ldots, L_{h-1}$, which separate the subsets, $S_0, S_1, \ldots, S_{h-1}$, of S. At a high level, our combine-step method can be viewed as a quantum adaptation of a classical algorithm by Goodrich [15] for computing the upper hull of a partially sorted set of points.

As mentioned above, by point-line duality, bridge finding is dual to two-dimensional linear programming. In more detail, let S be a set of n points in the plane, and L be a vertical line in this plane. Let S' be the same set of points after shifting the plane horizontally so that L is the y-axis. We define a dual plane by taking each point $p = (a, b)$ and mapping it to a line $y = ax - b$. A standard result in computational geometry is that the points in the upper hull of S' correspond to the lower envelope of the set of lines in the dual plane; see, e.g., [11, 30, 33]. Furthermore, by the duality transformation, the highest point in the lower envelope is the intersection of two lines in the envelope whose slopes transition from positive to negative. Mapping this point to the primal plane gives us the bridge edge we are looking for. Thus, we can reduce the problem of finding a bridge edge to the problem of finding the largest point in the intersection of n lower-halfplanes. We also use the following lemma due to Sadakane et al. showing that the problem of computing the highest point in the intersection of n halfplanes efficiently using a quantum algorithm.

▶ **Lemma 4** (Sadakane et al. [32]). *The highest point in the intersection of n lower-halfplanes can be computed in $O(\sqrt{n}\log^2 n)$ time, using a quantum computer, with probability $1 - n^{-c}$ for any constant c given the linear constraints in superposition.*

The linear constraints in superposition refers to a state in the following form:

$$|\psi\rangle := \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle\,|p_i\rangle \tag{5}$$

where p_i are the coordinates of each of the n points. By point-line duality, the coordinates fully describe the halfplanes in the dual plane.

The output element is a point in the dual plane, which maps back to a line in the primal plane. To determine which points this line intersects, we can run two iterations of minimum finding to find the points in S that are on the resulting line by minimizing by the distance to the line. We combine the above two lemmas to a function with the signature,

$$\texttt{qLP}(L,\ i,\ j,\ h), \tag{6}$$

and it returns the endpoints $p_s, p_f \in S$ of the bridge edge as a tuple. This process will also call `qPrep`(i,h) as needed.

We are now ready to describe the full algorithm. The first step will be to divide our input set into h blocks, each containing n/h points. We define a **bridge edge** to be an edge in the upper hull of S whose endpoints are in two different blocks.

Algorithm 4 Bridge(i, j, h).

Require: i, j: The indices of blocks to find the bridge edge over.
Require: h: The total number of blocks that we are partitioning into.
1: Let L be a vertical line drawn between the last point of S_i and the first point of S_j. This can be computed by querying U_{in} at the final and first index of blocks i and j respectively.
2: (p_s, p_f) =qLP(L, i, j, h)
3: Return the two endpoints $p_s \in S_i$ and $p_f \in S_j$ of the bridge edge.

To find the bridge edges, we take inspiration from the classical Graham scan algorithm for computing the convex hull. We will be using the fact that if we traverse the convex hull in the clockwise direction, each edge makes a right turn from its previous edge. Since bridge edges are edges in the upper hull, this is true for them as well. We will compute bridge edges between consecutive pairs, but if at any point a left turn is formed between two bridge edges, we know that these edges will not be a part of the upper hull. Critically, this allows us to ignore an entire block of points, and move on to find the bridge edge between the remaining blocks. The details of the above approach are outlined in Algorithm 5.

Algorithm 5 FindBridgeEdges(h).

1: Initialize an empty stack Σ.
2: Let L_1 :=Bridge$(0, 1, h)$, and push this onto Σ. ▷ L_i will represent the bridge edge that ends in block i.
3: **for** $i \in [2, h-1]$ **do**
4: Let L_i := Bridge$(i-1, i, h)$.
5: **while** L_{i-1} and L_i form a left turn **do**
6: Pop the stack
7: Let t be the index of the item on top of the stack.
8: Let L_i := Bridge(t, i, h).
9: Push L_i on the stack
10: Return Σ as an array.

▶ **Lemma 5.** *Let S be a set of points divided into h blocks such that $p_i \leq p_j$ for all $p_i \in S_i$ and $p_j \in S_j$ where $i \leq j$. Then, the set of bridge edges of this partition can be found in $\tilde{O}(\sqrt{nh})$ time.*

Proof. Each block's pending contribution to the bridge edges can only be popped at most once. Thus, the total time it takes for the loop starting at line 3 of Algorithm 5 will be proportional to $O(h)$. For each edge, we must compute the Bridge between sets S_j and S_i, which will take $\tilde{O}(\sqrt{n/h})$ time by lemma 3 and 4. Thus, the total time it takes to find the bridge edges is $\tilde{O}(\sqrt{nh})$. ◀

Once the bridge edges are identified, what remains is to find the edges of the convex hull whose end points are in the same block. To do this, we use a quantum assisted Jarvis march. Recall that the Jarvis march is another standard convex hull algorithm in which we start with an edge (p, q) on the convex hull, then do a search over the remaining points to find the point r such that the angle formed between (p, q) and (q, r) is maximized.

In each block, S_i, rather than performing a full Jarvis march, however, we start the search from the bridge edge entering S_i from the left, and perform up to h Grover searches for the point maximizing the angle until we find the starting point of the bridge edge leaving S_i. We summarize this step in Algorithm 6 and Lemma 6. See also Figure 4.

Algorithm 6 QuantumBlockJarvisMarch(j, h, B).

Require: B is the set of bridges of S computed in Algorithm 5.

Define f_c to be a function such that $f_c(p) = 1$ iff there is a bridge in B that starts at p.
Define f_f to be a function such that $f_f(p) = 1$ iff there is a bridge in B that ends at p.
$p_c = \texttt{qMax}(f_c, \preceq_{Lex}, j, h)$
$p_f = \texttt{qMax}(f_f, \preceq_{Lex}, j, h)$
If neither are present, Return.
Let p_s, p_c be the two end points of the bridge edge containing p_c.
while $p_c \neq p_f$ **do**
    Define $\preceq_{deg(p_s,p_c)}$ to be the ordering induced by the angle formed between (p_s, p_c) and (p_c, p) for $p \in S$.
    $p_r = \texttt{qMax}(f, \preceq_{deg(p_s,p_c)}, j, h)$ where $f(x) = 1$ for all x.
    Add p_r to the set of output points associated with block j.
    Let $(p_s, p_c) = (p_c, p_r)$.

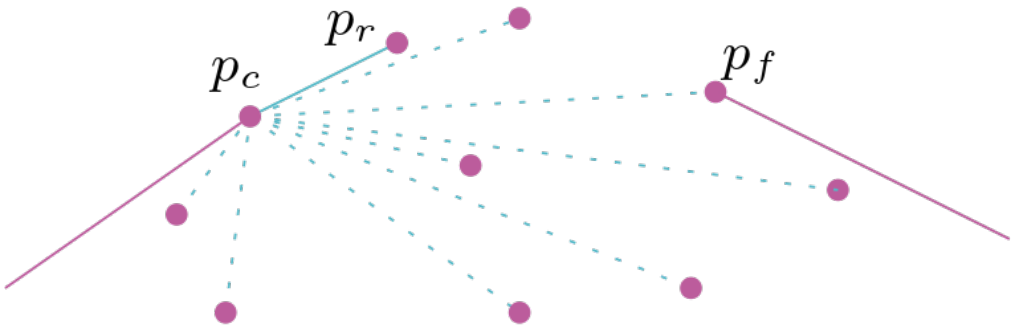


Figure 4 An example of our restricted Jarvis March convex hull algorithm. The purple lines are edges that we know are on the convex hull, and the points are the content of some block S_j. In a Jarvis March algorithm, we start our search from p_c which is known to be on the convex hull, then search for the point that forms the maximum angle with the incoming edge containing p_c. In the above example, the edges we check are in blue, and the solid line denotes the edge that is added to the convex hull. We repeat the search again starting at p_r, until we connect to p_f.

▶ **Lemma 6.** *Let S be a set of m points and p_s, $p_f \in S$ be points on the convex hull of S. The part of the convex hull from p_s to p_f in the clockwise direction can be computed in $\tilde{O}(h\sqrt{m})$ time, where h is the number of points on the convex hull from p_s to p_f.*

We now combine the above pieces to describe our complete quantum convex hull algorithm. See Figure 5. As was the case in the maxima set algorithm, we begin with a guess for the upper bound of h, then during the conquer step detect whether or not we have exceeded this bound. A convex hull requires at least 3 points, so we use 4 as our initial guess, as it is the smallest power of 2 greater than 3. If we exceed our guess, then we double our estimate for h and rerun the algorithm. Thus, the running times across all our guesses form a geometric sum that is $\tilde{O}(\sqrt{nh})$. See Algorithm 7.

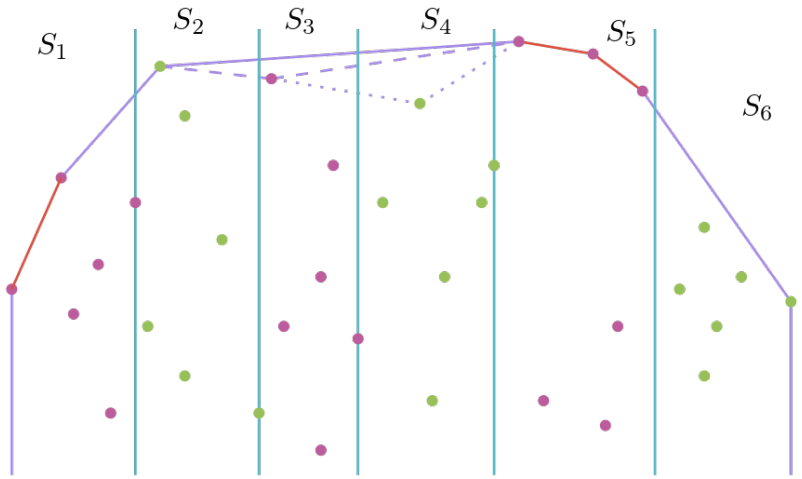


Figure 5 An upper hull with $n = 36$ and $h = 6$. Consecutive blocks of points are colored in alternating purple and green. The bridge edges discovered are shown in purple, and the dotted and dashed lines starting from block S_2 indicate how the algorithm handles left turns. The bridge edge between S_4 and S_5 forms a left turn relative to the previous dotted bridge edge, so both are popped and a new bridge edge is found between S_3 and S_5. This is repeated until no left turns are formed, giving the solid purple line. Finally, the bridge edges do not close the hull in some blocks, which is where we run the quantum Jarvis march to find the edges of the upper hull shown in red.

Algorithm 7 QuantumConvexHull(U_{in}).

Require: U_{in}: A unitary that encodes the lexicographically sorted data $[d_0, \ldots, d_{n-1}]$

Let $h = 4$.
while True **do**
  $B =$ `FindBridgeEdges`(h)
  for $j \in [0, 1, \ldots, h-1]$ **do**
    `QuantumBlockJarvisMarch`(j, h, B)
  if the total number of points found exceeds h **then**
    $h = 2h$
  else
    Print all the points in the output register of each block and terminate the loop.

▶ **Theorem 7.** *There is a quantum algorithm for finding the convex hull of a set of n points in lexicographic order that runs in $\tilde{O}(\sqrt{nh})$ time, where h is the size of the output.*

Proof. As observed, the outer loop will run at most $\log n$ times. By lemma 5, `FindBridgeEdges` takes $\tilde{O}(\sqrt{nh})$ time. By lemma 6, `QuantumJarvisMarch` takes $\tilde{O}(h_j\sqrt{n/h})$ time for block S_j, where h_j is the number of points from S_j that lie on the convex hull of the entire points set S. Since $\sum_j h_j \leq h$, the total running time for all the calls to `QuantumJarvisMarch` is $\tilde{O}(\sqrt{nh})$. Finally, printing the h discovered points takes $O(h)$ time. ◀

In an appendix, we give a lower bound for quantum convex hull that shows our algorithm is optimal up to polylogarithmic factors.

5 Discussion

Afshani, Peyman, and Chan [1] give an instance-optimal classical algorithm for finding 2D convex hulls, which adapts the algorithm by Kirkpatrick and Seidel [22] to run in $O(n(\mathcal{H}(S)+1))$ time, where $\mathcal{H}(S)$ is a classical-computing structural entropy measure that is never more than $O(\log h)$. An interesting direction for future work would be to determine if there is a quantum convex hull algorithm that is instance optimal based on a quantum structural entropy measure.

References

1 Peyman Afshani, Jérémy Barbay, and Timothy M Chan. Instance-optimal geometric algorithms. *Journal of the ACM*, 64(1):1–38, 2017. doi:10.1145/3046673.

2 Shyan Akmal and Ce Jin. Near-optimal quantum algorithms for string problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2791–2832, 2022. doi:10.1137/1.9781611977073.109.

3 Jonathan Allcock, Jinge Bao, Aleksandrs Belovs, Troy Lee, and Miklos Santha. On the quantum time complexity of divide and conquer, 2023. doi:10.48550/arXiv.2311.16401.

4 Israel F Araujo, Daniel K Park, Francesco Petruccione, and Adenilton J da Silva. A divide-and-conquer algorithm for quantum state preparation. *Scientific Reports*, 11(1):6329, 2021.

5 Omer Berkman, Baruch Schieber, and Uzi Vishkin. A fast parallel algorithm for finding the convex hull of a sorted point set. *International Journal of Computational Geometry & Applications*, 06(02):231–241, 1996. doi:10.1142/S0218195996000162.

6 Ibrahim Cameron, Teague Tomesh, Zain Saleem, and Ilya Safro. Scaling up the quantum divide and conquer algorithm for combinatorial optimization, 2024. arXiv:2405.00861.

7 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.

8 Daniel Tzu Shiuan Chen, Zain Hamid Saleem, and Michael Alexandrovich Perlin. Quantum circuit cutting for classical shadows. *ACM Transactions on Quantum Computing*, 5(2):1–21, 2024. doi:10.1145/3665335.

9 Andrew M. Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. Quantum divide and conquer, 2022. doi:10.48550/arXiv.2210.06419.

10 John A. Cortese and Timothy M. Braje. Loading Classical Data into a Quantum Computer, March 2018. arXiv:1803.01958.

11 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

12 Christoph Durr and Peter Hoyer. A Quantum Algorithm for Finding the Minimum, January 1999. arXiv:quant-ph/9607014.

13 Mujtaba R Ghouse and Michael T Goodrich. In-place techniques for parallel convex hull algorithms. In *3rd ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 192–203, 1991.

14 Li-Hua Gong, Wei Ding, Zi Li, Yuan-Zhi Wang, and Nan-Run Zhou. Quantum k-nearest neighbor classification algorithm via a divide-and-conquer strategy. *Advanced Quantum Technologies*, 7(6):2300221, 2024.

15 Michael T. Goodrich. Constructing the convex hull of a partially sorted set of points. *Computational Geometry*, 2(5):267–278, 1993. doi:10.1016/0925-7721(93)90023-Y.

16 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996. doi:10.1145/237814.237866.

17 Sariel Har-Peled. On the Expected Complexity of Random Convex Hulls. *arXiv e-prints*, page arXiv:1111.5340, November 2011. doi:10.48550/arXiv.1111.5340.

18 T. Hayashi, K. Nakano, and S. Olarlu. An $O((\log \log n)^2)$ time algorithm to compute the convex hull of sorted points on reconfigurable meshes. *IEEE Transactions on Parallel and Distributed Systems*, 9(12):1167–1179, 1998. doi:10.1109/71.737694.

19 Peter HÃÿyer, Jan Neerbek, and Yaoyun Shi. *Quantum Complexities of Ordered Searching, Sorting, and Element Distinctness*, pages 346–357. Springer Berlin Heidelberg, 2001. doi:10.1007/3-540-48224-5_29.

20 David G. Kirkpatrick and Seidel Raimund. Marriage before conquest: A variation on the divide and conquer paradigm. Technical Report 83-7, University of British Columbia, 1983. URL: https://www.cs.ubc.ca/sites/default/files/tr/1983/TR-83-07.pdf.

21 David G. Kirkpatrick and Raimund Seidel. Output-size sensitive algorithms for finding maximal vectors. In *1st Symposium on Computational Geometry (SoCG)*, pages 89–96, 1985. doi:10.1145/323233.323246.

22 David G. Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM Journal on Computing*, 15(1):287–299, 1986. doi:10.1137/0215021.

23 Marco Lanzagorta and Jeffrey Uhlmann. Quantum algorithmic methods for computational geometry. *Mathematical Structures in Computer Science*, 20(6):1117–1125, 2010. doi:10.1017/S0960129510000411.

24 Marco Lanzagorta and Jeffrey K. Uhlmann. Quantum computational geometry. In *Quantum Information and Computation II*, volume 5436, pages 332–339. SPIE, 2004.

25 Junde Li, Mahabubul Alam, and Swaroop Ghosh. Large-scale quantum approximate optimization via divide-and-conquer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6):1852–1860, 2023. doi:10.1109/TCAD.2022.3212196.

26 Masaya Nakagawa, Duhu Man, Yasuaki Ito, and Koji Nakano. A simple parallel convex hulls algorithm for sorted points and the performance evaluation on the multicore processors. In *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 506–511, 2009. doi:10.1109/PDCAT.2009.56.

27 Koji Nakano. Computation of the convex hull for sorted points on a reconfigurable mesh. *Parallel Algorithms and Applications*, 8(3-4):243–250, 1996. doi:10.1080/10637199608915555.

28 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010.

29 Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

30 Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer, 2012.

31 H. Raynaud. Sur lâĂŹenveloppe convexe des nuages de points aleatoires dans $\mathbb{R}^n$. i. *Journal of Applied Probability*, 7(1):35–48, 1970. doi:10.2307/3212146.

32 Kunihiko Sadakane, Norito Sugawara, and Takeshi Tokuyama. Quantum computation in computational geometry. *Interdisciplinary Information Sciences*, 8(2):129–136, 2002.

33 Raimund Seidel. Linear programming and convex hulls made easy. In *6th Symposium on Computational Geometry (SoCG)*, pages 211–215, 1990. doi:10.1145/98524.98570.

34 Raimund Seidel. Convex hull computations. In *Handbook of Discrete and Computational Geometry*, pages 687–703. Chapman and Hall/CRC, 2017.

35 Teague Tomesh, Zain H. Saleem, Michael A. Perlin, Pranav Gokhale, Martin Suchara, and Margaret Martonosi. Divide and conquer for combinatorial optimization and distributed quantum computation. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 01, pages 1–12, 2023. doi:10.1109/QCE57702.2023.00009.

36 Cheng Wang and Ri-Gui Zhou. A quantum search algorithm of two-dimensional convex hull. *Communications in Theoretical Physics*, 73(11):115102, September 2021. doi:10.1088/1572-9494/ac1da0.

37 Qisheng Wang. A note on quantum divide and conquer for minimal string rotation, 2022. doi:10.48550/arXiv.2210.09149.

Uniform Bounds on Product Sylvester-Gallai Configurations

Abhibhav Garg
University of Waterloo, Canada

Rafael Oliveira
University of Waterloo, Canada

Akash Kumar Sengupta
University of Waterloo, Canada

Abstract

In this work, we explore a non-linear extension of the classical Sylvester-Gallai configuration. Let $\mathbb{K}$ be an algebraically closed field of characteristic zero, and let $\mathcal{F} = \{F_1, \ldots, F_m\} \subset \mathbb{K}[x_1, \ldots, x_N]$ denote a collection of irreducible homogeneous polynomials of degree at most d, where each F_i is not a scalar multiple of any other F_j for $i \neq j$. We define $\mathcal{F}$ to be a product Sylvester-Gallai configuration if, for any two distinct polynomials $F_i, F_j \in \mathcal{F}$, the following condition is satisfied:

$$\prod_{k \neq i,j} F_k \in \mathrm{rad}\,(F_i, F_j)\,.$$

We prove that product Sylvester-Gallai configurations are inherently low dimensional. Specifically, we show that there exists a function $\lambda : \mathbb{N} \to \mathbb{N}$, independent of $\mathbb{K}$, N, and m, such that any product Sylvester-Gallai configuration must satisfy:

$$\dim(\mathrm{span}_{\mathbb{K}}(\mathcal{F})) \leq \lambda(d).$$

This result generalizes the main theorems from (Shpilka 2019, Peleg and Shpilka 2020, Oliveira and Sengupta 2023), and gets us one step closer to a full derandomization of the polynomial identity testing problem for the class of depth 4 circuits with bounded top and bottom fan-in.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Computational geometry

Keywords and phrases Sylvester-Gallai theorem, arrangements of hypersurfaces, algebraic complexity, polynomial identity testing, algebraic geometry, commutative algebra

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.52

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2025/037`

Acknowledgements The authors would like to thank Amir Shpilka and Shir Peleg for several helpful conversations in the course of this work.

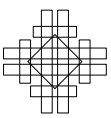
1 Introduction

In 1893, Sylvester posed the following basic problem in extremal combinatorial geometry.

▶ **Problem 1** ([49]). *Suppose a finite set of real points are arranged such that any line through two points of the set contains a third point in the set. Prove that the set of points lies on a straight line.*

The first complete affirmative answer to Problem 1 was given by Melchior [35]. The same problem was independently posed in 1943 by Erdős [17], and was answered by Gallai [20]. The above statement is now known as the Sylvester-Gallai theorem.

© Abhibhav Garg, Rafael Oliveira, and Akash Kumar Sengupta;
licensed under Creative Commons License CC-BY 4.0



It is important to note that the answer to Sylvester's problem depends on the base field. For instance, it was known (quite possibly even to Sylvester) that the answer to Problem 1 is negative when the base field is $\mathbb{C}$, as the nine inflection points of any non-singular planar cubic form a 2-dimensional complex configuration satisfying Sylvester's conditions. In 1966, Serre asked in [46] whether any complex configurations satisfying Sylvester's conditions must be 2-dimensional – that is, whether the configurations arising from non-singular planar cubics are extremal. An affirmative answer to Serre's question was given in 1983 by Hirzebruch [29], using highly non-elementary algebro-geometric tools.

Ever since the solution of Sylvester's problem, several generalizations and variations have been studied in combinatorial geometry, theoretical computer science (TCS) and coding theory [15, 27, 42, 16, 7, 13, 39, 40] – see [9, 12] for brief surveys on these generalizations. The usual setup of such generalizations is as follows: given a *finite* collection of geometric objects (points, in the case of Sylvester's problem) satisfying *enough* **local conditions** (collinearity of certain triples of points, in the case of Problem 1), one wants to know if such collection of geometric objects must be "*low dimensional*" (all points must be in one line, in the case of Problem 1). As is usual in the literature, any configuration satisfying the proposed local conditions are called *Sylvester-Gallai configurations*, and the result stating that such configurations are low dimensional is referred to as a *Sylvester-Gallai* type theorem.

One line of generalizations of Problem 1 arises from projective duality, which we now discuss. By projective duality, any point $P = (p_1, \ldots, p_N)$ gives rise to a dual hyperplane, defined by the zero set of the linear form $\ell_P := p_1 x_1 + \cdots + p_N x_N$, which we denote by $V(\ell_P)$. Given three points P, Q, R, the condition that R is in the line defined by P, Q is equivalent to $V(\ell_P, \ell_Q) \subset V(\ell_R)$ in the dual space. Lastly, given points $P_1, \ldots, P_m$, the condition that they are collinear is equivalent to $\dim \operatorname{span}_{\mathbb{R}} \{\ell_{P_1}, \ldots, \ell_{P_m}\} = 2$. Thus, we can recast Sylvester's problem as follows: given a finite set of distinct hyperplanes defined by the set of linear forms $\mathcal{L} := \{\ell_1, \cdots, \ell_m\} \subset \mathbb{R}[x_1, \ldots, x_N]$ such that for any $i \neq j \in [m]$, there is $k \neq i, j$ such that $V(\ell_i, \ell_j) \subset V(\ell_k)$, then it must be the case that $\dim \operatorname{span}_{\mathbb{R}} \{\mathcal{L}\} = 2$. For simplicity, if one applies projective duality in two dimensions, the foregoing statement becomes quite natural: given a finite set of distinct lines (the set $\mathcal{L}$) such that for any two lines $\ell_i, \ell_j \in \mathcal{L}$, there must be a third line in $\mathcal{L}$ which passes through the intersection of ℓ_1, ℓ_2 (i.e., $V(\ell_1, \ell_2)$), then it must be the case that the set of lines forms a pencil (i.e., all lines intersect at a common point). In fact, this dual formulation was precisely the setting treated in [29].

Motivated by questions in algebraic complexity theory, Gupta [24] proposed non-linear, algebro-geometric generalizations to the above (dual) formulation of Sylvester's problem (and their set variants, such as [15]). In this work, we make progress on Gupta's program and fully resolve one such generalization of Sylvester's problem. Before we describe our main result, in the following subsection we describe the connection between the Sylvester-Gallai configurations that we study and the Polynomial Identity Testing (PIT) problem, a fundamental problem in algebraic complexity theory.

1.1 Polynomial Identity Testing (PIT) and Sylvester-Gallai configurations

The Polynomial Identity Testing (PIT) problem is the task of determining whether a given algebraic circuit computes the zero polynomial. While there are simple and efficient randomized algorithms for the PIT problem, developing an efficient deterministic solution remains a significant open problem in theoretical computer science. The PIT problem is

closely tied to fundamental topics such as lower bounds for algebraic circuits [28, 2, 30] and derandomizing key problems in mathematics and computer science [3, 19, 33, 36, 26, 18]. For a deeper exploration of PIT and its applications, see [43, 48, 44].

The recent works [1, 25, 50] have shown that the PIT problem for general circuits can be reduced to solving it for low-depth circuits, such as unrestricted depth-3 or homogeneous depth-4 circuits, which has made these circuit classes a focus of recent study. Notably, progress has been made towards deterministic, polynomial-time PIT for depth-3 circuits of constant top fanin via the connection between this problem and linear Sylvester-Gallai configurations arising from depth-3 identities [14, 32, 45].

Let us now explain how Sylvester-Gallai configurations arise in the context of depth-3 and depth-4 PIT. Let $\mathbf{x} := (x_1, \ldots, x_N)$ be a tuple of N variables and $S := \mathbb{C}[\mathbf{x}]$ be the polynomial ring in N variables. Suppose we are given a depth three circuit with top fan-in three (denoted $\Sigma^3\Pi\Sigma$) that computes a polynomial P. Such a polynomial has the form

$$P = \prod_{i=1}^{m} \ell_i(\mathbf{x}) + \prod_{j=1}^{m} g_j(\mathbf{x}) + \prod_{k=1}^{m} h_k(\mathbf{x}), \tag{1}$$

where ℓ_i, g_j, h_k are linear forms. When $P \equiv 0$, that is, when we have an identity, and moreover this representation of the identity is "efficient,"[1] we may wonder whether the linear forms in the identity must be "low dimensional" (i.e., all such linear forms must "depend on few variables"). To capture the dimensionality of the circuit given by Equation (1), [14] defined the *rank* of the circuit as being the linear span of the forms that appear in the circuit. In case of Equation (1), we have that the rank is $\dim \operatorname{span}_{\mathbb{C}} \{\ell_i, g_j, h_k\}_{i,j,k\in[m]}$.

Consider a linear form ℓ_i from the first gate and a second linear form g_j from the second gate. Since $P \equiv 0$, we have that for any $\mathbf{a} \in \mathbb{C}^N$ such that $\ell_i(\mathbf{a}) = g_j(\mathbf{a}) = 0$, it must be the case that $\prod h_k(\mathbf{a}) = 0$. In other words, we have that $V(\ell_i, g_j) \subset V\left(\prod h_k\right) = \bigcup_k V(h_k)$. Since the algebraic set $V(\ell_i, g_j)$ is irreducible, we must have $V(\ell_i, g_j) \subset V(h_a)$ for some $a \in [m]$. Note that this last condition (combined with the symmetry among the gates) is exactly the local constraint arising from the dual formulation of Sylvester's problem. In [14], the authors asked whether sets $\{\ell_i\}_{i\in[m]}, \{g_j\}_{j\in[m]}, \{h_k\}_{k\in[m]}$ arising from Equation (1) which satisfy the above conditions must be low-dimensional – more precisely, whether $\dim \operatorname{span}_{\mathbb{C}} \{\ell_i, g_j, h_k\}_{i,j,k} = O(1)$.

As it turns out, the Sylvester-Gallai like condition in the previous paragraph can be seen as a set-version of Sylvester's problem, which was studied by [15]. This work showed that such configurations must have constant rank. Thus, we deduce that $\operatorname{span}_{\mathbb{C}} \{\ell_i, g_j, h_k\} = O(1)$. This shows that any $\Sigma^3\Pi\Sigma$ identity essentially depends on constantly many variables. Combined with the results of [31], this gives a black box deterministic PIT algorithm for such circuits. While significantly extra complexity and subtleties arise when one works with $\Sigma^k\Pi\Sigma$ circuits for $k > 3$, the linear forms in "efficient" identities arising from such circuits satisfy enough local relationships to deduce that they have low rank. This approach was carried out by [32, 45], who generalized the above approach and showed an intrinsic and elegant connection between $\Sigma^k\Pi\Sigma$ identities and Sylvester-Gallai configurations (see [45, Theorem 1.4]). As a result of this connection, Sylvester-Gallai theorems imply deterministic, polynomial-time PIT algorithms for $\Sigma^k\Pi\Sigma$ circuits, when $k = O(1)$.

Motivated by these results, Gupta [24] studied depth four circuits with constant top and bottom fan-ins. This circuit family is denoted $\Sigma^k\Pi\Sigma\Pi^d$, and it computes polynomials which can be written as a sum of k products of polynomials of degree at most d.

[1] The formal definition of efficient is simple and minimal, as defined in [32].

Consider a polynomial Q computed by a $\Sigma^3\Pi\Sigma\Pi^d$ circuit. It has the form

$$Q = \prod_{i=1}^{m_1} A_i(\mathbf{x}) + \prod_{j=1}^{m_2} B_j(\mathbf{x}) + \prod_{k=1}^{m_3} C_k(\mathbf{x}),$$

where A_i, B_j, C_k are polynomials of degree at most d. If $Q \equiv 0$ and the representation is efficient, as in the previous case, we have $V(A_i, B_j) \subseteq V\left(\prod C_k\right)$.[2] However, as the forms are not necessarily linear, we have that $V\left(A_i, B_j\right)$ is not necessarily irreducible, and thus we no longer have the same Sylvester-Gallai type configuration, as we cannot guarantee the existence of $k \in [m]$ such that $V\left(A_i, B_j\right) \subset V\left(C_k\right)$. Nevertheless, as there are many constraints of the form $V(A_i, B_j) \subseteq V\left(\prod C_k\right)$, [24] asked whether such Sylvester-Gallai type conditions would be enough to prove that $\dim \operatorname{span}_{\mathbb{C}} \{A_i, B_j, C_k\}_{i,j,k}$ is small enough.

Given any "efficient" $\Sigma^3\Pi\Sigma\Pi^d$ circuit (not necessarily computing the zero polynomial), [24] called it a Sylvester-Gallai circuit if the above constraints on the zerosets (i.e. $V(A_i, B_j) \subseteq V\left(\prod C_k\right)$) hold. The above discussion shows that all such circuits that compute 0 are Sylvester-Gallai circuits, but the converse is not true: for instance, multiplying a single form by a constant can change whether or not the circuit computes 0 without changing the above condition.

Non Sylvester-Gallai circuits are relatively simple compared to Sylvester-Gallai circuits. Non Sylvester-Gallai circuits do not have too many cancellations (the above discussion shows that they can never compute 0), and the circuit structure can be preserved by certain linear variable reduction maps, which is the main tool used in the works [24, 23] to handle these circuits. This allows these works to certify the non-zeroness of non-Sylvester-Gallai circuits.

The cancellations in Sylvester-Gallai circuits on the other hand are a lot more subtle, and it is not clear if they are preserved by the linear maps of [24, 23]. Similar to how linear Sylvester-Gallai configurations have constant rank, Gupta conjectured that sets of higher degree forms satisfying the local condition $V(A_i, B_j) \subset V\left(\prod_k C_k\right)$ must "depend on constantly many variables." If this is true, then we can efficiently test if Sylvester-Gallai circuits are nonzero using the methods of [8]. A simpler form of Gupta's main conjecture is the following generalization of the configurations studied in [15]:

▶ **Conjecture 2** (Gupta's main conjecture - simple form). *Let $\{A_i\}_{i\in[m_1]} \cup \{B_j\}_{j\in[m_2]} \cup \{C_k\}_{k\in[m_3]}$ be a set of irreducible polynomials of degree $\leq d$ such that for any $i \in [m_1], j \in [m_2]$, we have that $V\left(A_i, B_j\right) \subseteq V\left(\prod_k C_k\right)$ (and the same relations hold when exchanging the roles of the polynomials A, B, C). Then there exists a function $\lambda : \mathbb{N} \to \mathbb{N}$ such that*

$$\dim \operatorname{span}_{\mathbb{C}} \{A_i, B_j, C_k\}_{i,j,k} \leq \lambda(d).$$

A first step towards establishing the above conjecture is to establish the Sylvester-Gallai version of the above conjecture. This motivates us to consider the Sylvester-Gallai configurations we study in this article, which we now define. By way of the algebraic-geometric dictionary, we will henceforth replace the geometric condition $V\left(A, B\right) \subset V\left(C\right)$ by the equivalent algebraic condition $\operatorname{rad}\left(C\right) \subset \operatorname{rad}\left(A, B\right)$.

▶ **Definition 3** (Product Sylvester–Gallai configurations). *A finite set $\mathcal{F} \subset \mathbb{C}\left[x_1, \ldots, x_N\right]$ of irreducible forms of degree at most d is a* d-product-SG-configuration *if the following hold.*

- *$F_i \notin (F_j)$ for any $F_i, F_j \in \mathcal{F}$.* *(each form encodes a different hypersurface)*
- *For each $F_i, F_j \in \mathcal{F}$, we have* *(Sylvester-Gallai condition)*

$$\prod_{F\in\mathcal{F}\setminus\{F_i,F_j\}} F \;\in\; \operatorname{rad}\left(F_i, F_j\right).$$

[2] By symmetry among the gates, we also have $V(A_i, C_k) \subseteq V\left(\prod B_j\right)$ and $V(B_j, C_k) \subseteq V\left(\prod A_i\right)$.

As stated above, the reader may find it strange that the product constraint no longer seems local, since the condition for each F_i, F_j involves every other form in the configuration. However, by a standard Bézout type argument it follows that the condition $\prod_{F\in\mathcal{F}\setminus\{F_i,F_j\}} F \in \operatorname{rad}(F_i, F_j)$ is equivalent to the (local) condition that there are indices $k_1, \ldots, k_{d^2}$ different from i, j such that

$$\prod_{a=1}^{d^2} F_{k_a} \in \operatorname{rad}(F_i, F_j).$$

Since d is a constant, each (local) condition now only depends on constantly many forms in the configuration as expected. As this equivalent condition is more cumbersome to state, we prefer to work with the above definition.

In a similar fashion to [24], we can also conjecture that such "product" configurations are "low dimensional." This is the content of the following conjecture:

▶ **Conjecture 4** (Product Sylvester-Gallai Conjecture). *There is a function $\lambda : \mathbb{N} \to \mathbb{N}$ such that if $\mathcal{F}$ is a d-*product-SG-configuration *then* $\dim \operatorname{span}_{\mathbb{C}} \{\mathcal{F}\} = \lambda(d)$.

The above conjecture was first considered by [39], where the authors proved Conjecture 4 for $d = 2$. Our main theorem, which we now state, generalizes their main result to forms of every degree, thus fully settling Conjecture 4.

▶ **Theorem 5** (Product Sylvester-Gallai Theorem). *There is a function $\lambda : \mathbb{N} \to \mathbb{N}$ such that if $\mathcal{F}$ is a d-*product-SG-configuration *then* $\dim \operatorname{span}_{\mathbb{C}} \{\mathcal{F}\} = \lambda(d)$.

Our proof techniques build upon the techniques introduced in [47, 38] to study radical Sylvester-Gallai configurations, which we define in Section 1.3. In Section 1.3 we also discuss related and previous works on non-linear Sylvester-Gallai configurations and the PIT problem for depth-4 circuits. In Section 1.4, we give a high level outline of our proof, highlighting the technical difficulties that need to be overcome, as well as the new conceptual contributions of this work. We will now discuss the main contributions of this work, where we state our main technical result, which is of independent interest. Formal statements and proofs of all claims can be found in the full version of this paper.

1.2 Contributions of this paper

This paper has two main contributions. The first is the proof of Theorem 5, which states that product Sylvester-Gallai configurations of degree bounded by d have dimension uniformly upper bounded by a function which depends only on d. This result simultaneously generalizes [47, 39, 38], putting us one step closer towards a proof of Conjecture 2, and also to the main conjecture of Gupta [24]. To achieve this, we adapt the inductive approach from [38] to work with product configurations. This adaptation, as we discuss in Section 1.4, requires us to deal with several subtleties, one of them being to establish a useful property of the general quotients of [47, 38]: we prove that such general quotients behave well with respect to (a generalized notion of) absolute irreducibility. More precisely, we prove that that these quotients send *absolutely reducible* forms to *reducible* elements, and that forms which are *absolutely irreducible* over the vector space being quotiented out will *remain absolutely irreducible* after a general quotient.

The second contribution is our main technical result which relates absolute irreducibility of a polynomial P with an *effective and uniform bound* on the non-primality of certain ideals containing P. In a simplified form, it can be stated as:

▶ **Theorem 6** (Prime bound - informal). *Let* $S := \mathbb{C}[x_1, \ldots, x_N, y_1, \ldots, y_n]$ *and* $P \in S \setminus (x_1, \ldots, x_N)$ *be a form of degree* d *that is irreducible over* $\overline{\mathbb{C}(x_1, \ldots, x_N)}[y_1, \ldots, y_n]$. *The number of non-associate irreducible forms* $Q \in \mathbb{C}[x_1, \ldots, x_N]$ *such that* (P, Q) *is not prime is bounded above by a function of* d.

We believe that the above theorem, and its more general version (which can be found in the full version), together with its proof, are of independent interest. We now briefly discuss the ingredients needed in the proof of the above theorem. If one only combines results from Gröbner basis theory and elimination theory with techniques from algebraic geometry, one can obtain an effective bound for the above theorem which depends on N, n and d. Since the goal of a Sylvester-Gallai theorem is to obtain bounds *independent of the number of variables*, such a bound will not be good enough for our purposes. By combining the Stillman's uniformity bounds of [5] with results from elimination theory and algebraic geometry, we are able to obtain an effective bound (depending only on n, d) on the number of non-associate irreducible forms Q such that (P, Q) is not prime.

Once the dependence on N has been removed and a finite bound on the number of (non-associate) bad forms Q has been established, we can then use Bertini's theorem and the Ananyan-Hochster principle to further drop the dependency on n, finally obtaining a bound which simply depends on the degree d. To achieve this, we give very effective bounds on the degrees of generators of Gröbner bases for ideals generated by a small number of variables of low degree, and also for certain elimination ideals of such ideals. These bounds could be of independent interest.

An important point to note is that [38] builds on the Ananyan-Hochster principle to show their transfer theorems, which we also use. However, to prove the above theorem, we also need to apply the Ananyan-Hochster principle in a different manner: combined with results from Gröbner basis theory, we use it to find low degree polynomials in elimination ideals of ideals generated by a small number of low degree forms.

1.3 Related Work

1.3.1 Sylvester-Gallai configurations

In [24], as a first step of his plan, Gupta proposed the study of a direct generalization of the linear Sylvester-Gallai configurations, which are captured by the following definition.

▶ **Definition 7** (Radical Sylvester–Gallai configuration). *Let* $\mathcal{F} \subset \mathbb{C}[x_1, \ldots, x_N]$ *be a finite set of irreducible forms of degree at most* d. *We say that* $\mathcal{F}$ *is a* d*-radical Sylvester-Gallai configuration if the following hold.*

- $F_i \notin (F_j)$ *for any* $F_i, F_j \in \mathcal{F}$.
- *For every* $F_i, F_j \in \mathcal{F}$, *we have* $F_k \in \operatorname{rad}(F_i, F_j)$ *for some* $F_k \in \mathcal{F} \setminus \{F_i, F_j\}$.

The following conjecture is a strengthening of [24, Conjecture 2].

▶ **Conjecture 8** (Radical Sylvester-Gallai conjecture). *There is a function* $\lambda : \mathbb{N} \to \mathbb{N}$ *such that if* $\mathcal{F}$ *is a* d*-radical Sylvester-Gallai configuration then* $\dim \operatorname{span}_{\mathbb{C}} \{\mathcal{F}\} = \lambda(d)$.

Conjecture 8 was proved by [47] in the special case when $d = 2$. Subsequently, [37] proved Conjecture 8 for $d = 3$, and [38] fully resolved the above conjecture in the affirmative. Note that the condition $F_k \in \operatorname{rad}(F_i, F_j)$ for forms of degree d is a natural generalization of the condition $\ell_k \in (\ell_i, \ell_j)$: by the Nullstellensatz, these conditions are equivalent to $V(F_k) \supset V(F_i, F_j)$ and $V(\ell_k) \supset V(\ell_i, \ell_j)$ respectively. Therefore Conjecture 8 is a direct generalisation of the usual Sylvester-Gallai theorem for forms of higher degrees.

The first work to consider product Sylvester-Gallai configurations was the work of [39], where the authors settled Conjecture 4 for quadratic forms. In follow up work [40], the same authors settled Conjecture 2 for the case when $d = 2$, therefore proving that Gupta's PIT algorithm runs in deterministic, poly-time for $\Sigma^3\Pi\Sigma\Pi^2$ circuits. Note that the above path in the proofs of Sylvester-Gallai type results is a natural path towards resolving the PIT problem for $\Sigma^3\Pi\Sigma\Pi^d$ circuits, since Conjecture 8 follows from Conjecture 4, and it can be shown that Conjecture 2 would follow from a robust version of Conjecture 4.

Robust and higher dimensional generalizations of the radical Sylvester-Gallai theorems have also been studied by the works [41, 22, 21]. Aside from settling such interesting questions in extremal combinatorial geometry, such variations are also motivated by the PIT problem for depth-4 circuits, in the hope that these more general versions may be useful towards obtaining a (potentially simpler) proof of Gupta's conjectures.

1.3.2 PIT for depth four circuits

Depth-4 circuits with bounded top fan-in are among the "easiest" classes for which we do not have deterministic, poly-time PIT algorithms. There has thus been some work on deterministic PIT for these circuits using methods other than the Sylvester-Gallai based methods discussed above.

In [11], the authors give a quasipolynomial time PIT algorithm for depth 4 circuits of bounded top and bottom fanins, via the Jacobian method of [4]. By using the logarithmic derivative and its power series, they are able to modify the top sum gate in the circuit to a powering gate.

The breakthrough work of [34] on lower bounds for bounded depth circuits gives another approach to PIT for this model. Hardness-randomness tradeoffs have been well studied in algebraic complexity [2, 30], and the work of [10] showed that these tradeoffs also hold in the bounded depth setting. Combining these results gives a subexponential time PIT algorithm for depth four circuits. Based on the lower bounds of [34], a hitting set generator for bounded depth circuits was constructed by [6], which gives another subexponential time PIT algorithm for depth four circuits, with improved parameters.

It is important to note that neither of the above methods seem to be able to give a truly polynomial time algorithm for the PIT problem for $\Sigma^k\Pi\Sigma\Pi^d$ circuits. The only currently known method that might do so is the approach of [24].

1.4 High-level proof overview

In this subsection we provide an overview of the proof of our main result, Theorem 5. We begin by outlining the high-level approach used in previous works to bound the dimension of the span of radical Sylvester-Gallai configurations and discuss the challenges of applying this method in our context. Next, we describe the tools that we employ to address these challenges. Finally, we provide a concise summary of how these components are combined to establish our main theorem.

To illustrate the ideas in this overview, we use a special case of product Sylvester-Gallai configurations with a simplifying assumption as an example. In the final part of this subsection, we explain how this assumption can be eliminated to extend the argument to the general case.

1.4.1 High-level approach

At a high level, our proof uses a similar approach as the previous works on higher degree Sylvester-Gallai configurations [47, 39, 40, 41, 21, 38], which we now summarize.

Let $R := \mathbb{C}[x_1, \ldots, x_N]$, and $S := R[y_1, \ldots, y_n]$ be two polynomial rings and $\mathcal{F} \subset S$ be a radical (or product) Sylvester-Gallai configuration, where the forms in $\mathcal{F}$ have degree at most d. Thus, we can write $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_d$, where $\mathcal{F}_e$ are the forms in $\mathcal{F}$ of degree e. We start with the assumption that each form only depends on constantly many variables in S. A number of key ideas can already be highlighted in this easier setting.

We want to control the highest degree forms $\mathcal{F}_d$ in our configuration, with the goal of reducing to the case when the highest degree is $d-1$, where we can proceed inductively.

If many ideals of the form (F_i, F_j), where $F_i, F_j \in \mathcal{F}_d$, are radical (or prime), then the set $\mathcal{F}_d$ is essentially a (robust) linear Sylvester-Gallai configuration. In this case, the linear Sylvester-Gallai theorems imply that constantly many forms $F_1, \ldots, F_a$ are a basis for $\operatorname{span}_{\mathbb{C}}\{\mathcal{F}_d\}$, and if $x_1, \ldots, x_r$ is the union of the set of variables of $F_1, \ldots, F_a$, then $\mathcal{F}_d \subset \mathbb{C}[x_1, \ldots, x_r]$. This is sufficient to control the forms of $\mathcal{F}_d$ and apply our inductive step. The interesting case is thus when there are many ideals generated by forms in $\mathcal{F}_d$ that are not radical (or prime). In this case, the goal is to show that the forms in $\mathcal{F}_d$ must share many variables in common. The steps listed below show how previous works have dealt with this case.

1. First, one devises a structure theorem on ideals that are not radical (or prime). In principle, such structure theorems show that if an ideal generated by two polynomials is not radical (or not prime), then the variables appearing in the polynomials must have some special "dependencies".
2. Next, combine the structure theorem with the local conditions to show the existence of constantly many "good variables" $x_1, \ldots, x_b$ such that $\mathcal{F}_d \subset (x_1, \ldots, x_b)$.
   In this step, our approach differs from previous works, as in a product configuration, it may not be possible to directly achieve this high level of control on the forms in $\mathcal{F}_d$. We discuss this in more detail in Step 2 below.
3. Finally, we "randomly project" (i.e., apply a general quotient) the special variables $x_1, \ldots, x_b$ to a new variable z in order to reduce the degree of the forms in $\mathcal{F}_d$.
   Also, in this step our approach will differ from previous works. Instead of reducing degree directly, we will make the forms in $\mathcal{F}$ "*factor more*."

We now elaborate on this approach, pointing out the new difficulties we face in this work, and sketch how we deal with these issues.

1.4.1.1 Step 1 – Structure theorems

Earlier works ([47, 37]) gave fairly strong classification of ideals that are not radical in the first step. In particular, they consider the possible minimal primes that a non-radical ideal of the form (P, Q) can have, and possible multiplicities that these minimal primes can have, giving a structural result for each of these cases. However this strategy is hard to generalise beyond the case of ideals generated by two cubics, since these ideals have very high degree and their primary decomposition becomes much more complex as the degree grows.

Subsequent work by [38] used a significantly weaker (albeit more general) structure theorem, and showed that this suffices to carry out the strategy above. The structure theorem in [38] is the following. If $P \in S$ is an irreducible form of degree d that is not in the ideal $(x_1, \ldots, x_N)$, then there are at most $3d^3$ square-free forms $Q_i \in R$ such that (P, Q_i) is not radical. Informally, this statement says that if a form P depends non-trivially on some

variable(s) y_i, then (P, Q) must be radical for almost every polynomial Q that only depends on $x_1, \ldots, x_N$.

For product Sylvester-Gallai configurations, the ideals of interest are those that are not prime. One could hope to generalize the structural result from [38] to this setting. However, a statement of the above form is no longer true here. Take the form $P = y_1^4 - x_1 x_2 y_2^2$. This form depends on the variables y_1, y_2. However, for every irreducible linear form $Q \in R$, the ideal $\left(P, x_1 x_2 - Q^2\right)$ is not prime, even though P and $x_1 x_2 - Q^2$ are irreducible. This is our first technical difficulty.

We overcome this by showing that the above only happens because P is reducible as a form in $\overline{\mathbb{C}\left(x_1, \ldots, x_N\right)}\left[y_1, y_2\right]$, since it factors as $P = \left(y_1^2 - \sqrt{x_1 x_2} y_2\right)\left(y_1^2 + \sqrt{x_1 x_2} y_2\right)$. This is where our first key conceptual and technical contribution comes in. We show that for any degree d form in $R[y_1, \ldots, y_m]$ that is irreducible in $\overline{\mathbb{C}\left(x_1, \ldots, x_N\right)}\left[y_1, \ldots, y_n\right]$ (such forms are called absolutely irreducible over $R[y_1, \ldots, y_m]$), there are only finitely many $Q_i \in R$ such that (P, Q_i) is not prime.

A fairly subtle issue remains in the above statement. In order to solve the product Sylvester-Gallai problem, we need quantitative bounds on the number of "bad forms" Q_i. Moreover, we need a bound which is solely a function of $\deg(P)$. If we use standard techniques from commutative algebra and algebraic geometry, we obtain bounds which depend on the number of variables (that is: N, n) and on $\deg(P)$. This is not sufficient for us, as we want to show that $\dim \operatorname{span}_{\mathbb{C}} \{\mathcal{F}\}$ does not depend on the number of variables in the polynomial ring. To prove that the number of bad forms is in fact *independent* of N and n, we need to work a bit harder, and we need to make use of two additional tools: Bertini-type theorems (to remove the dependence on n), and the Ananyan-Hochster construction of small strong subalgebras (this will remove the dependence on N). This yields, in its basic form, Theorem 6, which we discussed in Section 1.2.

1.4.1.2 Step 2 – Finding a small set of "good common variables"

Suppose $\mathcal{F}$ is a radical Sylvester-Gallai configuration. By the above, we can assume that $\mathcal{F}_d$ is not a robust linear Sylvester-Gallai configuration. Our assumption implies that for many pairs F_i, F_j, the ideal (F_i, F_j) is not radical. In particular, for a small $0 < \varepsilon < 1$, we can find $3d^3$ forms $F_1, \ldots, F_{3d^3} \in \mathcal{F}_d$ such that $(\mathcal{F}_i, \mathcal{F}_j)$ is not radical for $i \leq 3d^3$ and $j \leq (1-\varepsilon)\left|\mathcal{F}_d\right|$. The radical bound mentioned in Step 1 shows that there is $a \in O(1)$ and an ideal generated by linear forms $(\ell_1, \ldots, \ell_a)$ that contains every such F_j. Repeating this argument, we can find an ideal $(\ell_1, \ldots, \ell_b)$ with b constant such that $\mathcal{F}_d \subset (\ell_1, \ldots, \ell_b)$.

When $\mathcal{F}$ is a product Sylvester-Gallai configuration a number of issues arise: first, our structure theorem deals with prime ideals (instead of radical) and our assumptions are that the forms are absolutely irreducible. Hence, we will only be able to draw the following weaker conclusion: there are constantly many variables $x_1, \ldots, x_b$ such that every $F \in \mathcal{F}_d$ is either in $(x_1, \ldots, x_b)$, or absolutely reducible in the ring $S' := \mathbb{C}\left(x_1, \ldots, x_b\right)\left[x_{b+1}, \ldots, x_N, y_1, \ldots, y_n\right]$. Therefore, when we apply the general quotient (i.e. a random projection), we can no longer guarantee that the degree of our forms reduce. We deal with this issue by changing the way we "decompose" the configuration $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_d$. This is a key difference between our approach and previous approaches, since this issue does not arise when studying radical Sylvester-Gallai configurations. This new decomposition also allows us to solve an issue that arises when we apply the general quotient. We now state this issue, and discuss the solution.

1.4.1.3 Step 3 – General quotients & degree reduction/lowering degree

The image of an irreducible form under a projection map can be reducible. In order to apply induction, the definition of product Sylvester-Gallai configurations has to be generalised to allow the forms in $\mathcal{F}$ to be reducible. Hence, we update our definition of product Sylvester-Gallai configurations to the following.

▶ **Definition 9** (Product Sylvester–Gallai configurations). *A finite set $\mathcal{F} \subset \mathbb{C}[x_1, \ldots, x_N]$ of forms of degree at most d is a* d-product-SG-configuration *if the following hold.*

- $\gcd(F_i, F_j) = 1$ *for all* $i \neq j$.
- *For every* $F_i, F_j \in \mathcal{F}$, *we have* $\displaystyle\prod_{F \in \mathcal{F} \setminus \{F_i, F_j\}} F \in \operatorname{rad}(F_i, F_j)$.

This generalisation already appears in [38], where reducibility is not an issue when dealing with radical ideals, since the radical bound also applies to reducible forms. The prime bound however can no longer apply, irrespective of any assumption: if P is reducible then (P, Q) can never be prime. In particular, if every form in $\mathcal{F}_d$ is reducible, then we can draw no conclusion about $\mathcal{F}_d$ using the above ideas.

We tackle both these issues by turning this reducibility into an advantage: instead of trying to reduce the degree as in the inductive approaches of [47, 38, 39], we will try and make the forms in our configuration "factor more".

In order to formalise what we mean by "factor more", we introduce the notion of factor sets of a product configuration. Given a (generalized) product Sylvester-Gallai configuration $\mathcal{F}$, we define the factor set $\mathcal{I}$ of $\mathcal{F}$ to be the set of irreducible factors of all forms in $\mathcal{F}$. As before, we write $\mathcal{I} = \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_c$, where $\mathcal{I}_j \subset S_j$.

If every form in $\mathcal{F}$ factors a lot, then all the forms in $\mathcal{I}$ will have low degree. We will induct on the highest integer c such that $\mathcal{I}_c$ is nonzero. Now, the set of "common variables" we look for will not be variables common to $\mathcal{F}_d$, but variables common to $\mathcal{I}_c$. The absolute irreducibility condition remains, and we end up with variables $x_1, \ldots, x_b$ such that every form in $\mathcal{I}_c$ is either in the ideal $(x_1, \ldots, x_b)$ or absolutely reducible. We postpone the discussion of how we do this step to the next subsection. We now show how these common variables are used in the rest of the induction step.

Step 2 lets us find the "common variables" $x_1, \ldots, x_b$. Define a projection map φ that maps each x_i for $1 \leq i \leq b$ to a random multiple of a new variable z, and acts as identity on the other variables. If $F \in (x_1, \ldots, x_b)$ then $\varphi(F) = zF'$, where $\deg(F') < \deg(F)$. We define $\varphi(\mathcal{F})$ to be the image of $\mathcal{F}$ under φ, after factoring out powers of z. Such a projection map preserves radical and product Sylvester-Gallai structure, and also allows bounds on the dimension of $\varphi(\mathcal{F})$ to be lifted back to bounds on the dimension of $\mathcal{F}$.

If we have $\mathcal{F}_d \subset (x_1, \ldots, x_b)$ (as in the case of radical Sylvester-Gallai configurations) then $\varphi(\mathcal{F})$ is a Sylvester-Gallai configuration with forms of degree at most $d - 1$, and induction can be applied. If on the other hand $\mathcal{F}$ is a product Sylvester-Gallai configuration, then we have the weaker condition that every $F \in \mathcal{I}_c$ is either in $(x_1, \ldots, x_b)$ or absolutely reducible in S'. We show that the image of any F that is absolutely reducible in S' under the map φ is reducible. Thus, in either case we conclude that $\mathcal{I}'_c = \emptyset$, where $\mathcal{I}'$ is a factor set of $\varphi(\mathcal{F})$, and we can apply induction.

In the next subsection we show how we find these "common variables" in our setting. Note that we are still under the assumption that every form in $\mathcal{F}$ (and thus in $\mathcal{I}$) depends on only constantly many variables. We will also show how this assumption is removed.

1.4.2 Putting everything together

Let $\mathcal{F}$ be a product Sylvester-Gallai configuration of forms of degree at most d. We now allow forms in $\mathcal{F}$ to be reducible, but still require that they are square-free and are pairwise relatively prime. Let $\mathcal{I} := \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_d$ be the set of irreducible factors of forms in $\mathcal{F}$. We will induct on the largest integer c such that $\mathcal{I}_c \neq \emptyset$. We first go over the base case in our induction.

Our base case is when every form $F_i \in \mathcal{F}$ is a product of linear forms, say $F_i = \prod_a \ell_{ia}$. In this case, $\mathcal{I} = \mathcal{I}_1 = \{\ell_{ia}\}_{i,a}$. Now suppose we pick ℓ_{ia} and ℓ_{jb} with $i \neq j$. The ideal (ℓ_{ia}, ℓ_{jb}) is a minimal prime of (F_i, F_j). Therefore, the product Sylvester-Gallai condition implies that $F_k \in (\ell_{ia}, \ell_{jb})$ for some k, which in turn implies that $\ell_{ke} \in (\ell_{ia}, \ell_{jb})$ for some choice of index e. This is a linear Sylvester-Gallai relationship among the elements of $\mathcal{I}_1$. The choice of ℓ_{ia}, ℓ_{jb} was arbitrary, except the condition that $i \neq j$. Therefore, the set $\mathcal{I}_1$ is a robust linear Sylvester-Gallai configuration, and therefore has bounded rank. This in turn implies that the forms in $\mathcal{F}$ have bounded rank, since they are spanned by monomials of degree at most d in a basis for $\mathcal{I} = \mathcal{I}_1$.

We now sketch the induction step. Suppose c is the largest integer such that $\mathcal{I}_c \neq \emptyset$. Our induction step is further split into two steps, as discussed in the previous subsection.

1. First we show that there are variables $x_1, \ldots, x_b$ such that every form in $\mathcal{I}_c$ is either in the ideal $(x_1, \ldots, x_b)$ or absolutely reducible in $S' := \mathbb{C}(x_1, \ldots, x_b)[x_{b+1}, \ldots, x_N, y_1, \ldots, y_n]$. This is a relaxed version of step 2 in the previous approach, applied to $\mathcal{I}_c$ instead of $\mathcal{F}_d$.
2. We then show that projecting $x_1, \ldots, x_b$ to a new variable z results in a product Sylvester-Gallai configuration with $\mathcal{I}_e = \emptyset$ for any $e \geq c$.

We elaborate on these in the special case when $c = 2$, since this already captures all the main ideas. As before, if $\mathcal{I}_2$ is a robust linear Sylvester-Gallai configuration, then step 2 follows easily by picking a basis for $\mathcal{I}_2$. Consider $A, B \in \mathcal{I}_2$, and suppose $A|F_i$ and $B|F_j$ for some $F_i, F_j \in \mathcal{F}$. Suppose (A, B) is prime. Then (A, B) is a minimal prime of (F_i, F_j). The product Sylvester-Gallai configuration implies $F_k \in (A, B)$ for some k, and therefore $C \in (A, B)$ for some $C|F_k$. Further, by homogeneity, it must be that $C \in \mathcal{I}_2$. This gives us a linear Sylvester-Gallai relationship between the elements $A, B, C \in \mathcal{I}_2$. The assumption that $\mathcal{I}_2$ is not a robust linear Sylvester-Gallai configuration will therefore imply that (A, B) is not prime for many pairs A, B. Since the forms in $\mathcal{I}_2$ are irreducible (even though the forms in $\mathcal{F}$ might not be), by a combinatorial argument and Theorem 6, we deduce the existence of variables $x_1, \ldots, x_b$, completing step 1 above.

We now move on to second step. Suppose φ is a projection map that sends $x_1, \ldots, x_b$ to random multiples of a new variable z. The map φ sends elements in $(x_1, \ldots, x_b)$ to multiples of z. We also prove that φ sends elements of $\mathcal{I}_2 \setminus (x_1, \ldots, x_b)$ that are absolutely reducible in S' to reducible forms. Since $c = 2$, the degree of each of these forms has to be 1. Therefore, if we apply φ to some $F \in \mathcal{F}$, every factor of F will be linear, even though it is possible that $\deg(F) = \deg(\varphi(F))$. This reduces the problem to the base case, and the properties of φ allow us to lift the resulting bound on the dimension of $\varphi(\mathcal{F})$ back to a bound on the dimension of $\mathcal{I}$.

1.4.2.1 Removing the initial assumption

We started with the assumption that every form in $\mathcal{F}$ depended only on constantly many variables. In general, it is of course possible that the forms depend on many or even all the variables. This issue was overcome in [38], building on the seminal work of [5]. They show that forms $H_1, \ldots, H_a$ of high enough strength behave essentially like variables, in

the sense that $\mathbb{C}[H_1, \ldots, H_a]$ is isomorphic to a polynomial ring. Further, the extension $\mathbb{C}[H_1, \ldots, H_a] \subset S$ has many of the properties that the extension $\mathbb{C}[x_1, \ldots, x_a] \subset S$ has, the most useful of them being that this extension preserves arbitrary intersection of ideals.

Motivated by this, [38] define the notion of strong vector spaces and algebras, which are vector spaces of small dimension spanned by forms of high rank, and the algebras they generate. They show that in the general case of the radical Sylvester-Gallai problem (when the forms depend on more than constantly many variables), while there might not be variables $x_1, \ldots, x_b$ such that $\mathcal{F}_d \subset (x_1, \ldots, x_b)$, we can find a strong vector space V such that $\mathcal{F}_d \subset (V)$.

The above change complicates the projection step, since random restriction of higher degree forms is no longer well defined. To fix this, [38] defined generalised projection maps, and again proved that such maps have all the properties that the linear projection maps have. In particular, they preserve the radical Sylvester-Gallai structure, and they allow bounds on the dimension of the image of $\mathcal{F}$ to be lifted back to bounds on $\mathcal{F}$. With these tools, they set up an inductive framework to show radical Sylvester-Gallai configurations have bounded dimension, with linear Sylvester-Gallai configurations acting as the base case.

Extending our results to the setting of strong algebras in order to apply this framework requires two technical results. First we define the notion of absolute reducibility with respect to strong vector spaces. This allows us to extend the prime bound to strong algebras. We then show that general projection maps send forms that are absolutely reducible with respect to strong vector spaces to reducible forms. These two extensions to the above framework allow us to prove our main theorem in the general case.

2 Conclusion

In this work, we prove that product Sylvester-Gallai configurations of forms have bounded dimension, generalising the result of [38], and getting one step closer towards a PIT algorithm for $\Sigma^k\Pi\Sigma\Pi^d$ circuits. To achieve this, we give a novel, and very effective, sufficient condition for the primality of ideals of the form (P, Q), where P "depends on more variables" than Q, via the transfer principle from [38]. This turns out to be far more subtle, and to require more tools from algebraic geometry, than giving a sufficient condition for the ideal to be radical.

Our work leaves two important questions, in order to bridge the gap between our results and a complete analysis for a polynomial-time, deterministic PIT algorithm for $\Sigma^k\Pi\Sigma\Pi^d$ circuits. The first question is to extend this work and prove a higher codimension product Sylvester-Gallai theorem for all degrees, simultaneously generalising the main theorems of [21] and this work. The second question is to provide a tight connection between such Sylvester-Gallai problems and rank bounds for identities in $\Sigma^k\Pi\Sigma\Pi^d$, thereby generalizing the beautiful result in [45, Theorem 1.4] to circuits of depth four.

References

1 M. Agrawal and Manindra. Vinay. Arithmetic circuits: A chasm at depth four. *In 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28*, 2008.

2 Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science: 25th International Conference, Hyderabad, India, December 15-18, 2005. Proceedings 25*, pages 92–105. Springer, 2005. doi:10.1007/11590156_6.

3 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

4 Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian hits circuits: Hitting sets, lower bounds for depth-d occur-k formulas and depth-3 transcendence degree-k circuits. *SIAM Journal on Computing*, 45(4):1533–1562, 2016. doi:10.1137/130910725.

5 Tigran Ananyan and Melvin Hochster. Small subalgebras of polynomial rings and stillman's conjecture. *Journal of the American Mathematical Society*, 33(1):291–309, 2020.

6 Robert Andrews and Michael A. Forbes. Ideals, determinants, and straightening: proving and using lower bounds for polynomial ideals. In *54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 389–402, 2022. doi:10.1145/3519935.3520025.

7 Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 519–528, 2011. doi:10.1145/1993636.1993705.

8 M. Beecken, J. Mittmann, and N. Saxena. Algebraic independence and blackbox identity testing. *Information and Computation*, 222:2–19, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011). doi:10.1016/j.ic.2012.10.004.

9 Peter Borwein and William OJ Moser. A survey of sylvester's problem and its generalizations. *Aequationes Mathematicae*, 40(1):111–135, 1990.

10 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing*, 15(1):1–34, 2019. doi:10.4086/TOC.2019.V015A013.

11 Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In *Proceedings of the 36th Computational Complexity Conference*, CCC '21, 2021. doi:10.4230/LIPIcs.CCC.2021.11.

12 Zeev Dvir. Incidence theorems and their applications. *arXiv preprint arXiv:1208.5073*, 2012. arXiv:1208.5073.

13 Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Improved rank bounds for design matrices and a new proof of kelly's theorem. In *Forum of Mathematics, Sigma*, volume 2. Cambridge University Press, 2014.

14 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM Journal on Computing*, 36(5):1404–1434, 2007. doi:10.1137/05063605X.

15 Michael Edelstein and Leroy M Kelly. Bisecants of finite collections of sets in linear spaces. *Canadian Journal of Mathematics*, 18:375–380, 1966.

16 N. Elkies, L. Pretorius, and K. Swanepoel. Sylvester–gallai theorems for complex numbers and quaternions. *Discrete Comput Geom*, 35:361–373, 2006. doi:10.1007/S00454-005-1226-7.

17 Paul Erdos, Richard Bellman, Hubert S Wall, James Singer, and Victor Thébault. Problems for solution: 4065-4069. *The American Mathematical Monthly*, 50(1):65–66, 1943.

18 Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. A deterministic parallel algorithm for bipartite perfect matching. *Communications of the ACM*, 62(3):109–115, 2019. doi:10.1145/3306208.

19 Michael A Forbes and Amir Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 527–542. Springer, 2013. doi:10.1007/978-3-642-40328-6_37.

20 Tibor Gallai. Solution of problem 4065. *American Mathematical Monthly*, 51:169–171, 1944.

21 Abhibhav Garg, Rafael Oliveira, Shir Peleg, and Akash Kumar Sengupta. Radical Sylvester-Gallai Theorem for Tuples of Quadratics. In *38th Computational Complexity Conference (CCC 2023)*, pages 20:1–20:30, 2023. doi:10.4230/LIPIcs.CCC.2023.20.

22 Abhibhav Garg, Rafael Oliveira, and Akash Kumar Sengupta. Robust Radical Sylvester-Gallai Theorem for Quadratics. In *38th International Symposium on Computational Geometry (SoCG 2022)*, pages 42:1–42:13, 2022. doi:10.4230/LIPIcs.SoCG.2022.42.

23 Zeyu Guo. Variety Evasive Subspace Families. In Valentine Kabanets, editor, *36th Computational Complexity Conference (CCC 2021)*, volume 200 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:33, 2021. doi:10.4230/LIPIcs.CCC.2021.20.

24 Ankit Gupta. Algebraic geometric techniques for depth-4 pit & sylvester-gallai conjectures for varieties. In *Electron. Colloquium Comput. Complex.*, volume 21, page 130, 2014.

25 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM Journal on Computing*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

26 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 821–830, 2017. doi:10.1145/3055399.3055440.

27 Sten Hansen. A generalization of a theorem of sylvester on the lines determined by a finite point set. *Mathematica Scandinavica*, 16(2):175–180, 1965.

28 Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute. In *Proceedings of the twelfth annual ACM Symposium on Theory of Computing*, pages 262–272, 1980.

29 Friedrich Hirzebruch. Arrangements of lines and algebraic surfaces. In *Arithmetic and geometry*, pages 113–140. Springer, 1983.

30 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *computational complexity*, 13:1–46, 2004. doi:10.1007/S00037-004-0182-6.

31 Zohar S. Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. In *2008 23rd Annual IEEE Conference on Computational Complexity*, pages 280–291, 2008. doi:10.1109/CCC.2008.15.

32 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 198–207. IEEE, 2009. doi:10.1109/FOCS.2009.67.

33 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24:295–331, 2015. doi:10.1007/S00037-015-0102-Y.

34 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 804–814. IEEE, 2022.

35 Eberhard Melchior. Uber vielseite der projektiven ebene. *Deutsche Math*, 5:461–475, 1940.

36 Ketan Mulmuley. Geometric complexity theory v: Efficient algorithms for noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017.

37 Rafael Oliveira and Akash Sengupta. Radical sylvester-gallai theorem for cubics. *FOCS*, 2022.

38 Rafael Oliveira and Akash Kumar Sengupta. Strong algebras and radical sylvester-gallai configurations. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 95–105, 2024. doi:10.1145/3618260.3649617.

39 Shir Peleg and Amir Shpilka. A Generalized Sylvester-Gallai Type Theorem for Quadratic Polynomials. In *35th Computational Complexity Conference (CCC 2020)*, pages 8:1–8:33, 2020. doi:10.4230/LIPIcs.CCC.2020.8.

40 Shir Peleg and Amir Shpilka. Polynomial time deterministic identity testing algorithm for $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits via edelstein-kelly type theorem for quadratic polynomials. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 259–271. ACM, 2021. doi:10.1145/3406325.3451013.

41 Shir Peleg and Amir Shpilka. Robust sylvester-gallai type theorem for quadratic polynomials. In *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224, pages 43:1–43:15, 2022. doi:10.4230/LIPIcs.SoCG.2022.43.

42 L. M. Pretorius and K. J. Swanepoel. The sylvester-gallai theorem, colourings and algebra. *Discret. Math.*, 2009.

43 Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99:49–79, 2009.

44 Nitin Saxena. Progress on polynomial identity testing-ii. *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, pages 131–146, 2014.

45 Nitin Saxena and Comandur Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *Journal of the ACM (JACM)*, 60(5):1–33, 2013. doi:10.1145/2528403.

46 Jean-Pierre Serre. Advanced problem 5359. *Amer. Math. Monthly*, 73(1):89, 1966.

47 Amir Shpilka. Sylvester-gallai type theorems for quadratic polynomials. *Discrete Analysis*, page 14492, 2020.

48 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010. doi:10.1561/0400000039.

49 James Joseph Sylvester. Mathematical question 11851. *Educational Times*, 59(98):256, 1893.

50 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Information and Computation*, 240:2–11, 2015. doi:10.1016/J.IC.2014.09.004.

A PTAS for TSP with Neighbourhoods over Parallel Line Segments

Benyamin Ghaseminia
Department of Computing Science, University of Alberta, Edmonton, Canada

Mohammad R. Salavatipour
Department of Computing Science, University of Alberta, Edmonton, Canada

Abstract

We consider the Travelling Salesman Problem with Neighbourhoods (TSPN) on the Euclidean plane ($\mathbb{R}^2$) and present a Polynomial-Time Approximation Scheme (PTAS) when the neighbourhoods are parallel line segments with lengths between $[1, \lambda]$ for any constant value $\lambda \geq 1$. In TSPN (which generalizes classic TSP), each client represents a set (or neighbourhood) of points in a metric and the goal is to find a minimum cost TSP tour that visits at least one point from each client set. In the Euclidean setting, each neighbourhood is a region on the plane. TSPN is significantly more difficult than classic TSP even in the Euclidean setting, as it captures group TSP. A notable case of TSPN is when each neighbourhood is a line segment. Although there are PTASs for when neighbourhoods are fat objects (with limited overlap), TSPN over line segments is **APX**-hard even if all the line segments have unit length. For parallel (unit) line segments, the best approximation factor is $3\sqrt{2}$ from more than two decades ago. The PTAS we present in this paper settles the approximability of this case of the problem. Our algorithm finds a $(1+\varepsilon)$-factor approximation for an instance of the problem for n segments with lengths in $[1, \lambda]$ in time $n^{O(\lambda/\varepsilon^3)}$.

2012 ACM Subject Classification Theory of computation → Routing and network design problems; Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Scheme, TSP Neighbourhood, Parallel line segments

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.53

Related Version *Full Version*: `https://arxiv.org/abs/2504.02190` [21]

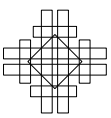
Funding *Mohammad R. Salavatipour*: NSERC

1 Introduction

The Travelling Salesman Problem (TSP) is one of the most fundamental and well-studied problems in combinatorial optimization due to its wide range of applications. In TSP, one is given a set of points in a metric space and the goal is to find a (closed) tour (or walk) of minimum length visiting all the points. For several decades, the classic algorithm by Christofides [10] and independently by Serdyukov [32] which implies a $\frac{3}{2}$-approximation was the best-known approximation for TSP until a recent result in [24] which shows a slight improvement. Several generalizations (or special cases) of TSP have been studied as well, the most notable is when the points are given in fixed dimension Euclidean space. Arora and Mitchell [4, 30] presented different PTASs for (fixed dimension) Euclidean TSP. There have been many papers that have extended these results. Arkin and Hassin [3] introduced the notion of TSP with neighbourhoods (TSPN).

An instance of TSPN is a set of neighbourhoods (or regions) given in a metric space, and the goal is to find a minimum length (or cost) tour that visits all these regions. Each region can be a single point or could be defined by a subset of points. They gave several $O(1)$-approximations for the geometric settings where each region is some well-defined shape on the plane, e.g. disks, and parallel unit length segments. Several papers have studied TSPN for various classes of neighbourhoods and under different metrics.

© Benyamin Ghaseminia and Mohammad R. Salavatipour;
licensed under Creative Commons License CC-BY 4.0



TSPN is much more difficult than TSP in general and in special cases, just as group Steiner tree is much more difficult than Steiner tree (one can consider each neighbourhood as a group/set from which at least one point needs to be visited). In group Steiner tree or group TSP, one is given a metric along with groups of terminals (each group is a finite set). The goal is to find a minimum cost Steiner tree (or a tour) that contains (or visits) at least one terminal from each group. TSPN generalizes group TSP by allowing infinite size groups. Using the hardness result for group Steiner tree [22], it follows that general TSPN is hard to approximate within a factor better than $\Omega(\log^{2-\varepsilon} n)$ for any $\varepsilon > 0$ even on tree metrics. The algorithms for group Steiner tree on trees in [20], and embedding of metrics onto tree metrics in [18], imply an $O(\log^3 n)$-approximation for TSPN in general metrics. Unlike Euclidean TSP (which has a PTAS), TSPN is **APX**-hard on the Euclidean plane (i.e. $\mathbb{R}^2$) [11]. The special case when each region is an arbitrary finite set of points in the Euclidean plane (group TSP) has no constant approximation [31] and the problem remains **APX**-hard even when each region consists of exactly two points [13].

Focusing on Euclidean metrics, most of the earlier work have studied the cases where the regions (or objects) are *fat*. Roughly speaking, it usually means the ratio of the smallest enclosing circle to the largest circle fitting inside the object is bounded. There are some work on when regions are *not* fat, most notably when the regions are (infinite) lines or line segments or in higher dimensions when they are hyperplanes. For the case of infinite line segments on $\mathbb{R}^2$, the problem for n lines can be solved exactly in $O(n^4 \log n)$ time by a reduction to the Shortest Watchman Route Problem (see [12, 23]). For the same setting, Dumitrescu and Mitchell [14] presented a linear time $\frac{\pi}{2}$-approximation, which was improved to $\sqrt{2}$ by Jonsson [23] (again in linear time). For infinite lines in higher dimensions (i.e. ≥ 3), the problem is proved to be **APX**-hard (see [2] and references there). For neighbourhoods being hyperplanes and dimension being $d \geq 3$, Dumitrescu and Tóth [15] present a constant factor approximation (which grows exponentially with d). For arbitrary d, they present an $O(\log^3 n)$-approximation. For any fixed $d \geq 3$, authors of [1] present a PTAS.

For parallel (unit) line segments on the plane ($\mathbb{R}^2$) Arkin and Hassin [3] presented a $(3\sqrt{2}+1)$-approximation which was improved to $3\sqrt{2}$ by [14], which remains the best-known approximation for this case as far as we know for over two decades. Authors of [17] proved that TSPN for unit line segments (in arbitrary orientation) is **APX**-hard. In this paper, we settle the approximability of TSPN when regions are parallel line segments of similar length (unit length is a special case) and present a PTAS for it. As mentioned above, the best-known approximation for unit length parallel segments has ratio $3\sqrt{2}$ [14]. We first focus on the case of unit line segments and show how our result extends to when line segments have bounded length ratio. This is in contrast with the **APX**-hardness of [17] for unit line segments with arbitrary orientation. Our result also implies a $(2+\varepsilon)$-approximation for the case where the segments can be both vertical and horizontal.

1.1 Related Work

The work on TSPN is extensive, we list a subset of the most notable and relevant work here and refer to the references of them for earlier works. All works listed are for $\mathbb{R}^2$ metric. Arkin and Hassin [3] presented constant factor approximations for several TSPN cases including when the neighbourhoods are parallel unit-length line segments (with ratio $3\sqrt{2}+1$). Very recently, PTASs were proposed for the case of unit disks and unit squares in [5]. Mata and Mitchell [25] presented $O(\log n)$-approximation for general connected polygonal neighbourhoods. Mitchell [28] gave a PTAS the case that regions are *disjoint* fat objects. This built upon his earlier work on PTAS for Euclidean TSP [30]. However, it is still open

whether the problem is **APX**-hard for disjoint (general) shapes. Dumitrescu and Mitchell [14] presented several results, including $O(1)$-approximation for TSPN where objects are connected regions of the same or similar diameter. This implies $O(1)$-approximation for line segments of the same length in arbitrary orientation. They also improved the bounds for cases of parallel segments of equal length, translates of a convex region, and translates of a connected region. For parallel (unit) line segments they obtain a $3\sqrt{2}$-approximation which remained the best ratio for this class until now. Feremans and Grigoriev [19], gave a PTAS for the case that regions are disjoint fat polygons of similar size. A similar result was obtained by [7] which gave a PTAS for TSPN for disjoint fat regions of similar sizes. Mitchell [29] presented an $O(1)$-approximation for planar TSPN with pairwise-disjoint connected neighbourhoods of any size or shape (see also [11]). Subsequently, [16] gave $O(1)$-approximation for discrete setting where regions are overlapping, convex, and fat with similar size.

These results were further generalized by Chan and Elbassioni [8] who presented a QPTAS for fat weakly disjoint objects in doubling dimensions (this allows limited amount of overlapping of the objects). This was further improved to a PTAS for the same setting [9]. The TSPN problem is even harder if the neighbourhoods are disconnected. See the surveys of Mitchell [26, 27]. If the metric is planar and each group (or neighbourhood) is the set of vertices around a face, [6] presents a PTAS for the group Steiner tree and a $(2+\varepsilon)$-approximation for TSPN.

1.2 Our Results and Technique

The main result of this paper is the following theorem.

▶ **Theorem 1.** *Given a set of n parallel (say vertical) line segments with lengths in $[1, \lambda]$ for a fixed λ as an instance of TSPN, there is an algorithm that finds a $(1+\varepsilon)$-approximation solution in time $n^{O(\lambda/\varepsilon^3)}$.*

The algorithm we present is randomized but can be easily derandomized (like the PTAS for classic TSP). To simplify the presentation, we give the proof for the case of unit line segments first.

This problem generalizes the classic (point) TSP (at a loss of $(1+\varepsilon)$ factor). Given a point TSP instance, scale the plane so that the minimum distance between the points is at least $1/\varepsilon$; call this instance $\mathcal{I}$. Obtain instance $\mathcal{I}'$ for line TSP by placing a vertical unit line segment over each point. It is easy to see that the optimum solutions of $\mathcal{I}$ and $\mathcal{I}'$ differ by at most an ϵ factor.

The difficult cases for line TSP are when the line segments are not too far apart (for e.g. they can be packed in a box of size $O(\sqrt{n})$ or smaller). There are two key ingredients to our proof that we explain here. One may try to adapt the hierarchical decomposition by Arora [4] to this setting. Following that hierarchical decomposition, the first issue is that some line segments might be crossing the horizontal dissecting lines, and so we don't have independent sub-instances, and it is not immediately clear in which subproblem these crossing segments must be covered. Note that the number of line segments crossing a dissecting line can be large. Our first insight is the following:

Insight 1: *At a loss of $(1+\varepsilon)$, we can drop the line segments crossing horizontal dissecting lines and instead requiring a subset of portals of each square (of the dissection) to be visited, provided we continue the quad-tree decomposition until each square has size $\Theta(1/\varepsilon)$.*

In other words, assuming all the squares in the decomposition have height at least $\Omega(1/\varepsilon)$, then at a small loss we can show that a solution for the modified instance where line segments on the boundary of the squares are dropped, can be extended to a solution

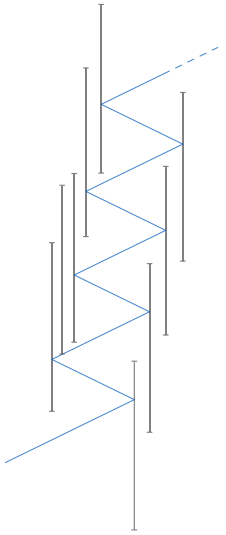


Figure 1 A potential arrangement of line segments where the solution has a large shadow.

for the original instance. So, proving this property allows us to work with the hierarchical (quad-tree) decomposition until squares of size $\Theta(1/\varepsilon)$. This can be proved by a proper packing argument. But then we need to be able to solve instances where the height of the sub-problem instance is bounded by $O(1/\varepsilon)$. Let's define the notion of shadow of a solution (or in general, shadow of a collection of paths on the plane) as the maximum number of times a vertical sweeping line Γ (that moves from left to right) intersects any of these paths. Our second insight is the following:

Insight 2: *If we consider a window that is a horizontal strip of height $O(h)$ and move this window vertically anywhere over an optimum solution, then the shadow of the parts of optimum visible in this strip is at most $O(h)$.*

In other words, one expects that in the base case of the decomposition (where squares have height $\Theta(1/\varepsilon)$), the shadow is bounded by $O(1/\varepsilon)$. Despite our efforts, proving this appears to be more difficult than thought and it seems there are examples where, even in the unit length segments, the shadow may be large (see Figure 1). We were not able to prove this nor come up with an explicit counterexample. However, we are able to prove the following slightly weaker version that still allows us to prove the final result:

(Revised) Insight 2: *There is a $(1+\varepsilon)$-approximate solution such that the shadow of any strip of height h over that solution is bounded by $O(f(\varepsilon)\cdot h)$. for some function $f(\cdot)$.*

The proof of this insight forms the bulk of our work. We characterize specific structures that would be responsible for having a large shadow in a solution and show how we can modify the solution to a $(1+\varepsilon)$-approximate one with shadow $O(1/\varepsilon)$. Consider the unit line segments case, and suppose opt is the cost of an optimum solution.

▶ **Theorem 2.** *Given any $\varepsilon > 0$, there is a solution $\mathcal{O}'$ of cost at most $(1+\varepsilon)\cdot$ opt such that in any strip of height 1, the shadow of $\mathcal{O}'$ is $O(1/\varepsilon)$.*

We will show that this near-optimum solution has in fact further structural properties that allow us to solve the bounded height cases at the base cases of the hierarchical decomposition using a Dynamic Program (DP) which, later on, is referred to as the inner DP. Proof of this theorem is fairly long and involves multiple steps that gradually proves structural properties for specific configurations.

To get a very high level idea of the proof of this theorem, consider some fixed optimum solution OPT. We decompose the problem into horizontal *strips* by drawing horizontal lines, called *cover-lines* that are 1-unit apart. The region of the plane between two consecutive cover-lines is called a *strip*. Note that each line segment crosses one cover-line (or might touch exactly two consecutive cover-lines). Let's consider one strip, say S, and consider the intersection of OPT with this strip. This intersection looks like a collection of paths that enter/exit this strip. We define the shadow of this strip similarly: consider a hypothetical vertical sweep line that moves left to right along the x-axis, the maximum number of intersections of this sweep line with these pieces of OPT restricted to S is the shadow in S. We show that we can modify OPT to a near-optimum solution of cost at most $(1+\varepsilon)$ times that of OPT so that the shadow in each strip is at most $O(1/\varepsilon)$. To prove this, we show that there are certain potential structures that can cause OPT having a large shadow in a strip, one of which we call a zig-zag (see Figure 5). We show that we can modify OPT (at a small increase to the cost) so that the shadow becomes bounded along each zig-zag (or similar structures).

Organization of the paper. We start by some preliminaries in the next section. In Section 3, we present some structural properties of a near-optimum solution and prove Theorem 2. We describe the main algorithm in Section 4, which includes the outer DP and inner DP. All the missing proofs appear in the full version of the paper [21].

2 Preliminaries

Suppose we are given n vertical line segments $s_1, \ldots, s_n$ of lengths in the range $[1, \lambda]$ for some constant $\lambda \geq 1$, and the top and bottom points of each s_i are denoted by s_i^t and s_i^b, respectively. These end-points are also called *tips* of the segment. For any point p, let $x(p)$ and $y(p)$ denote the x and y-coordinates of p, respectively. Similarly, for any segment or vertical line s, let $x(s)$ denote its x-coordinate. For two points p, q, we use $||pq||$ to denote the Euclidean distance between them. A feasible (TSP) tour is specified by a sequence of points where each of these points is on one of the segments of the instance and each line segment has at least one such point, and the tour visits these points consecutively using straight lines. The line that connects two consecutive points in a tour is called a *leg* of the tour. In our problem, the goal is to find a TSP tour of minimum total length. As mentioned earlier, we focus on the case where all the line segments have length 1 and then show how the proof easily extends to the setting where they have lengths in $[1, \lambda]$. Fix an optimum solution, which we refer to by OPT and use opt to refer to its cost. Our goal is to show the existence of a near-optimum (i.e. $(1+\varepsilon)$-approximate) structured solution that allows us to find it using dynamic programming.

First, we show at a small loss we can assume all the line segments have different x-coordinates. We assume that the minimal bounding box of these line segments has length L and height H. For now, assume $H > 3$ (case of $H \leq 3$ is easier, see Theorem 4). Let $B = \max\{L, H-2\}$. So opt $\geq 2B$; we can also assume $B \leq \frac{n}{\varepsilon}$, because otherwise opt $\geq 2n/\varepsilon$ and if we consider an arbitrary point on each line segment (say the lower tip) and use a PTAS for the classic TSP for these points, then it will be a PTAS for our original instance as well (because we pay at most an extra $+2$ for each line for a total of $2n$ which is $O(\varepsilon \cdot \text{opt})$). For a given $\epsilon > 0$, consider a grid on the plane with side length $\frac{\epsilon B}{n^2}$. Now move each line segment (parallel to the y-axis) so that the lower tip of each s_i is moved to the nearest grid point where there is no other line segment s_i with that x-coordinate. By doing this,

all the segments will have different x-coordinates and each segment would move at most $\frac{\sqrt{2}}{2} \cdot \frac{\varepsilon B}{n} < \frac{\epsilon B}{n}$, and in total, all segments would move at most a distance of ϵB. So the optimum value of the new instance has cost at most $(1+\varepsilon) \cdot \text{opt}$. For simplicity of notations, from now on we assume the original instance has this property and let OPT (and opt) refer to an optimum (and its value) of this modified instance.

3 Properties Of A Structured Near-Optimum Solution

One special instance of the problem is when there is a horizontal line that crosses all the input segments. This special case can be detected and solved easily. Otherwise, any optimum solution will visit at least 3 points that are not colinear. In such cases, like in the classic (point) TSP [4], we can assume the optimum does not cross itself, i.e. there are no two legs of the optimum ℓ (between points p, q) and ℓ' (between points p', q') that intersect, as otherwise removing these two and adding the pair of $pq', p'q$ or pp', qq' will be a feasible solution of smaller cost.

▶ **Definition 3.** *Given a collection $\mathcal{P}$ of paths on the plane and a vertical line at point $x_0 \in \mathbb{R}$, the **shadow** at x_0 is the number of legs of the paths in $\mathcal{P}$ that have an intersection with the vertical line at x_0. The shadow of a given range $[a, b]$ is defined to be the maximum shadow of values $x_0 \in [a, b]$.*

Note that if a solution is self-crossing, the operation of uncrossing (which reduces the cost) does not increase the shadow

Suppose the sequence of points of OPT is $p_1, p_2, \ldots, p_\sigma$ and the straight lines connecting these points (i.e. legs of OPT) are $\ell_1, \ell_2, \ldots, \ell_\sigma$ where ℓ_i connects two points p_i, p_{i+1} (with $p_{\sigma+1} = p_1$), and each s_i has at least one point p_j on it. We consider OPT oriented in this order, i.e. going from p_i to p_{i+1}. Since all segments have distinct x-coordinates, we can assume No two consecutive points p_i, p_{i+1} are on the same line segment by short-cutting (so no leg ℓ_i is vertical) and all points p_i on distinct line segments have distinct x-coordinates.

As mentioned before, let the length of the sides of the minimal bounding box of an instance of the problem be $L \times H$. By proving the following special case, we show we can instead focus on the cases that H is large.

▶ **Theorem 4.** *If $H \leq 3$, then the shadow of an optimum solution is at most 2.*

From now on we assume that $H > 3$. Our main goal is to prove Theorem 2.

▶ **Definition 5.** *Given a segment s and a leg ℓ incident to a point on s, we say ℓ is to the **left** of s if ℓ is entirely in the subplane $x \leq x(s)$; and ℓ is to the **right** of s if ℓ is entirely in the subplane $x \geq x(s)$.*

Since there are no vertical legs, there is no leg that is both to the left and to the right of a segment of the instance at the same time. Consider any segment s_i and suppose that ℓ_j, ℓ_{j+1} are the two legs of OPT with common end-point p_j that is on s_i. Let s_i^t and s_i^b denote the top and the bottom tips of s_i. We consider 3 possible cases for the location of p_j and the arrangement of ℓ_j, ℓ_{j+1}. Informally, one possibility is that the two legs ℓ_j, ℓ_{j+1} form a straight line that crosses s_i at p_j; one possibility is that the two legs are touching s_i at one of its tips (i.e. $p_j = s_i^t$ or $p_j = s_i^b$) such that one is to the left and one is to the right of s_i and they don't make a straight line, and the third possibility is that the two legs ℓ_j, ℓ_{j+1} are on the same side (both left or both right) of s_i.

▶ **Observation 6.** *Consider any segment s_i (with top/bottom points s_i^t,s_i^b) and suppose that ℓ_j, ℓ_{j+1} are the two legs of OPT with common end-point p_j that is on s_i. Then either:*

- *The subpath of OPT going through p_{j-1}, p_j, p_{j+1} forms a straight line (i.e. $\angle \ell_j p_j \ell_{j+1} = \pi$), and ℓ_j, ℓ_{j+1} are on two sides (left/right) of s_i; then we call p_j a **straight point**, or*
- *p_j is a tip of s_i (i.e. $p_j = s_i^t$ or $p_j = s_i^b$), $\angle \ell_j p_j \ell_{j+1} \neq \pi$ and ℓ_j and ℓ_{j+1} are on two sides of s_i (one left and one right); in this case p_j is called a **break point**, or*
- *both ℓ_j, ℓ_{j+1} are on the left or on the right of s_i; in this case p_j is called a **reflection point**.*

For the case of a reflection point p_j with two legs ℓ_j, ℓ_{j+1}, if both legs are to the left of the segment it is called a *left reflection* point and otherwise it is a *right reflection* point. Also note that if ℓ_j, ℓ_{j+1} are on the two sides of s_i and $\angle \ell_i p_j \ell_{i+1} \neq \pi$, then p_j must be a tip, or else we could move p_j slightly up or down and reduce the length of OPT (see Figure 2).

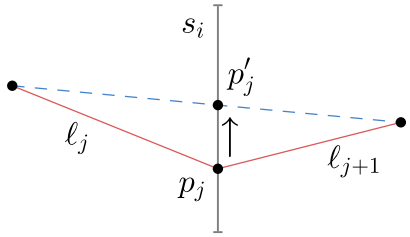


Figure 2 If p_j isn't a tip of s_i, then ℓ_j, ℓ_{j+1} must be collinear.

▶ **Lemma 7.** *If P is a subpath of OPT with end-points p, q where both are to the right of a vertical line Γ, and if P crosses Γ, then the left-most point on P to the left of Γ is a right reflection point (symmetric statement holds for opposite directions).*

▶ **Definition 8.** *Consider an arbitrary reflection point r on a segment s. Let the two legs of OPT incident to r visited before and after r (on the orientation of OPT) be ℓ^- and ℓ^+, respectively. ℓ^- is said to be **on top of** ℓ^+ if all the points of ℓ^- have larger y-coordinate than all of the points of ℓ^+. In this case we also call ℓ^- the* upper leg *and ℓ^+ the* lower leg. *Also, in this case, r is called a **descending** reflection point. If ℓ^+ is on top of ℓ^-, then r is called an **ascending** reflection point.*

If ℓ_j, ℓ_{j+1} are two legs incident to a reflection point p on a segment s, if the angle between ℓ_j and s is the same as the angle between ℓ_{j+1} and s (i.e. ℓ_{j+1} is like the reflection of a ray ℓ_j on mirror s) then p is called a *pure reflection* point.

▶ **Lemma 9.** *Any reflection point that is not a tip of a segment is a pure reflection point.*

We say a subpath contains a reflection point p_j if p_j is not the start or end vertex of the subpath (i.e. both legs of incident to p_j belong to that subpath.)

▶ **Lemma 10.** *If a sweeping vertical line Γ moves left to right on the x-axis, the only values of x for which the shadow at Γ changes will be when Γ hits a reflection point on that x-coordinate. Specifically, this means that any subpath of OPT that doesn't contain a reflection point, must have a shadow of 1 throughout its length.*

▶ **Definition 11.** *Let P_1 and P_2 be any two subpaths of OPT. We say P_1 is **above** P_2 in range $I = [x_0, x_1]$ if for every vertical line Γ with $x(\Gamma) \in I$, the top-most intersection of Γ with these two paths is a point on P_1. We say P_2 is **below** P_1 if the bottom-most intersection of Γ*

*with P_1, P_2 is a point on P_2. Similarly, we say L_1 is to the **left** of L_2 in range $I' = [y_0, y_1]$ if for every horizontal line Λ with $y(\Lambda) \in I'$, the left-most intersection point of Λ with L_1, L_2 (i.e. one with the least x value) always belongs to L_1. We say L_2 is to the **right** of L_1 if the right-most intersection of Λ is with L_2.*

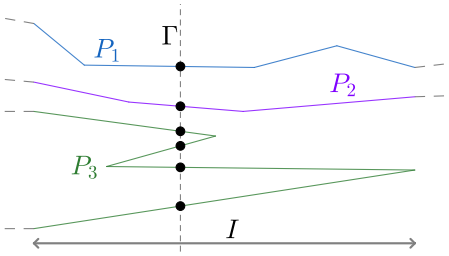


Figure 3 In range I, P_1 is above P_2, P_3, and P_2 is above P_3.

▶ **Lemma 12.** *For any distinct points p_j and $p_{j'}$ on OPT, following OPT according to its orientation, either the path from p_j to $p_{j'}$ or the path from $p_{j'}$ to p_j must contain at least one reflection point.*

▶ **Lemma 13.** *Among the set of points visited by OPT following its orientation, suppose $p_j, p_{j'}$, $j < j'$ (on segments s_i, $s_{i'}$, respectively) are two consecutive reflection points (i.e. no other reflection point exists in between them). Then p_j and $p_{j'}$ cannot be both left or both right reflection points. Furthermore, if s_i is to the left of $s_{i'}$ then p_j is a right reflection and $p_{j'}$ is a left reflection (the opposite holds if $s_{i'}$ is to the left of s_i).*

▶ **Corollary 14.** *Consecutive reflection points in OPT alternate between left and right reflections.*

3.1 Strips, Zig-zag, Sink

We decompose the problem into horizontal *strips* by drawing some parallel horizontal lines, which we call *cover-lines*. Starting from the bottom tip of the top-most segment, draw parallel horizontal lines that are 1-unit apart, these are our *cover-lines*. Each input segment is considered "covered" by the top-most (i.e. the first) cover-line that intersects it. Let's label these cover-lines by $C_1, C_2, \ldots$.

▶ **Definition 15** (strip, top/bottom segments)**.** *The region of the plane between two consecutive cover-lines $C_\tau, C_{\tau+1}$ is called a **strip** and denoted by S_τ. We consider $C_\tau, C_{\tau+1}$ part of S_τ as well. The input line segments that are intersecting the top cover-line of S_τ (C_τ) are called **top segments,** and the segments covered by the bottom cover-line ($C_{\tau+1}$) are called **bottom segments** of the strip.*

We show the near-optimum solution guaranteed by Theorem 2 has more structural properties that will be defined later. Note that once we prove that theorem, it follows that if we restrict a solution to $h > 1$ many strips, then the shadow is bounded by $O(h/\varepsilon)$ as well.

For now, let us focus on an (arbitrary) strip S_τ and imagine we cut the plane along $C_\tau, C_{\tau+1}$ and look at the pieces of line segments of the instance left inside this strip, along with pieces of OPT inside S_τ. Each top segment is now a partial segment in S_τ that has one end on C_τ; and each bottom segment has one end on $C_{\tau+1}$. Let OPT_τ be the restriction of OPT to S_τ. For each leg of OPT that intersects C_τ or $C_{\tau+1}$, we add a dummy point at

the intersection(s) of that leg with C_τ and $C_{\tau+1}$ (so that the components of OPT_τ become consistent with our definition of legs). So OPT_τ can be seen as a collection of subpaths within S_τ (possibly along C_τ or $C_{\tau+1}$). Following the orientation of OPT, each subpath of OPT_τ is formed by it intersecting with S_τ, traveling within S_τ (possibly along one of the cover-lines), and then exiting S_τ. Using the dummy points added, each path in OPT_τ is a subpath of OPT that is between two points on cover-lines (these are called the entry points of the path with the strip. A formal definition is provided below).

▶ **Definition 16** (entry points, loops, ladders). *For each subpath P_j of OPT_τ, let e_j and o_j be the first and last intersections of P_j with the interior of S_τ. Points e_j and o_j are called the **entry points** of P_j.*
*If both e_j and o_j lie on the same cover-line (either C_τ or $C_{\tau+1}$), then P_j is called a **loop**, otherwise it's called a **ladder**. If a subpath of OPT_τ enters S_τ at e_j on a cover-line and follows on that cover-line to point o_j and exits the strip, it is a special case of loop that we refer to as a **cover-line loop**.*

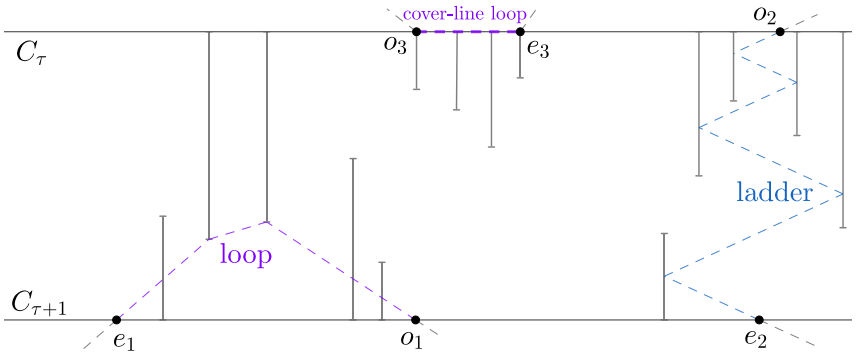


Figure 4 An example of loops and ladders in a strip S_τ (i.e. area between cover-lines $C_\tau, C_{\tau+1}$).

Since we're assuming $H > 3$ (see Theorem 4), we get that OPT is not limited to a single strip, and that it has to indeed enter and exit any given strip that it intersects with (i.e. there is no strip that OPT completely lies inside it). Note that if a path of OPT_τ is a cover-line loop, i.e. a section of the line C_τ or $C_{\tau+1}$, then the entry points of that path must be the two end-points of this section. In other words, if for a cover-line loop of OPT_τ the first point is e_j on (say) C_τ, and the last point is o_j on C_τ, then this subpath must be travelling straight from e_j to o_j without any change of direction. This is true because otherwise, that cover-line loop would have to go back and forth on some portion on a cover-line, which is only possible if it's self-intersecting; but this is against our assumption that OPT is not self-crossing. The two structures defined below (called a zig-zag and a sink) are the two configurations that can cause a large shadow.

▶ **Definition 17** (Zig-zag/Sink). *Consider any loop or ladder of OPT_τ, call it P. Let $\mathcal{R} = r_1, r_2, \ldots, r_t$ be the sequence of points of P that are reflection points (indexed by the order they're visited). Consider any maximal sub-sequence $r_j, r_{j+1}, \ldots, r_q$ of $\mathcal{R}$ (with $q \geq j+1$) such that all are ascending or all are descending reflections, and the segments containing them alternate between top and bottom segments; then the subpath P that starts at r_j and ends at r_q is called a **zig-zag**.*
*If $r_j, r_{j+1}, \ldots, r_q$ is a maximal sub-sequence of $\mathcal{R}$ that are all ascending or all descending, and all belong to top segments or all belong to bottom segments; then the subpath P that starts at r_j and ends at r_q is called a **sink** (see Figure 5).*

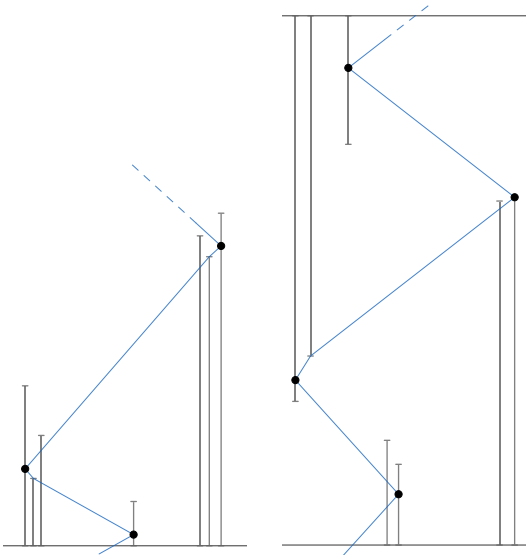


Figure 5 Examples of a sink (left) and a zig-zag (right). Bold dots indicate the reflection points.

Note that each zig-zag has at least two reflection points (or else it will be a sink). Also, using Corollary 14, the reflection points in a zig-zag or sink should alternate between left and right reflection. The next important lemma is used critically to show that very specific structures (namely zig-zags and sinks) are responsible for having large shadow along a ladder or loop in OPT_τ. Additionally, we can partition each ladder/loop into parts (subpaths), such that the shadow of the ladder/loop is equal to the maximum shadow among these parts, and each part is a path that consists of up to three sinks and/or zig-zags. So the shadow of a loop/ladder is within $O(1)$ of the maximum shadow of zig-zag/sinks along it.

▶ **Lemma 18.** *Consider any strip S_τ and any ladder or loop $P \in OPT_\tau$ within S_τ. Suppose the sequence of reflection points of P is $r_1, \ldots, r_q$. Then these reflection points can be partitioned into disjoint parts, say part i consists of reflection points $r_{a_i}, r_{a_i+1}, \ldots, r_{a_j}$, where the subpath of P from r_{a_i} to r_{a_j} is concatenation of up to three sections in the following order:* ***a)*** *A sink , followed by* ***b)*** *A zig-zag , followed by* ***c)*** *A sink, where any of these three sections can possibly be empty, and the last reflection of a section is common with the first reflection of the next section. Furthermore, for any vertical line Γ, there is at most one of these parts (of the partition) that intersects with it, i.e. the shadow of the ladder/loop is the maximum shadow among the parts plus 2.*

The proof of this lemma is rather involved (see [21] for details). To give an idea of the proof, we essentially show that for any loop/ladder in any strip, the vertical line at which the largest shadow for that loop/ladder happens, can intersect with at most two sinks and a zig-zag. So the shadow of a loop or ladder is within $O(1)$ of the maximum shadow of the zig-zags and sinks along itself. The following lemma is one of the main components of proof of Theorem 2.

▶ **Lemma 19.** *Consider OPT_τ for an arbitrary strip S_τ, and let opt_τ be the total cost of OPT_τ. Given any $\varepsilon > 0$, we can change OPT_τ to a solution of cost at most $(1 + O(\varepsilon)) \cdot opt_\tau$ where the shadow of each zig-zag and sink is at most $O(1/\varepsilon)$.*

The following corollary immediately follows from Lemmas 18 and 19:

▶ **Corollary 20.** *There is a $(1+\varepsilon)$-approximate solution in which all loops/ladders have shadow $O(1/\varepsilon)$.*

▶ **Definition 21.** *Let $\mathcal{R} = p_i, p_{i+1}, \ldots, p_q$ be any sequence of consecutive points in OPT such that p_i and p_q are reflection points. If none of p_j's $(i < j < q)$ in $\mathcal{R}$ is a tip of a segment, then $\mathcal{R}$ is called a* ***pure reflection sequence****.*

So each point in $\mathcal{R}$ is either a straight point or a pure reflection according to Lemma 9.

▶ **Lemma 22.** *Consider OPT_τ for an arbitrary strip S_τ and suppose the total length of legs of OPT_τ is opt_τ. Given $\varepsilon > 0$, we can change OPT_τ to a solution of cost at most $(1+\varepsilon)\cdot opt_\tau$ in which the size of any pure reflection sequence is bounded by $O(\frac{1}{\epsilon})$.*

Our next goal is to show that for any vertical line, it can intersect at most $O(1)$ many loops or ladders of OPT_τ in a strip S_τ. This together with the above corollary implies there is a $(1+\varepsilon)$-approximate solution where the shadow in each strip S_τ is $O(1/\varepsilon)$.

▶ **Definition 23.** *A collection of loops and or ladders are said to be* overlapping *with each other if there is a vertical line that intersects all of them.*

▶ **Lemma 24.** *Consider OPT_τ, the restriction of OPT to any strip S_τ. We can modify the solution (without increasing the shadow or the cost) such that there are at most $O(1)$ loops or ladders in OPT_τ that all are overlapping with each other.*

Assuming the correctness of main lemmas defined above (i.e. Lemmas 18, 19, 22, and 24), we can now prove Theorem 2.

Proof of Theorem 2. If the height of the bounding box is at most 3, simply use Theorem 4. Otherwise, consider any strip S_τ (to be more precise, S_τ can be any arbitrary strip of height 1 in the plane). Using Lemma 19, there is a solution $\mathcal{O}''$ of cost at most $(1+\varepsilon)\cdot \mathrm{opt}$ where the shadow of each sink and zig-zag is bounded by $O(1/\varepsilon)$. By Lemma 18, each loop or ladder in S_τ has a shadow that is at most 3 times the maximum shadow of a sink or zig-zag in it, plus two. So each loop or ladder has shadow $O(1/\varepsilon)$. Finally, Lemma 24 shows that there can be at most $O(1)$ overlapping loops or ladders in a strip. Thus, the overall shadow of $\mathcal{O}''$ in S_τ is bounded by $O(1/\varepsilon)$. Furthermore, we apply Lemma 22 on O'' to get a solution $\mathcal{O}'$. This new solution has the property that with an additional cost of factor $(1+O(\varepsilon))$ compared to $\mathcal{O}''$, the size of any pure reflection sequence is bounded by $O(1/\varepsilon)$. The total cost of $\mathcal{O}''$ is at most $(1+O(\varepsilon))\cdot \mathrm{opt}$ and the shadow is bounded by $O(1/\varepsilon)$. ◀

Note. Although having a bound on the length of pure reflection sequences is not in the statement of the theorem, we use this extra property crucially in designing our DP to find a near-optimum solution.

4 Main Algorithm and Reduction to Structured Bounded Height Instances

As mentioned in the introduction, we follow the paradigm of Arora [4] for designing a PTAS for classic Euclidean TSP with some modifications. We focus more on defining the modifications that we need to make to that algorithm. First, we describe the main algorithm and how it reduces the problem into a collection of instances with a constant-height bounding box. We show how those instances can be solved using another DP (referred to as the *inner DP*), and how we can combine the solutions for them using another DP (referred to as the

outer DP) to find a near-optimum solution of the original instance. Recall that in Section 2, we assumed the minimal bounding box of the instance has length L and height H and we defined $B = \max\{L, H-2\}$, and also we can assume that $B \leq \frac{n}{\varepsilon}$. We moved each line segment to be aligned with a grid point with side length $\frac{\epsilon B}{n^2}$ (at a loss of $(1+\varepsilon)$ at approximation). Now, we scale the grid (as well as the line segments of the instance) by a factor of $\rho = \frac{4n^2}{\epsilon B}$ so that each grid cell has size 4. We obtain an instance where each line segment has length ρ, all have even integer coordinates, any two segments are at least 4 units apart, and the bounding box has size $N = O(n^2/\varepsilon)$. Let this new instance be $\mathcal{I}$. Note that if we define cover-lines as before but with a spacing of ρ, all the arguments for the existence of a near-optimum solution with a bounded shadow in any strip (the area between two consecutive cover-lines) still hold. We will present a PTAS for this instance. It can be seen that this implies a PTAS for the original instance of the problem. From now on, we use OPT to refer to an optimum solution of instance $\mathcal{I}$, and opt to refer to its cost. Note that since the bounding box has side length N, then opt $\geq 2N$. Similar to Arora's approach, we do the hierarchical dissectioning of the instance into nested squares using random axis-parallel dissectioning lines, and put portals at these dissecting lines. We continue this dissectioning process until the distances between horizontal (and so vertical) dissecting lines is $h \cdot \rho$ for $h = \lceil 1/\varepsilon \rceil$. So at the leaf nodes of our recursive decomposition quad-tree, each square is $(h \cdot \rho) \times (h \cdot \rho)$, and the height of the decomposition is $\log(N/\rho h) = O(\log n)$ since $B \leq \frac{n}{\varepsilon}$. We choose vertical dissecting lines only at odd x-coordinates so no line segment of the instance will be on a vertical dissecting line.

We define our cover-lines C_τ based on these horizontal dissecting lines carefully. Consider the first (horizontal) dissecting line we choose, this will be a cover-line, and then moving in both up and down directions from this line, we draw horizontal lines that are ρ apart. These will be all the cover-lines. Label the cover-lines from the top to bottom by $C_1, C_2, \ldots, C_\sigma$ in that order. As before, the smallest index τ such that C_τ crosses a line segment is the cover-line that "covers" that line segment. We partition the cover-lines into h groups based on their indices: Group G_j contains all those cover-lines with index τ where $j = \tau \pmod h$. Let G_{j*} be the group of cover-lines that includes the first horizontal dissecting line, and hence all the other horizontal dissecting lines as well. The arguments for the case of unit-length line segments to show there is a near-optimum solution in which the shadow in each strip of height 1 is $O(1/\varepsilon)$ (Theorem 2), also imply the same for the scaled instance $\mathcal{I}$. Furthermore, if we consider h consecutive strips, i.e. the area between two consecutive cover-lines in the same group G_j, then there is a near-optimum solution that has shadow $O(h/\varepsilon) = O(1/\varepsilon^2)$. Our goal is to show that, at a $(1+\varepsilon)$-factor loss, we can simply drop the line segments that are intersecting the horizontal dissecting lines (i.e. all those intersecting cover-lines in G_{j^*}) with appropriate consideration of portals (to be described). Removing the line segments that cross the dissecting lines allows us to decompose the instance into "independent" sub-instances that interact only via portals. The details of how to remove these segments which leads to proof of Lemma 25 below are deferred to the full version [21].

Similar to Arora's scheme for TSP, for $m = O(\frac{1}{\varepsilon}\log(N/\rho h))$, we place portals at all 4 corners of a square in the decomposition, plus an additional $m-1$ equally distanced portals along each side (so a total of $4m$ portals on the perimeter of a square of the dissection). For simplicity, we assume m is a power of 2 and at least $\frac{4}{\varepsilon}\log(N/\rho h)$. We say a tour is *portal respecting* if it crosses between two squares in our decomposition only via portals of the squares. A tour is *r-light* if it crosses the portals on each side of a square of the dissection at most r times. For classic (point) TSP, it can be shown that there is a near-optimum solution that is portal respecting and r-light for $r = O(1/\varepsilon)$. Our goal is to show a similar statement,

except that we want the restriction of the tour to each "base" square of side length $O(h \cdot \rho)$ to have bounded (by $O(h/\varepsilon) = O(1/\varepsilon^2)$) shadow as well. We then show that we can find an optimum solution with a bounded shadow for the base cases using a DP. This will be our *inner DP*. We then show how the solutions of for the 4 sub-squares of a square in our decomposition can be combined into a solution for the bigger subproblem using the outer DP. We show that at a small loss in approximation (i.e. $O(\varepsilon \cdot \text{opt})$), we can drop all the line segments of input that are intersecting the horizontal dissecting lines (i.e. covered by a cover-line in group G_{j^*}), solve appropriate subproblems, and then extend the solutions to cover those dropped segments. This modification requires certain portals of each square in the decomposition to be visited in the solution for that square. We show there is a feasible solution that visits all the remaining segments as well as the "required" portals, of total cost at most $(1+\varepsilon) \cdot \text{opt}$, and that such a solution can be extended to a feasible solution visiting all the segments of the original instance (i.e. including the ones that we dropped) at an extra cost of $O(\varepsilon \cdot \text{opt})$.

▶ **Lemma 25.** *Given instance $\mathcal{I}$, there is another instance $\mathcal{I}'$ that is obtained by removing all the segments that are crossing cover-lines in G_{j^*} (i.e. intersecting horizontal dissecting lines), and instead some of the portals around (more precisely, the top and bottom sides of) each square of quad-tree dissection are required to be covered (visited); such that there is a solution for $\mathcal{I}'$ of cost at most $(1+O(\varepsilon)) \cdot opt$, and such a solution can be extended to a feasible solution of $\mathcal{I}$ of cost at most $(1+O(\varepsilon)) \cdot opt$. Furthermore, the shadow of the solution for $\mathcal{I}'$ between any two consecutive cover-lines in G_{j^*} is at most 4 more than the shadow of OPT between those two lines.*

The outer DP based on the quad-tree dissection is similar to the classic PTAS for Euclidean TSP. One can show that for $r = O(1/\varepsilon)$, there is a r-light portal respecting tour for $\mathcal{I}'$ with cost at most $(1+\varepsilon) \cdot \text{opt}'$ where opt$'$ is the cost of an optimum solution fo $\mathcal{I}'$. The DP will also "guess" in the recursion for each square, which portals are the "required" portals around it. The base case of this DP will be instances with bounding box of size $\rho \cdot h$. For such instances, we solve the problem using an inner DP. Informally, the inner DP is a nontrivial generalization of the DP for the classic (and textbook example) bitonic TSP in which the shadow is 2. In our case, the shadow is $O(1/\varepsilon^2)$. We defer the details of both the inner and outer DP to the full version [21].

References

1. Antonios Antoniadis, Krzysztof Fleszar, Ruben Hoeksma, and Kevin Schewior. A ptas for euclidean tsp with hyperplane neighborhoods. *ACM Trans. Algorithms*, 16(3), June 2020. doi:10.1145/3383466.
2. Antonios Antoniadis, Sándor Kisfaludi-Bak, Bundit Laekhanukit, and Daniel Vaz. On the approximability of the traveling salesman problem with line neighborhoods. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022*, volume 227 of *LIPIcs*, pages 10:1–10:21, Tórshavn, Faroe Islands, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.SWAT.2022.10.
3. Esther M. Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discret. Appl. Math.*, 55(3):197–218, 1994. doi:10.1016/0166-218X(94)90008-6.
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 Sayan Bandyapadhyay, Katie Clinch, William Lochet, Daniel Lokshtanov, Saket Saurabh, and Jie Xue. *PTASes for Euclidean TSP with Unit Disk and Unit Square Neighborhoods*, pages 2326–2356. SIAM, 2025. doi:10.1137/1.9781611978322.78.

6 MohammadHossein Bateni, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dániel Marx. A ptas for planar group steiner tree via spanner bootstrapping and prize collecting. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 570–583, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897549.

7 Hans L. Bodlaender, Corinne Feremans, Alexander Grigoriev, Eelko Penninkx, René Sitters, and Thomas Wolle. On the minimum corridor connection problem and other generalized geometric problems. *Comput. Geom.*, 42(9):939–951, 2009. doi:10.1016/J.COMGEO.2009.05.001.

8 T.-H. Hubert Chan and Khaled M. Elbassioni. A QPTAS for TSP with fat weakly disjoint neighborhoods in doubling metrics. *Discret. Comput. Geom.*, 46(4):704–723, 2011. doi:10.1007/S00454-011-9337-9.

9 T.-H. Hubert Chan, Shuguang Hu, and Shaofeng H.-C. Jiang. A PTAS for the steiner forest problem in doubling metrics. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016*, pages 810–819, Hyatt Regency, New Brunswick, New Jersey, USA, 2016. IEEE Computer Society. doi:10.1109/FOCS.2016.91.

10 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

11 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.

12 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003*, pages 473–482, San Diego, CA, USA, 2003. ACM. doi:10.1145/780542.780612.

13 Moshe Dror and James B. Orlin. Combinatorial optimization with explicit delineation of the ground set by a collection of subsets. *SIAM J. Discret. Math.*, 21(4):1019–1034, 2008. doi:10.1137/050636589.

14 Adrian Dumitrescu and Joseph S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *J. Algorithms*, 48(1):135–159, 2003. doi:10.1016/S0196-6774(03)00047-6.

15 Adrian Dumitrescu and Csaba D. Tóth. The traveling salesman problem for lines, balls, and planes. *ACM Trans. Algorithms*, 12(3), April 2016. doi:10.1145/2850418.

16 Khaled M. Elbassioni, Aleksei V. Fishkin, Nabil H. Mustafa, and René Sitters. Approximation algorithms for euclidean group TSP. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 1115–1126, Lisbon, Portugal, 2005. Springer. doi:10.1007/11523468_90.

17 Khaled M. Elbassioni, Aleksei V. Fishkin, and René Sitters. Approximation algorithms for the euclidean traveling salesman problem with discrete and continuous neighborhoods. *Int. J. Comput. Geom. Appl.*, 19(2):173–193, 2009. doi:10.1142/S0218195909002897.

18 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. doi:10.1016/J.JCSS.2004.04.011.

19 Corinne Feremans and Alexander Grigoriev. Approximation schemes for the generalized geometric problems with geographic clustering. In *(Informal) Proceedings of the 21st European Workshop on Computational Geometry, March 9-11, 2005*, pages 101–102, Eindhoven, The Netherlands, 2005. Technische Universiteit Eindhoven. URL: `http://www.win.tue.nl/EWCG2005/Proceedings/26.pdf`.

20 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998*, pages 253–259, San Francisco, California, USA, 1998. ACM/SIAM. URL: `http://dl.acm.org/citation.cfm?id=314613.314712`.

21 Benyamin Ghaseminia and Mohammad R. Salavatipour. A ptas for travelling salesman problem with neighbourhoods over parallel line segments of similar length, 2025. `arXiv:2504.02190`.

22 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003*, pages 585–594, San Diego, CA, USA, 2003. ACM. `doi:10.1145/780542.780628`.

23 Håkan Jonsson. The traveling salesman problem for lines in the plane. *Inf. Process. Lett.*, 82(3):137–142, 2002. `doi:10.1016/S0020-0190(01)00259-9`.

24 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved bound on the integrality gap of the subtour LP for TSP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, October 31 - November 3, 2022*, pages 832–843, Denver, CO, USA, 2022. IEEE. `doi:10.1109/FOCS54457.2022.00084`.

25 Cristian S. Mata and Joseph S. B. Mitchell. Approximation algorithms for geometric tour and network design problems (extended abstract). In Jack Snoeyink, editor, *Proceedings of the Eleventh Annual Symposium on Computational Geometry, June 5-12, 1995*, pages 360–369, Vancouver, B.C., Canada, 1995. ACM. `doi:10.1145/220279.220318`.

26 Joseph S. B. Mitchell. Geometric shortest paths and network optimization. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. North Holland / Elsevier, Amsterdam, NL, 2000. `doi:10.1016/B978-044482537-7/50016-4`.

27 Joseph S. B. Mitchell. Shortest paths and networks. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 607–641. Chapman and Hall/CRC, London, UK, 2004. `doi:10.1201/9781420035315.CH27`.

28 Joseph S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, January 7-9, 2007*, pages 11–18, New Orleans, Louisiana, USA, 2007. SIAM. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283385`.

29 Joseph S. B. Mitchell. A constant-factor approximation algorithm for TSP with pairwise-disjoint connected neighborhoods in the plane. In David G. Kirkpatrick and Joseph S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry, June 13-16, 2010*, pages 183–191, Snowbird, Utah, USA, 2010. ACM. `doi:10.1145/1810959.1810992`.

30 Joseph SB Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on computing*, 28(4):1298–1309, 1999. `doi:10.1137/S0097539796309764`.

31 Shmuel Safra and Oded Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003, 11th Annual European Symposium, September 16-19, 2003, Proceedings*, volume 2832 of *Lecture Notes in Computer Science*, pages 446–458, Budapest, Hungary, 2003. Springer. `doi:10.1007/978-3-540-39658-1_41`.

32 A. I. Serdyukov. O nekotorykh ekstremal'nykh obkhodakh v grafakh. *Upravlyaemye sistemy 17*, pages pp. 76–79, 1978. URL: `http://nas1.math.nsc.ru/aim/journals/us/us17/us17_007.pdf`.

The Fréchet Distance Unleashed: Approximating a Dog with a Frog

Sariel Har-Peled
Department of Computer Science, University of Illinois, Urbana, IL, USA

Benjamin Raichel
Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA

Eliot W. Robson
Department of Computer Science, University of Illinois, Urbana, IL, USA

Abstract

We show that a variant of the continuous Fréchet distance between polygonal curves can be computed using essentially the same algorithm used to solve the discrete version. The new variant is not necessarily monotone, but this shortcoming can be easily handled via refinement.

Combined with a Dijkstra/Prim type algorithm, this leads to a realization of the Fréchet distance (i.e., a morphing) that is locally optimal (aka locally correct), that is both easy to compute, and in practice, takes near linear time on many inputs. The new morphing has the property that the leash is always as short as possible. These matchings/morphings are more natural, and are better than the ones computed by standard algorithms – in particular, they handle noise more graciously. This should make the Fréchet distance more useful for real world applications.

We implemented the new algorithm, and various strategies to obtain fast practical performance. We performed extensive experiments with our new algorithm, and released publicly available (and easily installable and usable) `Julia` and `Python` packages. In particular, the `Julia` implementation, for computing the regular Fréchet distance, seems to be **significantly faster** than other currently available implementations. See Table 2.2. Our algorithms can be used to compute the almost-exact Fréchet distance between polygonal curves.

Implementations and numerous examples are available here: frechet.xyz.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Curve similarity, Fréchet distance

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.54

Related Version *Full Version*: `https://arxiv.org/abs/2407.03101` [14]

Supplementary Material *Audiovisual (Examples)*: `https://frechet.xyz`
Software (Python Implementation): `https://github.com/eliotwrobson/FrechetLib` [17]
archived at `swh:1:dir:cf434ca17d61d3d43c67e7bc40fd2eb2b507c9d9`
Software (Julia Implementation): `https://github.com/sarielhp/FrechetDist.jl` [13]
archived at `swh:1:dir:9600673d8c20bc49890fd55c71bd6e5c841dc7df`

Funding *Sariel Har-Peled*: Work on this paper was partially supported by a NSF AF award CCF-2317241.
Benjamin Raichel: Work on this paper was partially supported by NSF AF Award CCF-2311179.
Eliot W. Robson: Work on this paper was partially supported by a NSF AF award CCF-2317241.

© Sariel Har-Peled, Benjamin Raichel, and Eliot W. Robson;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

1.1 Definitions

Given two polygonal curves, their Fréchet distance is the length of a leash that a person needs, if they walk along one of the curves, while a dog connected by the leash walks along the other curve, assuming they synchronize their walks so as to minimize the length of this leash. (I.e. they walk so as to minimize their maximum distance apart during the walk.) Our approach is slightly different than the standard approach, and we define it carefully first.

1.1.1 Free space diagram and morphings

▶ **Definition 1.** *For a (directed) curve $\pi \subseteq \mathbb{R}^d$, its* **uniform parameterization** *is the bijection $\pi : [0, \|\pi\|] \to \pi$, where $\|\pi\|$ is the length of π, and for any $x \in [0, \|\pi\|]$, the point $\pi(x)$ is at distance x (along π) from the starting point of π.*

▶ **Definition 2.** *The free space diagram of two curves π and σ is the rectangle $R = R(\pi, \sigma) = [0, \|\pi\|] \times [0, \|\sigma\|]$. Specifically, for any point $(x, y) \in R$, we associate the* **elevation function** *$e(x, y) = \|\pi(x) - \sigma(y)\|$.*

The free space diagram R is partitioned into a non-uniform grid, where each cell corresponds to all leash lengths when a point lies on a fixed segment of one curve, and the other lies on a fixed segment of the other curve, see Figure 1.1. For a given value $\delta \geq 0$, the *sublevel set* of a real valued function consists of all inputs whose function value is $\leq \delta$. It is known that for any value $\delta \geq 0$, the sublevel set of the elevation function inside such a grid cell is a clipped ellipse.

▶ **Definition 3.** *A* **morphing**[1] *m between π and σ is a (not self-intersecting) curve $m \subseteq R(\pi, \sigma)$ with endpoints $(0, 0)$ and $(\|\pi\|, \|\sigma\|)$. The set of all morphings between π and σ is $\mathcal{M}_{\pi,\sigma}$. A morphing that is a segment inside each cell of the free space diagram that it visits, is* well behaved[2]*.*

Intuitively, a morphing is a reparameterization of the two curves, encoding a synchronized motion along the two curves. That is, for a morphing $m \in \mathcal{M}_{\pi,\sigma}$, and $t \in [0, \|m\|]$, this encodes the configuration, with a point $\pi\big(x(m(t))\big) \in \pi$ matched with $\sigma\big(y(m(t))\big) \in \sigma$. The ***elevation*** of this configuration is $e(t) = e(m(t)) = \big\|\pi\big(x(m(t)\big) - \sigma\big(y(m(t)\big)\big\|$.

1.1.2 Fréchet distance

▶ **Definition 4.** *The* **width** *of a morphing m between π and σ is $\omega(m) = \max_{t \in [0, \|m\|]} e(t)$. The* **Fréchet distance** *between the two curves π and σ is*

$$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) = \min_{m \in \mathcal{M}^{+}_{\pi,\sigma}} \omega(m),$$

where $\mathcal{M}^{+}_{\pi,\sigma} \subseteq \mathcal{M}_{\pi,\sigma}$ is the set of all x/y-monotone morphings.

[1] A morphing induces a natural homotopy between the two curves.
[2] All the morphings we deal with in this paper are well behaved.

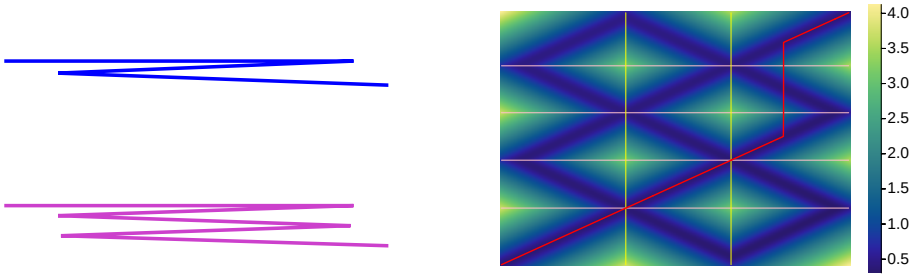
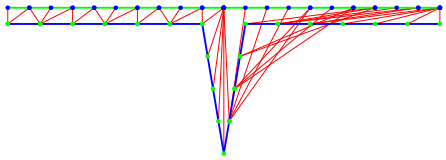
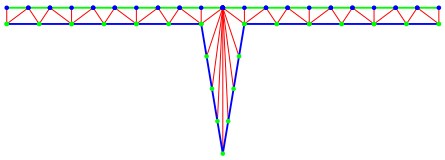


Figure 1.1 Two curves, their free space diagram (and the associated elevation function), and the optimal Fréchet morphing between the two curves encoded as an x/y-monotone curve. More illustrations and animations of this example are available here.



The classical (discrete) Fréchet morphing, caring only about the maximum leash length.



The retractable discrete Fréchet morphing, using the shortest leash possible at each point.

Figure 1.2 A comparison between the classical and retractable Fréchet distances. Observe that the morphing generated by the classical distance can be quite loose in many places. An animation of both morphings is available here.

Conceptually, the Fréchet distance is the problem of computing the minimum bottleneck matching between two curves, respecting the order and continuity of the curves.[3] Alternatively, it is an L_∞-norm type measure of the similarity between two curves. It thus suffers from sensitivity to outliers. Furthermore, even if only a small portion of the morphing requires a long leash, the measure, and the algorithms computing it, may use this long leash in large portions of the walk, generating a matching that is loose in many places, see Figure 1.2.

Observe that the Fréchet distance is the minimum value such that the sublevel set of the elevation function has an x/y-monotone path from $(0,0)$ to $(\|\pi\|, \|\sigma\|)$ in R.

1.2 Background

Alt and Godau [1] presented a rather involved $O(n^2 \log n)$ time algorithm to compute the Fréchet distance using parametric search. The parametric search can be removed by using randomization, giving a simpler algorithm as shown in [12]. Buchin *et al.* [7] presented an alternative algorithm for computing the Fréchet distance that replaces the decision procedure by using a data-structure to maintain appropriate lower envelopes. Despite some simplifications, all these algorithms are somewhat involved.

Unfortunately, it is believed this problem requires quadratic time in the worst case, although a logarithmic speedup is possible; see [5] and references therein. The quadratic time can be improved for realistic inputs by assuming that the input is "nice", and introducing approximation, but the resulting algorithms are still not simple [9].

[3] Formally, since the reparameterization is not one-to-one, this is not quite a matching. One can restrict to using only such bijections, with no adverse effects, but it adds a level of tediousness, which we avoid for the sake of simplicity of exposition.

1.2.1 Variants of the Fréchet distance

Weak Fréchet distance. The ***weak Fréchet distance*** allows morphings where the agents are allowed to go back and traverse portions of the curve visited previously (i.e., the morphing does not have to be x/y-monotone). Since the weak Fréchet distance allows considering more parameterizations, it is potentially smaller, and we have that $\mathsf{d}^w_{\mathcal{F}}(\pi, \sigma) \leq \mathsf{d}_{\mathcal{F}}(\pi, \sigma)$, for any two curves π, σ. Elegantly, Alt and Godau [1] showed the weak Fréchet distance can be reduced to computing the minimum spanning tree of an appropriate graph. Unfortunately, there does not seem to be a natural way to overcome this non-monotonicity (and thus get the "strong" version).

Discrete Fréchet distance. The complexity of these algorithms, together with sensitivity of the Fréchet distance to noise, lead to using "easier" related measures, such as the discrete version of the problem, and dynamic time-warping (discussed below). In the discrete version, you are given two sequences of points $p_1, \ldots, p_n$, and $q_1, \ldots, q_m$, and the purpose is for two "frogs" starting at p_1 and q_1, respectively, to jump through the points in the sequence until reaching p_n, q_m, respectively, while minimizing the maximum distance between the two frogs during this traversal. At each step, only one of the frogs can jump from its current location to the next point in its sequence (no jumping back). Computing the optimal distance under this measure can be done by dynamic-programming, similar to the standard approach to edit-distance. Indeed, the configuration space here is the grid $H = [\![n]\!] \times [\![m]\!]$, where $[\![n]\!] = \{1, \ldots, n\}$.

To use the discrete version in the continuous case, one sprinkles enough points along both input curves, and then solves the discrete version of the problem. Beyond the error this introduces, to get a distance that is close to the standard Fréchet distance, one has to sample the two curves quite densely in some cases.

For the (monotone) discrete Fréchet distance the induced graph on the grid H is a DAG, and the task of computing the Fréchet distance is to find a minimum bottleneck path from $(1, 1)$ to (n, m), where the weights are on the vertices. Here, the weight on the vertex (i, j) is the distance $\|p_i - q_j\|$. In particular, a Fréchet morphing is an x/y-monotone path in H from $(1, 1)$ to (n, m). The standard algorithm to do this traverses the grid, say, by increasing rows i, and in each row by increasing column j, such that the value at (i, j) is the maximum of the length of the leash of this configuration, together with the minimum solution for $(i - 1, j)$ and $(i, j - 1)$. This algorithm leads to a straightforward, $O(nm)$ time algorithm for the discrete Fréchet distance. However, the morphing computed might be inferior, see Figure 1.2 for such a bad example.

Retractable Fréchet. For simplicity, assume that the pairwise distances between all pairs of points in the two sequences are unique. We would like to imagine that we have a retractable leash, that can become shorter at times, and the leash "resists" being longer than necessary. It is thus natural to ask for a morphing where the leash as short as possible at any point in time.

Informally, the optimal *retractable Fréchet morphing* between the two sequences includes the bottleneck configuration, realizing the Fréchet distance, in the middle of its path, and the two subpaths from the endpoints to this configuration have to be also recursively optimal. This is formally defined and described in the Full version of the paper [14]. This concept was introduced by Buchin *et al.* [6]. Interestingly, they show that the discrete version can be computed in $O(nm)$ time, but unfortunately, the algorithm is quite complicated. They also show that the continuous retractable Fréchet can be computed in $O(n^3 \log n)$ time.

Buchin *et al.* [6] refers to this Fréchet distance as *locally correct*, but we prefer the *retractable* labeling. The term "retractable Fréchet" was used by Buchin *et al.* [7], but in a different (and not formally defined) context than ours.

(Continuous) Dynamic Time Warping. One way to get less sensitivity for noise is to compute the total area "swept" by the leash as the walk is being performed. In the discrete case, we simply add up the lengths of the leashes during the configurations in the walk. There is also work on extending this to the continuous setting [15, 2]. For the continuous case this intuitively boils down to computing (or approximating) an integral along the morphing. See full version of the paper [14] for more details.

1.2.2 Critical events

The standard algorithm for computing the Fréchet distance works by performing a "binary" search for the Fréchet distance. Given a candidate distance, it constructs a "parametric" diagram that is a grid, where inside each grid cell the feasible region is a clipped ellipse. The task is then to decide if there is an x/y-monotone path from the bottom left corner to the top right corner, which is easily doable. The critical values the search is done over are:

- **(I)** *Vertex-vertex events*: The distance between two vertices of the two curves,
- **(II)** *Vertex-edge events*: The distance between a vertex of one curve, and an edge of the other.
- **(III)** *Monotonicity events*: This is the minimum distance between a point on one edge e of the curves, and (maximum distance to) two vertices u, v of the other curve. Specifically, it is realized by the point on e with equal distance to u and v.

The first two types of events are easy to handle, but the monotonicity events are the bane of the algorithms for the Fréchet distance.

1.2.3 Algorithm engineering the Fréchet distance

Given the asymptotic complexity and involved implementations of the aforementioned algorithms, there has been substantial work on practical aspects of computing the Fréchet distance. In particular, in 2017, ACM SIGSPATIAL GIS Cup had a programming challenge to implement algorithms for computing the Fréchet distance. See [19] for details.

More recently, Bringmann *et al.* [3] presented an optimized implementation of the decider for the Fréchet distance. Somewhat informally, Bringmann *et al.* [3] builds a decider for the Fréchet distance using a *kd*-tree over the free space diagram, keeping track of the reachable regions on the boundary of each cell, refining cells by continuing down the (virtual) *kd*-tree if needed.

1.3 Our results

1.3.1 Result I: A new algorithm for retractable discrete Fréchet

We observe that a natural approach to compute the retractable Fréchet morphing is to modify Dijkstra's/Prim's algorithm so that it solves the minimum bottleneck path problem. This observation leads to a simpler (but log factor slower) algorithm for computing it. The only modification of Dijkstra necessary is that one always handles the cheapest edge coming out of the current cut induced by the set of vertices already handled. (In the discrete Fréchet case, the weights are on the vertices, but this is a minor issue.) This modified version of

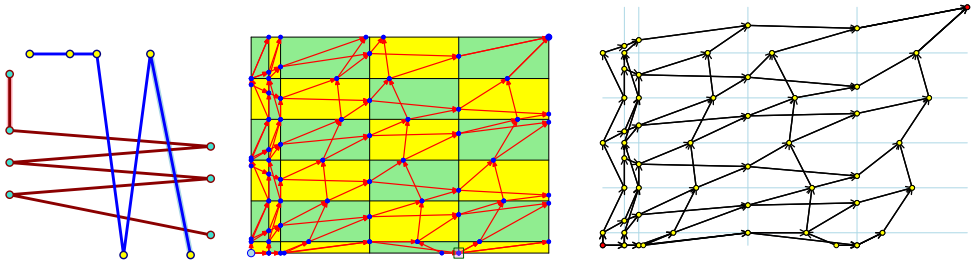


Figure 1.3 Two curves, and their associated VE-Fréchet graph. Note, that every internal edge of the grid contains a portal (i.e., vertex of the graph), but in many cases the portals of two edges are co-located on their common vertex (see the squared marked vertex in the middle figure) For the edges adjacent to the starting and ending corners we set their portals to lie at the corners themselves. See here for more info about a similar example.

Dijkstra is well known, but we include, for the sake of completeness, the proof showing that it indeed computes the recursively optimal path, which is also a retractable Fréchet morphing between the two sequences. This immediately leads to better and more natural morphings, see Figure 1.2.

Maybe more importantly, in practice, one does not need to explore the whole space of nm configurations (since we are in the discrete case, a configuration $(i, j) \in [\![n]\!] \times [\![m]\!]$ encodes the matching of p_i with q_j), as the algorithm can stop as soon as it arrives at the destination configuration (n, m). Informally, if the discrete Fréchet distance is "small" compared to the vast majority of pairwise distances (i.e., the two sequences are similar), then the algorithm only explores a small portion of the configuration space. Thus, this leads to an algorithm that is faster than the standard algorithm in many natural cases, while computing a significantly better output morphing.

1.3.2 Result II: A new distance and algorithm: VE-Fréchet

It is not clear how to extend the above to the continuous case. A natural first step is to consider the continuous Fréchet distance, where one restricts the solution inside each cell of the free space diagram to be a segment (which is already the case for the morphings computed by existing algorithms), but more importantly, insisting that the shape of this segment must be determined only locally, thus facilitating a greedy strategy compatible with the retractable approach. In practical terms, we throw away the (global) monotonicity events.

Traveling only through vertex-edge events. Because of the continuity and strict convexity of the elevation function, the function has a unique minimum on each edge of the free space diagram grid – geometrically, this is the minimum distance between a vertex of one curve, and an edge of the other curve (a *vertex-edge event*). We restrict our solution to enter and leave a cell only through these *portals* (which are easy to compute). The continuous configuration space now collapses to a discrete graph that is somewhat similar to the natural grid graph on $[\![n]\!] \times [\![m]\!]$. Indeed, a grid cell has four portals on its boundary edges. Specifically, there are directed edges from the portal on the bottom edge, to the portal on the top and right edges of the cell. Similarly, there are edges from the portal on the left edge, to the portals

lying on the top and right edges. As before, the values are on the portals, and our purpose is to compute the optimal bottleneck path in this grid-like graph. Examples of this graph are depicted in Figure 1.3, Figure 1.4 and Figure 1.5.

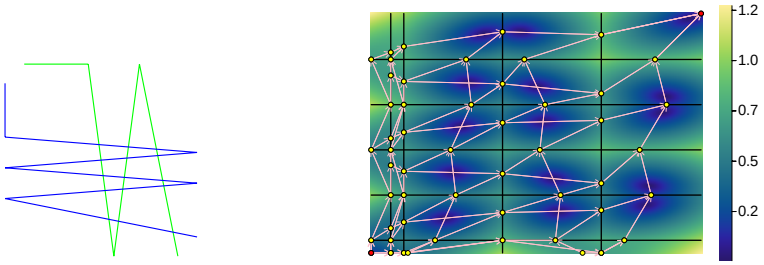


Figure 1.4 Left: Two curves. Right: Their elevation function, and the associated graph. See here for more info.

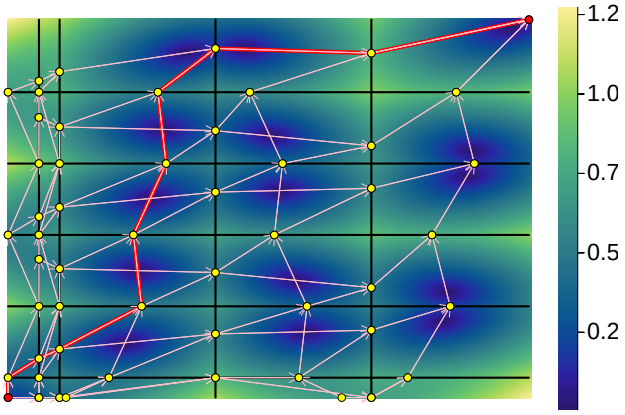


Figure 1.5 The VE-Fréchet morphing for the two curves from Figure 1.3. Note, that the solution is "slightly" not x/y-monotone. For animation of this morphing, follow the link.

▶ Remark 5. A somewhat similar idea was used by Munich and Perona [15], but they used it in the other direction – namely, in defining a better CDTW distance for two discrete sequences. However, this idea was already present (implicitly) in the original work of Alt and Godau [1] – indeed, their algorithm for the Weak Fréchet distance uses only the Vertex-Edge events (i.e., edges in the free space diagram). This boils down to solving the bottleneck shortest path problem in an undirected graph. In this case, this problem can be solved by computing the minimum spanning tree (e.g., by Prim's algorithm, which is a variant of Dijkstra's algorithm), as Alt and Godau do. For the directed case, one needs to use a variant of Dijkstra's algorithm [11] – see full version of the paper [14]. See also Buchin *et al.* [4] who also used a similar idea.

We can now run the retractable bottleneck shortest-path algorithm (i.e., the variant of Dijkstra described above) on this implicitly defined graph computing the vertices and edges of it, as they are being explored. For many natural inputs, this algorithm does not explore a large fraction of the configuration space, as it involves distances that are significantly larger than the maximum leash length needed. The algorithm seems to have near linear running time for many natural inputs. The ***VE-Fréchet*** morphing is the one induced by

the computed path in this graph. Unfortunately, the VE-Fréchet morphing might allow the agents to move backwards on an edge, but importantly, the motion across a vertex is monotone. Namely, the VE-Fréchet is monotone for vertices, but not necessarily monotone on the edges. A vertex is thus a *checkpoint* that once passed, cannot be crossed back.

1.3.3 Result III: New algorithm for the regular Fréchet distance

The natural question is how to use the (easily computable) VE-Fréchet morphing to compute the optimal (regular) continuous Fréchet distance. We next describe how this can be done in practice.

The hunt for a monotone morphing. We denote the VE-Fréchet distance between two curves π and σ by $\mathsf{d}^{ve}_{\mathcal{F}}(\pi,\sigma)$. Clearly, we have that $\mathsf{d}^{w}_{\mathcal{F}}(\pi,\sigma) \leq \mathsf{d}^{ve}_{\mathcal{F}}(\pi,\sigma) \leq \mathsf{d}_{\mathcal{F}}(\pi,\sigma)$.

One can of course turn any morphing into a monotone one by staying put instead of moving back. This is appealing for VE-Fréchet, as the corresponding VE morphing $\boldsymbol{m}$, say between two curves π and σ, never backtracks over vertices (only edges), so we already expect the error this introduces to be relatively small. Let $\boldsymbol{m}^+$ denote the monotone morphing resulting from this simple strategy. Observe that

$$\omega(\boldsymbol{m}) = \mathsf{d}^{ve}_{\mathcal{F}}(\pi,\sigma) \leq \mathsf{d}_{\mathcal{F}}(\pi,\sigma) \leq \omega(\boldsymbol{m}^+).$$

In particular, if $\omega(\boldsymbol{m}) = \omega(\boldsymbol{m}^+)$, then $\mathsf{d}_{\mathcal{F}}(\pi,\sigma)$ is realized by $\boldsymbol{m}^+$, and we have computed the Fréchet distance between π and σ.

A less aggressive approach is to introduce new vertices in the middle of the edges of π and σ as to enforce monotonicity. Indeed, clearly, if we refine both curves by repeatedly introducing vertices into them, the VE-Fréchet distance between the two curves converges to the Fréchet distance between the original curves, as introducing a vertex in the middle of an edge does not change the regular Fréchet distance, while preventing the VE-Fréchet morphing from backtracking over this point.

We refer to this process of adding vertices to the two curves as ***refinement***. (See Figure 1.6.) In practice, in many cases, one or two rounds of (carefully implemented) refinement are enough to isolate the maximum leash in the morphing from the non-monotonicity, and followed by the above brute-force monotonization leads to the (practically) optimal Fréchet distance. Even for pathological examples, after a few more rounds of refinement, this process computes the almost-exact Fréchet distance. That is, the computed lower bound, which is the VE-Fréchet distance, is equal to the width of the computed monotone morphing, which is the Fréchet distance.

▶ Remark 6 (Almost-exact: Floating point issues). As we are implementing our algorithm using floating point arithmetic, and the calculation of the optimal Fréchet distance involves distances, impreciseness is unavoidable. A slight improvement in preciseness can be achieved by using squared distances (and also slightly faster code) – but for simplicity we have not used this idea in our code. In particular, we take the somewhat pragmatic view, that an approximation to the optimal up to a factor of (say) 1.00001 can be considered as computing the "optimal" solution. We refer to such solutions as being ***almost-exact***.

Note that Fréchet morphings are somewhat less sensitive to numerical issues than other geometric problems – indeed, once a morphing is computed, one can compute its width directly.

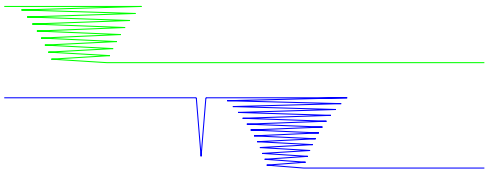


Figure 1.6 For these two curves, the solution involves an agent stopping at one point on one curve while the other agent traverses is zig-zag, and vice versa. The algorithm enforces monotonicity by refining the two curves by introducing new vertices. For the results, see here.

▶ Remark 7 (What if one wants the exact distance?). As pointed above, our algorithm computes the almost-exact Fréchet distance, and in practice there is no difference to the exact Fréchet distance (and in many cases they seem to coincide). Nevertheless, what if one insists on the exact Fréchet distance?

The refinement process can be modified to compute the monotonicity events on the regions on the curves where monotonicity is being violated. This would readily lead to an exact algorithm – from implementation point of view such a modification seems pointless, and we did not pursue it any further.

1.3.4 Result IV: Computing the Fréchet distance quickly for real inputs

The above leaves us with a natural strategy for computing the Fréchet distance between two given curves. Compute quickly, using simplification, a morphing between the two input curves, and maintain (using VE-Fréchet, for example) both upper and lower bounds on the true Fréchet distance. By carefully inspecting the morphing, (re)simplifying the curves in a way that is sensitive to their (local) Fréchet distance, and recomputing the above bounds, one can get an improved morphing. Repeat this process potentially several times till the upper and lower bounds meet, at which point the optimal Fréchet distance has been computed.

This seems somewhat overkill, but it enables us to compute (in practice) the almost-exact Fréchet distance between huge polygonal curves quickly.

1.3.5 Main contribution: Implementation in `Julia` and `Python`

We implemented the above algorithms as official packages in `Python` and `Julia`. The `Python` package is available at `https://github.com/eliotwrobson/FrechetLib`, and the `Julia` package is available at `https://github.com/sarielhp/FrechetDist.jl`. Animations and examples computed by the new algorithm are available at `https://frechet.xyz/`.

1.3.6 Additional results

Sweep distance. We demonstrate how one can convert our algorithm for computing VE-Fréchet to an algorithm that computes a variant of the CDTW distance, which we call the *sweep distance*. One can then use refinement to approximate the CDTW distance. One can also compute a lower bound on this quantity, and the two quantities converge. See the full version of the paper [14] for details.

Fast simplification. We show how to preprocess a curve π with n vertices, in $O(n \log n)$ time, such that given a query w, one can quickly extract a simplification of $\overline{\pi}$ of π, such that $\mathsf{d}_{\mathcal{F}}(\pi, \overline{\pi}) \leq w$. Importantly, the time to extract $\overline{\pi}$ is proportional to its size (i.e., the

extraction is output sensitive). More importantly, in practice, this works quite well – the size of $|V(\overline{\pi})|$ is reasonably close (by a constant factor) to the optimal approximation. Combined with greedy simplification, this yields a very good simplification result. See the full version of the paper [14] for details.

1.4 Significance

We demonstrate in this paper that a minor variant of the **continuous** Fréchet distance can be computed by a strikingly simple Dijkstra type algorithm. Furthermore, for many real world inputs it runs in near linear time (out of the box). Similarly, for many real world inputs, the computed morphing is monotone, thus realizing the (standard) Fréchet distance. More importantly, because of the *retractable* nature of the morphing computed, the matching computed is more natural, and can handle noise/outliers more gracefully than the matchings computed by the current algorithms for the Fréchet distance. Indeed, areas that are noise/outliers are isolated in the morphing to a small interval which can be easily identified and handled, see Figure 1.2. We believe that this makes the morphings computed by our new algorithms significantly better for real world applications than previous algorithms. There is also previous work on other variants of Fréchet distance that are more robust to uncertainty and outliers [10], and shortcutting [8].

We then show how to modify this algorithm to compute the (monotone) Fréchet distance (in cases when the morphing was not already monotone), and how to make it handle large inputs quickly via simplifications (handling all the technical difficulties this gives rise to).

Finally, we implemented our new algorithms in `Julia` and `Python`, and made them publicly available as standard packages, thus making the computation of the Fréchet distance accessible to a wider audience. Using such packages in `Python`/`Julia` is significantly easier than using any not pre-compiled code in `C++`.

Summary. The combination of simplification, new (and not so new) algorithmic ideas (such as retractablity via Dijkstra, and VE-Fréchet, among others), and careful implementation, leads to fast performance in practice beating other implementations. We discuss our implementation next – the algorithmic ideas described above are covered in detail in the appendices.

Full version. Due to space limitations, all the low-level details are omitted from this version. See the full version of the paper [14] (`https://arxiv.org/abs/2407.03101`) for these details.

2 Implementation and experiments

We include here some tables with information about the performance of our implementation. For details about the implementation, see the full version of the paper [14].

2.1 Discussion

Overall, it is clear that our `Julia` implementation is faster than other currently available implementations for computing the Fréchet distance. Beyond that, `Julia` seems like a great programming language to develop geometric algorithms – providing a combination of a safe, multi-threaded, high-level programming language with the performance of `C++`, without the pain involved in working in `C++`.

Table 2.1 The inputs tested and where they are taken from, `birds` referring to the stork migration dataset [18], the GeoLife dataset [20], and `Pigeons` referring to a dataset from [16].

Input	Description
1	`birds: 1787_1 / 1797_1`
2	`birds: 2307_3 / 2859_3`
3	`birds: 2322_2 / 1793_4`
4	`GeoLife 20080928160000 / 20081219114010`
5	`GeoLife 20090708221430 / 20090712044248`
6	`Pigeons RH887_1 / RH887_11`
7	`Pigeons C369_5 / C873_6`
8	`Pigeons C360_10 / C480_9`

Table 2.2 `Julia`/`C++` comparison, demonstrating that the `Julia` implementation is significantly faster. The `C++` implementation is from Bringmann *et al.* [3]. Running times are in seconds. `Julia` MT stands for the multi-threaded implementation – since multi-threading is easy in Julia, and the task at hand is easily parallizable, we tested this, but of course this should not be taken as a direct (or remotely fair) comparison to the `C++` implementation, and is provided for the reader amusement. In particular, the multi-threading done is pretty naive, and better performance should be possible by better fine-grained partition of the tasks. The tests were performed on a Linux system with 64GB memory with Intel `i7-11700` CPU, with a decent 16 threads CPU, but far from the fastest hardware currently available. The "# pairs" column is the number of pairs of curves that had their Fréchet distance compared during this test. The "avg # points" is the average number of vertices per curve in this input set – underlying the benefit of simplification for the GeoLife input set.

			Total RT in seconds			
Dataset	# pairs	avg # points	`C++`	`Julia`	`Julia` MT	`C++` / `Julia`
sigspatial	322,000	247.8	145	102	25	**1.42**
Characters	253,000	120.9	90	72	18	**1.25**
Geolife	322,000	1080.4	646	70	30	**9.22**
All tests combined			881	244	73	**3.61**

One optimization used by the `Julia` code, that should be generally useful, is the following. As a preprocessing step, precompute a hierarchy of simplified curves for each input curve. This improves the query process, but makes the preprocessing more expensive. Thus, storing such precomputed hierarchies might be a good idea if input curves are going to be used repeatedly for performing distance queries. We emphasize that the reported running times include this (light) preprocessing stage (interestingly, the SIGSPTIAL [19] competition allowed such preprocessing to not be included in the overall running time).

References

1 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995. `doi:10.1142/S0218195995000064`.

2 Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k, l)-medians clustering of trajectories using continuous dynamic time warping. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '20, pages 99–110, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3397536.3422245`.

Table 2.3 `Julia` performance on various inputs. All running times are is seconds. The # columns specify the number of vertices in each input. The single missing running time is for a case where the program ran out of memory. The quality of approximation is in the title of the column. The VER column is for running the VE-Fréchet (retractable) algorithm on the original input curves, which is much slower than the exact algorithm, which computes the exact Fréchet distance, but uses simplification internally for speed.

Input	P #	Q #	≈4	≈1.1	≈1.01	≈1.001	Exact	VER
1	10,406	11,821	0.106	0.256	4.164	16.639	0.476	27.743
2	16,324	14,725	0.114	0.428	0.985	4.762	3.522	19.734
3	22,316	4,613	0.151	0.651	1.207	3.262	0.882	7.138
4	56,730	91,743	0.578	1.207	4.215	28.109	4.259	--
5	6,103	9,593	0.043	0.040	0.305	0.519	0.484	5.678
6	2,702	1,473	0.034	0.084	1.367	1.387	0.160	1.081
7	1,068	1,071	0.026	0.066	0.048	0.049	0.044	0.011
8	864	1,168	0.018	0.044	0.142	0.124	0.069	0.080

Table 2.4 Python performance in seconds on the inputs given in Table 2.1. The missing runtime is for a case where the test code ran out of memory.

Input	P #	Q #	≈4	≈1.1	≈1.01	Exact	VER
1	10,400	11,815	0.071	0.250	5.372	0.593	92.869
2	16,318	14,719	0.091	0.240	1.142	69.525	66.844
3	22,310	4,607	0.087	0.232	1.041	49.426	25.032
4	56,730	91,743	0.307	0.563	1.892	2.390	--
5	6,103	9,593	0.042	0.032	0.175	3.657	19.618
6	2,696	1,467	0.077	0.294	4.913	2.946	283.538
7	1,062	1,065	0.145	0.275	0.455	0.666	382.455
8	858	1,162	0.034	0.100	1.537	0.159	0.709

3 Karl Bringmann, Marvin Künnemann, and André Nusser. Walking the dog fast in practice: Algorithm engineering of the fréchet distance. *J. Comput. Geom.*, 12(1):70–108, 2021. doi:10.20382/JOCG.V12I1A4.

4 Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristán, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In Erik G. Hoel, Shawn D. Newsam, Siva Ravada, Roberto Tamassia, and Goce Trajcevski, editors, *Proc. 25th ACM SIGSPATIAL Int. Conf. Adv. Geog. Info. Sys., GIS*, pages 14:1–14:10. ACM, 2017. doi:10.1145/3139958.3139964.

5 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, 2017. doi:10.1007/s00454-017-9878-7.

6 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann. Locally correct fréchet matchings. *Comput. Geom. Theory Appl.*, 76:1–18, 2019. doi:10.1016/J.COMGEO.2018.09.002.

7 Kevin Buchin, Maike Buchin, Rolf van Leusden, Wouter Meulemans, and Wolfgang Mulzer. Computing the fréchet distance with a retractable leash. *Discret. Comput. Geom.*, 56(2):315–336, 2016. doi:10.1007/S00454-016-9800-8.

8 A. Driemel and S. Har-Peled. Jaywalking your dog – computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

9 A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete Comput. Geom.*, 48:94–127, 2012. doi:10.1007/s00454-012-9402-z.

10 Emily Fox, Amir Nayyeri, Jonathan James Perry, and Benjamin Raichel. Fréchet edit distance. In *40th International Symposium on Computational Geometry (SOCG)*, volume 293 of *LIPIcs*, pages 58:1–58:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.58.

11 Harold N. Gabow and Robert Endre Tarjan. Algorithms for two bottleneck optimization problems. *J. Algorithms*, 9(3):411–417, 1988. doi:10.1016/0196-6774(88)90031-4.

12 S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algo.*, 10(1):3:1–3:22, 2014. doi:10.1145/2532646.

13 Sariel Har-Peled. FrechetDist.jl. Software, version 2.0., swhId: swh:1:dir:9600673d8c20bc49890fd55c71bd6e5c841dc7df (visited on 2025-04-24). URL: https://github.com/sarielhp/FrechetDist.jl, doi:10.4230/artifacts.22985.

14 Sariel Har-Peled, Benjamin Raichel, and Eliot Wong Robson. The fréchet distance unleashed: Approximating a dog with a frog. *CoRR*, abs/2407.03101, 2024. doi:10.48550/arXiv.2407.03101.

15 Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pages 108–115. IEEE Computer Society, 1999. doi:10.1109/ICCV.1999.791205.

16 E Pollonara, T Guilford, M Rossi, VP Bingman, and A Gagliardo. Data from: Right hemisphere advantage in the development of route fidelity in homing pigeons, 2017. doi:10.5441/001/1.245kb7r6.

17 Eliot Robson. FrechetLib. Software, version v0.1.1., swhId: swh:1:dir:cf434ca17d61d3d43c67e7bc40fd2eb2b507c9d9 (visited on 2025-04-24). URL: https://github.com/eliotwrobson/FrechetLib, doi:10.4230/artifacts.22984.

18 S Rotics, M Kaatz, SF Turjeman, D Zurell, M Wikelski, N Sapir, U Eggers, W Fiedler, F Jeltsch, and R Nathan. Data from: Early arrival at breeding grounds: causes, costs and a trade-off with overwintering latitude, 2018. doi:10.5441/001/1.v8d24552.

19 Martin Werner and Dev Oliver. ACM SIGSPATIAL GIS cup 2017: range queries under fréchet distance. *ACM SIGSPATIAL Special*, 10(1):24–27, 2018. doi:10.1145/3231541.3231549.

20 Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset – User Guide*, geolife gps trajectories 1.1 edition, July 2011. URL: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.

A Practical Algorithm for Knot Factorisation

Alexander He
401 Mathematical Sciences, Oklahoma State University, Stillwater, OK, USA

Eric Sedgwick
School of Computing, DePaul University, Chicago, IL, USA

Jonathan Spreer
School of Mathematics and Statistics, The University of Sydney, Australia

Abstract

We present an algorithm for computing the prime factorisation of a knot, which is practical in the following sense: using Regina, we give an implementation that works well for inputs of reasonable size, including prime knots from the 19-crossing census. The main new ingredient in this work is an object that we call an "edge-ideal triangulation", which is what our algorithm uses to represent knots. As other applications, we give an alternative proof that prime knot recognition is in coNP, and present some new complexity results for triangulations. Beyond knots, our work showcases edge-ideal triangulations as a tool for potential applications in 3-manifold topology.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Geometric topology

Keywords and phrases Prime and composite knots, (crushing) normal surfaces, edge-ideal triangulations, co-NP certificate, triangulation complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.55

Related Version *Full Version*: `https://arxiv.org/abs/2504.03942` [18]

Supplementary Material *Software (Source Code)*: `https://github.com/AlexHe98/idealedge` [19] archived at `swh:1:dir:ebcbe72831246d6d99ed7ff98f3cc00940ed808c`

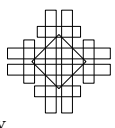
Funding *Alexander He*: Supported by an Australian Government Research Training Program Scholarship for part of this project.
Jonathan Spreer: Supported by the Australian Research Council under the Discovery Project scheme (grant number DP220102588).

Acknowledgements This work was inspired by discussions at the Dagstuhl workshop "Triangulations in Geometry and Topology" [7]. The authors would like to thank the participants for interesting conversations about knots and normal surfaces. Moreover, the authors thank Nathan Dunfield for suggesting a way to generate hard diagrams of composite knots on which to test our algorithm. We also thank the anonymous reviewers for their helpful comments.

1 Introduction

Computational problems in 3-manifold topology often exhibit a gap between theory and practice. For example, consider the Unknot Recognition problem, which takes a knot K as input, and asks whether K is equivalent to the unknot (see Section 2 for a review of all technical terminology used in this introduction). Although Regina [12] offers an Unknot Recognition algorithm that experimentally exhibits polynomial-time behaviour [15], in theory this algorithm is exponential-time in the worst case. There are other theoretical upper bounds on the complexity of Unknot Recognition, but these often use very different (and

© Alexander He, Eric Sedgwick, and Jonathan Spreer;
licensed under Creative Commons License CC-BY 4.0



in some cases, much more complicated) techniques: it is in both NP [17] and coNP [25, 26], and Lackenby has announced a quasipolynomial-time algorithm [27]. It remains unknown whether there is a (deterministic) polynomial-time algorithm for Unknot Recognition.

We are interested in algorithmically recognising other types of knots as well. Baldwin and Sivek give some complexity results for such other recognition problems in [4]: they show that a range of natural problems are in NP and/or coNP (some unconditionally, others conditional on the Generalised Riemann Hypothesis). Prior to the work in the present paper, none of these problems had algorithms with practical software implementations.

In this paper, we focus on the problem of deciding whether a knot is prime or composite, and in the composite case on computing its prime factorisation. This problem arises from the classical result of Schubert [30] that every knot has a unique prime factorisation. We study variants of this problem, in decreasing order of specificity:

▶ **Problem 1** (Knot Factorisation)**.**
INPUT: *A knot K.*
OUTPUT: *A collection of knots giving the prime factorisation of K.*

▶ **Problem 2** (k-Summands)**.**
INPUT: *A knot K, and a positive integer k.*
QUESTION: *Does the prime factorisation of K have at least k nontrivial summands?*

▶ **Problem 3** (Composite Knot Recognition)**.**
INPUT: *A knot K.*
QUESTION: *Is K a composite knot?*

Most commonly, the input knot K is presented as a knot diagram. However, a diagram is difficult to work with in many settings, and the first step of many algorithms is to convert the diagram into a more useful form. Often, the conversion is to a triangulation of the **knot exterior**: the 3-manifold obtained by deleting a small open neighbourhood of K from the ambient 3-sphere. This is fine for decision problems like Problems 2 and 3; however, for Knot Factorisation, if we require the output to be in the form of knot diagrams – rather than triangulations of knot exteriors – then we have the additional challenge of converting these triangulations to diagrams, which is itself a highly nontrivial problem [16].

Our approach is to encode knots as **edge-ideal triangulations** (**H-triangulations** in [3]). For a knot K, an edge-ideal triangulation is a pair $(\mathcal{T}, \ell)$, where $\mathcal{T}$ is a triangulation of the 3-sphere $\mathbb{S}^3$, and ℓ is an embedded loop (called an **ideal loop**) of edges (called **ideal edges**) in $\mathcal{T}$, such that ℓ is topologically equivalent to K. This idea extends to triangulations $\mathcal{T}$ of arbitrary compact 3-manifolds, and collections of loops in the 1-skeleton of $\mathcal{T}$.

Embedding a knot in the 1-skeleton of a triangulation is, on its own, not a new idea. For knots in 3-manifolds this is a common choice of combinatorial encoding; e.g., see [1, 26]. But even for knots in $\mathbb{S}^3$, this setting has been described and used in many publications with various backgrounds; e.g., see [3, 5, 6, 14, 20, 28] and the references therein.

The novelty of our approach is to view an ideal loop as representing a boundary component of a 3-manifold, loosely analogous to how an ideal triangulation (in the usual sense) has boundary components given by the ideal vertices. More precisely, given an edge-ideal triangulation $(\mathcal{T}, \ell)$ of a knot K, we consider a small open solid torus neighbourhood N of ℓ, and we view $(\mathcal{T}, \ell)$ as a combinatorial presentation of the 3-manifold $\mathcal{M}$ given by deleting N from $\mathcal{T}$. In this article, $\mathcal{T}$ is a 3-sphere and hence $\mathcal{M}$ is just the exterior of K.

To see why this perspective is useful, consider a normal surface S in $\mathcal{T}$. The intersection of S with the solid torus N is a disjoint union of meridian discs, with one such disc for each point in $S \cap \ell$. After deleting N, S becomes a surface that intersects the torus boundary of

$\mathcal{M}$ in curves that all have the same slope (in our case, the slope of the meridian of the knot). Hence, the edge-ideal triangulation carries more information than just the topology of $\mathcal{M}$: it also prescribes a slope along which normal surfaces must intersect the boundary of $\mathcal{M}$.

We use this setting to factorise knots in the 3-sphere. However, we believe other applications for edge-ideal triangulations exist. For instance, consider the problem of recognising Seifert fibred spaces (SFS) with nonempty boundary. This was shown to be in NP by Jackson [22], but the techniques are not well-suited to practical implementation. SFS with boundary fit well into the framework of edge-ideal triangulations: any such SFS $\mathcal{M}$ can be characterised using "vertical" essential annuli, which intersect components of $\partial\mathcal{M}$ in prescribed slopes. Exploring this idea is work in progress.

The utility of edge-ideal triangulations goes beyond the observation about boundary slopes of surfaces. It also allows us to apply the tool of **crushing** a normal surface, discussed in detail below. Crushing plays a central role in a number of practical 3-manifold algorithms provided by Regina [9, 10, 12], such as unknot recognition or 3-sphere recognition.

Our work extends this to give practical algorithms for Problems 1–3. In Section 5.3, we give a straightforward linear-time procedure to convert a diagram of a knot K into an edge-ideal triangulation of K, so our techniques apply equally to either method of encoding the input knot. However, converting an edge-ideal triangulation back into a knot diagram is sufficiently difficult to be beyond the scope of this paper. Thus, we settle for the following:

▶ **Theorem 4.** *There is an algorithm for* Knot Factorisation *that returns the prime factorisation of the input knot K as a collection of edge-ideal triangulations.*

The algorithm (see Algorithm 13) relies heavily on the tool of crushing normal surfaces. For our implementation, we use Regina's [12] longstanding crushing functionality, which is already known to be effective in practice. As it turns out, this practicability extends to our implementation: we are able to run our algorithm on prime knots from the census up to 19 crossings [11], as well as composite knots constructed by summing up to 8 knots sampled from the census (hence, knots with up to 152 crossings), including some "hard" diagrams of composite knots with up 70 crossings. See Section 5.4 for details on these experiments.

In addition to the practical implications, we also adapt our techniques to obtain theoretical upper bounds on computational complexity. Specifically, we prove the following:

▶ **Theorem 5.** *Problem 2 (k-Summands) is in* NP.

Since Problem 3 is a special case of Problem 2, this yields an alternative proof of one of Baldwin and Sivek's results [4, Theorem 7.3]: Composite Knot Recognition is in NP.

The operation of crushing a normal surface is crucial to both the practicality of Algorithm 13 and the proof of Theorem 5, so we now elaborate on its role in our work.

Inspired by unpublished work of Casson, crushing was first studied in detail by Jaco and Rubinstein [23], and later reformulated into a more implementation-friendly framework by Burton [10]. To date, one limitation of crushing is that the theory remains relatively poorly developed for surfaces other than 2-spheres and discs (though see [13, 21]). At first glance, this is an obstacle for the applications we have mentioned. For a knot K, testing whether K is composite entails finding a certain essential annulus (in the exterior of K). The same applies to recognising SFS with boundary.

By working with an edge-ideal triangulation $(\mathcal{T}, \ell)$, we circumvent this issue: the annuli show up in $\mathcal{T}$ as 2-spheres that intersect ℓ in two points. Thus, instead of crushing annuli, we are able to work in the better-understood setting of crushing 2-spheres. In Section 4, we develop the theory required to apply crushing in the presence of ideal loops.

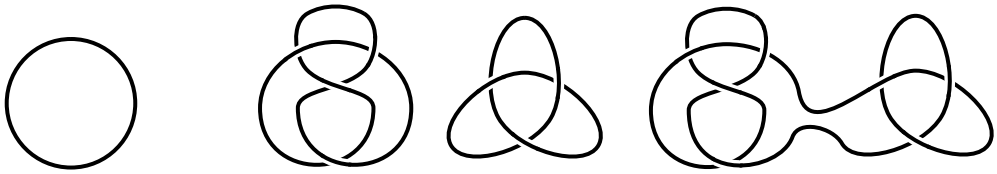
The reason crushing is useful for practical computations is that it never increases the number of tetrahedra in a triangulation. This allows us to factor a knot efficiently: we can repeatedly crush surfaces to progressively "decompose" the knot, whilst simultaneously keeping tight control on the amount of data needed to encode the resulting factorisation.

Another feature of our work is that we work exclusively with **quad vertex** 2-spheres in our edge-ideal triangulations; we show that suitable 2-spheres exist in Section 3. Theoretically, this is necessary to control how crushing affects the ideal loop; see Section 4. But even if this theoretical challenge were not present, working with quad (instead of standard) coordinates is often crucial for improving the practical performance of algorithms involving normal surfaces.

We combine the aforementioned ideas to formulate our main algorithm (Algorithm 13) in Section 5; we prove correctness of the algorithm, discuss some implementation details, and summarise our experimental results. In Section 6, we show how our techniques also give a proof of Theorem 5. Finally, we discuss further applications in relation to triangulation complexity in Section 7.

2 Background

A **knot** is an embedding of a circle $K : S^1 \to \mathbb{R}^3$ into 3-space. We sometimes embed knots into the 3-sphere $\mathbb{S}^3$, instead of $\mathbb{R}^3$; since $\mathbb{S}^3$ is the one-point compactification of $\mathbb{R}^3$, this has the benefit of making our topological space compact, but otherwise makes very little difference to the theory. Two knots are **equivalent**, or of the **same type**, if there is an ambient isotopy $\Phi : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ taking the image of one to the image of the other; intuitively, two knots are equivalent if one can be deformed into the other without passing the knot through itself. If a knot is equivalent to the trivial unknotted circle, we call it the **unknot**, or the **trivial knot**; otherwise, the knot is **nontrivial**.



(a) The unknot. **(b)** Two nontrivial knots (left and middle), and a connected sum of them (right).

Figure 1 Some examples of knot diagrams.

We typically represent a knot by a **diagram**, obtained by generically projecting the knot into a plane, while keeping information about which strand goes over and which goes under at **crossings** of the projection. Given two knots K_1 and K_2, we can construct a **connected sum** $K = K_1 \# K_2$ by cutting both K_1 and K_2 and gluing the ends together to form a new knot. A knot is **composite** if it is the connected sum of two nontrivial knots; otherwise, the knot is **prime**. Every nontrivial knot K has a **factorisation** $K = K_1 \# \ldots \# K_\ell$ into nontrivial prime knots K_i, $1 \leqslant i \leqslant \ell$, and this factorisation is unique up to re-ordering [30].

2.1 Triangulations of manifolds

A **triangulation** $\mathcal{T}$ (of **size** $|\mathcal{T}| := n$) is given by taking a finite set of disjoint tetrahedra $\Delta_1, \ldots, \Delta_n$, and gluing them together along their $4n$ boundary triangles in pairs. Let $\Phi_1, \ldots, \Phi_{2n}$ denote the gluings. The quotient space $\mathcal{T} = \{\Delta_1, \ldots, \Delta_n\}/\{\Phi_1, \ldots, \Phi_{2n}\}$ forms a closed 3-manifold, in which case we call $\mathcal{T}$ a **3-manifold triangulation**, if and only if: *(a)* no edge is identified to itself in reverse; and *(b)* for each vertex v of $\mathcal{T}$, the **link** of v – i.e., the frontier of a small regular neighbourhood of v – forms a 2-sphere.

As a result of the gluings, multiple lower-dimensional faces of $\{\Delta_1, \dots, \Delta_n\}$ become identified and we refer to the equivalence class of such faces as a single face of the triangulation $\mathcal{T}$. In fact, we often consider **1-vertex triangulations**, where all $4n$ vertices of $\{\Delta_1, \dots, \Delta_n\}$ are identified to a single vertex. Two 3-manifold triangulations are said to be **combinatorially isomorphic** if one of them can be turned into the other by a relabelling of its faces.

Let $\mathcal{M}$ be a 3-manifold with non-empty boundary $\partial\mathcal{M}$ a collection of tori. An **ideal triangulation** of $\mathcal{M}$ is a triangulation with one vertex per component of $\partial\mathcal{M}$ such that its vertex link triangulates the boundary component, and $\mathcal{M}$ is homeomorphic to the underlying set of the triangulation with a neighbourhood of the vertices removed.

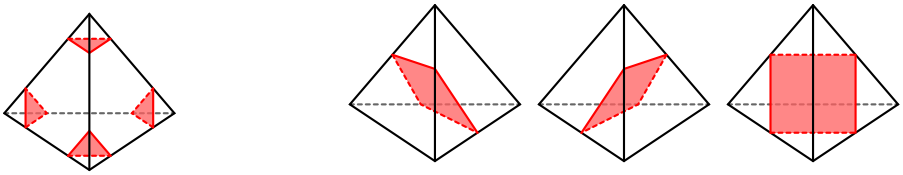
2.2 Normal surface theory and crushing

Consider a disc D properly embedded in a tetrahedron Δ so that its boundary γ is in *general position* – i.e., γ is disjoint from the vertices of Δ, and intersects the edges of Δ transversely. The **weight** of D in Δ is the number of points in which D intersects the edges of Δ.

A **normal surface** S in a triangulation $\mathcal{T}$ is a properly embedded surface such that:

- S is in general position: it is disjoint from the vertices of $\mathcal{T}$, and it intersects the edges and triangular faces of $\mathcal{T}$ transversely.
- For each tetrahedron Δ of $\mathcal{T}$, the intersection $\Delta \cap S$ consists of a (possibly empty) disjoint union of finitely many positive-weight discs, called **elementary discs**, such that the boundary of each elementary disc intersects each edge of Δ at most once.

The combinatorics of a normal surface remains unchanged under a **normal isotopy**: an ambient isotopy that preserves each vertex, edge, face and tetrahedron of the triangulation. Up to normal isotopy, each tetrahedron admits seven types of elementary discs: four **triangle** types and three **quadrilateral** (or **quad**) types (see Figure 2).



(a) The four triangle types. **(b)** The three quad types.

Figure 2 The seven types of elementary disc.

The link of a vertex v is an example of a normal surface. It is built entirely from elementary triangles (see Figure 3); conversely a normal surface consisting only of triangles must be a union of vertex-linking components. With this in mind, we call a normal surface **trivial** if it only has triangles, and **nontrivial** if it has at least one quad.

Normal surfaces can be described purely combinatorially by **coordinates**, counting the number of elementary disc types contained in them. These coordinates are subject to linear **matching equations** forcing the elementary discs to "match up" across triangular faces of $\mathcal{T}$. They must also have at most one quad type per tetrahedron, a condition called the **quad constraint**, to ensure that the surface is embedded. See [18, Section 2.3] for more details.

Using coordinates allows us to form sums of normal surfaces (subject to the quad constraints). More generally, it means that for an n-tetrahedron triangulation $\mathcal{T}$, all normal surface coordinate vectors lie inside a polyhedral cone in $\mathbb{R}^{7n}$, which can be described by providing one normal surface – called a **(standard) vertex** (normal) surface – per extremal ray of this cone. Vertex surfaces can be computed in finite time, and described efficiently [17].

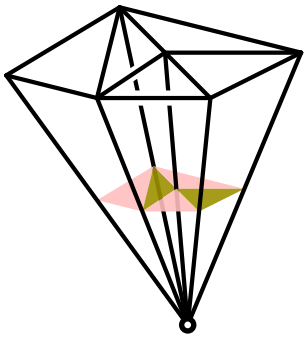


Figure 3 The link of a vertex is a normal surface built entirely from triangles.

Moreover, in all of this, it can be shown that it suffices to keep track of quadrilateral discs, and hence a polyhedral cone in $\mathbb{R}^{3n}$, leading to a (smaller) set of **quad vertex** (normal) surfaces that still describes the space of all normal surfaces in $\mathcal{T}$. See [18, Section 2.4] for more details, as well as some useful characterisations of vertex surfaces.

Given a normal surface S in a triangulation $\mathcal{T}$ of a 3-manifold $\mathcal{M}$, we can cut along S, shrink S to a point, and collapse cells that are not simplices until we recover a (possibly disconnected) triangulation $\mathcal{T}'$. This operation is called **crushing (S in $\mathcal{T}$)** and, in general, destroys much valuable information about $\mathcal{T}$ and $\mathcal{M}$. However, if done carefully, crushing provides an efficient way to "decompose" $\mathcal{T}$ into a new triangulation $\mathcal{T}'$ that represents the manifold $\mathcal{M}'$ obtained from $\mathcal{M}$ by cutting along S and then shrinking S to a point. Moreover, crushing has an extremely important algorithmic advantage: whenever S is nontrivial, we have $|\mathcal{T}'| < |\mathcal{T}|$. See [18, Section 2.5] for more details about crushing.

3 Existence of suitable quad vertex normal spheres

In this section we prove that, in any edge-ideal triangulation $(\mathcal{T}, \ell)$ of a composite knot, we can always find a normal 2-sphere that we can use to either reduce the size of our input, or to witness a (possibly trivial) decomposition of the knot. Crucially, we can ensure that these 2-spheres are quad vertex, which allows us to: *(a)* enumerate candidate 2-spheres efficiently, and *(b)* control the effect of crushing such 2-spheres.

Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a knot K. The ideal loop ℓ consists of a sequence of edges of $\mathcal{T}$, which we denote by $\ell_1, \ldots, \ell_m$. For any edge e of $\mathcal{T}$ and any normal surface S in $\mathcal{T}$, let $w_e(S)$ denote the **edge-weight** of S at e – i.e., the number of times S intersects e. Moreover, we define $w_\ell(S) = \sum_{i=1}^{m} w_{\ell_i}(S)$ – i.e., the number of times S intersects the loop ℓ. We have the following two lemmas (see [18, Section 3] for proofs):

▶ **Lemma 6.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a knot K. If K is a composite knot, then $\mathcal{T}$ contains a nontrivial normal 2-sphere S with $w_\ell(S) = 2$.*

▶ **Lemma 7.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a knot. Moreover, let Ω denote the set of all nontrivial normal 2-spheres S in $\mathcal{T}$ such that either $w_\ell(S) = 0$ or $w_\ell(S) = 2$. If Ω is nonempty, then there must be a quad vertex normal surface in Ω.*

▶ **Corollary 8.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a knot K. If K is composite, then $\mathcal{T}$ contains a quad vertex 2-sphere S such that either $w_\ell(S) = 0$ or $w_\ell(S) = 2$.*

Proof. To apply Lemma 7, we simply note that Ω is nonempty, by Lemma 6. ◀

▶ **Corollary 9.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a knot K. If K is composite, then $\mathcal{T}$ contains a standard vertex 2-sphere S such that either $w_\ell(S) = 0$ or $w_\ell(S) = 2$.*

Proof. This follows from [8, Lemma 4.5]: any quad vertex surface is also *standard* vertex. ◀

4 Tracking ideal edges under crushing

Consider an edge-ideal triangulation $(\mathcal{T}, \ell)$ and a quad vertex 2-sphere S intersecting ℓ twice. It is known that crushing S only mildly changes the topology of $\mathcal{T}$ [10]. In this section, we use a new characterisation of quad vertex 2-spheres to show that crushing S also only mildly changes the ideal loop ℓ. Moreover, using work of Agol, Hass, and Thurston [1, Sections 4 and 6], we can track how ℓ changes in time polynomial in $|\mathcal{T}|$; this is purely theoretical, and in practice a much simpler and more naive tracking method is efficient enough.

4.1 Orbits of segments

Let S be a normal surface in an edge-ideal triangulation $(\mathcal{T}, \ell)$. We say that S splits an (ideal) edge into **(ideal) segments**. After crushing S, each segment either gets destroyed or gets identified with a **surviving** segment that becomes an edge of the new triangulation. To determine how crushing S changes the ideal loop, we track how it causes ideal and surviving segments to be identified. Here, we outline how this can be done using the Agol-Hass-Thurston weighted orbit-counting algorithm from [1]; see [18, Section 4.1] for details.

We begin by defining the **induced orbit** $o(\sigma)$ of a segment σ to be a cell complex comprised of all the parts of $\mathcal{T}$ that get collapsed and identified with σ as a result of crushing S; this is not a subcomplex of $\mathcal{T}$, but rather a subcomplex of the cell decomposition obtained by subdividing $\mathcal{T}$ along S. We say that a segment is **type-k**, $k \in \{0, 1, 2\}$, if S intersects exactly k endpoints of the segment; all the segments in an induced orbit must have the same type, and hence we often say the orbit itself is type-k, for some k.

An induced orbit containing an ideal segment is called an **ideal orbit**, and is of particular interest to us. Indeed, our goal is to algorithmically determine which surviving segments belong to an ideal orbit. The challenge is that an induced orbit can be very large: the number of segments in a type-2 orbit can be of the same order as the size of the normal coordinates of S; this is exponential in $|\mathcal{T}|$ if S is quad vertex [17]. The Agol-Hass-Thurston machinery enables us to avoid the cost of explicitly tracing through such large orbits.

Abstractly, the Agol-Hass-Thurston weighted orbit-counting algorithm [1, Section 6] involves: k **pairings** between sequences of consecutive integers in $[1, N] := \{n \in \mathbb{Z} : 1 \leqslant n \leqslant N\}$ (i.e., bijections that either preserve or reverse the order between sequences); and a **weight function** $w : [1, N] \to \mathbb{Z}^d$. For each orbit in $[1, N]$ under the pairings, the algorithm computes the sum of the weights of the orbit's elements.

To apply this to our setting, we take each triangular face F of $\mathcal{T}$, and use F to define three pairings between sequences of consecutive segments, as illustrated in Figure 4. Two segments then belong to the same orbit under these pairings if and only if they belong to the same induced orbit. For the weights, we assign a coordinate to each surviving or ideal segment, and essentially use the weight vectors as indicator functions. The upshot is that after using the Agol-Hass-Thurston algorithm to compute the total weight of each orbit, we can read off: *(a)* which orbits contain an ideal segment, and *(b)* which surviving segments belong to each ideal orbit. Moreover, if S is a vertex surface, then plugging the bounds from [17] into the running time for the Agol-Hass-Thurston algorithm gives a polynomial in $|\mathcal{T}|$.

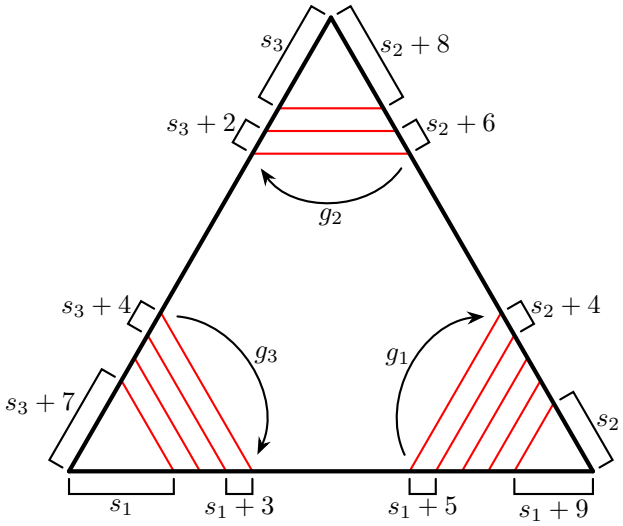


Figure 4 This triangle defines three pairings: $g_1 : [s_1+5, s_1+9] \to [s_2, s_2+4]$, $g_2 : [s_2+6, s_2+8] \to [s_3, s_3+2]$, and $g_3 : [s_3+4, s_3+7] \to [s_1, s_1+3]$. Adapted from Figure 4 of [1].

4.2 Quad vertex 2-spheres and trivial induced orbits

Call a type-1 induced orbit **trivial** if it forms a cone over a homotopically trivial subcomplex of S built from normal arcs and triangles. Call a type-2 induced orbit **trivial** if it forms a homotopically trivial product I-bundle. Our algorithm relies on the fact that quad vertex 2-spheres have trivial induced orbits. In fact, we can prove not just this, but also its converse:

▶ **Lemma 10.** *A normal 2-sphere is quad vertex if and only if every type-1 and every type-2 segment has a trivial induced orbit.*

Proof outline. Jaco and Tollefson gave a characterisation of vertex 2-spheres in terms of an object called an **exchange annulus** [24, Theorem 4.1]. As hinted in Figure 5, exchange annuli are closely related to type-2 orbits; type-1 orbits come into play because we are working with *quad* vertex 2-spheres. We prove the lemma by adapting the techniques from [24] to deal with both types of induced orbit. See [18, Sections 4.2 and 4.3] for details. ◀

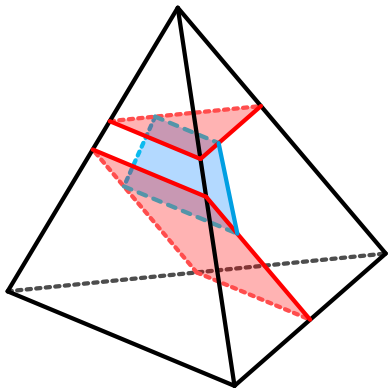


Figure 5 In each tetrahedron, an exchange annulus (blue) is always "close to" a type-2 segment.

4.3 The effect of crushing on ideal loops

Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a (possibly trivial) knot K. We now study the effect of crushing quad vertex 2-spheres that intersect the ideal loop ℓ in at most two points.

▶ **Lemma 11.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a (possibly trivial) knot K, and let S be a quad vertex 2-sphere with $w_\ell(S) = 0$.*

Crushing S yields a new triangulation $\mathcal{T}'$ consisting of either one or two 3-sphere components. Moreover, all the edges in ℓ are left untouched, and together form an embedded loop in a component $\mathcal{T}'_0$ of $\mathcal{T}'$, and $(\mathcal{T}'_0, \ell)$ forms an edge-ideal triangulation of K.

Proof outline. Let $\mathcal{T}'$ denote the triangulation given by crushing S. Because S and ℓ are disjoint, crushing S only changes parts of $\mathcal{T}$ that are sufficiently "far away" from ℓ to ensure that ℓ survives untouched inside a component of $\mathcal{T}'$; in particular, $\mathcal{T}'$ is nonempty. Moreover, since S is quad vertex, it is a consequence of Lemma 10 that $\mathcal{T}'$ has at most two components, each of which must be a 3-sphere. See [18, Lemma 21] for a complete proof. ◀

▶ **Lemma 12.** *Let $(\mathcal{T}, \ell)$ be an edge-ideal triangulation of a (possibly trivial) knot K, and let S be a quad vertex 2-sphere with $w_\ell(S) = 2$, so that S cuts ℓ into two ideal arcs α_1 and α_2, corresponding respectively to two (possibly trivial) knots K_1 and K_2 such that $K = K_1 \# K_2$.*

Crushing S yields a new triangulation $\mathcal{T}'$ consisting of either one or two 3-sphere components. Moreover, for each new knot K_i, $i = 1, 2$, we have either:

1. *all the segments in α_i survive the crushing to become an embedded loop ℓ'_i in a component $\mathcal{T}'_i$ of $\mathcal{T}'$, so that $(\mathcal{T}'_i, \ell'_i)$ forms an edge-ideal triangulation of K_i;*
2. *α_i consists of two type-1 segments, and these two segments merge to form a single non-loop edge ℓ'_i in a component $\mathcal{T}'_i$ of $\mathcal{T}'$; or*
3. *α_i is deleted.*

Cases 2 and 3 can only happen if K_i is trivial. If K_1 and K_2 both fall under Case 1, then the corresponding components $\mathcal{T}'_1$ and $\mathcal{T}'_2$ are distinct.

Proof outline. We give some intuition for the three cases. See [18, Lemma 22] for details.

Since S is a quad vertex 2-sphere, it is a consequence of Lemma 10 that crushing S gives a triangulation $\mathcal{T}'$ with at most two components, each of which is a 3-sphere. Moreover, since S is a separating 2-sphere, the two ideal arcs α_1 and α_2 must end up in different components of $\mathcal{T}'$. In the case where $\mathcal{T}'$ has fewer than two components, this simply means that the corresponding ideal arc α_i was deleted entirely (this is Case 3); it can be shown that this only occurs if the corresponding knot K_i is trivial.

It remains to consider what happens for an ideal arc α_i that survives. Since crushing involves collapsing S to a point, we know that the endpoints of α_i become identified in $\mathcal{T}'$. If these are the only points of α_i that get identified, then the result is an ideal loop corresponding to the knot K_i (this is Case 1). Otherwise, if crushing causes further identifications in α_i, then it can be shown that α_i falls under Case 2, and that K_i is trivial. ◀

To summarise, Lemmas 11 and 12 show that if we crush suitable quad vertex 2-spheres, then this changes the ideal loop in a topologically meaningful way. Moreover, as a consequence of the Agol-Hass-Thurston machinery described in Section 4.1, we can also efficiently keep track of how the ideal loop changes. These two pieces of machinery are, together, what allow us to give the algorithmic applications that follow in Sections 5 and 6.

5 Algorithm and implementation

The algorithm below takes a knot K as input, and produces the prime factorisation of K as a collection of edge-ideal triangulations. As stated, the input is a diagram of K, but by skipping Step 1 the algorithm can also be seen as taking an edge-ideal triangulation as input.

▶ **Algorithm 13.** *Given a diagram of a knot K, compute the prime factorisation of K (represented as edge-ideal triangulations) as follows:*

1. *Build an edge-ideal triangulation $(\mathcal{T}_K, \ell_K)$ of K; i.e., build a triangulation $\mathcal{T}_K$ of $\mathbb{S}^3$ such that K is embedded in $\mathcal{T}_K$ as an ideal loop ℓ_K. Section 5.3 describes how this can be done.*
2. *Create an empty list $\mathcal{P}$ (which will eventually contain all the prime summands). Also, create a list $\mathcal{L}$ of edge-ideal triangulations to process, initially containing just $(\mathcal{T}_K, \ell_K)$.*
3. *If $\mathcal{L}$ is nonempty, then pick and remove one edge-ideal triangulation $(\mathcal{T}, \ell)$ from $\mathcal{L}$ and **continue to Step 4**. Otherwise, **terminate with output** $\mathcal{P}$.*
4. *Search for a quad vertex normal 2-sphere S in $\mathcal{T}$ with either $w_\ell(S) = 0$ or $w_\ell(S) = 2$.*
   - *If S exists, crush it and keep track of the resulting ideal loops, as discussed in Section 4. Append all components of the resulting triangulation to $\mathcal{L}$ that contain an ideal loop. Discard the others.*
   - *If S does not exist, test whether ℓ forms a nontrivial knot. If so, add $(\mathcal{T}, \ell)$ to $\mathcal{P}$.*

   *In either case, **return to Step 3**.*

5.1 Correctness of the algorithm

▶ **Theorem 14.** *Algorithm 13 terminates and correctly factorises K into prime knots.*

Proof outline. To see that the algorithm terminates, note that only Steps 3 and 4 repeat. Together they replace (in $\mathcal{L}$) a triangulation $(\mathcal{T}, \ell)$ with (any of the nontrivial) triangulations obtained by crushing $(\mathcal{T}, \ell)$ along a normal sphere. As crushing strictly reduces the number of tetrahedra, the total number of tetrahedra contained in $\mathcal{L}$ decreases, and the number of iterations is at most the number of tetrahedra in the initial triangulation.

Each time before we execute Step 3 we verify that: *(a)* items in $\mathcal{L}$ are edge-ideal triangulations of (possibly trivial or composite) knots; *(b)* items in $\mathcal{P}$ are edge-ideal triangulations of nontrivial prime knots; and *(c)* K is given by composing all knots in $\mathcal{P} \cup \mathcal{L}$. This is true the first time we execute Step 3. For every other non-final iteration, we remove a summand from $\mathcal{L}$ that is processed by Step 4. If a suitable sphere is found, it is crushed and any nontrivial subsummands (by Lemmas 11 and 12) are returned to $\mathcal{L}$. If no sphere is found, then by Corollary 8 the summand is prime and moved to $\mathcal{P}$. In either case, *(a)–(c)* continue to hold.

In the final execution of Step 3, $\mathcal{L}$ is empty and thus the output list $\mathcal{P}$ contains the prime summands of K. See [18, Theorem 24] for details omitted from this proof outline. ◀

5.2 Optimisations and other implementation details

The most obvious optimisation is to simplify triangulations before attempting any other computations. For ideal triangulations of the knot complement, we can use the simplification tools available in Regina [12]. But for *edge-ideal* triangulations, we must instead use custom simplification tools to ensure that the ideal loop is preserved; see [18, Section 5.4] for details.

Furthermore, the algorithm has some potential bottlenecks that provide opportunity for optimisation. We found it effective to run multiple techniques in parallel to determine the best performing technique for any particular input.

1. We use different techniques for turning a knot into an edge-ideal triangulation. The most reliable one directly constructs an edge-ideal triangulation with $9n$ tetrahedra from an n-crossing input diagram, before simplifying it; it is illustrated in Figure 6. Other heuristics may be faster, or produce smaller triangulations. See Section 5.3 for details.

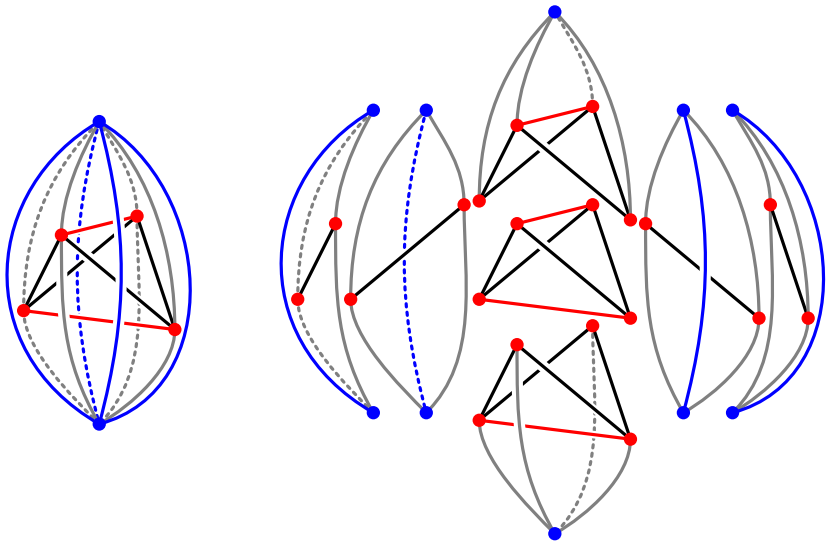


Figure 6 The 9 tetrahedron crossing gadget. Left: fully assembled gadget; red edges correspond to over- and under-crossing strands. Right: exploded view showing the individual tetrahedra.

2. In practice, the quad vertex surface enumeration in Step 4 usually terminates quickly. In the rare cases where the enumeration is not so quick, our remedy is to run the following in parallel with the enumeration: repeatedly randomise the triangulation, and attempt a (hopefully quick) enumeration on the new triangulation. See [18, Section 5.5] for details.
3. We can conclusively determine whether a knot is nontrivial using solid torus recognition, but this can be slow because it requires normal surface enumeration. In practice, running the algorithm on a nontrivial knot never seems to produce unknotted ideal loops. Thus, the main opportunity for optimisation here is to use fast heuristics that are usually good at certifying that a knot is nontrivial. See [18, Section 5.6] for details.

5.3 Building an edge-ideal triangulation from a knot

When the input knot K is presented as a knot diagram, our first task is to convert this into an edge-ideal triangulation of K. We use two techniques in parallel to achieve this task.

The first technique takes a knot diagram with n crossings, and guarantees to produce an edge-ideal triangulation of size $9n$ with an ideal loop of length $2n$. This is excellent in theory since the cost of the conversion is only linear, but as we discuss shortly 9 is a large enough constant to be a challenge in practice.

To describe the construction, view the input knot diagram as a 4-regular graph embedded in a 2-sphere S, with over- and under-crossing information recorded at each vertex of the graph. The dual graph decomposes S into 4-sided faces, with one crossing per face. Viewing the ambient 3-sphere as a suspension of S (i.e., two cones over S, glued together along S), we obtain a decomposition of the 3-sphere into 4-sided footballs: each football is a suspension of one of the 4-sided faces of S. We build an edge-ideal triangulation of the knot by triangulating each football with 9 tetrahedra, with two edges corresponding respectively to the over- and under-crossing strands inside that football; see Figure 6.

This construction is fast to perform, but the resulting triangulation is too large to give practical running times. Using combinatorial moves, our implementation reduces the length of the ideal loop to 1, and usually produces a significantly smaller triangulation. However, this simplification step is sufficiently slow to be a bottleneck for our implementation.

Our second technique is a heuristic construction that, though not guaranteed to terminate, typically produces small edge-ideal triangulations. Starting with a knot diagram, *(1)* triangulate the knot exterior, *(2)* modify this triangulation so that the meridian and longitude appear as boundary edges, and *(3)* attach a snapped 3-ball (a 1-tetrahedron gadget) to ensure that the meridian bounds a disc. Topologically, this realises a $\frac{1}{0}$ Dehn filling, and the longitude becomes the ideal loop. Regina [12] provides functionality for each of these steps, but the overall procedure is not guaranteed to terminate because we only have a *heuristic* method for *(2)*. See [18, Section 5.3] for more details on this second technique.

In practice, the second technique is often very fast, but sometimes prohibitively slow. Thus, running both techniques in parallel is an essential feature of our implementation.

5.4 Experimental results

In this section we demonstrate that Algorithm 13 is practical – and in some cases even quite efficient – for typical inputs. The code and the experimental data can be accessed from `https://github.com/AlexHe98/idealedge`.

On a laptop with an Intel Core i5-7200U processor, we ran Algorithm 13 over: *(a)* all 122 torus knots with 15 to 100 crossings; *(b)* all 376 satellite knots from the census [11] of prime knots with 15 to 19 crossings; *(c)* 100 hyperbolic knots with 15 or 16 crossings; and *(d)* a sample of composite knots with up to 8 prime summands and 152 crossings.

Algorithm 13 certified primeness of: all 122 torus knots in 310.02 seconds (2.54 seconds per knot); all 376 satellite knots in 404.66 seconds (1.08 seconds per knot); and all 100 hyperbolic knots in 138.48 seconds (1.38 seconds per knot). Note that for hyperbolic knots, the crushing part of our algorithm rarely has to run because heuristics for verifying hyperbolicity are often very efficient.

To obtain composite knots, we randomly sample census knots and form their connected sum. Thirty of these composite knots were sums of two prime knots, with diagrams deliberately constructed to "look prime"; Algorithm 13 factorised all of these in 124.82 seconds (4.16 seconds per knot). We also generated 430 composite knots with diagrams constructed in the standard way; timings for these are summarised in the table below.

# prime summands	# of samples	avg. running time per knot
2	100	4.71
3	100	10.95
4	100	24.06
5	100	33.06
6	10	47.75
7	10	70.21
8	10	88.68

Finally, we used the algorithm from [2] to randomly sample 1-vertex triangulations of 3-spheres, and look at the knot types of their edges. To maximise the variety of knots encountered, we started at a triangulation with no unknotted edges from [14]. We sampled 285 knots in triangulations of size 25–30, with running times of ≈ 0.23 seconds per knot.

6 NP certificate for composite knots

We now adapt our techniques to prove Theorem 5: Problem 2 is in NP, and hence Composite Knot Recognition is in NP. Recall that Problem 2 asks whether a given knot K has at least k summands in its prime factorisation.

Our certificate consists of a sequence of edge-ideal triangulations $(\mathcal{T}_0, L_0), \dots, (\mathcal{T}_m, L_m)$ and quad vertex normal surfaces S_i of $\mathcal{T}_i$, $i \in \{0, \dots, m-1\}$, such that: *(a)* $(\mathcal{T}_0, L_0)$ is obtained from K via the procedure outlined in Section 5.3; *(b)* every $(\mathcal{T}_{i+1}, L_{i+1})$ is obtained from $(\mathcal{T}_i, L_i)$ by crushing a suitable quad vertex normal sphere S_i of $\mathcal{T}_i$; and *(c)* for all $i \in \{0, \dots, m\}$, $(\mathcal{T}_i, L_i)$ is a union of edge-ideal triangulations of knots in $\mathbb{S}^3$, such that the input knot K is equal to the connected sum of the knots represented by the components of L_i. The certificate is completed by a set of k components of L_m together with a certificate that they form nontrivial knots in their respective components.

Since crushing reduces the size of a triangulation, the S_i are quad vertex normal surfaces, and knottedness lies in NP, the certificate is of polynomial size due to [17, 26].

The procedure from Section 5.3 to build an edge-ideal triangulation $(\mathcal{T}_0', L_0')$ of K, and the verification that $(\mathcal{T}_0', L_0')$ is combinatorially isomorphic to $(\mathcal{T}_0, L_0)$ are both polynomial time procedures (see, e.g., [29, Lemma 3.1 and Remark 3.2]). The same applies to checking for each $i \in \{0, \dots, m-1\}$, that either $w_{\ell_i}(S_i) = 0$ or $w_{\ell_i}(S_i) = 2$, and certifying that S_i is a quad vertex 2-sphere. For this latter task, we must show that type-1 and type-2 induced orbits are trivial, and that S_i is connected; see [18, Section 6.1] for details.

Crushing $\mathcal{T}_i$ along S_i to obtain triangulation $\mathcal{T}'$ can be done in polynomial time; see [18, Section 2.5]. The same is true for identifying the new collection of ideal loops L_{i+1}', following Section 4; and for verifying that $(\mathcal{T}_{i+1}', L_{i+1}')$ and $(\mathcal{T}_{i+1}, L_{i+1})$ are combinatorially isomorphic. Verifying knottedness of components of L_m is polynomial time due to [26].

To prove correctness of the above algorithm, it suffices to check that condition *(c)* above is valid for all $i \in \{0, \dots, m\}$. It then follows from the uniqueness of prime factorisation [30], that this is enough to verify that the input knot K has at least k summands.

First note that *(c)* is clearly satisfied for $(\mathcal{T}_0, L_0)$. To proceed by induction, assume that *(c)* is satisfied for some $(\mathcal{T}_i, L_i)$, $0 \leqslant i < m$. Let C_i denote the component of $\mathcal{T}_i$ containing S_i and with ideal loop ℓ_i. Lemmas 11 and 12 imply that crushing S_i replaces (C_i, ℓ_i) with either a new edge-ideal triangulation of the same knot, or a pair of edge-ideal triangulations giving a pair of knots obtained by a (possibly trivial) connected sum decomposition of ℓ_i. Either way, *(c)* is also satisfied for $(\mathcal{T}_{i+1}, L_{i+1})$. See [18, Section 6.3] for more details.

For the converse, suppose the verification algorithm succeeds. It follows that $(\mathcal{T}_0, L_0)$ must be an edge-ideal triangulation of K. Moreover, it follows that the S_i are quad vertex normal surfaces, and hence Lemmas 11 and 12 imply that *(c)* holds for all $i \in \{0, \dots, m\}$. Hence, it follows that $(\mathcal{T}_m, L_m)$ contains k nontrivial knot summands. By uniqueness of the prime factorisation of K, this verifies that K has at least k summands.

7 Triangulation complexity

In this section we leverage the fact that crushing a nontrivial normal surface reduces the size of its ambient triangulation to study **triangulation complexity** – i.e., the minimum number of tetrahedra necessary to triangulate a given manifold within a fixed class of triangulations.

Let $\mathcal{M}$ be the exterior of a composite knot K in $\mathbb{S}^3$, and let $\tilde{c}(\mathcal{M})$ be the triangulation complexity of $\mathcal{M}$ over the class of edge-ideal triangulations. Write $K = K_1 \# \cdots \# K_m$ for the unique prime factorisation of K. Starting with an edge-ideal triangulation $(\mathcal{T}, \ell)$ for K, our results from Section 4.3 ensure that we can iteratively find quad vertex 2-spheres to

crush at least $m-1$ times to obtain edge-ideal triangulations for each of the prime knots $K_1, \ldots, K_m$. If we only crush $m-1$ times, then every crush performs a nontrivial connected sum decomposition; thus, by Lemma 12, each crush increases the number of ideal edges by exactly 2. Due to the excess ideal edges, it can be shown (see [18, Lemma 25]) that we can crush an additional 2-sphere. So in fact, we can always crush at least m times, and each crush strictly decreases the overall number of tetrahedra, which establishes the bound

$$\tilde{c}(\mathbb{S}^3 \setminus K) \geqslant m + \sum_{i=1}^{m} \tilde{c}(\mathbb{S}^3 \setminus K_i). \tag{1}$$

▶ **Example 15.** There are two knot complements of ideal edge complexity 1: the unknot and the trefoil 3_1 (both are represented by edges in the unique 1-tetrahedron 1-vertex 3-sphere triangulation). The only knot complement of ideal-edge complexity 2 is 5_1. The knot complements of ideal-edge complexity 3 are 4_1, 7_1, 8_{19} and 10_{124}.

It follows from Equation (1) that the ideal-edge complexity of the complements of $3_1\#3_1$ and $3_1\#5_1$ are at least 4 and 5, respectively. However, such triangulations of $\mathbb{S}^3$ do not exist in the census. Instead, the smallest triangulations of $\mathbb{S}^3$ containing an ideal edge of type $3_1\#3_1$ and $3_1\#5_1$ have sizes 5 and 6, respectively.

Example 15 motivates the following question.

▶ **Question 16.** *Is there a composite knot that realises tightness in Equation* (1)*?*

We can also adapt our methods to obtain complexity results about ideal triangulations of knot complements; see [18, Section 7.2] for details.

References

1 I. Agol, J. Hass, and W. Thurston. The computational complexity of knot genus and spanning area. *Trans. Amer. Math. Soc.*, 358(9):3821–3850, 2006. doi:10.1090/S0002-9947-05-03919-X.

2 E. G. Altmann and J. Spreer. Sampling triangulations of manifolds using Monte Carlo methods. *Exp. Math.*, 0(0):1–15, 2025. doi:10.1080/10586458.2024.2433506.

3 F. B. Aribi, F. Guéritaud, and E. Piguet-Nakazawa. Geometric triangulations and the teichmüller tqft volume conjecture for twist knots. *Quantum Topol.*, 14(2):285–406, 2023.

4 J. A. Baldwin and S. Sivek. On the complexity of torus knot recognition. *Trans. Amer. Math. Soc.*, 371:3831–3855, 2019. doi:10.1090/tran/7394.

5 B. Benedetti. Discrete Morse theory for manifolds with boundary. *Trans. Amer. Math. Soc.*, 364(12):6631–6670, 2012. doi:10.1090/S0002-9947-2012-05614-5.

6 B. Benedetti and F. H. Lutz. Knots in collapsible and non-collapsible balls. *Electron. J. Combin.*, 20(3):Paper 31, 29, 2013. doi:10.37236/3319.

7 M. Buchin, J. Cardinal, A. de Mesmay, J. Spreer, and A. He. Triangulations in geometry and topology (dagstuhl seminar 24072). *Dagstuhl Reports*, 14(2):120–163, 2024. doi:10.4230/DAGREP.14.2.120.

8 B. A. Burton. Converting between quadrilateral and standard solution sets in normal surface theory. *Alg. Geom. Topol.*, 9:2121–2174, 2009. doi:10.2140/agt.2009.9.2121.

9 B. 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.

10 B. A. Burton. A new approach to crushing 3-manifold triangulations. *Discrete Comput. Geom.*, 52(1):116–139, 2014. doi:10.1007/s00454-014-9572-y.

11 B. 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SoCG.2020.25.

12 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2023. Version 7.3. URL: https://regina-normal.github.io.

13 B. A. Burton, T. de Paiva, A. He, and C. O. Y. Hui. Crushing surfaces of positive genus. arXiv:2403.11523, 2024. To appear in *Algebr. Geom. Topol.*.

14 B. A. Burton and A. He. Finding large counterexamples by selectively exploring the Pachner graph. 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 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SoCG.2023.21.

15 B. A. Burton and M. Özlen. A fast branching algorithm for unknot recognition with experimental polynomial-time behaviour. arXiv:1211.1079, 2012. To appear in *Math. Program.*

16 N. M. Dunfield, M. Obeidin, and C. G. Rudd. Computing a Link Diagram from Its Exterior. 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 37:1–37:24, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.37.

17 J. Hass, J. C. Lagarias, and N. Pippenger. The computational complexity of knot and link problems. *J. ACM*, 46(2):185–211, 1999. doi:10.1145/301970.301971.

18 A. He, E. Sedgwick, and J. Spreer. A practical algorithm for knot factorisation. arXiv:2504.03942, 2025.

19 Alexander He. AlexHe98/idealedge. Software, swhId: swh:1:dir:ebcbe72831246d6d99ed7ff98f3cc00940ed808c (visited on 2025-06-02). URL: https://github.com/AlexHe98/idealedge, doi:10.4230/artifacts.23286.

20 D. Ibarra, D. V. Mathews, J. S. Purcell, and J. Spreer. Triangulations of the 3-sphere with knotted edge. arXiv:2411.18938, 2024.

21 K. Ichihara, Y. Nishimura, and S. Tani. The computational complexity of classical knot recognition. *Journal of Knot Theory and Its Ramifications*, 32(11):2350069, 2023. doi:10.1142/S0218216523500694.

22 A. Jackson. Recognition of Seifert fibered spaces with boundary is in NP. *Math. Ann.*, 2024. doi:10.1007/s00208-024-02920-x.

23 W. Jaco and J. H. Rubinstein. 0-efficient triangulations of 3-manifolds. *J. Differential Geom.*, 65(1):61–168, 2003. doi:10.4310/jdg/1090503053.

24 W. Jaco and J. L. Tollefson. Algorithms for the complete decomposition of a closed 3-manifold. *Illinois J. Math.*, 39(3):358–406, 1995.

25 G. Kuperberg. Knottedness is in NP, modulo GRH. *Adv. Math.*, 256:493–506, 2014. doi:10.1016/j.aim.2014.01.007.

26 M. Lackenby. The efficient certification of knottedness and Thurston norm. *Adv. Math.*, 387, 2021. Article ID: 107796. doi:10.1016/j.aim.2021.107796.

27 M. Lackenby. Unknot recognition in quasi-polynomial time. https://www.maths.ox.ac.uk/node/38304, 2021.

28 W. B. R. Lickorish. Unshellable triangulations of spheres. *European J. Combin.*, 12(6):527–530, 1991. doi:10.1016/S0195-6698(13)80103-5.

29 S. Schleimer. Sphere recognition lies in NP. In *Low-dimensional and symplectic topology*, volume 82 of *Proc. Sympos. Pure Math.*, pages 183–213. Amer. Math. Soc., Providence, RI, 2011. doi:10.1090/pspum/082/2768660.

30 H. Schubert. *Die eindeutige Zerlegbarkeit eines Knotens in Primknoten.* Sitzungsberichte der Heidelberger Akademie der Wissenschaften. Springer, 1949. doi:10.1007/978-3-642-45813-2.

Sublinear Data Structures for Nearest Neighbor in Ultra High Dimensions

Martin G. Herold
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Danupon Nanongkai
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Joachim Spoerhase
University of Liverpool, UK

Nithin Varma
University of Cologne, Germany

Zihang Wu
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Geometric data structures have been extensively studied in the regime where the dimension is much smaller than the number of input points. But in many scenarios in Machine Learning, the dimension can be much higher than the number of points and can be so high that the data structure might be unable to read and store all coordinates of the input and query points.

Inspired by these scenarios and related studies in feature selection and explainable clustering, we initiate the study of geometric data structures in this ultra-high dimensional regime. Our focus is the *approximate nearest neighbor* problem.

In this problem, we are given a set of n points $C \subseteq \mathbb{R}^d$ and have to produce a *small* data structure that can *quickly* answer the following query: given $q \in \mathbb{R}^d$, return a point $c \in C$ that is approximately nearest to q, where the distance is under ℓ_1, ℓ_2, or other norms. Many groundbreaking $(1+\epsilon)$-approximation algorithms have recently been discovered for ℓ_1- and ℓ_2-norm distances in the regime where $d \ll n$.

The main question in this paper is: *Is there a data structure with sublinear ($o(nd)$) space and sublinear ($o(d)$) query time when $d \gg n$?* This question can be partially answered from the machine-learning literature: [1]

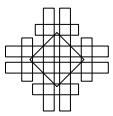
- For ℓ_1-norm distances, an $\tilde{O}(\log(n))$-approximation data structure with $\tilde{O}(n \log d)$ space and $O(n)$ query time can be obtained from explainable clustering techniques [Dasgupta et al. ICML'20; Makarychev and Shan ICML'21; Esfandiari, Mirrokni, and Narayanan SODA'22; Gamlath et al. NeurIPS'21; Charikar and Hu SODA'22].
- For ℓ_2-norm distances, a $(\sqrt{3}+\epsilon)$-approximation data structure with $\tilde{O}(n \log(d)/\mathrm{poly}(\epsilon))$ space and $\tilde{O}(n/\mathrm{poly}(\epsilon))$ query time can be obtained from feature selection techniques [Boutsidis, Drineas, and Mahoney NeurIPS'09; Boutsidis et al. IEEE Trans. Inf. Theory'15; Cohen et al. STOC'15].
- For ℓ_p-norm distances, a $O(n^{p-1}\log^2(n))$-approximation data structure with $O(n\log(n)+n\log(d))$ space and $O(n)$ query time can be obtained from the explainable clustering algorithms of [Gamlath et al. NeurIPS'21].

An important open problem is whether a $(1+\epsilon)$-approximation data structure exists. This is not known for any norm, even with higher (e.g. $\mathrm{poly}(n)\cdot o(d)$) space and query time.

In this paper, we answer this question affirmatively. We present $(1+\epsilon)$-approximation data structures with the following guarantees.

[1] All data structures discussed here are randomized and answer each query correctly with $\Omega(1)$ probability. The space complexity refers to the number of words and the time is in the word-RAM model. $\tilde{O}(f(n))$ hides $\mathrm{polylog}(f(n))$ factors

© Martin G. Herold, Danupon Nanongkai, Joachim Spoerhase, Nithin Varma, and Zihang Wu;
licensed under Creative Commons License CC-BY 4.0



41st International Symposium on Computational Geometry (SoCG 2025).
Editors: Oswin Aichholzer and Haitao Wang; Article No. 56; pp. 56:1–56:15

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

- For ℓ_1- and ℓ_2-norm distances: $\tilde{O}(n \log(d)/\text{poly}(\epsilon))$ space and $\tilde{O}(n/\text{poly}(\epsilon))$ query time. We show that these space and time bounds are tight up to $\text{poly}(\log n/\epsilon)$ factors.
- For ℓ_p-norm distances: $\tilde{O}(n^2 \log(d)(\log\log(n)/\epsilon)^p)$ space and $\tilde{O}\left(n(\log\log(n)/\epsilon)^p\right)$ query time.

Via simple reductions, our data structures imply sublinear-in-d data structures for some other geometric problems; e.g. approximate orthogonal range search (in the style of [Arya and Mount SoCG'95]), furthest neighbor, and give rise to a sublinear $O(1)$-approximate representation of k-median and k-means clustering. We hope that this paper inspires future work on sublinear geometric data structures.

2012 ACM Subject Classification Theory of computation → Nearest neighbor algorithms; Theory of computation → Data structures design and analysis; Theory of computation → Computational geometry; Mathematics of computing → Probabilistic algorithms

Keywords and phrases sublinear data structure, approximate nearest neighbor

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.56

Related Version *Full Version*: `https://arxiv.org/pdf/2503.03079`

Funding *Martin G. Herold*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 399223600.
Joachim Spoerhase: Part of this work was done when the author was a researcher at Max Planck Institute for Informatics & Saarland Informatics Campus, Saarbrücken, Germany, and at Aalto University, Finland. Partially supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557).
Nithin Varma: Part of this work was done when the author was a researcher at Max Planck Institute for Informatics & Saarland Informatics Campus, Saarbrücken, Germany.

1 Introduction

Nearest Neighbor. The *nearest neighbor* problem in geometric spaces is defined as follows: We are given a set of n points $C \subseteq \mathbb{R}^d$ to preprocess and then have to produce a data structure that can answer the following queries: given $q \in \mathbb{R}^d$, return the point c^* in C whose distance to q is smallest, where the distance can be induced by ℓ_1-, ℓ_2-, and other norms.

In the so-called *low-dimensional regime* [4] – when $d = o(\log n)$ and thus the data structure's space and time complexities can be *exponential* in d – classical data structures can solve this problem efficiently (e.g. [15, 25, 26, 8]). Recently, novel techniques such as locality-sensitive hashing and sketching (e.g. [22, 2, 3, 5, 23]) have led to a new wave of data structures in the *high-dimensional regime* that solve the problem approximately: for any $\alpha \geq 1$, a α-approximation data structure is the one that returns a point $c \in C$ whose distance to q is at most α times the distance between c^* and q. These recent data structures typically require space and time complexities that are linear or super linear in d, which is acceptable in most applications. In fact, many papers start with an assumption that $d = O(\log n)$ which can be assumed due to Johnson-Lindenstrauss lemma [24] at the cost of additional $\tilde{O}(d)$ space and query time.

Ultra-High Dimensions. But what if we cannot read all coordinates of the query q and cannot store all coordinates of all n points in C? There can be various reasons for such a situation to arise. For example, we may be unable to access the whole q due to the high-cost concern of obtaining the data (e.g. high-cost medical tests or genome sequencing) and privacy concerns (e.g. providing only the necessary medical or genetic information while protecting

unrelated sensitive data). More crucially, the size of the data, i.e., d might be prohibitively large to query all of its attributes (as is usually the case for genetic data) and, yet, n can be much smaller than d. For example, each point in C may be a center of a cluster among some small number of clusters obtained from prior training and we aim to classify a new data point $q \in \mathbb{R}^d$ by assigning it to the nearest representative. In this case, to accurately classify q without accessing all attributes of q, we need to exploit the fact that there are typically a small number of clusters. Additionally, classifying q based on a few attributes is crucial for *explainability* – when it is important for human users to know which factors play a role in the classifying decision.

Related Work and Existing Results. Scenarios like the above are a motivation behind at least two lines of research in machine learning. **(i)** In *Feature Selection*, the goal is to select a subset of relevant features of the input dataset. For example, Boutsidis, Drineas, and Mahoney in NeurIPS'09 [9] (also see [10]) show that given a set $S \subseteq \mathbb{R}^d$ and a parameter $k \ll d$, we can select roughly k coordinates such that we are required to only access these coordinates when computing k-means clustering, at the expense of a $(3+\epsilon)$ multiplicative approximation factor. The approximation factor was subsequently improved to $(1+\epsilon)$ in the STOC'15 paper by Cohen et al. [16]. **(ii)** In *Explainable Clustering*, given data that are already classified into k clusters via cluster centers, the goal is to construct a decision tree with some desired properties that can classify the input data points into clusters without too much loss in the clustering cost compared to the original clustering. For example, Makarychev and Shan [27] show that given a k-median clustering on ℓ_1 distance, one can construct such a decision tree that returns a clustering whose cost is $\tilde{O}(\log k)$ times the original cost. Since the number of accessed coordinates intuitively plays an important role in explainability, it is not surprising that one of the desired properties of the decision tree is that it never accesses more than $O(k)$ coordinates.

The feature selection and explainable clustering problems are *static problems*; i.e., the algorithms for these problems receive all of their inputs before returning the output. This is in contrast to the nearest neighbor problem which is a *data structure problem*. Nevertheless, given the connections between clustering and nearest neighbor problems, techniques from some of the previous works on these static problems can be extended to the nearest neighbor problem. (All data structures discussed here are randomized and answer each query correctly with $\Omega(1)$ probability. The space complexity refers to the number of words and the time is in the word-RAM model. $\tilde{O}(f(n))$ hides polylog$(f(n))$ factors.)

- For ℓ_1-norm distances, an $O(\log(n))$-approximation data structure with $O(n\log(n) + n\log d)$ space and $O(n)$ query time can be obtained by adapting the explainable clustering algorithms of [20, 29]. This is because these algorithms only require the center of each cluster, and not the remaining input data points. Weaker results can also be obtained from [18, 27, 31, 19].
- For ℓ_2-norm distances, a $(\sqrt{3}+\epsilon)$-approximation data structure with $\tilde{O}(n\log(d)/\epsilon^2)$ space and $O(n\log(n)/\epsilon^2)$ query time can be obtained by reducing to feature selection algorithms [9, 10, 16] (see Appendix F).[2] Weaker results for the case of ℓ_2-norm distances can also be obtained from explainable clustering algorithms of [28, 27, 18, 19, 11, 31].

[2] In Appendix F we also argue why a natural reduction to feature selection [16] fails to give a $(1+\epsilon)$-approximate nearest neighbor data structure.

- For ℓ_p-norm distances, an $O(n^{p-1}\log^2(n))$-approximation data structure with $O(n\log(n)+n\log(d))$ space and $O(n)$ query time can be obtained from the explainable clustering algorithms of [19].

In contrast to other regimes, where many efficient $(1+\epsilon)$-approximation data structures are known, no $(1+\epsilon)$-approximation data structure is known for the ultra-high dimensional regime for any norm, and even when we allow higher space and query time (e.g. $\text{poly}(n)\cdot o(d)$ space and query time).

Our Results. We first observe that every $(1+\epsilon)$-approximate nearest neighbor data structure requires $\Omega(n)$ space and must make $\Omega(n)$ queries (see Theorem G.1). Our main contributions are data structures with nearly matching space and query bounds in the ultra-high dimensional regime. Our data structures require polynomial preprocessing time, answer each query correctly with $1-\delta$ probability, and guarantee the following bounds.

- For ℓ_1-norm distances: $O(n\log^2(n/\delta)\log(d)/(\epsilon^5\delta^4))$ space and $O(n\log(n/\delta)/(\epsilon^3\delta^2))$ query time. See Theorem 4.2.
- For ℓ_2-norm distances: $O(n\log^2(n/\delta)\log(d)/(\epsilon^6\delta^4))$ space and $O(n\log(n/\delta)/(\epsilon^3\delta^2))$ query time. See Theorem B.3.
- For ℓ_p-norm distances: $O(n^2\log(n/\delta)\log(d)(\log\log n)^p/\epsilon^{p+2})$ space and $O(n\log(n/\delta)(\log\log n)^p/\epsilon^{p+2})$ query time. See Theorem D.1.

Our data structure for ℓ_1- and ℓ_2-norm distances uses essentially the same space and query time as the previously best result and provides a stronger approximation guarantee. For other norms, our data structure needs higher (but sublinear) space and gives strong approximation guarantees.

Connection to Explainable Clustering. A recent line of research investigates explainable clustering [18, 27, 31, 19, 20, 29]. The input is a set C of k cluster centers. The output is a (random) decision tree assigning any query point to a cluster represented by a label in $[k]$. The current best result [20] for explainable k-median clustering under ℓ_1-distances guarantees that for *any* set P of data points the clustering generated by the decision tree is in expectation within a factor $O(\log k)$ of the optimal assignment of P to the centers in C, that is, of the nearest-center assignment. The decision tree has k leaves. Therefore, it can be interpreted as $O(\log k)$-approximate representation of the clustering C with sublinear $\tilde{O}(k)$ space and query complexity. There are two aspects why this representation is considered explainable: (i) the decision tree is easy to interpret for humans (ii) it is compact as it has only sublinear $\tilde{O}(k)$ space and query complexity. When applied to a single query point these data structures constitute a $O(\log k)$-approximate nearest neighbor data structure in expectation.

Our techniques have implications in the converse direction. We argue (see Section C) that our techniques give a sublinear $\tilde{O}(k)$ space and query complexity data structure for an $O(1)$-approximate nearest neighbor in *expectation*. This implies a sublinear $O(1)$-approximate representation for k-median clustering improving upon the $O(\log k)$-approximation obtained via decision trees. Notice that while our data structure does not produce decision trees it does constitute a compact representation of the clustering. Our techniques also imply a sublinear $O(1)$-approximate representation of k-means clustering in Euclidean space improving upon the $O(\log^2 k)$-bound implied by explainable k-means clustering [28].

Perspective: Sublinear Data Structures. Geometric data structures are fundamental computational objects that have been extensively studied and used for decades. To the best of our knowledge, there was no prior result on geometric data structure problems in the ultra-high dimensional regime. Showing that it is possible to design sublinear geometric data structures is the main conceptual contribution of our paper.

As another example of where our techniques are applicable, consider the orthogonal range search problem. Here, we are given a set $C \subseteq \mathbb{R}^d$ to preprocess and need to answer the following query: Given q_1 and q_2 in $\mathbb{R}^d$, return all points $p \in C$ such that the i^{th} attribute of p has value between the i^{th} attributes of q_1 and q_2, for every i. This problem can be solved using range trees and k-d trees [7, 30, 12, 13] in the low-dimensional regime. For higher dimensions, some approximation algorithms have been proposed where some other points can be returned as long as they are not "far" from the query [6, 14, 17], i.e., the distance between the point returned and the range is within an ϵ fraction of the distance between q_1 and q_2 – we call such a range search as $(1+\epsilon)$-approximate range search. We approach this problem, as before, from a data-structure point of view. Using similar techniques, we can design a data structure of size $\tilde{O}(n)$ that processes n centers such that given any "valid" orthogonal range (q_1, q_2) (i.e., a range which contains at least one center point), queries only $\tilde{O}(n/\epsilon^2)$ coordinates of q_1 and q_2 to report all centers that are within the $(1+\epsilon)$-approximate range. The details of this can be found in Appendix E.

Besides orthogonal range search, simple reductions also lead to data structures for other proximity problems such as the furthest neighbor problem, where we can guarantee the same bounds as the nearest neighbor problem.

We hope that this paper inspires future work on sublinear geometric data structures.

Organization. Section 2 gives an overview of the main technical contributions of the paper. Section 4 describes our $(1+\epsilon)$-approximate nearest neighbor data structures with $\tilde{O}(n)$ space and query complexity for the ℓ_1- and ℓ_2-norm.

Appendix A contains the outline of a simple data structure with space $\tilde{O}(n^2)$ and query time $\tilde{O}(n)$ for ℓ_1-norm, with some simplifying assumptions as warmup. Appendix B contains missing proofs of Section 4 and extension to ℓ_2 case. Appendix C describes our expected constant-approximate nearest neighbor data structure, and shows the connection to explainable clustering. Appendix D describes our $(1+\epsilon)$-approximate nearest neighbor data structure with space $\tilde{O}(n^2)$ and query time $\tilde{O}(n)$ that works for general ℓ_p-norms. Appendix E shows the implications of our results for Range Search problem. Appendix F shows the connection between approximate nearest neighbor and feature selection for clustering. Appendix G shows $\Omega(n)$ space and query tight lower bound.

2 Technical Overview

Warmup: Quadratic Space via Pairwise Sampling. Let c be an input point, let q be an unknown query point, and let $\|c-q\|_1 = \sum_b |c^{(b)} - q^{(b)}|$ denote their ℓ_1 distance. First, observe that approximating this distance requires memorizing *all* coordinates of c because c and q may differ in only a single coordinate. Our strategy is to *compare* distances to multiple input points without knowing the (approximate) distances. For intuition, consider when we have two input points c_1 and c_2. Consider an extreme case when c_1 and c_2 differ only in one coordinate b. In this case, keeping only this coordinate suffices to tell for any query q if it is

closer to c_1 or c_2. To extend this intuition to when c_1 and c_2 differ in many coordinates, a natural idea is to sample each coordinate b independently with probability proportional to the difference $|c_1^{(b)} - c_2^{(b)}|$; more precisely, the probability that we sample coordinate b is[3]

$$p^{(b)} = \frac{|c_1^{(b)} - c_2^{(b)}|}{\|c_1 - c_2\|_1}.$$

We further scale the sample entry by a factor of $1/p^{(b)}$. The expected number of sampled coordinates is then $\sum_b p^{(b)} = \sum_b |c_1^{(b)} - c_2^{(b)}|/\|c_1 - c_2\|_1 = 1$. Let $\epsilon \in (0,1)$ be the approximation parameter, and let $\delta \in (0,1)$ be a bound on the error probability. We repeat this process to store a set I of $O(\log(1/\delta)/\epsilon^2)$ coordinates in total. For $x \in \mathbb{R}^d$, we denote by $x^{(I)}$ the restriction of x to coordinates in I. We denote by $S = \mathsf{diag}((1/p^{(b)})_{b\in I})$ denote the diagonal matrix representing the scaling factors. Then $Sx^{(I)}$ is the vector obtained from applying the above sampling process to x.

Under what we call *bounding box assumption* (which we will remove later), where $\min(c_1^{(b)}, c_2^{(b)}) \le q^{(b)} \le \max(c_1^{(b)}, c_2^{(b)})$ for each coordinate b, we can find a $(1+\epsilon)$-approximate nearest neighbor between c_1 and c_2. This is due to the following property. (Details in Section A.)

> *Comparator Property:* Let $\epsilon, \delta \in (0,1)$ and assume we sample a set I of $O(\log(1/\delta)/\epsilon^2)$ coordinates as described above. If $\|c_2 - q\|_1 > (1+\epsilon)\|c_1 - q\|_1$ then $\|Sc_2^{(I)} - Sq^{(I)}\|_1 > (1+\epsilon/2)\|Sc_2^{(I)} - Sq^{(I)}\|_1$ with probability $1-\delta$.

With this idea, we can construct an $\tilde{O}(n^2)$-space data structure by repeating the above for every pair (c_i, c_j) of input points. It is not hard to find the approximately nearest neighbor to q with $n-1$ comparisons: this is very simple if the comparator is exact, and the same idea can be extended for our approximate comparator as well; see Section A for details (also see [1] for previous work on imprecise comparisons).

Towards Linear Space via Global Sampling. The drawback of the previous algorithm is its $\tilde{O}(n^2)$ space complexity. To avoid this, a natural idea is to reduce the number of pairs for which we perform the above sampling process. However, it is unclear to us if this is possible at all. Another idea is to do the above process *more globally* instead of pairwise. Here is a simple way to do this: Let $C = \{c_1, \ldots, c_n\}$ be the set of input points. We (independently) sample each coordinate b with probability $p^{(b)} = \max_{ij} p_{ij}^{(b)}$ where $p_{ij}^{(b)} = |c_i^{(b)} - c_j^{(b)}|/\|c_i - c_j\|_1$ denotes the probability of sampling b under the pairwise sampling process for $c_i, c_j \in C$.

At least two questions arise from the idea above: (i) For any pair c_i and c_j and query q in the bounding box can we still tell (approximately) if c_i or c_j is nearer to q as in the pairwise process (in particular, are we creating noise by sampling some coordinates with higher probility than $p_{ij}^{(b)}$)? This is not so hard to address: Despite sampling each coordinate with probability higher than $p_{ij}^{(b)}$, the expectation does not change due to rescaling by $1/p^{(b)}$, and the concentration behavior is only better. More precisely, the comparator property can still be proven.

A more unclear point is: (ii) How many coordinates do we sample in total? For example, why is it not possible that in the worst case every pair c_i and c_j needs different coordinates and thus we are forced to select $\Omega(n^2)$ coordinates in total. We show that this bad situation

[3] This can be viewed as an application of so-called importance sampling – a technique for variance reduction. The independent sampling process described here differs slightly from the one in Section A.

will not happen. To give an idea, we consider the special case $C \subseteq \{0,1\}^d$. The overall expected number of coordinates sampled is $S(C) = \sum_b p^{(b)} = \sum_b \max_{ij} p_{ij}^{(b)}$. (We ignore sampling repetition, which increases our bound only by $O(\log n)$.) Assume w.l.o.g. that (c_1, c_2) is the closest pair in C. Notice that $p^{(b)} = p_{12}^{(b)}$ for every coordinate b in the set D of coordinates where c_1 and c_2 differ; they minimize $\|c_i - c_j\|_1$ and hence maximize $p_{ij}^{(b)}$. Therefore $S(C)$ drops by at most $\sum_b p_{12}^{(b)} = \sum_b |c_1^{(b)} - c_2^{(b)}|/\|c_1 - c_2\|_1 = 1$ if we remove the coordinates in D from all input points. On the other hand, we can identify c_1 and c_2 after removing D obtaining a set C' with at most $n-1$ points. Thus $S(C) \leq S(C') + 1$. Using induction we can upper bound $S(C)$ by n. This inductive proof strategy can be extended to the general ℓ_1 metric although the proof is notably more technical. Even though ℓ_1 and ℓ_2 metrics are equivalent on the Boolean cube, we do not know how to extend the above proof strategy to the general ℓ_2 metric. To obtain a linear bound for ℓ_2 we rely on an entirely different, linear-algebraic approach leveraging a latent connection between our sampling probabilities and the singular-value decomposition. Our proof is based on a connection to the feature selection approach by Boustidis et al. [10] for k-Means clustering. In fact, our sampling probabilities can be upper bounded by theirs, which are based on SVD. Notice, however, that we achieve a $(1+\epsilon)$-approximation for nearest neighbor as compared to their $(3+\epsilon)$-approximation for k-Means.[4]

Let I denote the set of coordinates sampled by our global scheme. Memorizing these coordinates for all input points would still require $n|I| = \tilde{O}(n^2)$ space. To obtain $\tilde{O}(n)$ space we apply standard dimension reduction tools such as the Johnson-Lindenstrauss transform for ℓ_2 or the technique by Indyk [21] for ℓ_1. Specifically, we sample a suitable random $m \times |I|$ matrix M, where $m = O(\log n)$, and store the set $C' = \{\, MSc_i^{(I)} \mid c_i \in C \,\}$ requiring $mn = \tilde{O}(n)$ space. Storing M itself requires $m|I| = \tilde{O}(n)$ space, too, as does storing I itself. Upon receiving a query q, we compute $MSq^{(I)}$ and output the nearest neighbor in C', which takes $\tilde{O}(n)$ coordinate queries. Here we use that the comparator property guarantees $\|Sq^{(I)} - Sc_j^{(I)}\|_1 > (1+\epsilon/2)\|Sq^{(I)} - Sc_i^{(I)}\|_1$ if $\|q - c_j\|_1 > (1+\epsilon)\|q - c_i\|_1$. Therefore, if we bound the distance distortion of the dimension reduction by, say, $1 + \epsilon/10$ then $F(MSq^{(I)}, MSc_j^{(I)}) > (1+\epsilon/5)F(MSq^{(I)}, MSc_i^{(I)})$ for the function F from the ℓ_1 dimension reduction by Indyk [21]. This is sufficient for approximately comparing the distances.

Removing the Bounding Box Assumption. If a query point q does not satisfy the bounding box property for two points $c_i, c_j \in C$ the comparator property may fail with too high probability. Specifically, it may happen that $\|q - c_j\|_1 > (1+\epsilon)\|q - c_i\|_1$ and yet $\|Sq^{(I)} - Sc_j^{(I)}\|_1 = (1+\xi)\|Sq^{(I)} - Sc_i^{(I)}\|_1$ for some positive $\xi \ll \epsilon$. In such a scenario, applying dimension reduction via matrix M may result in $F(MSq^{(I)}, MSc_j^{(I)}) < F(MSq^{(I)}, MSc_i^{(I)})$ because it distorts distances by factor $1 + \Theta(\epsilon)$.

A natural attempt to resolve this issue is to "truncate" the query point so it comes sufficiently close to the bounding box. This approach is described in Section D and can in fact be applied to general ℓ_p metrics. However, it requires explicitly storing the projected points and therefore $\tilde{O}(n^2)$ space.

To retain our near-linear space bound, we do not actually change the sampling or the query procedure. Rather, we relax in our analysis on the requirement that the comparator property needs to hold for all point pairs. Specifically, we argue (using Markov) that with

[4] In Section F, we provide a formal reduction to feature selection by Boustidis et al. [10]. There is follow-up work achieving even guarantee $1+\epsilon$ for feature selection in clustering [16]. However, we do not know how to apply their approach to our setting. In particular, the above reduction is not applicable.

sufficiently high probability our distance estimate to the (unknown) nearest neighbor o is within a bounded factor (*). We argue that under this condition all comparisons with o do satisfy the comparator property, which is enough to satisfy correctness. Towards this, we distinguish for each $c \in C$ between *good* coordinates b where $q^{(b)}$ is relatively close to the interval $[c^{(b)}, o^{(b)}]$. For the set of good coordinates, strong concentration properties hold – similar to those under the bounding box assumption. For the remaining (bad) coordinates, we use two properties: First, the query point is indifferent between c and o on the set of bad coordinates as it is sufficiently distant and thus indifferent between o and c on these coordinates. This and property (*) automatically implies that the total contribution of the bad coordinates cannot be too high. Together, these properties imply that (a relaxation of) the comparator property holds for all comparisons with o thereby implying the $(1+\epsilon)$-approximation guarantee also after removing the bounding box assumption. For details, we refer to the full technical sections below.

3 Preliminaries

In this section, we list some basic terminology, definitions and notation that we use throughout.

▶ **Definition 3.1** (Approximate Notation). *Given real numbers $a, b > 0$ and $0 < \epsilon < 1$, we say that $a \approx_\epsilon b$ if $(1-\epsilon)b \le a \le (1+\epsilon)b$.*

▶ **Definition 3.2** (Coordinate Index Notation). *For a point $a \in \mathbb{R}^d$, we use $a^{(i)}$ to represent the i-th coordinate of a. For a set or multiset $I = \{i_1, i_2, \ldots, i_m\} \subseteq [d]$, we denote $a^{(I)}$ to be $\left(a^{(i_1)}, a^{(i_2)}, \ldots, a^{(i_m)}\right) \in \mathbb{R}^m$.*

In this paper, we look into the approximate nearest neighbor problem, defined as follows.

▶ **Definition 3.3** (α-Approximate Nearest Neighbor Problem). *Given a set $C = \{c_1, c_2, \ldots, c_n\}$ of n points in a metric space (X, dist), build a data structure that given any query point $q \in X$, returns index $i \in [n]$, such that $\mathsf{dist}(q, c_i) \le \alpha \cdot \min_{j \in [n]} \mathsf{dist}(q, c_j)$.*

When C only has 2 points, we use the shorthand *2-point comparator* to denote such an approximate nearest neighbor data structure.

▶ **Definition 3.4** (2-point Comparator). *Given a set $C = \{c_1, c_2\}$ of 2 points in a metric space (X, dist) and parameters $0 \le \epsilon, \delta < 1$, we call a data structure as a 2-point comparator, if for any query point $q \in X$, with probabilty $1-\delta$, it returns index $i \in \{1, 2\}$, such that $\mathsf{dist}(q, c_i) \le (1+\epsilon) \cdot \min_{j \in \{1,2\}} \mathsf{dist}(q, c_j)$.*

We use the following Chernoff bounds in this paper.

▶ **Theorem 3.5** (Chernoff Bound). *Let $X_1, X_2, \ldots, X_n$ be independent random variables, and $X_i \in [0, c]$. Define $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then*

$$\mathbb{P}\left[X \ge (1+\delta)\mu\right] \le \exp(-\delta^2\mu/(3c)), \text{ for } 0 < \delta \le 1$$

$$\mathbb{P}\left[X \ge (1+\delta)\mu\right] \le \exp(-\delta\mu/(3c)), \text{ for } \delta \ge 1$$

$$\mathbb{P}\left[X \le (1-\delta)\mu\right] \le \exp(-\delta^2\mu/(2c)), \text{ for } \delta \ge 0.$$

4 Near-linear Space Data Structures for ℓ_1 and ℓ_2 Metrics

In this section, we design a data structure with near-linear space and query complexity for approximate nearest neighbor under ℓ_1 and ℓ_2 metrics. We prove the following theorem.

▶ **Theorem 4.1.** *Consider a set C of n points $c_1, \ldots, c_n \in \mathbb{R}^d$ equipped with ℓ_1 or ℓ_2 metric. Given $0 < \epsilon < 1/4$ and $0 < \delta < 1$, we can efficiently construct a randomized data structure such that for any query point $q \in \mathbb{R}^d$ the following conditions hold with probability $1 - \delta$. (i) The data structure reads $O(n \log(n)/(\text{poly}(\epsilon, \delta)))$ coordinates of q. (ii) It returns $i \in [n]$ where c_i is a $(1+\epsilon)$-approximate nearest neighbor of q in C. (iii) The data structure has a size of $O(n \log^2(n) \log(d)/(\text{poly}(\epsilon, \delta)))$ words.*

The proof of Theorem 4.1 for the case of ℓ_1-metric is presented in Section 4.1 with missing proofs deferred to Appendix B.1, where as the proof for the case of ℓ_2-metric is in Appendix B.2. Both proofs follow the same structure although they differ in details.

4.1 Data Structure for ℓ_1-metric with Near-linear Space and Query Time

The preprocessing and querying procedures of our data structure are described in Section 4.1. The algorithm receives a set C of n points and parameters $\epsilon, \delta \in (0, 1)$ as inputs for its preprocessing phase. During preprocessing, it iteratively samples coordinates from $[d]$ to form a multiset I.[5] Let R denote the set C of n points restricted to I. An ℓ_1 dimension reduction is then applied to R to further reduce the space. In the query phase, the algorithm receives a query $q \in \mathbb{R}^d$. It probes q on all coordinates in I and applies the same ℓ_1 dimension reduction to the coordinate-selected query. The output, identifying which point in C is the nearest neighbor of q, is determined by which point, after coordinate selection and dimension reduction, is closest to the query.

Algorithm 1 An approximate nearest neighbor data structure for n points in $\mathbb{R}^d$ for ℓ_1 metric.

Preprocessing (C, ϵ, δ) `// Inputs:` $C = \{c_1, \ldots, c_n\} \subseteq \mathbb{R}^d, \epsilon \in (0, \frac{1}{4}), \delta \in (0, 1)$
  Let $I \leftarrow \emptyset$ be a multiset and $T \leftarrow O(\log(n/\delta)/(\epsilon^3\delta^2))$;
  for $t \in [T]$ **do**
    for $b \in [d]$ **do**
      Add b to I with probability $p^{(b)} \triangleq \max_{(i,j)\in\binom{n}{2}} \frac{\left|c_i^{(b)} - c_j^{(b)}\right|}{\|c_i - c_j\|_1}$;
  Let $R = \{r_1, r_2, \ldots, r_n\} \subseteq \mathbb{R}^I$, where for $i \in [n], b \in I$, we have $r_i^{(b)} = c_i^{(b)}/p^{(b)}$;
  Let $m \leftarrow O(\log(n/\delta)/(\epsilon^2\delta^2))$;
  Sample $M \in \mathbb{R}^{[m]\times I}$, where each entry of M follows the Cauchy distribution, whose density function is $c(x) = \frac{1}{\pi(1+x^2)}$. Notice that $M(\cdot)\colon \mathbb{R}^I \to \mathbb{R}^m$ is an oblivious linear mapping [21];
  store I, $M(R) = \{Mr_i | i \in [n]\}$, $\{p^{(b)} | b \in I\}$, M
Query (q) `// Inputs:` $q \in \mathbb{R}^d$
  Query $q^{(b)}, \forall b \in I$;
  Let $u \in \mathbb{R}^I$, where $\forall b \in I, u^{(b)} = q^{(b)}/p^{(b)}$;
  Let $F((x_1, \ldots, x_m), (y_1, \ldots, y_m)) \coloneqq \mathsf{median}(|x_1 - y_1|, \ldots, |x_m - y_m|)$;
  Let $\hat{i} = \arg\min_{i\in[n]} F(Mr_i, Mu)$;
  return $\hat{i}$

[5] Multiple coordinates may be added to I in the same iteration, and I may contain repeated coordinates.

▶ **Theorem 4.2.** *Consider a set C of n points $c_1, \ldots, c_n \in \mathbb{R}^d$ equipped with ℓ_1 metric. Given $0 < \epsilon < 1/4$ and $0 < \delta < 1$, with probability $(1-\delta)$, we can efficiently construct a randomized data structure (See Section 4.1) such that for any query point $q \in \mathbb{R}^d$ the following conditions hold with probability $1-\delta$. (i) The data structure reads $O(n \log(n/\delta)/(\epsilon^3\delta^2))$ coordinates q. (ii) It returns $i \in [d]$ where c_i is a $(1+\epsilon)$-approximate nearest neighbor of q in C. (iii) The data structure has a size of $O(n \log^2(n/\delta) \log(d)/(\epsilon^5\delta^4))$ words.*

To prove Theorem 4.2, we consider the following four events and prove that each of them holds with probability $1-\delta'$, where $\delta' = \delta/4$. We establish the correctness of the data structure by conditioning on the first three events. By conditioning on the last event, we demonstrate the space and query complexity.

1. The first event is that the distance between a query and its nearest neighbor is overestimated at most by the factor $4/\delta$.
2. The second event is that for a query point the estimated distance to its nearest neighbor is significantly smaller than the estimated distance to any center that is not an approximate nearest neighbor.
3. The third event is that the dimension reduction preserves the distances between the coordinate-selected points and the query.
4. The fourth event is that at most near-linear many coordinates are sampled in the preprocessing and accessed in the query phase.

Finally, by applying a union bound, we show that all four events hold with probability $1-\delta$, which implies that the requirements of Theorem 4.1 are met. We show the proof in the following subsections in line with the structure described above.

4.1.1 Upper Bound for Estimate of Distance between Query and its Nearest Neighbor

We first show that the distance between a query point and its nearest neighbor is overestimated at most by a factor $1/\delta'$ with probability $1-\delta'$.

▷ Claim 4.3. Let $q \in \mathbb{R}^n$ be a query and let $i^* \in [n]$ be the index of its nearest neighbor. It holds that $\frac{1}{T}\|r_{i^*} - u\|_1 \leq \frac{1}{\delta'}\|c_{i^*} - q\|_1$, with probability $1-\delta'$.

4.1.2 Coordinate Selection Preserves Distance Ratio

In this section, we show that, the following holds: for any query point $q \in \mathbb{R}^n$, whose nearest neighbor is c_{i^*}, with probability $1-2\delta'$, for all $c_i \in C$ such that $\frac{\|c_i - q\|_1}{\|c_{i^*} - q\|_1} \geq 1+\epsilon$, the estimated ratio is significantly larger than one, that is $\frac{\|r_i - u\|_1}{\|r_{i^*} - u\|_1} \geq 1 + \epsilon\delta'/4$. This statement is formalized in Lemma 4.8.

To prove the statement, we first partition the coordinates in $[d]$ into good and bad regions for each pair of points. In the good region are the coordinates on which the query is not too far away from one of the points. The remaining coordinates are in the bad region. The intuition is that, restricted to the coordinates in the good region, the distance between one of the points and the query is bounded. This implies that we can estimate these distances using a Chernoff bound. For the bad coordinates the intuition is that the distances from the query to the 2 points are so large that the ratio between these distances is close to one. Additionally these distances are not overestimated because of the previous event in Section 4.1.1.

▶ **Definition 4.4** (Good region and bad region). *Consider two points $c_1, c_2 \in \mathbb{R}^d$, and a query point $q \in \mathbb{R}^d$. We define the good region* $\mathsf{Good}(c_1, c_2, q)$ *w.r.t.* (c_1, c_2, q) *as*

$$\{z \in [d] \mid \min\{c_1^{(z)}, c_2^{(z)}\} - \frac{8}{\epsilon\delta'}|c_1^{(z)} - c_2^{(z)}| \leq q^{(z)} \leq \max\{c_1^{(z)}, c_2^{(z)}\} + \frac{8}{\epsilon\delta'}|c_1^{(z)} - c_2^{(z)}|\}.$$

We define the bad region w.r.t. (c_1, c_2, q) *as* $\mathsf{Bad}(c_1, c_2, q) = [d] \setminus \mathsf{Good}(c_1, c_2, q)$. *When there is no ambiguity, we use* Good *for* $\mathsf{Good}(c_1, c_2, q)$, *and* Bad *for* $\mathsf{Bad}(c_1, c_2, q)$.

Additionally, we define the distance between points restricted to subsets of coordinates.

▶ **Definition 4.5** (Restricted distance). *Given $x, y \in \mathbb{R}^d$, we define the distance between x and y restricted to region $J \subseteq [d]$ as* $\mathsf{dist}_J(x, y) = \sum_{z \in J} \left|x^{(z)} - y^{(z)}\right|$. *Furthermore, for $u, v \in \mathbb{R}^I$, where I is a multiset, whose elements are from $[d]$, we define* $\mathsf{dist}_J(u, v) = \sum_{z \in I} \mathbb{1}_{z \in J} \cdot \left|x^{(z)} - y^{(z)}\right|$.

The following lemma shows that for two points, whose distances to the query differ significantly, then the majority of the distance between the two points is present in the good region. The intuition of the lemma is that if the bad region contributes a lot to the distance $\|b - a\|_1$, it implies that the bad region contributes much to $\|b - q\|_1$ and $\|a - q\|_1$, then the ratio between the two distances should be very close to 1.

▶ **Lemma 4.6.** *Consider three points $a, b, q \in \mathbb{R}^d$. If $\frac{\|b-q\|_1}{\|a-q\|_1} \geq 1+\epsilon$, then $\frac{\mathsf{dist}_{\mathsf{Good}(a,b,q)}(b,a)}{\|b-a\|_1} \geq \frac{13}{16}$.*

The following lemma shows that for two points c_i, c_j and a query q such that q is significantly closer to c_j, we can estimate the distances, restricted to the good region, between one of the points and the query.

▶ **Lemma 4.7.** *Let c_i, c_j where $i, j \in [n]$ be such that for the query point $q \in \mathbb{R}^d$ it holds that $\|c_i - q\|_1 \geq (1 + \epsilon)\|c_j - q\|_1$. Let* Good *be shorthand for* $\mathsf{Good}(c_i, c_j, q)$ *and* Bad *for* $\mathsf{Bad}(c_i, c_j, q)$. *The following statements are true:*

1. $\mathsf{dist}_{\mathsf{Good}}(c_i, q) \geq (1 + \epsilon)\mathsf{dist}_{\mathsf{Good}}(c_j, q)$.
2. *If* $\mathsf{dist}_{\mathsf{Good}}(c_j, q) \geq \frac{1}{8}\mathsf{dist}_{\mathsf{Good}}(c_i, c_j)$, *then with probability* $1 - \delta'/n$,

$$\mathsf{dist}_{\mathsf{Good}}(r_i, u) \approx_{\frac{\epsilon}{16}} T \cdot \mathsf{dist}_{\mathsf{Good}}(c_i, q),$$
$$\mathsf{dist}_{\mathsf{Good}}(r_j, u) \approx_{\frac{\epsilon}{16}} T \cdot \mathsf{dist}_{\mathsf{Good}}(c_j, q).$$

3. *If* $\mathsf{dist}_{\mathsf{Good}}(c_j, q) < \frac{1}{8}\mathsf{dist}_{\mathsf{Good}}(c_i, c_j)$, *then* $\frac{\mathsf{dist}_{\mathsf{Good}}(c_i,q)}{\mathsf{dist}_{\mathsf{Good}}(c_j,q)} > 7$. *With probability* $1 - \delta'/n$,

$$\mathsf{dist}_{\mathsf{Good}}(r_i, u) \approx_{\frac{\epsilon}{16}} T \cdot \mathsf{dist}_{\mathsf{Good}}(c_i, q),$$
$$\frac{\mathsf{dist}_{\mathsf{Good}}(r_i, u)}{\mathsf{dist}_{\mathsf{Good}}(r_j, u)} \geq 3.$$

In the following lemma we state that our bounds on the distance estimates are good enough. Concretely, for a point that is not an approximate nearest neighbor, the estimated distance between this point and the query is significantly larger than the estimated distance between the nearest neighbor and the query.

▶ **Lemma 4.8.** *For any query point $q \in \mathbb{R}^d$, with probability $1 - 2\delta'$, it holds for all $i \in [n]$, if $\frac{\|c_i-q\|_1}{\|c_{i^*}-q\|_1} \geq 1 + \epsilon$, then $\frac{\|r_i-u\|_1}{\|r_{i^*}-u\|_1} \geq 1 + \epsilon\delta'/4$.*

4.1.3 ℓ_1 Dimension Reduction Preserves the Distance Ratio

▶ **Theorem 4.9** (Theorem 1 in [21]). *Consider 2 points $a, b \in \mathbb{R}^d$ and parameters $0 < \epsilon, \delta < 1$. We sample a random matrix $M \in \mathbb{R}^{m \times d}$, where $m = O(\log(1/\delta)/(\epsilon^2))$ and each entry of M is sampled from Cauchy distribution, whose density function is $c(x) = \frac{1}{\pi(1+x^2)}$. Then with probability $1 - \delta$, $F(Ma, Mb) \approx_\epsilon \|a - b\|_1$, where $F((x_1, \ldots, x_m), (y_1, \ldots, y_m)) \coloneqq \mathsf{median}(|x_1 - y_1|, \ldots, |x_m - y_m|)$.*

▷ Claim 4.10. Let $q \in \mathbb{R}^n$ be a query. Furthermore, let $i^* \in [n]$ be the index of its nearest neighbor. Section 4.1 outputs an index i such that

$$\frac{\|c_i - q\|_1}{\|c_{i^*} - q\|_1} \leq 1 + \epsilon$$

with probability $1 - 3\delta'$.

Proof. Applying Theorem 4.9 with $m = O(\log(n/\delta')/(\epsilon^2\delta'^2))$, given any $v \in R$, it holds with probability $1 - \delta'/(n+1)$, that $F(Mv, Mu) \in [1 - \epsilon\delta'/100, 1 + \epsilon\delta'/100]\|v - u\|_1$. By union bound, with probability $1 - \delta'$, it holds for all $v \in R$. By Lemma 4.8 it holds with probability $1 - 2\delta'$ that for all $i \in [n]$, if $\frac{\|c_i - q\|_1}{\|c_{i^*} - q\|_1} \geq 1 + \epsilon$, then $\frac{\|r_i - u\|_1}{\|r_{i^*} - u\|_1} \geq 1 + \epsilon\delta'/4$, thus $\frac{F(Mr_i, Mu)}{F(Mr_{i^*}, Mu)} > 1$ with probability $1 - 3\delta'$, i.e. the algorithm will not output i as the index of the approximate nearest neighbor. ◁

4.1.4 Space and Query Complexity

Next, we argue the space and query complexity. The main result in this section is that for any set C of centers the sum of probabilities $\sum_{b \in [d]} p^{(b)}$ is upper bounded by the number of centers.

▶ **Lemma 4.11.** $1 \leq \sum_{b \in [d]} p^{(b)} \leq n$.

Proof. The lower bound for $\sum_{b \in [d]} p^{(b)}$ can be seen as follows: Fix arbitrary $i_1, j_1 \in [n]$ s.t. $\|c_{i_1} - c_{j_1}\|_1 \neq 0$. We have, $\sum_{b \in [d]} p^{(b)} \geq \sum_{b \in [d]} \frac{\left|c_{i_1}^{(b)} - c_{j_1}^{(b)}\right|}{\|c_{i_1} - c_{j_1}\|_1} = 1$.

Next we prove the upper bound for $\sum_{b \in [d]} p^{(b)}$. We first introduce an operation that we call "collapse". For real numbers $\tau_1 \leq \tau_2$,

$$\mathsf{collapse}_{\tau_1, \tau_2}(x) \coloneqq \begin{cases} x, & x \leq \tau_1, \\ \tau_1, & \tau_1 < x \leq \tau_2, \\ x - (\tau_2 - \tau_1), & x > \tau_2. \end{cases}$$

For $\tau_1 > \tau_2$, $\mathsf{collapse}_{\tau_1, \tau_2}(x) \coloneqq \mathsf{collapse}_{\tau_2, \tau_1}(x)$.

We also slightly abuse the notation, and define for $a, b, c \in \mathbb{R}^d$,

$$\mathsf{collapse}_{a,b}(c) \triangleq (\mathsf{collapse}_{a_1, b_1}(c_1), \mathsf{collapse}_{a_2, b_2}(c_2), \ldots, \mathsf{collapse}_{a_d, b_d}(c_d)) \in \mathbb{R}^d.$$

▷ Claim 4.12. Let $x, y, \tau_1, \tau_2 \in \mathbb{R}$, where $x \geq y$. We have (i) $x - |\tau_1 - \tau_2| \leq \mathsf{collapse}_{\tau_1, \tau_2}(x) \leq x$ and (ii) $x - y - |\tau_1 - \tau_2| \leq \mathsf{collapse}_{\tau_1, \tau_2}(x) - \mathsf{collapse}_{\tau_1, \tau_2}(y) \leq x - y$.

Proof. Denote $x' = \mathsf{collapse}_{\tau_1, \tau_2}(x)$ and $y' = \mathsf{collapse}_{\tau_1, \tau_2}(y)$. Assume w.l.o.g. that $\tau_1 \leq \tau_2$, otherwise we switch them.

For the first argument, $\forall x \in \mathbb{R}, x - |\tau_1 - \tau_2| \leq \mathsf{collapse}_{\tau_1, \tau_2}(x) \leq x$, this is true because of the definition of collapse.

For the second argument, $\forall x \geq y \in \mathbb{R}$, $x-y-|\tau_1-\tau_2| \leq \mathsf{collapse}_{\tau_1,\tau_2}(x)-\mathsf{collapse}_{\tau_1,\tau_2}(y) \leq x-y$, consider

$$f(x) = x - \mathsf{collapse}_{\tau_1,\tau_2}(x) = \begin{cases} 0, & x \leq \tau_1, \\ x-\tau_1, & \tau_1 \leq x \leq \tau_2, \\ \tau_2-\tau_1, & x > \tau_2. \end{cases}$$

Notice that $f(x)$ is a non-decreasing function, thus $f(x) \geq f(y)$, which proves $x-y \geq x'-y'$. Notice that $0 \leq f(x) \leq \tau_2-\tau_1$, thus $f(x) \leq f(y)+(\tau_2-\tau_1)$, which proves $x'-y' \geq x-y-(\tau_2-\tau_1)$. ◁

▷ Claim 4.13. Let c_{i_0}, c_{j_0} be the closest pair of points in a set $C = \{c_1, c_2, \ldots, c_n\}$ of points, i.e., $(i_0, j_0) = \arg\min_{(i,j):\, i \neq j} \|c_i - c_j\|_1$. Define $P_n(C)$ to be $\sum_{b\in[d]} \max_{i\neq j} \frac{|c_i^{(b)} - c_j^{(b)}|}{\|c_i - c_j\|_1}$. The set $C' = \{c'_i = \mathsf{collapse}_{c_{i_0},c_{j_0}}(c_i) | i \in [n]\}$ is such that (i) it contains $n-1$ points and (ii) $P_n(C) \leq P_{n-1}(C') + 1$.

Proof. Note, by the definition of the collapse operation, that $c'_{i_0} = c'_{j_0}$. Hence, C' has at most $n-1$ points, proving part (i) of the claim.

To prove (ii), define $p_b = \max_{c_i,c_j\in C, i\neq j} \frac{|c_i^{(b)} - c_j^{(b)}|}{\|c_i - c_j\|_1}$ and $p'_b = \max_{c'_i,c'_j\in C', i\neq j} \frac{|c_i'^{(b)} - c_j'^{(b)}|}{\|c'_i - c'_j\|_1}$. We prove that $p_b - p'_b \leq \frac{|c_{i_0}^b - c_{j_0}^b|}{\|c_{i_0} - c_{j_0}\|_1}, \forall b \in [d]$. The claim follows, since $P_k(C) - P_{k-1}(C') = \sum_b (p_b - p'_b) \leq \sum_b \frac{|c_{i_0}^{(b)} - c_{j_0}^{(b)}|}{\|c_{i_0} - c_{j_0}\|_1} = 1$.

Fix some b, choose $(u, v) = \arg\max_{(i,j):\, c_i^{(b)} \neq c_j^{(b)}} \frac{|c_i^{(b)} - c_j^{(b)}|}{\|c_i - c_j\|_1}$, i.e. $p_b = \frac{|c_u^{(b)} - c_v^{(b)}|}{\|c_u - c_v\|_1}$.

If $c_u'^{(b)} \neq c_v'^{(b)}$, $p_b' \geq \frac{|c_u'^{(b)} - c_v'^{(b)}|}{\|c'_u - c'_v\|_1} \geq \frac{|c_u^{(b)} - c_v^{(b)}| - |c_{i_0}^{(b)} - c_{j_0}^{(b)}|}{\|c_u - c_v\|_1}$. The second inequality follows from Claim 4.12. Thus $p_b - p'_b \leq \frac{|c_{i_0}^{(b)} - c_{j_0}^{(b)}|}{\|c_u - c_v\|_1} \leq \frac{|c_{i_0}^{(b)} - c_{j_0}^{(b)}|}{\|c_{i_0} - c_{j_0}\|_1}$. If $c_u'^{(b)} = c_v'^{(b)}$, it means $|c_u^{(b)} - c_v^{(b)}| \leq |c_{i_0}^{(b)} - c_{j_0}^{(b)}|$, thus $p_b - p'_b \leq p_b = \frac{|c_u^{(b)} - c_v^{(b)}|}{\|c_u - c_v\|_1} \leq \frac{|c_{i_0}^{(b)} - c_{j_0}^{(b)}|}{\|c_{i_0} - c_{j_0}\|_1}$. ◁

Solving the recurrence given by Claim 4.13 with the boundary condition that for a set C'' of two points, $P_2(C'') = 1$, we get that $P_n(C) \leq n$. Since $P_n(C)$, for the set of n points, is by definition, equal to $\sum_{b\in[d]} p^{(b)}$, the statement of the lemma follows. ◀

▷ Claim 4.14. With probability $1-\delta'$, we have $|I| \leq 2Tn$.

Proof. We define $X_{t,z}$ for $t \in [T], z \in [d]$ to be the indicator random variable for the event that we add z to I at iteration t. Note that $\Pr[X_{t,z} = 1] = p^{(z)}$. We can see that $|I| = \sum_{t\in[T], z\in[d]} X_{t,z}$. Hence, $\mathbb{E}[|I|] = \sum_{t\in[T],z\in[d]} p^{(z)} \leq T \cdot \sum_{b\in[d]} p^{(b)} \leq T \cdot n$, where the last inequality follows from Lemma 4.11. Similarly, we can see that $\mathbb{E}[|I|] \geq T$. Using Chernoff bound, we have $\Pr[|I| - \mathbb{E}[|I|] \geq \mathbb{E}[|I|]] \leq \exp(-\mathbb{E}[|I|]/3) \leq \exp(-T/3) \leq \delta/4$. Thus, with probability $1-\delta/4$, we have $|I| \leq 2E[|I|] \leq 2Tn$. ◁

▶ **Lemma 4.15.** *With prob. $1-\delta'$, the space complexity of Section 4.1 is $O(\frac{n\log^2(n/\delta)\log(d)}{\epsilon^5\delta^4})$ wordsize, and the number of coordinates that we query is $O(\frac{n\log(n/\delta)}{\epsilon^3\delta^2})$.*

Proof. By Claim 4.14, we have that the number of coordinates of q that we query is $|I| = O(Tn) = O(\frac{n\log(n/\delta)}{\epsilon^3\delta^2})$. Next we analyze the space complexity of the data structure. Matrix M is of wordsize $O(m|I|)$, $M(R)$ is of wordsize $O(mn)$, I is of wordsize $O(|I|\log(d))$. Thus the space complexity of the our data strcuture is $O(m|I| + mn + |I|\log(d)) = O(\frac{n\log^2(n/\delta)\log(d)}{\epsilon^5\delta^4})$ wordsize. ◀

Using union bound, the success probability of the data structure is at least $1-4\delta' = 1-\delta$.

References

1 Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. *ACM Trans. Algorithms*, 12(2):19:1–19:19, 2016. doi:10.1145/2701427.

2 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008. doi:10.1145/1327452.1327494.

3 Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1018–1028. SIAM, 2014. doi:10.1137/1.9781611973402.76.

4 Alexandr Andoni, Huy L. Nguyen, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *(STOC'17)*, pages 902–913. ACM, 2017. doi:10.1145/3055399.3055418.

5 Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *(STOC'15)*, pages 793–801. ACM, 2015. doi:10.1145/2746539.2746553.

6 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.

7 Jon Louis Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980. doi:10.1145/358841.358850.

8 Alina Beygelzimer, Sham M. Kakade, and John Langford. Cover trees for nearest neighbor. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML'06)*, volume 148 of *ACM International Conference Proceeding Series*, pages 97–104. ACM, 2006. doi:10.1145/1143844.1143857.

9 Christos Boutsidis, Petros Drineas, and Michael W Mahoney. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems(NeurIPS'09)*, volume 22. Curran Associates, Inc., 2009.

10 Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Trans. Inf. Theory*, 61(2):1045–1062, 2015. doi:10.1109/TIT.2014.2375327.

11 Moses Charikar and Lunjia Hu. Near-optimal explainable k-means for all dimensions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, (SODA'22)*, pages 2580–2606. SIAM, 2022. doi:10.1137/1.9781611977073.101.

12 Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM J. Comput.*, 15(3):703–724, 1986. doi:10.1137/0215051.

13 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988. doi:10.1137/0217026.

14 Bernard Chazelle, Ding Liu, and Avner Magen. Approximate range searching in higher dimension. *Comput. Geom.*, 39(1):24–29, 2008. doi:10.1016/J.COMGEO.2007.05.008.

15 Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. *Discret. Comput. Geom.*, 22(1):63–93, 1999. doi:10.1007/PL00009449.

16 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, (STOC'15)*, pages 163–172. ACM, 2015. doi:10.1145/2746539.2746569.

17 Guilherme Dias da Fonseca and David M. Mount. Approximate range searching: The absolute model. *Comput. Geom.*, 43(4):434–444, 2010. doi:10.1016/J.COMGEO.2008.09.009.

18 Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. Almost tight approximation algorithms for explainable clustering. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, (SODA'22)*, pages 2641–2663. SIAM, 2022. doi:10.1137/1.9781611977073.103.

19 Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. Nearly-tight and oblivious algorithms for explainable clustering. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, (NeurIPS'21)*, pages 28929–28939, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/f24ad6f72d6cc4cb51464f2b29ab69d3-Abstract.html.

20 Anupam Gupta, Madhusudhan Reddy Pittu, Ola Svensson, and Rachel Yuan. The price of explainability for clustering. In *64th IEEE Annual Symposium on Foundations of Computer Science, (FOCS'23)*, pages 1131–1148. IEEE, 2023. doi:10.1109/FOCS57990.2023.00067.

21 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006. doi:10.1145/1147954.1147955.

22 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 604–613. ACM, 1998. doi:10.1145/276698.276876.

23 Piotr Indyk and Tal Wagner. Approximate nearest neighbors in limited space. In *Proc. Conference On Learning Theory (COLT'18)*, volume 75, pages 2012–2036, 2018. URL: http://proceedings.mlr.press/v75/indyk18a.html.

24 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemp. Math.*, pages 189–206. Amer. Math. Soc., Providence, RI, 1984.

25 David R. Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 741–750. ACM, 2002. doi:10.1145/509907.510013.

26 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, (SODA'04)*, pages 798–807. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982913.

27 Konstantin Makarychev and Liren Shan. Near-optimal algorithms for explainable k-medians and k-means. In *(ICML'21)*, volume 139 of *Proceedings of Machine Learning Research*, pages 7358–7367. PMLR, 2021. URL: http://proceedings.mlr.press/v139/makarychev21a.html.

28 Konstantin Makarychev and Liren Shan. Explainable k-means: don't be greedy, plant bigger trees! In *Proc. 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC'22)*, pages 1629–1642, 2022. doi:10.1145/3519935.3520056.

29 Konstantin Makarychev and Liren Shan. Random cuts are optimal for explainable k-medians. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, (NeurIPS'23)*, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/d3408794e41dd23e34634344d662f5e9-Abstract-Conference.html.

30 Edward M. McCreight. Priority search trees. *SIAM J. Comput.*, 14(2):257–276, 1985. doi:10.1137/0214021.

31 Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. Explainable k-means and k-medians clustering. In *(ICML'20)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7055–7065. PMLR, 2020. URL: http://proceedings.mlr.press/v119/moshkovitz20a.html.

Snap Rounding: A Cautionary Tale

John Hershberger
Siemens EDA, Wilsonville, OR, USA

Abstract

This paper describes the author's experience using and modifying the technique of snap rounding to meet the needs of a circuit verification system. The interplay of theory and practice illuminates the subtle challenges of both.

2012 ACM Subject Classification Theory of computation → Computational geometry; Software and its engineering → Designing software; Applied computing → Computer-aided design

Keywords and phrases Snap rounding, implementation, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.57

1 Introduction

As a discipline, computational geometry is based on real-world problems. Where is the nearest post office? What is the shortest path to grandma's house? What route should a delivery truck follow? To solve these problems, we abstract them, focusing on the key features and ignoring ones that seem irrelevant. We can (we assume) ignore the specifics of post offices, grandparents, and the kinds of goods delivered.

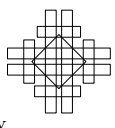
If computational geometry is to remain relevant, the abstract solutions it develops have to be brought to bear on the specific problems that inspired them. The problem features that were ignored may turn out to matter after all. The difference between ice cream and oriental rugs may matter a lot when planning a delivery route! This paper describes one example of applying the computational geometry technique of snap rounding [7, 8, 12] to a problem in the design of integrated circuits.

The specific problem to be solved is computing Boolean operations on polygonal regions. This is a key subroutine in representing the layers of metal, polysilicon, diffusion, etc., used to design a modern IC (integrated circuit) [3, 13, 15]. For example, given a collection of possibly overlapping polygons drawn by a circuit designer, a design system may need to compute the boundary of the union of the polygons. Or alternatively, given two collections of polygons, the system may need to find the points inside the first collection and not the second. (These operations are called single-layer OR and two-layer NOT, respectively.)

Computing the overlay of two subdivisions of the plane is a classic computational geometry problem, with a classic plane sweep algorithm to solve it. From the overlay one can compute any of the Boolean operations simply by noting which input polygons overlap each region of the overlay [5].

Like many computational geometry problems, the overlay problem is stated and solved in the Real RAM model, in which point coordinates are specified and computed with arbitrary precision. On a computer in the real world, the Real RAM is not realistic. Coordinates and computations can use only a limited number of bits. In this problem of polygon Boolean operations, the vertices of the input polygons are specified with k-bit integer coordinates ($k = 64$), and the result polygons must use the same format. This immediately presents a problem of accuracy. If the endpoints of two intersecting line segments are specified with k-bit integers, then rough calculation shows that the intersection point may need up to $5k$ bits to represent each of its coordinates precisely as a rational number. The vertices of the Boolean result must be rounded to the integer grid to satisfy the requirements of the IC design system.

© John Hershberger;
licensed under Creative Commons License CC-BY 4.0



Rounding the result of a geometric computation raises problems of accuracy and topological correctness. How much inaccuracy is introduced by rounding? Is it possible to guarantee that the rounded result faithfully represents the unrounded one? *Snap rounding* was invented to round geometric arrangements as accurately and correctly as possible. Section 2 describes this technique in more detail.

This paper tells the story of my experience using snap rounding in an IC design system. Features of snap rounding that seemed natural and unobjectionable in theory turned out to be ugly stumbling blocks in practice. Resolving these issues led to new theory, further practical challenges, and (eventually) a deeper understanding of the interplay between theory and practice.

2 Definitions

Snap rounding solves the problem of rounding the vertices of an arrangement of line segments to the integer grid. The algorithm is simple to define and relatively simple to implement; it comes with guarantees on the quality of its results [2, 6, 7, 8, 9, 10, 12, 14].

The input to snap rounding is a collection of *ursegments* ("ur" is short for "UnRounded" and is also a German prefix meaning "original"). In the IC design application, ursegment endpoints are 64-bit integer grid points, but snap rounding itself does not restrict its inputs. The vertices of the arrangements of ursegments, both endpoints and intersections, determine the rounding. The Cartesian plane is divided into *pixels*, unit squares centered on the integer grid points. Pixels are closed on their left and bottom sides and open on their right and top sides – thus for integer x and y, the corner $(x - \frac{1}{2}, y - \frac{1}{2})$ belongs to the pixel centered at (x, y), and the other three corners do not. Snap rounding is defined by the interaction of ursegments and pixels:

> Every pixel that contains a vertex of the ursegment arrangement becomes a *hot pixel*. Every ursegment s is replaced by a polygonal path joining the centers of the hot pixels that s intersects.

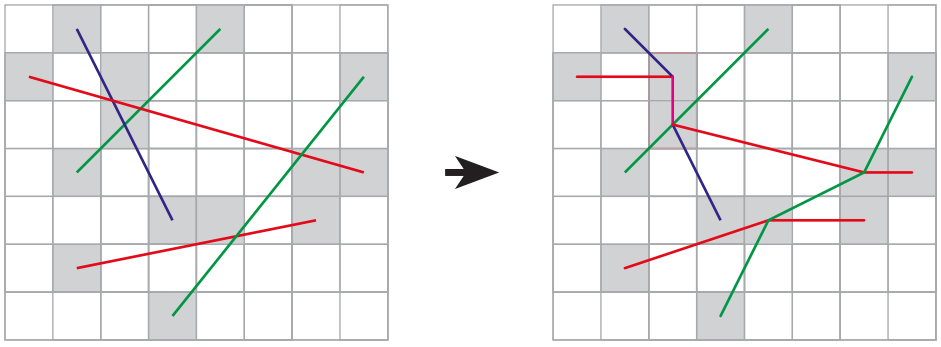


Figure 1 Hot pixels are created by vertices of the ursegment arrangement. Each ursegment is replaced by a polygonal path connecting hot pixel centers.

This simple rounding scheme, illustrated in Figure 1, has two desirable features, as proven by Guibas and Marimont [8]: no segment moves by more than half a pixel, and the topology of the input arrangement is preserved up to collapsing of features. (Here the word "topology" is used to refer to adjacency and ordering relationships.) The order of intersections between ursegments is preserved in the rounded segments, the order of intersection of ursegments with a vertical or horizontal line is preserved, and triangle orientations are preserved, all up

to collapsing of features. This reflects the fact that the rounded arrangement is a homotopic deformation of the arrangement of ursegments. The proofs are sketched below because they will be needed later.

▶ **Lemma 1.** *No segment moves by more than half a pixel. That is, the L_∞ Hausdorff distance between any ursegment and its rounded representation is at most $\frac{1}{2}$.*

Proof. For every ursegment $s = \overline{ab}$, we pick a point $p_i \in s$ belonging to each hot pixel that s intersects, with the first and last points equal to the segment endpoints: $a = p_1, \ldots, p_k = b$. Thus s is trivially represented by the polygonal path $(p_1, \ldots, p_k)$. Let q_i denote the center of the hot pixel containing each p_i, and define $\Delta_i = p_i - q_i$. Since p_i belongs to the hot pixel centered at q_i, $||\Delta_i||_\infty \leq \frac{1}{2}$. The rounded version of s is the polygonal path $(q_1, \ldots, q_k)$. Any point $p \in \overline{p_i p_{i+1}}$ is a convex combination of the endpoints, $p(t) = (1-t) \cdot p_i + t \cdot p_{i+1}$, for $t \in [0, 1]$. The corresponding point of $\overline{q_i q_{i+1}}$ is $q(t) = (1-t) \cdot q_i + t \cdot q_{i+1}$. By linearity, $p(t) = q(t) + ((1-t) \cdot \Delta_i + t \cdot \Delta_{i+1})$, so $||p(t) - q(t)||_\infty \leq \frac{1}{2}$. This one-to-one mapping between points of s and points of the rounded segment shows that the Hausdorff distance between s and the rounded segment is at most $\frac{1}{2}$. See Figure 2. ◀

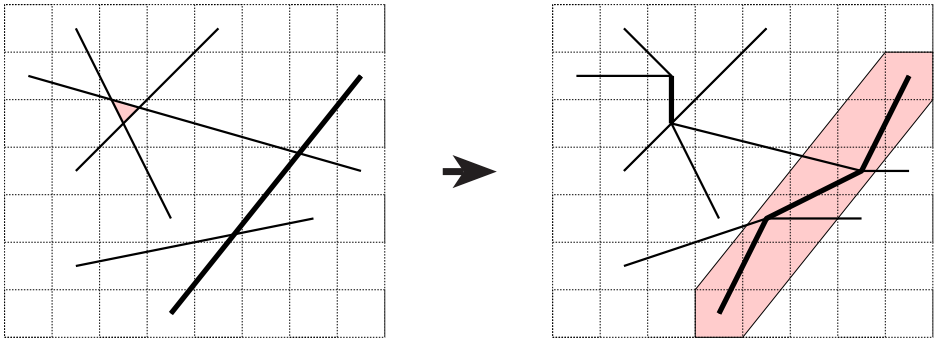


Figure 2 The rounded version of the lower right ursegment is contained in the region within L_∞ distance $\frac{1}{2}$ of the ursegment.

▶ **Lemma 2.** *The topology of the arrangement of ursegments is preserved in the rounded arrangement, up to collapsing of features.*

Proof. (Summary) Guibas and Marimont's proof uses a continuous deformation of the arrangement that maps the input arrangement to the rounded one. Topology is preserved at every instant of the deformation, and hence it is preserved over the whole deformation.

The first step of the argument introduces internal vertices on each ursegment at its first and last contacts with hot pixel boundaries. The deformation shrinks each hot pixel to its center, first by a uniform horizontal contraction to a line segment and then by a vertical contraction to a point. All the hot pixels shrink simultaneously, at the same rate. The arrangement inside each hot pixel shrinks uniformly (affinely) with the hot pixel, so all spatial and topological relationships inside the pixel are preserved (up to collapsing of features). As the hot pixels shrink, the segments outside the hot pixels move, dragged along by their endpoints on the hot pixel boundaries. Guibas and Marimont's key observation is that no external segment can pass over another segment or vertex without crossing into a hot pixel (because all endpoints and ursegment crossings are inside hot pixels), but the deformation is defined such that the hot pixel corners move away from the segments at least as fast as the segments approach – the segments never catch up with the corners.

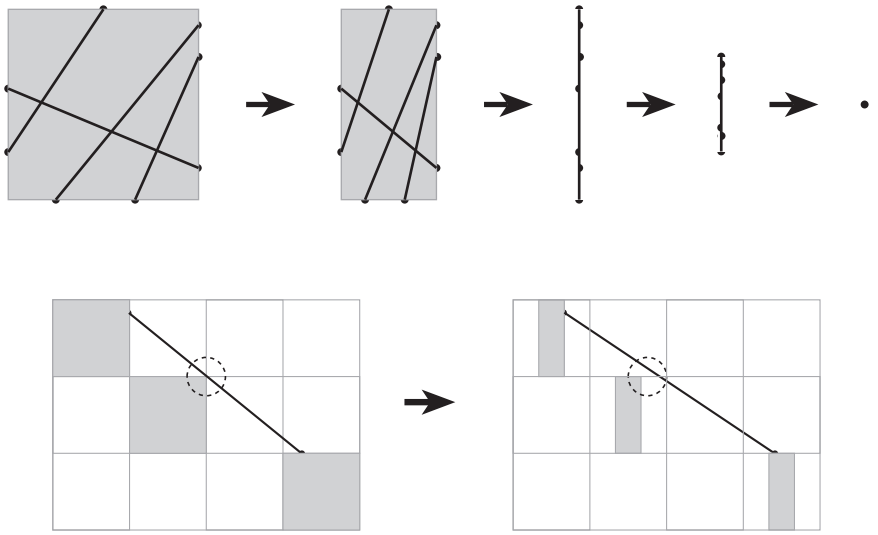


Figure 3 Contraction of a hot pixel does not affect topological relationships inside it. A deforming edge that starts outside all hot pixels never crosses into any hot pixel.

Specifically, let p_0 and p_1 be two consecutive breakpoints along some ursegment, and let $p(\tau)$ be a point on the segment between them, with $p(\tau) = (1-\tau) \cdot p_0 + \tau \cdot p_1$ for $\tau \in [0,1]$. During the horizontal contraction phase, let $p_0(t)$, $p_1(t)$, and $p(\tau,t)$ be the corresponding points at time t of the contraction, where t is the hot pixel half-width, decreasing uniformly from $\frac{1}{2}$ to 0. As t decreases, the corners of all the hot pixels move inward (left or right) at speed 1. Endpoints $p_0(t)$ and $p_1(t)$ move left or right at speed at most 1 (exactly 1 if the endpoint is on the left or right side of a hot pixel; less if it is on the bottom or top). It follows by linearity that $p(\tau,t)$ also moves left or right at speed at most 1, and therefore, because $p(\tau)$ starts outside all hot pixels, $p(\tau,t)$ never crosses into the interior of one. The same argument also works for the vertical contraction phase. See Figure 3. ◀

3 Theory meets practice

I implemented snap rounding as part of a C++ application for physical verification of integrated circuits. Because I wanted to apply snap rounding to solve a variety of problems, I used C++ templates to separate the rounding algorithm from the applications that use it in a scanline sweep.

The scanline package is partitioned into six policies, each with its own role and interface. The *input policy* extracts polygon edge data from a compressed in-memory format and presents them in left-to-right, bottom-to-top order; the *scanline representation* holds the edges intersected by the scanline at its current x-position; the *rounding policy* receives edge events in sorted order and maintains the scanline representation while generating events that describe the rounded subsegments; the *region characteristic* maintains an application-specific characterization of the region above each edge in the scanline (e.g., the depth of coverage); the *output policy* receives rounded segments and consults the scanline representation and region characteristics to produce application-specific output (e.g., polygon Boolean results); and the *event manager* coordinates the interactions between all the other subsystems, including maintaining event queues for the rounding policy and updating the region characteristics

exactly where they need to be recalculated. This decomposition of scanline functionality into distinct roles allows each subsystem to be upgraded independently of the others and allows for substantial reuse between applications. Most applications can use off-the-shelf versions of the input policy, scanline representation, rounding policy, and event manager. The region characteristic can often be reused. Only the output policy always needs a custom implementation.

This structure is both a strength and a weakness of the scanline package. It allows snap rounding to be deployed in a variety of applications with very good reliability, but the strict adherence to interface communication protocols means that out-of-band communication between subsystems is difficult or impossible, say if one subsystem needs to compensate for a problem in another.

Note that application design has already influenced the choice of algorithm. I wanted to use snap rounding to find incidences between polygons and/or edges, as one of the output policies, so the rounding policy needs to identify all the ursegments that contribute to each rounded segment. (Multiple ursegments may be rounded to the same segment in the rounded arrangement.) That means the rounding policy must provide more information than is present in the rounded arrangement of ursegments. That is, the number of notifications is asymptotically larger than the size of the rounded arrangement in the worst case. See Figure 4. Hershberger proposed an output-sensitive snap rounding algorithm that deals with these configurations efficiently [10], but the algorithm is more complicated than necessary for the inputs that arise in practice.

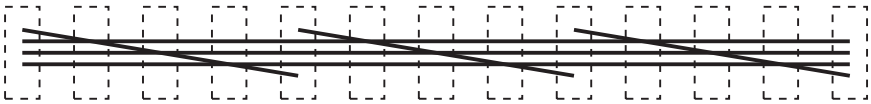


Figure 4 A pathological example with $O(n^2)$ segments in the rounded arrangement, each with $O(n)$ ursegments contributing (rounding) to it. Pixels are stretched vertically for clarity.

The implementation of the rounding policy uses a simple Bentley-Ottmann scanline [1]. The algorithm sweeps a vertical line over the arrangement of ursegments from left to right, maintaining the intersection of the line with the arrangement at each position. Whenever an ursegment intersection or endpoint is detected, it makes the containing pixel *hot*. A local search in the scanline finds all the ursegments that intersect each hot pixel.

This high-level picture becomes a bit more complicated when we look at the details. The implementation gains efficiency by exploiting the knowledge that the result of the Bentley-Ottmann scan is a rounded arrangement. Unlike a classic Bentley-Ottmann scanline, this one does not need to know the precise relative x-order of the ursegment intersections. It is enough to identify the hot pixel containing each intersection and determine whether the intersection lies in the left or right half of the pixel. The crossings inside each half-pixel can then be processed in arbitrary order. Ignoring the precise x-ordering of crossing events reduces the polynomial degree of the required arithmetic computations from quintic to cubic.

Note that the Bentley-Ottmann scanline is a suboptimal algorithmic choice, at least in theory. The scanline processes every vertex of the input arrangement $\mathcal{A}_{in}$, giving complexity $O(|\mathcal{A}_{in}| \log n)$. But $|\mathcal{A}_{in}|$ may be $\Omega(n^2)$ while the output complexity $|\mathcal{A}_{out}|$ is $O(n)$, as shown in Figure 5. It is possible to round such an arrangement in $O(n \log n)$ time, using an algorithm that spends $O(\log n)$ time for each incidence between an ursegment and a hot pixel [6]. However, this is unimportant in the IC design application: arrangements like the one in Figure 5 do not occur in practice. Simplicity of implementation beats theoretical worst-case efficiency.

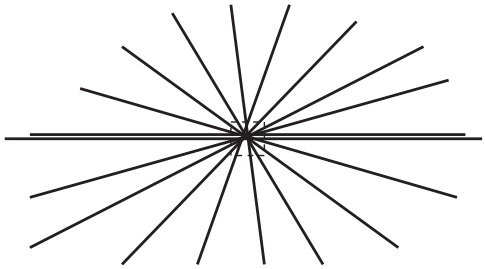


Figure 5 $O(n^2)$ ursegment intersections occur inside a single hot pixel.

The output policy for Boolean operations on polygons maintains a "depth of coverage" (winding number) above each segment in the scanline. Every edge is assigned a value of $+1$ or -1, depending on whether the polygon interior it bounds lies above or below it. The sum of the edge values along the scanline from $-\infty$ to each point on the scanline determines how many polygons cover that point. In the OR operation, for example, a region of the rounded arrangement of segments belongs to the result if its depth of coverage is positive. When the rounding policy notifies the event manager of the left/right end of a rounded segment, it also provides a pointer into the scanline. The event manager uses these pointers to update the depth of coverage counts exactly where they have been changed by scanline updates. When the right end of a rounded segment is processed, the segment gets written to the result by the output policy if and only if the segment forms the boundary between a region inside and a region outside the desired Boolean result. Because of rounding, two segments that are classified as being on the result boundary by depth of coverage may snap to be collinear. When that happens, the two segments annihilate – both are removed from the result. Correspondingly, two regions of the result interior are separated or joined, depending on whether the result interior lies between the two annihilating edges or not.

The design system requires each connected polygon to have a unique polygon number. Here "connected" refers to the polygon interior – polygons may be nonsimple (have holes), but are not connected by contact at vertices. Polygon numbers are assigned on the fly using a disjoint set (Union-Find) data structure [4].[1] Each edge in the output gets its own number; the sets for two edge numbers are joined if the edges are adjacent on the boundary of the result, or if they are vertically visible through the result interior. (Both of these conditions can be checked in constant time per edge.) At the end of the scan, the disjoint set structure is condensed to map the edge numbers to a dense set of polygon numbers. One pass through the output then suffices to replace every edge number with its corresponding polygon number.

The implementation of Boolean operations based on snap rounding replaced an earlier implementation that used a heuristic rounding scheme. That algorithm also detected intersections with a scanline, breaking each segment at rounded intersections. However, the right-hand side of each broken segment was subject to future rounding, so it was possible, though unlikely, for rounded segments to drift arbitrarily far. A second weakness became apparent in the use of Boolean operations to partition and reconstruct polygonal regions in

[1] The disjoint set structure is a simple array, with all joins linking an element to a predecessor. That means that height-balancing (linking the shorter tree to the taller) is not possible, and hence the theoretical performance bound of $O(n\alpha(n))$ is not achievable. However, path compression gives an $O(n \log n)$ worst-case performance bound, and performance in practice is excellent.

IC design. When a polygon is fractured into pieces, geometric rounding may leave tiny gaps (less than a pixel wide) between them. It is desirable for the OR of the pieces to fill in those gaps, but the previous algorithm was inconsistent in achieving that goal.

The new implementation of Boolean operations based on snap rounding has proven to be very successful. Performance is generally good, and snap rounding gives much more predictable, dependable results than the previous heuristic algorithm. Lemma 1 guarantees that rounded results are within half a pixel of the input ursegments – no segment drift is possible. Lemma 2 guarantees topological correctness. Furthermore, the fact that snap rounding moves all ursegments that hit a hot pixel is also beneficial: when snap rounding computes the OR of polygons with tiny gaps between them, the result heals up the gaps better than the earlier implementation.

4 Practice meets theory

The trouble with abstracting a practical problem to get something clean and simple is that some goals may get lost in the translation from practice to theory. This isn't an issue for basic problems like sorting, but becomes relevant for more complex problems. For example, in the problem of rounding an arrangement, symmetry is a practical goal that is difficult to achieve.[2]

In snap rounding, every ursegment that passes through a hot pixel is deformed to pass through the center of the hot pixel. But what if an ursegment merely grazes a hot pixel? What counts as "passing through"? A hot pixel is defined to be closed on its left and bottom sides, and open on its top and right sides. Thus out of the four corners of a hot pixel, only the lower left belongs to it. This means that for the configuration of segments on the left side of Figure 6, the rounded result is asymmetric, as shown on the right.

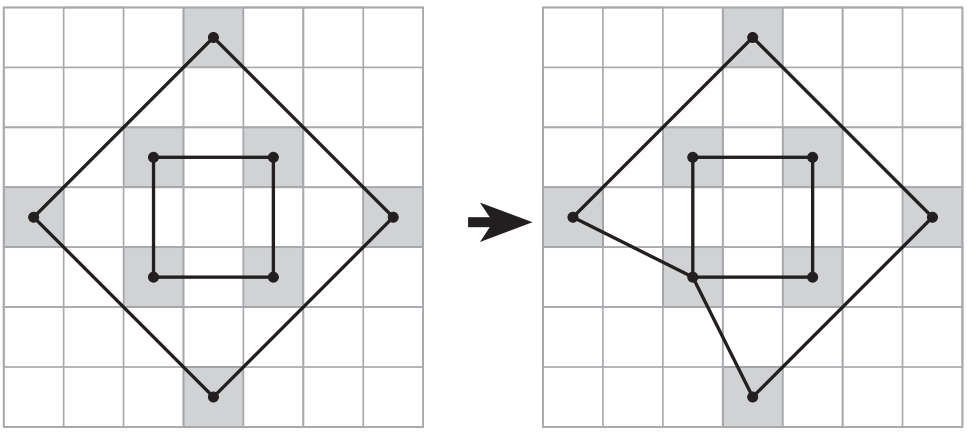


Figure 6 Snap rounding deforms one of the four segments of the outer diamond to pass through a vertex of the inner square.

[2] For efficiency's sake, modern ICs are designed hierarchically. Smaller "cells" (design components) are placed within larger cells. The result of instantiating all the placements, all the way down, is the *flat* design, which should be equivalent to the hierarchical design. However, in a hierarchical design, geometric rounding is performed in the frame of reference of individual cells. If those cells are rotated by a multiple of 90° before placement, as often occurs, any asymmetry in rounding creates a difference between the hierarchical design and the ideal flat design, in which rounding occurs after expansion of placements. Thus asymmetry is to be avoided if possible.

The previous rounding algorithm, the one that snap rounding replaced, did not perturb any of the segments in the left-hand configuration. Thus for this configuration and others like it, the results of snap rounding are perceived as worse by the users of the polygon Boolean application. Unfortunately, predictability and solid definitions are virtues that appeal to theoreticians more than to practitioners. The practitioner is likely to say, "The new algorithm gives worse results on this example. I want an algorithm that does *the right thing* on all of my test cases." The theoretician can say, loftily, "This undisciplined wish for a simultaneous solution to incompatible requirements is how we ended up with the original mess of an algorithm," but it's a tough argument to make.

5 Relaxed snap rounding (Theory gives in to practice)

Eventually, the complaints of the practitioners wore me down. Yes, by the rules of snap rounding, the rounded version of the configuration in Figure 6 has to be asymmetric, but perhaps there's *another* rounding scheme that would give better results.

For example, Hershberger's *stable snap rounding* algorithm might fit the bill [11]. Stable snap rounding has two kinds of hot pixels, magnets and pins, with magnets created by crossings or vertices not at the pixel center. An ursegment that hits a magnet is forced to pass through its center, but one that hits a pin is not – if the segment moves because of magnet vertices, the only restriction is that it may not pass over a pin, as illustrated in Figure 7. In the example of Figure 6, stable snap rounding would not move any of the segments, leaving a symmetric result.

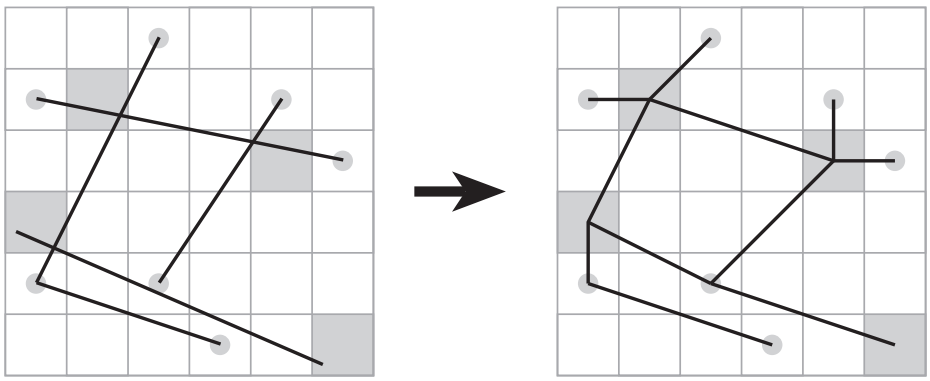


Figure 7 Shaded squares are magnets (pixels containing ursegment crossings or off-grid endpoints); shaded disks are pins (all other hot pixels). Stable snap rounding does not force ursegments to deform to pins, but maintains ursegment homotopy relative to pins.

I didn't want to implement stable snap rounding to solve the asymmetry problem. The behavior of stable snap rounding is much different from ordinary snap rounding. It was hard enough for our software testers to adapt to the differences between snap rounding and the previous algorithm – I didn't want to start the whole qualification process over again. Instead, I wanted a quick, small change that would fix this little asymmetry problem without breaking anything else. [Reader, if you are feeling uneasy about this goal – you have cause!]

I came up with the idea of *relaxed snap rounding*. This borrows an idea from stable snap rounding and applies it in one special circumstance. The rules of relaxed snap rounding are the same as those of ordinary snap rounding *except*

> If a hot pixel contains no ursegment intersections and has ursegment endpoints only at its center, then an ursegment that crosses the pixel's lower left corner *is not considered to contact the hot pixel*.

That is, if a segment crosses the lower left corner of a pixel that is hot only because of ursegment endpoints at its center, the segment is not required to deform to the pixel's center. All other rules remain the same – if an ursegment touches a hot pixel's interior, it *does* deform to the pixel center.

I was quite pleased when I came up with the idea of relaxed snap rounding. In fact, I even considered writing a short paper about it. Relaxed snap rounding solved the asymmetry problem, and I could prove that it had the same good properties as ordinary snap rounding.

- No ursegment moves by more than half a pixel. The original argument applies unchanged. An ursegment is deformed to pass through a subset of the hot pixels that it touches under ordinary snap rounding, so *a fortiori* its deformation is no greater than that produced by ordinary snap rounding.
- Topology of the input arrangement is preserved up to collapsing of features. This, too, can be proved by an adaptation of the original argument of Guibas and Marimont. Once again, we break each ursegment at its intersections with hot pixel boundaries, except that for each pixel made hot only by ursegment endpoints at its center, any segment passing only through its lower left corner is not broken there. Each hot pixel is collapsed to a point, first by uniform horizontal contraction of all hot pixels simultaneously, then by uniform vertical contraction of all the hot pixels (which are now just vertical segments). Each ursegment is split into subsegments at its breakpoints on the hot pixel boundaries, and as the hot pixels contract, the subsegments move with their breakpoint vertices. Every internal point of a subsegment is a convex combination of the subsegment endpoints. Since each endpoint moves left or right (during the first contraction) or up or down (during the second) with speed at most 1, the same is true of every internal point of a segment. Because the corners of each hot pixel move inward at speed 1, no subsegment that starts strictly outside all hot pixels can ever enter any hot pixel. Since all ursegment intersections and endpoints are inside hot pixels, the original arrangement's topology is preserved. In the worst case, a segment that passes through a hot pixel's lower left corner may stay in contact with that corner as it deforms, so the segment ends up passing through the hot pixel center. However, this still preserves topology up to collapsing of features.

Relaxed snap rounding was a big success! The asymmetry problem was resolved, with very few changes to behavior overall. It was rare for an ursegment to pass through the lower left corner of an endpoint-only hot pixel, and when the situation did occur, leaving the segment unbroken at that corner gave more desirable results than the alternative.

6 The fly in the ointment

It was not until a year later that the other shoe dropped. People began to notice that the polygon Boolean operations sometimes produced results that were not well-formed polygons. Some polygons were only weakly simple, like the one shown in Figure 8.

What was the cause of these errors? After several days of debugging, eventually I traced the problem back to a previously overlooked weakness in relaxed snap rounding: no snapping means no noticing. The output policy gets notified of a snapped segment at the segment's right end, when the segment is snapped to pass through a hot pixel's center. The polygon

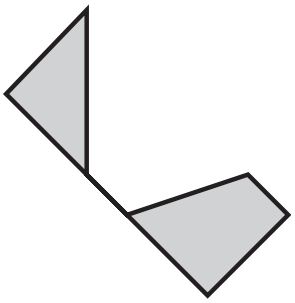


Figure 8 Relaxed snap rounding created polygons with disconnected interiors!

Boolean operation uses that notification to identify oppositely-oriented segments that cancel each other out, and to identify vertices where a polygon pinches off into two. With relaxed snap rounding, a snapped segment may pass through a hot pixel center, but the incidence is not noticed when the hot pixel is processed. The incidence is caused by snapping in another, later hot pixel. See Figure 9.

Note that this problem does not occur for ordinary snap rounding:

▶ **Lemma 3.** *Under ordinary snap rounding, an ursegment s is snapped to pass through the center of a hot pixel h if and only if s contacts h.*

Proof. Contact implies snapping, so the only thing to prove is that if s does not contact h, its snapped version cannot pass through the center of h.

Any point on s moves by at most $\frac{1}{2}$ in x and/or y during the snap rounding deformation, which means that any point of s strictly outside the closure of h (denoted $\overline{h}$) cannot deform to the center of h. The points of $\overline{h} \setminus h$ are the top and right edges and the three corners incident to those edges. If a point $p \in s$ is on the top edge of h and internal to that edge, then s is horizontal – otherwise it would intersect the interior of h. In this case all the hot pixels that s contacts are above h, so s will snap upward, away from h. A similar argument applies for s collinear with the right edge of h. If s passes through a non-contact corner of h (denoted by p) and deforms to pass through the center of h, let h_1 and h_2 be the hot pixels that s contacts immediately before and after p. For p to move to the center of h, the intersections of s with $\overline{h_1}$ and $\overline{h_2}$ must both move by $\frac{1}{2}$ in both x- and y-coordinates, in the same direction as from p toward the center of h. This implies that the contact between s and h_1 (resp. h_2) is at the same corner of h_1 (h_2) that p occupies relative to h. But contact is defined the same for all hot pixels, so since s does not contact h at p, it cannot contact h_1 or h_2, either, a contradiction. ◀

That is to say, in a scanline implementation of ordinary snap rounding, every incidence between a snapped ursegment and a hot pixel center is detected when the hot pixel is processed, but this is not true for relaxed snap rounding. As in Figure 9, it is possible for polygon Boolean operations using relaxed snap rounding to produce a pinch-off, e.g., to separate a single result polygon into two components, without detecting it.

7 The way forward

In hindsight, relaxed snap rounding seemed like a big mistake. It sacrificed the algorithmic guarantee of ordinary snap rounding for a small improvement in result quality. Unfortunately, once user expectations had been set for relaxed snap rounding's behavioral improvements, it was impossible to go backward.

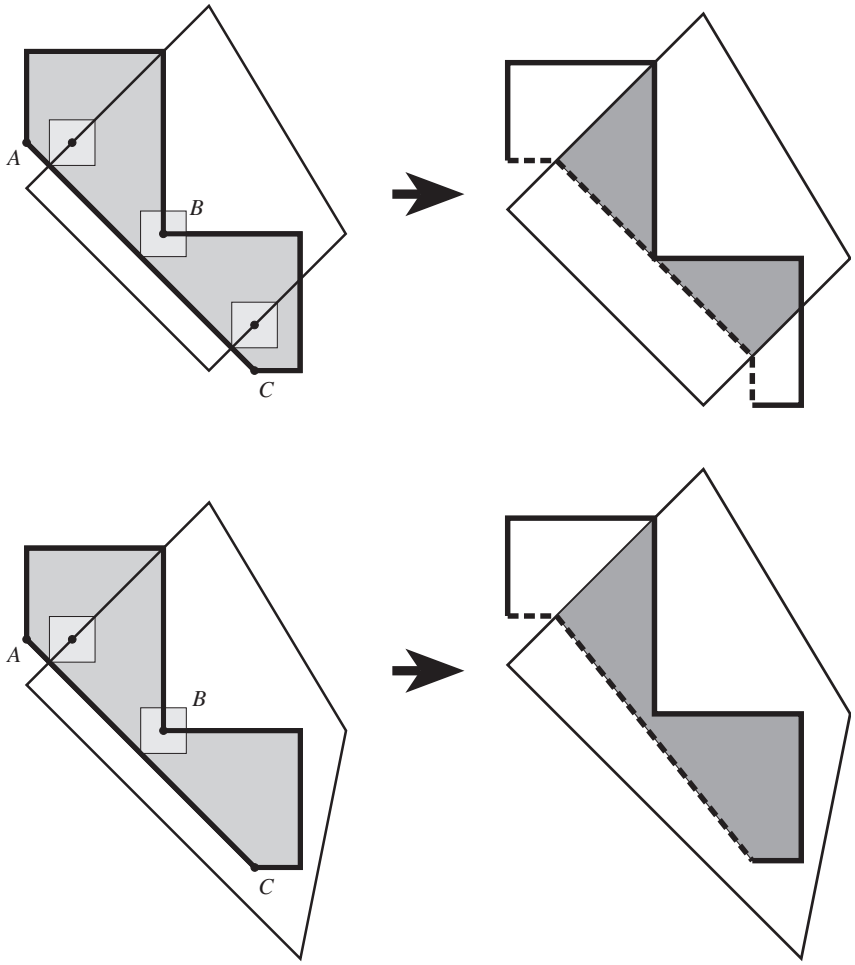


Figure 9 The shaded polygon is ANDed with the unshaded polygon. Depending on what occurs to the right of B, the segment $\overline{AC}$ may or may not snap to pass through B, creating either two or one polygons in the AND result. The relevant hot pixels are highlighted. The incidence between $\overline{AC}$ and the hot pixel at B in the upper figure is not detected by relaxed snap rounding. The snapping that creates the incidence does not occur until the scanline is past B. When the scanline is at position B, the configurations in the upper and lower figures are indistinguishable, even though the results of the AND near B are very different.

I had to find a way to reconstruct the incidences that relaxed snap rounding fails to detect. In this case, the separation of the rounding policy from the polygon Boolean output policy is a liability. The rounding policy knows when a contact between an endpoint-only hot pixel and an ursegment is being ignored, but "ignoring" means that the rounding policy does not generate an end-of-segment event for the non-incidence. The output policy is left to compensate for the rounding policy's omissions.

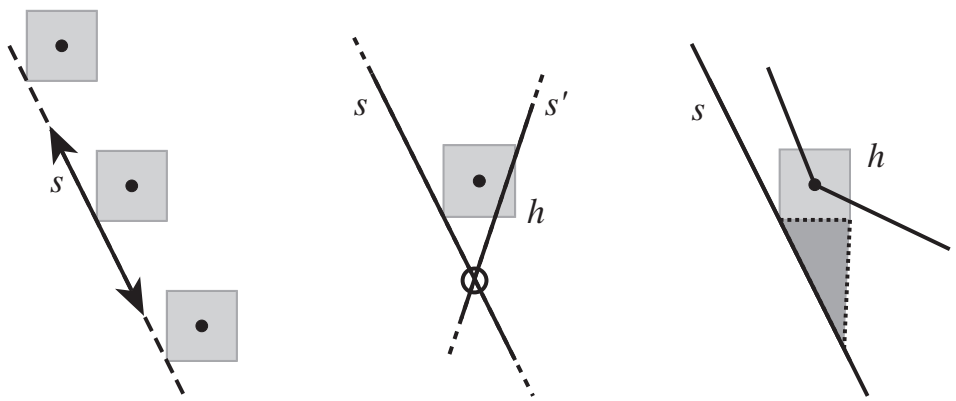
As a first step, I defined a *vulnerable vertex* as a hot pixel that contains no crossings and has an ursegment passing through its lower left corner. Vulnerable vertices can lead to undetected incidences, and every undetected incidence occurs at a vulnerable vertex. The following lemma characterizes these vulnerable vertices.

▶ **Lemma 4.** *Let s be an ursegment that passes through the lower left corner of a hot pixel h without touching the interior of h. If the contact between s and h is ignored under relaxed snap rounding, but s is snapped to pass through the center of h, then*

1. *The x-span of s contains the x-span of h in its interior.*
2. *If $s' \neq s$ is an ursegment that extends to the left of the x-span of h and rounds to pass through its center, then its leftmost intersection with h is on the top or left boundary.*
3. *If any ursegment $s' \neq s$ extends to the left of h and rounds to pass through its center, then s is immediately below the lowest such segment. Otherwise, s is immediately below the lowest segment with its left endpoint in h.*

Proof. Note that because s's contact with h is ignored, h has at least one ursegment vertex at its center, but contains no ursegment intersections.

(1) Let h_l and h_r be the hot pixels that s contacts before and after h. Since s snaps to pass through the center of h, s needs to move by $(\Delta x, \Delta y) = (\frac{1}{2}, \frac{1}{2})$ at both h_l and h_r, i.e., it must pass through their lower left corners. That means h_l must be strictly left of h, and h_r must be strictly to the right. See Figure 10(1).



(1) (2) (3)

Figure 10 (1) The hot pixels that s contacts before and after h are strictly left and right of h. (2) Ursegments s and s' intersect in the column of pixels below h. (3) The shaded triangle below h must be empty of ursegments (except possibly for ones that contact h).

(2) Let x_h be the x-coordinate of h's center. If s' first intersects the bottom of h, then it has positive slope. Segment s has negative slope. Consequently s and s' must intersect in the x-interval $(x_h - \frac{1}{2}, x_h + \frac{1}{2})$, somewhere below h. This means that h_r, the first hot pixel on s right of $x_h - \frac{1}{2}$, has its center on $x = x_h$. By (1), this means that s cannot snap to pass through the center of h. See Figure 10(2).

(3) If any ursegment that does not contact h intersects the interior of the triangle formed by s, the vertical line $x = x_h + \frac{1}{2}$, and the bottom of h, then either s has an intersection in the x-interval $(x_h - \frac{1}{2}, x_h + \frac{1}{2})$ or it crosses a hot pixel created by an ursegment endpoint in the column below h. In either case, h_r is not to the right of h, contradicting (1). Hence this triangle is free of all ursegments except possibly for ones that intersect h, implying that s is immediately below the lowest left or right segment in h, as claimed. See Figure 10(3). ◀

It is possible to detect vulnerable vertices in constant time apiece using the events that *are* passed to the output policy. Whenever the scanline passes the left or right end of an ursegment, it generates an event that is reported to the output policy with a pointer to the scanline location where the vertex occurs. Any segment with an undetected incidence lies just below one of the segments with a vertex at h's center, by Lemma 4(3). By looking at the neighboring edges in the scanline for each endpoint event, it is possible to identify all vulnerable vertices.

I used vulnerable vertices as the basis of a two-part scheme to restore correct behavior for Boolean operations. The first step uses vulnerable vertices to identify undetected incidences. The vulnerable vertices are identified in left-to-right order and stored in a table in that order. The detection scheme keeps a sliding window of vulnerable vertices. The width of the window is defined by the edges intersecting the current scanline. When the scanline reaches the right end of a snapped segment s that forms part of the result polygon, if s snaps up and to the right by $(\frac{1}{2}, \frac{1}{2})$, the scheme searches the active vulnerable vertices to see whether the snapping creates an incidence in the segment interior. If so, the identity of the output polygon containing s is recorded for later processing.

The second part of the scheme processes any incidences that have been discovered. Fortunately, incidences occur rarely enough in practice that fairly expensive cleanup is affordable. After the polygon Boolean operation completes, if any internal incidences were detected, the cleanup code extracts the edges of all polygons containing an incidence and performs an OR operation again on the extracted edges, without further snapping. Then these updated polygons are renumbered and integrated back into the rest of the output. This, finally, restores correct output.

This is quite ugly and potentially inefficient, as the reader will have observed. With the addition of more data structures, it could be made more efficient, but for now, it seems to be efficient enough for the application. I declared victory and slunk from the field.

8 Lessons learned

The most obvious take-away from this experience is no surprise at all: implementation is hard, and we theoreticians ignore that fact, or sweep it under the rug, at our peril. The requirements of an application often do not precisely match the assumptions of a theoretical algorithm. My mistake in "proving" the correctness of relaxed snap rounding was that I forgot the dependencies of the polygon Boolean operation on the scanline implementation of snap rounding. Although relaxed snap rounding meets its approximation guarantees, it fails to detect all incidences between snapped segments and hot pixel centers, and the polygon Boolean operation depends on incidence detection for correctness. Even if incidence detection is restored with a post-processing step, incidences are not detected in left-to-right order, so the simplicity of the scanline Boolean algorithm is compromised.

As someone who has worked in both the theoretical and applied communities for many years, I think that in some ways, the life of a theoretician is easy – if the problem as posed is too hard, we can solve a related problem and wave our hands to argue that the new problem is just as interesting as the original. The life of an implementor of applications is easy, too – there's no need for an optimal implementation, just one that is correct (enough) and fast enough on the data that occur in practice. However, the life of someone who wears both hats can be very challenging: finding (or inventing) an algorithm that solves the given problem efficiently and is simple enough to implement is hard to do repeatedly. Nevertheless, success, even only occasional success, is quite satisfying when it comes, and makes the struggle worthwhile.

That brings me to the key question of this section, "Have *I* learned *my* lesson?" For good or ill, the answer is *no*. I continue to struggle (actually, to play) at the boundary between theory and practice, trying to find or make theory that is practical enough to use, trying to find practical problems that are elegant enough to appeal to theoreticians, and doing my best to implement elegant, efficient solutions to real-world problems.

References

1. J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, September 1979. doi:10.1109/TC.1979.1675432.
2. M. de Berg, D. Halperin, and M. Overmars. An intersection-sensitive algorithm for snap rounding. *Comp. Geom.: Theory and Appl.*, 36:159–165, 2007. doi:10.1016/J.COMGEO.2006.03.002.
3. W.-K. Chen, editor. *The VLSI Handbook*. CRC Press LLC, 2nd edition, 2006.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.
5. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.
6. M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.
7. D. H. Greene. Integer line segment intersection. Unpublished Manuscript.
8. L. J. Guibas and D. H. Marimont. Rounding arrangements dynamically. *Int. J. Comp. Geometry & Appl.*, 8(2):157–176, 1998. doi:10.1142/S0218195998000096.
9. D. Halperin and E. Packer. Iterated snap rounding. *Comp. Geom.: Theory and Appl.*, 23:209–225, 2002. doi:10.1016/S0925-7721(01)00064-5.
10. J. Hershberger. Improved output-sensitive snap rounding. *Discr. Comput. Geom.*, 39:298–318, 2008. doi:10.1007/S00454-007-9015-0.
11. J. Hershberger. Stable snap rounding. *Comp. Geom.: Theory and Appl.*, 46(4):403–416, 2013. http://dx.doi.org/10.1016/j.comgeo.2012.02.011. doi:10.1016/J.COMGEO.2012.02.011.
12. J. D. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.*, 13(4):199–214, October 1999. doi:10.1016/S0925-7721(99)00021-8.
13. C. Mack. *Fundamental Principles of Optical Lithography: The Science of Microfabrication*. John Wiley and Sons, 2007.
14. E. Packer. Iterated snap rounding with bounded drift. *Comp. Geom.: Theory and Appl.*, 40(3):231–251, 2008. doi:10.1016/J.COMGEO.2007.09.002.
15. L. Scheffer, L. Lavagno, and G. Martin, editors. *Electronic Design Automation for Integrated Circuits Handbook*. Taylor & Francis, Boca Raton, Florida, 2006.

Tracking the Persistence of Harmonic Chains: Barcode and Stability

Tao Hou
University of Oregon, Eugene, OR, USA

Salman Parsa
DePaul University, Chicago, IL, USA

Bei Wang
University of Utah, Salt Lake City, UT, USA

Abstract

The persistence barcode is a topological descriptor of data that plays a fundamental role in topological data analysis. Given a filtration of data, the persistence barcode tracks the evolution of its homology groups. In this paper, we introduce a new type of barcode, called the harmonic chain barcode, which tracks the evolution of harmonic chains. In addition, we show that the harmonic chain barcode is stable. Given a filtration of a simplicial complex of size m, we present an algorithm to compute its harmonic chain barcode in $O(m^3)$ time. Consequently, the harmonic chain barcode can enrich the family of topological descriptors in applications where a persistence barcode is applicable, such as feature vectorization and machine learning.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Topology

Keywords and phrases Persistent homology, harmonic chains, topological data analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.58

Related Version *Full Version*: https://arxiv.org/abs/2412.15419

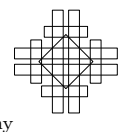
Funding *Tao Hou*: Supported by NSF CCF 2439255.
Salman Parsa: Supported by DOE DE-SC0021015.
Bei Wang: Supported by DOE DE-SC0021015, NSF IIS-2145499 and NSF DMS-2301361.

1 Introduction

There are two primary tasks in topological data analysis (TDA) [27, 30, 53]: reconstruction and inference. In a typical TDA pipeline, the data is given as a point cloud in $\mathbb{R}^N$. A "geometric shape" K in $\mathbb{R}^N$ is reconstructed from the point cloud, usually as a simplicial complex, and K is taken to represent the (unknown) space X from which the data is sampled. X is then studied using K as a surrogate for properties that are invariant under invertible mappings (technically, homeomorphisms). The deduced "topological shape" is not specific to the complex K or the space X, but is a feature of the homeomorphism type of K or X. For example, a standard round circle has the same topological shape as any closed loop such as a knot in the Euclidean space. Although topological properties of X alone are not sufficient to reconstruct X exactly, they are among a few global features of X that can be inferred from the data sampled from X.

An (ordinary) persistence barcode [10, 31] (or equivalently, a persistence diagram [14, 19, 29]) captures the evolution of homological features in a filtration constructed from a simplicial complex K. It consists of a multi-set of intervals in the extended real line, where the start and end points of an interval (i.e., a bar) are the birth and death times of a homological feature in the filtration. Equivalently, a persistence diagram is a multi-set of points in the extended plane, where a point in the persistence diagram encodes the birth and death time of a homological feature.

© Tao Hou, Salman Parsa, and Bei Wang;
licensed under Creative Commons License CC-BY 4.0



A main drawback of ordinary persistent homology is that it does not provide canonical choices of geometric representatives for each bar in the barcode. Specifically, there are two levels of choices in assigning representatives for bars in a persistence barcode: first, we choose a basis for persistent homology in a consistent way across the filtration; second, we choose a cycle representative inside each homology class in the basis. Both levels of choices involve choosing among significantly distinct geometric features of data to represent the same bar in the barcode. Since such choices are not canonical, it can be challenging to interpret their underlying geometric meaning. In addition, while existing works [7, 25, 51] compute optimal representatives for ordinary persistence, the optimality would rely on a choice of weights for the simplices and on the optimality criterion, leading to different shapes. Moreover, computing the optimal representatives is NP-hard in many interesting situations [17, 11, 12, 5, 32, 25].

As our first contribution, we introduce a new type of barcode called *harmonic chain barcode*. This barcode is obtained by tracking the birth and death of harmonic chains in an increasing filtration. Since there is always a *unique* harmonic chain in a homology class, the *harmonic representatives* in our harmonic chain barcode immediately remove the second level of choices in a natural way. The main idea is to take the harmonic chain groups along an increasing filtration, and to observe that the groups grow or shrink due to the birth and death of harmonic chains, resulting in a zigzag module with inclusion maps. Like any zigzag module [8], the module of harmonic chain groups decomposes into interval modules, which then form the harmonic chain barcode. Moreover, since the maps in this zigzag module are essentially inclusions, each representative for a bar in our harmonic chain barcode consists of a single harmonic chain which is alive over the entire bar. We emphasize that our harmonic chain barcode is distinct from the ordinary persistence barcode; see for example Figs. 1 and 2.

The ordinary persistence barcode is shown to be stable [19], which is crucial for applications. The stability means that small changes in the data imply only small changes in the barcode. For our second contribution, we show that our harmonic chain barcode is also stable in the same sense as the stability of persistence barcode [19].

Finally, we present an algorithm for computing the harmonic chain barcode in $O(m^3)$ time for a filtration of size m, matching the complexity of practical algorithms for ordinary persistence.

The interpretation of homological features is crucial for applications such as extracting hierarchical structures of amorphous solids [36] and quantifying the growing branching architectures of plants [42]. Such an interpretation is closely related to the existence of canonical chain representatives for a barcode. In our harmonic chain barcode, using orthogonality, each bar enjoys a canonical choice of representative (within a homology class) which *lives exactly at the time-interval of the bar*. This arguably promises a more interpretable data feature. Therefore, we expect our harmonic chain barcodes to enrich the family of topological descriptors in applications where ordinary persistence barcodes are used, such as feature vectorization and machine learning. Note that just from the fact that our barcode is different from the ordinary persistence one cannot argue in favor or against our suggested barcode. Such a comparison is only meaningful in a specific domain of application and when performed using experimental evaluations. We refer to [35] for an interesting pipeline of data analysis using extra properties of harmonic chains. [35] also contains a method of propagating the information from harmonic chains back to the original data points, which we find also relevant to our harmonic representatives.

2 Related work

Harmonic chains were first studied in the context of functions on graphs, where they were identified as the kernel of the Laplacian operator on graphs [40]. The graph Laplacian and its kernel are important tools in studying graph properties, see [48, 50] for surveys. Eckmann [28] introduced the higher-order Laplacian for simplicial complexes, and proved the isomorphism of harmonic chains and homology. The work [33] studied the stability of higher-order Laplacians. Horak and Jost [37] defined a weighted Laplacian for simplicial complexes. Already their theoretical results on Laplacian [37] anticipated the possibility of applications, as the harmonic chains are thought to contain important geometric information. They are eigenvectors of the 0 eigenvalue of the Laplace operator on simplicial complexes and the spectrum of the Laplace operator has significant geometric information, see e.g. [38]. This has been validated by recent results that use curves of eigenvalues of Laplacians in a filtration in data analysis [18, 52]. The Laplacian was applied to improve the mapper algorithm [49], and for coarsening triangular meshes [39]. The persistent Laplacian [47] and its stability [44] is an active research area. Due to the close relation of harmonic chains and Laplacians, harmonic chains could find applications in areas that Laplacians have been used. Computing reasonable representative cycles for persistent homology is also an active area of research. Here, usually an optimality criterion is imposed on cycles in a homology class to obtain a unique representative. For a single homology class, a number of works [16, 23, 25, 51, 7, 13, 17, 23, 41] consider different criteria for optimality of cycles. Hardness of computing optimal representatives has been studied by [17, 11, 12, 13, 5, 32, 25]. For persistent homology, the authors of [25] studied the hardness of choosing optimal cycles for persistence bars. Furthermore, the study in [20] uses harmonic cycles in a persistent homology setting to compute the persistence barcode. Lieutier [43] studied the harmonic chains in persistent homology classes, called persistent harmonic forms.

Relation to the work of Basu and Cox. The most relevant work to ours is the inspiring work of Basu and Cox [2]. Basu and Cox had a similar goal as ours, namely, to associate geometric information to each bar in order to obtain a more interpretable data feature. To that end, they introduced the notion of *harmonic persistent barcode*, by associating a subspace of harmonic chains to each bar in the *ordinary* persistence barcode. When the multiplicity of the bar is 1, the subspace is 1-dimensional. This is the space of harmonic chains that are born at s and die entering t. In general, this is a quotient subspace of harmonic chains at time s. Using orthogonality, they can represent this subquotient as a subspace of the harmonic cycles at time s. As a result, they successfully assign a canonical harmonic cycle to a bar in the ordinary persistence diagram. We note that, in general, a bar in the persistence barcode cannot be represented using a single harmonic chain that *remains harmonic during the lifetime of the bar*. The harmonic cycle associated to a bar using the Basu-Cox approach is the initial harmonic representative of the bar, that is, the harmonic cycle that represents the bar at its birth. Basu and Cox also used the *terminal harmonic cycle* in some of their arguments, thus showing that the choice is not entirely canonical. However, Basu and Cox proved significant properties of the initial cycles in terms of what they call *relative essential content*. The novelty of our result is that, in contrast to [2], we define a new barcode distinct from the ordinary persistence barcode, in which each bar has a harmonic cycle associated with it that is harmonic during the lifetime of the bar. Basu and Cox also proved stability for their harmonic persistent barcode, by considering subspaces as points of a Grassmannian manifold and measuring distances in the Grassmannian. Such a distance quantifies the angles

between subspaces, whereas our notions of stability are stronger in the sense that they use the classical bottleneck distance analogous to the ordinary persistence homology. The authors of [34] studied a method that permits the construction of stable persistence diagrams that are equipped with a canonical choice of representative cycles for filtrations over arbitrary finite posets. If the underlying poset is a finite subset of the real line, using persistent Laplacians, they obtained harmonic cycles connected (through a certain isomorphism) to those identified by Basu and Cox.

3 Background

In this section, we review the notion of harmonic chains and persistent homology. Homology and cohomology are defined with real coefficients $\mathbb{R}$ (instead of $\mathbb{Z}_2$). Let K be a simplicial complex and p the homology dimension (or equivalently, homology degree). $C_p(K)$, $Z_p(K)$, and $H_p(K)$ denote the p-th chain group, cycle group, and homology group of K, whereas $C^p(K)$, $Z^p(K)$, and $H^p(K)$ denote the p-th cochain group, cocycle group, and cohomology group of K, respectively. Groups across all dimensions are denoted as $C_*(K)$, $C^*(K)$, etc. We use ∂ and δ to denote boundary and coboundary operators, respectively.

3.1 Harmonic cycles

Based on the standard notions of homology and cohomology with $\mathbb{R}$ coefficients, we identify chains and cochains via duality, i.e., $C_p(K) = C^p(K) = \mathbb{R}^{n_p}$, where n_p is the number of p-simplices. Therefore, we can talk about coboundaries of cycles in $Z_p(K)$. We first introduce the notion of harmonic chains.

▶ **Definition 1.** *The* p-th harmonic chain group *of* K*, denoted* $\mathbb{H}_p(K)$*, is the group of* p*-cycles that are also* p*-cocylces. Equivalently,* $\mathbb{H}_p(K) := Z_p(K) \cap Z^p(K)$*. Each element in* $\mathbb{H}_p(K)$ *is called a* harmonic p-chain*. The harmonic chain group in all dimensions is the group* $\mathbb{H}(K) := \bigoplus_p \mathbb{H}_p(K)$*.*

We sometimes use *harmonic cycles* in place of *harmonic chains* to emphasize the fact that harmonic chains are cycles.

▶ **Lemma 2** ([28])**.** $\mathbb{H}_p(K)$ *is isomorphic to* $H_p(K)$ *and* $H^p(K)$*. Specifically, each homology and cohomology class has a unique harmonic cycle in it.*

Harmonic cycles enjoy certain geometric properties. As an example, we mention the following Proposition 3; see [21, Proposition 3] for a proof. In other words, a harmonic cycle is the chain with the least squared-norm in a cohomology class.

▶ **Proposition 3** ([21])**.** *Let* $\alpha \in C^p(K)$ *be a cochain. There is a unique solution* $\overline{\alpha}$ *to the least-squares minimization problem* $\arg\min_{\overline{\alpha}}\{||\overline{\alpha}||^2 \mid \exists \gamma \in C^{p-1}(K); \alpha = \overline{\alpha} + \delta\gamma\}$*. Moreover,* $\overline{\alpha}$ *is characterized by the relation* $\partial\overline{\alpha} = 0$*.*

There is an alternative definition of harmonic cycles. Consider the natural inner product on $C_p(K)$ given by $\langle \sigma_i, \sigma_j \rangle = \delta_{i,j}$. The harmonic chain group can be defined as $\mathbb{H}_p(K) = Z_p(K) \cap B_p(K)^\perp$. With this definition, the isomorphism of Lemma 2 is realized by a map that sends $z + B_p(K)$ to its projection to $B_p(K)^\perp$. In addition, the harmonic cycles satisfy $\mathbb{H}_p(K) = \ker(\partial_p) \cap \ker(\partial_{p+1}^\perp)$. For a short proof of the equivalence among the definitions of harmonic cycles above, see [2]. Importantly, our algorithm relies on the fact that harmonic cycles are cocycles whose boundaries are zero.

3.2 Ordinary persistence

Ordinary persistent homology takes a *filtration* of a simplicial complex K as input. A *continuous* filtration F assigns to each $r \in \mathbb{R}$ a subcomplex $K_r \subseteq K$ such that $K_r \subseteq K_s$ for $r \leq s$. Since K is finite, there are finitely many $t_0, t_1, \ldots, t_m \in \mathbb{R}$ where K_{t_i} changes. We then abuse the notations slightly by letting $K_i := K_{t_i}$ and have a *discrete* form of F,

$$F : \emptyset = K_0 \hookrightarrow K_1 \hookrightarrow \cdots \hookrightarrow K_{m-1} \hookrightarrow K_m = K, \tag{1}$$

where each $K_i \hookrightarrow K_{i+1}$ is an inclusion. Unless stated otherwise, we assume that F is *simplex-wise*, i.e., each two K_i and K_{i+1} differ by at most a single simplex. For simplicity of the exposition, complexes in F sometimes are subscripted by real-valued "timestamps" of the form K_{t_i} (e.g., in Sec. 6) or subscripted by integers of the form K_i (e.g., in Sec. 5), which should not cause any confusions. Applying homology functor to Eq. (1), we obtain a sequence of homology groups and connecting linear maps (homomorphisms), forming a *persistence module*:

$$\mathcal{M} : H_p(K_0) \to H_p(K_1) \to \cdots \to H_p(K_{m-1}) \to H_p(K_m). \tag{2}$$

For $s \leq t$, let $f_p^{s,t} : H_p(K_s) \to H_p(K_t)$ denote the map induced on the p-th homology by inclusion. The image of the map, $f_p^{s,t}(H_p(K_s)) \subseteq H_p(K_t)$, is called the p-th (s,t)-*persistent homology group*, denoted $H_p^{s,t}$. The group $H_p^{s,t}$ consists of homology classes which exist in K_s and survive until K_t. The dimensions of these vector spaces are the *persistent Betti numbers*, denoted $\beta_p^{s,t}$. An *interval module*, denoted $I = I[b,d)$, is a persistence module of the form

$$0 \to \cdots \to 0 \to \mathbb{R} \to \cdots \to \mathbb{R} \to 0 \to \cdots \to 0.$$

In the above, $\mathbb{R}$ is generated by a homology class and the connecting homomorphisms map generator to generator. We have $I_r = \mathbb{R}$ for $b \leq r < d$ and $I_r = 0$ for other r. Any persistence module can be decomposed into a collection of interval modules in a unique way [45]. The collection of $[b,d)$ for all interval modules is called the *persistence barcode*. When plotted as points in an extended plane, the result is the equivalent *persistence diagram*.

Stability of persistence diagram/barcode. The stability of persistence diagrams (or barcodes) is a crucial property for applications. It says that small changes in data lead to small changes in the persistence diagrams. We only review the stability for sublevel-set filtrations here. Let D and D' denote two persistence diagrams. Recall that a persistence diagram is a multi-set of points in the extended plane (each of which is a birth-death pair) which also contains all points on the diagonal. The *bottleneck distance* of D, D' is defined as

$$d_B(D, D') = \inf_\gamma \sup_{p \in D} ||p - \gamma(p)||_\infty,$$

where γ ranges over all bijections between D and D' and $||\cdot||_\infty$ is the largest absolute value of differences of the points' coordinates.

A function $\tilde{f} : |K| \to \mathbb{R}$ is called *simplex-wise linear* if it is linear on each simplex. Let $K_r = \{\sigma \in K \mid \forall x \in |\sigma|, \tilde{f}(x) \leq r\}$ be the *sublevel set complex* at value $r \in \mathbb{R}$. The complexes K_r and the inclusions between them form a *sublevel set filtration* $\tilde{F}$ (which is not necessarily simplex-wise). We denote the persistence diagram of $\tilde{F}$ as $\mathrm{Dgm}(\tilde{f})$. We refer to [19, 14] for proof of the following Theorem 4. See [19, 14] for the stability of ordinary persistence.

▶ **Theorem 4.** *Let $\tilde{f}, \tilde{g} : |K| \to \mathbb{R}$ be simplex-wise linear functions. Then*

$$d_B(\mathrm{Dgm}(\tilde{f}), \mathrm{Dgm}(\tilde{g})) \leq ||\tilde{f} - \tilde{g}||_\infty.$$

3.3 Zigzag persistence

We provide a brief overview of zigzag persistence; see [8, 9] for details. A zigzag module

$$\mathcal{M} : V_0 \xleftrightarrow{g_0} V_1 \xleftrightarrow{g_1} \cdots \xleftrightarrow{g_{k-1}} V_k$$

is a sequence of vector spaces connected by linear maps that could be forward or backward, i.e., each g_i could be $g_i : V_i \to V_{i+1}$ or $g_i : V_i \leftarrow V_{i+1}$. The module $\mathcal{M}$ decomposes into a direct sum of interval modules $I[b, d]$ of the form

$$0 \longleftrightarrow \cdots \longleftrightarrow 0 \longleftrightarrow \mathbb{R} \longleftrightarrow \cdots \longleftrightarrow \mathbb{R} \longleftrightarrow 0 \cdots \longleftrightarrow 0$$

with 1-dimensional vector spaces in the range $[b, d]$. The multi-set of intervals in the decomposition defines the *barcode* of $\mathcal{M}$, denoted as $\mathsf{B}(\mathcal{M})$.

Conventions. In this paper, we may omit the subscript/dimension of a homology group if there is no danger of ambiguity. Moreover, we use the terms persistence barcode and persistence diagram interchangeably.

4 Harmonic chain barcodes and representatives

As reviewed in Sec. 3.2, by directly taking the homology functor, an increasing filtration of simplicial complexes leads to an *ordinary persistence module* consisting of homology groups [15, 27]. These homology groups are connected by forward maps of the form $H(K_i) \to H(K_{i+1})$. The ordinary persistence module then decomposes into interval modules, which define the *ordinary persistence barcode*. In this section, we show that the harmonic chain groups of all the complexes in a filtration constitute an abstract *zigzag persistence module* (see Sec. 3.3). The main idea is that we take the harmonic chain groups along the filtration, where the groups could grow or shrink, resulting in a zigzag module. Moreover, the maps in this zigzag module are inclusions. Like any zigzag module [8], the module of harmonic chain groups decomposes into interval modules. These intervals form the harmonic chain barcode, the main object of interest in this paper.

Throughout the section, consider a simplex-wise filtration

$$F : \emptyset = K_0 \xhookrightarrow{\sigma_0} K_1 \xhookrightarrow{\sigma_1} \cdots \xhookrightarrow{\sigma_{m-1}} K_m = K,$$

where each K_{i+1} differs from K_i by the addition of a simplex σ_i. Recall that $K_i := K_{t_i}$ for $t_i \in \mathbb{R}$ where K_{t_i} is a complex from a *continuous* filtration indexed over $\mathbb{R}$; see Sec. 3.2.

▶ **Proposition 5.** *For each inclusion* $K_i \xhookrightarrow{\sigma_i} K_{i+1}$ *in* F*:*

$$\dim(\mathbb{H}(K_{i+1})) = \dim(\mathbb{H}(K_i)) \pm 1.$$

Proof. This follows from Lemma 2 and some well-known facts in persistence (see [27, 29]). ◀

▶ **Definition 6.** *A simplex* σ_i *inserted in* F *is called* positive *if* $\dim(\mathbb{H}(K_{i+1})) = \dim(\mathbb{H}(K_i)) + 1$, *and* negative *if* $\dim(\mathbb{H}(K_{i+1})) = \dim(\mathbb{H}(K_i)) - 1$.

We describe how we connect the harmonic chain groups of complexes in F by inclusions. For each K_i and each chain $c = \sum_{j=0}^{i-1} \alpha_j \sigma_j$ in $C_p(K_i)$, we identify c as a chain $c = \sum_{j=0}^{m-1} \alpha_j \sigma_j$ in $C_p(K)$, where $\alpha_j = 0$ for $j \geq i$. This makes both $C_p(K_i)$ and $C_p(K_{i+1})$ a subspace of $C_p(K)$. We then have the following inclusion:

$$Z_p(K_i) \hookrightarrow Z_p(K_{i+1}). \tag{3}$$

Recall that $\mathbb{H}_p(K_i) := Z_p(K_i) \cap Z^p(K_i)$, which means that $\mathbb{H}_p(K_i) \subseteq Z_p(K_i) \subseteq C_p(K)$. Hence, we also identify each harmonic cycle in $\mathbb{H}_p(K_i)$ as a chain in $C_p(K)$. We then observe in Theorem 7 a similar inclusion as in Eq. (3) between any harmonic chain groups $\mathbb{H}_p(K_i)$ and $\mathbb{H}_p(K_{i+1})$, with a possible flip on the direction.

▶ **Theorem 7.** *For each arrow $K_i \xhookrightarrow{\sigma_i} K_{i+1}$ in F, where σ_i is a p-simplex:*
- *There is an inclusion $\mathbb{H}_p(K_i) \hookrightarrow \mathbb{H}_p(K_{i+1})$ if σ_i is positive;*
- *And there is an inclusion $\mathbb{H}_{p-1}(K_i) \hookleftarrow \mathbb{H}_{p-1}(K_{i+1})$ if σ_i is negative.*

In addition, in each case the harmonic chain groups in other dimensions remain unchanged, i.e., $\mathbb{H}_q(K_i) = \mathbb{H}_q(K_{i+1})$ for any other $q \notin \{p, p-1\}$.

Proof. The only harmonic chain groups that might change from K_i to K_{i+1} are those in dimension p because cycle and cocycle groups in other dimensions do not change.

First consider Case I that σ_i is positive. Let $\iota : K_i \hookrightarrow K_{i+1}$ be the inclusion map. As noted above, we identify any $c \in C_*(K_i) = C^*(K_i)$ with $\iota_\sharp(c) \in C_*(K_{i+1}) = C^*(K_{i+1})$.

Case I.1: $\mathbb{H}_p(K_i) \subseteq \mathbb{H}_p(K_{i+1})$. Take $z \in \mathbb{H}_p(K_i)$. Obviously, $\iota_\sharp(z)$ is a cycle in K_{i+1}. For any $c \in C_{p+1}(K_{i+1})$, $\delta(\iota_\sharp(z))(c) = \iota_\sharp(z)(\partial c)$. Since $C_{p+1}(K_i) = C_{p+1}(K_{i+1})$, c and hence ∂c exist in K_i, meaning that $\iota_\sharp(z)(\partial c) = z(\partial c) = (\delta z)(c) = 0$. It follows that $\iota_\sharp(z)$ is a cocycle in K_{i+1}. Therefore, $\mathbb{H}_p(K_i) \subseteq \mathbb{H}_p(K_{i+1})$.

Case I.2: $\mathbb{H}_{p-1}(K_i) = \mathbb{H}_{p-1}(K_{i+1})$. Take $z \in \mathbb{H}_{p-1}(K_{i+1})$. Since σ_i is a positive p-simplex, we have $Z_{p-1}(K_{i+1}) = Z_{p-1}(K_i)$ and $B_{p-1}(K_i) = B_{p-1}(K_{i+1})$. So $z \in Z_{p-1}(K_i)$ and we consider z as a cochain in K_i. Then, for any $c \in C_p(K_i)$, $\delta(z)(c) = z(\partial c) = 0$, with the last equality due to $\partial c \in B_{p-1}(K_i) = B_{p-1}(K_{i+1})$. Therefore, $\mathbb{H}_p(K_{i+1}) \subseteq \mathbb{H}_p(K_i)$. Take $z \in \mathbb{H}_{p-1}(K_i)$. For any $c \in C_p(K_{i+1})$, $\delta(\iota_\sharp z)(c) = (\iota_\sharp z)(\partial c) = 0$, with the last equality due to $\partial c \in B_{p-1}(K_{i+1}) = B_{p-1}(K_i)$. Therefore, $\mathbb{H}_{p-1}(K_i) = \mathbb{H}_{p-1}(K_{i+1})$.

Now consider Case II that σ_i is negative.

Case II.1: $\mathbb{H}_p(K_{i+1}) = \mathbb{H}_p(K_i)$. Since σ_i is negative, $Z_p(K_{i+1}) = Z_p(K_i)$ and $B_p(K_{i+1}) = B_p(K_i)$. The verification for this case is then the same as the verification for Case I.2 with a shift on the homology degree.

Case II.2: $\mathbb{H}_{p-1}(K_{i+1}) \subseteq \mathbb{H}_{p-1}(K_i)$. Take $z \in \mathbb{H}_{p-1}(K_{i+1})$. We have $Z_{p-1}(K_{i+1}) = Z_{p-1}(K_i)$. Therefore, $z \in Z_{p-1}(K_i)$ and we consider z as a cochain in K_i. For any $c \in C_p(K_i)$, $\delta(z)(c) = z(\partial c) = 0$, with the last equality due to $\partial c \in B_{p-1}(K_i) \subseteq B_{p-1}(K_{i+1})$. Therefore, $\mathbb{H}_{p-1}(K_{i+1}) \subseteq \mathbb{H}_{p-1}(K_i)$. ◀

▶ **Definition 8** (Harmonic chain barcode). *Consider the following* harmonic zigzag module

$$\mathbb{H}(F) : \mathbb{H}(K_0) \leftrightarrow \mathbb{H}(K_1) \leftrightarrow \cdots \leftrightarrow \mathbb{H}(K_m), \tag{4}$$

where the harmonic chain groups are connected by either forward inclusions (e.g., $\mathbb{H}(K_i) \to \mathbb{H}(K_{i+1})$) or backward inclusions (e.g., $\mathbb{H}(K_i) \leftarrow \mathbb{H}(K_{i+1})$); see Theorem 7. Define the harmonic chain barcode $\mathsf{B}^{\mathbb{H}}(F)$ *of F as the barcode of the zigzag module $\mathbb{H}(F)$, that is,*

$$\mathsf{B}^{\mathbb{H}}(F) := \mathsf{B}(\mathbb{H}(F)).$$

In general, the harmonic chain barcode is different from the ordinary persistence barcode for a filtration; see Fig. 2. In this paper, we also consider the zigzag module $\mathbb{H}_p(F)$ derived by taking $\mathbb{H}_p(K_i)$ on each K_i in F. While the p-th harmonic chain groups in $\mathbb{H}_p(F)$ are still connected by forward or backward inclusions, we may have $\mathbb{H}_p(K_i) = \mathbb{H}_p(K_{i+1})$ for two consecutive groups in $\mathbb{H}_p(F)$. We therefore define the *p-th harmonic chain barcode* $\mathsf{B}^{\mathbb{H}}_p(F)$ of F as the barcode of $\mathbb{H}_p(F)$, i.e., $\mathsf{B}^{\mathbb{H}}_p(F) := \mathsf{B}(\mathbb{H}_p(F))$. Since $\mathbb{H}(F) = \bigoplus_p \mathbb{H}_p(F)$ (following from Theorem 7), we have $\mathsf{B}^{\mathbb{H}}(F) = \bigsqcup_p \mathsf{B}^{\mathbb{H}}_p(F)$.

Harmonic representatives. In the rest of the section, we define harmonic representatives.

▶ **Proposition 9.** *Let $[b,d]$ be an interval in $\mathsf{B}^{\mathbb{H}}(F)$. The inclusion $\mathbb{H}(K_{b-1}) \hookrightarrow \mathbb{H}(K_b)$ in $\mathbb{H}(F)$ is forward. Moreover, if $d < m$, then the inclusion $\mathbb{H}(K_d) \hookleftarrow \mathbb{H}(K_{d+1})$ is backward.*

Proof. This follows from Theorem 7 and the fact that $\dim(\mathbb{H}(K_b)) = \dim(\mathbb{H}(K_{b-1})) + 1$ and $\dim(\mathbb{H}(K_{d+1})) = \dim(\mathbb{H}(K_d)) - 1$. ◀

Representatives for general zigzag modules were introduced in [4, 46] (see also [26]), which consist of a sequence of cycles for each interval. However, since the harmonic chain groups are connected by inclusion maps in $\mathbb{H}(F)$, we have:

▶ **Proposition 10.** *Each representative for an interval in $\mathsf{B}^{\mathbb{H}}(F)$ contains a single harmonic cycle.*

We then adapt the definition of representatives for general zigzag modules [26, 46] and define harmonic representatives as follows:

▶ **Definition 11** (Harmonic representative)**.** *A* harmonic p-representative (*or simply* p-representative) *for an interval $[b,d] \in \mathsf{B}_p^{\mathbb{H}}(F)$ is a p-cycle $z \in \mathbb{H}_p(K_i)$ for $i \in [b,d]$ satisfying:*
Birth condition: *z is born in $\mathbb{H}_p(K_b)$, i.e., $z \in \mathbb{H}_p(K_b) \setminus \mathbb{H}_p(K_{b-1})$ (notice that $b > 0$);*
Death condition: *z dies leaving $\mathbb{H}_p(K_d)$, i.e., $z \in \mathbb{H}_p(K_d) \setminus \mathbb{H}_p(K_{d+1})$ if $d < m$.*

Sometimes we relax the restriction of $[b,d] \in \mathsf{B}_p^{\mathbb{H}}(F)$ and have a harmonic representative for an arbitrary integer interval $[b,d] \subseteq [0,m]$. For details of the original definition of zigzag representatives and justification of Proposition 10 and Definition 11, see the full version of the paper.

5 A cubic-time algorithm for computing harmonic chain barcode

In this section, we propose an $O(m^3)$ algorithm for computing the harmonic chain barcode and its harmonic representatives given a filtration containing m insertions. We first overview the algorithm and then describe the implementation. Although there have been algorithms [9, 22, 26, 46] for computing zigzag barcodes in the general case, these algorithms target zigzag modules induced from directly taking the homology functor on zigzag filtrations. Our algorithm targets the special type of zigzag module $\mathbb{H}(F)$, where harmonic chain groups are derived from an ordinary non-zigzag filtration and are connected by inclusions (on the chain level).

For simplicity, when describing the algorithm, complexes in a filtration are always indexed by integers instead of the real-valued timestamps (see Sec. 3.2). Again, we assume that the input is a simplex-wise filtration $F : \emptyset = K_0 \xhookrightarrow{\sigma_0} K_1 \xhookrightarrow{\sigma_1} \cdots \xhookrightarrow{\sigma_{m-1}} K_m = K$, where each K_{i+1} differs from K_i by the addition of a simplex σ_i.

Algorithm 1 provides an overview: our algorithm computes the harmonic chain barcode by iteratively maintaining the harmonic representatives. It processes the insertions in F one by one and finds *pairings* of the *birth indices* (starting points of the intervals) and *death indices* (ending points of the intervals) to form intervals in $\mathsf{B}^{\mathbb{H}}(F)$. When we encounter a new birth index, it is initially *unpaired*; when we encounter a new death index, an unpaired birth index is chosen to pair with the death index.

The following definition helps present the algorithm.

▶ **Definition 12.** *For an interval $[b,i] \subseteq [0,m]$, a* partial p-representative *for $[b,i]$ is a p-representative as in Definition 11 by ignoring the death condition.*

Algorithm 1 Computing the harmonic chain barcode.

Let U^p be the set of unpaired birth indices for each homology degree p, where $U^p = \emptyset$ initially. Before each iteration that processes the insertion $K_i \xhookrightarrow{\sigma_i} K_{i+1}$, we maintain a p-cycle z for each $b \in U^p$ that is a partial representative for the interval $[b, i]$. We use partial representatives to determine a finalized representative when a birth index is paired with a death index (by making sure the death condition is satisfied).

When processing the insertion of a p-simplex σ_i via $K_i \xhookrightarrow{\sigma_i} K_{i+1}$, we proceed as follows:

If σ_i is positive:

- Since $\dim(\mathbb{H}_p(K_{i+1})) = \dim(\mathbb{H}_p(K_i)) + 1$ (by Theorem 7), we add a new birth index $i+1$ to U^p.
- Find a harmonic p-cycle $z \in \mathbb{H}_p(K_{i+1}) \setminus \mathbb{H}_p(K_i)$, and let z be the partial representative for $i+1 \in U^p$.

If σ_i is negative:

- Since $\dim(\mathbb{H}_{p-1}(K_{i+1})) = \dim(\mathbb{H}_{p-1}(K_i)) - 1$, we have a new death index i.
- Let $U^{p-1} = \{b_j \mid j = 1, \dots, k\}$, where each b_j has a partial $(p-1)$-representative z_j.
- Let $U' = \{b_j \in U^{p-1} \mid z_j \notin \mathbb{H}_{p-1}(K_{i+1})\}$, i.e., U' contains all birth indices in U^{p-1} whose partial representatives do not persist to K_{i+1}.
- Pair the *smallest* (i.e., the "oldest") index $b_{j^*} \in U'$ with i which forms a new interval $[b_{j^*}, i] \in \mathsf{B}^{\mathbb{H}}_{p-1}(F)$. Assign z_{j^*} as the representative for $[b_{j^*}, i] \in \mathsf{B}^{\mathbb{H}}_{p-1}(F)$ and remove i from U^{p-1}.
- Consider each $b_j \in U' \setminus \{b_{j^*}\}$. We have that $z_j \notin \mathbb{H}_{p-1}(K_{i+1})$ because $\delta(z_j)$ becomes non-zero in K_{i+1} (∂z_j is always zero after being born). Also, the only reason that $\delta(z_j) \neq 0$ in K_{i+1} is because $\alpha_j := \delta(z_j)(\sigma_i) = z_j(\partial\sigma_i) \neq 0$. Let $\alpha_* = z_{j^*}(\partial\sigma_i)$. Update the partial representative for b_j as $z_j := z_j - (\alpha_j/\alpha_*) \cdot z_{j^*}$ so that z_j now persists to $\mathbb{H}_{p-1}(K_{i+1})$.

After processing all the insertions, for each b in each U^p with a partial representative z, let $[b, m]$ form an interval in $\mathbb{H}_p(F)$ with a representative z.

▶ **Example.** Fig. 1 provides an example of how Algorithm 1 computes harmonic chain barcode $\mathbb{H}_1(F)$ and representatives for a filtration F, with the resulting barcode in Fig. 2. For simplicity, K_0, K_1, K_2 are omitted and vertex insertions are ignored. In K_4, K_5, and K_7, new 1-cycles are born that are also harmonic cycles due to a lack of triangles. The partial representatives z_4, z_5, z_7 for the indices $U^1 = \{4, 5, 7\}$ all persist till K_7. When the triangle abe is inserted in K_8 (producing a death index 7), $\delta(z_4)(abe) = 1$, $\delta(z_5)(abe) = 1$, and $\delta(z_7)(abe) = 3$. We pair $4 \in U^1$ with 7 and have $[4, 7] \in \mathbb{H}_1(F)$ with a representative z_4. We also let $z_5 := z_5 - z_4$ and $z_7 := z_7 - 3z_4$, which now both persist to $\mathbb{H}_1(K_8)$. The remaining steps are similar. Notice that when abe is inserted, the "alive" bar represented by the boundary of abe (i.e., $ab + be - ae$) is not killed. This is a significant difference from ordinary persistence (which will kill the alive bar represented by $\partial(abe)$ when abe is inserted).

▶ Remark 13. Algorithm 1 chooses the "oldest" birth, among the options, to pair with a death while the ordinary persistence chooses the "youngest" one [29]. A brief explanation is that a summation of representatives *persists* to the next harmonic chain group in $\mathbb{H}(F)$, whereas a summation of representatives in the ordinary persistence *becomes trivial* in the next homology group (and cannot persist). We refer to [26, 46] for a formal definition of the different types of birth/death ends and how they impact the computation of persistence.

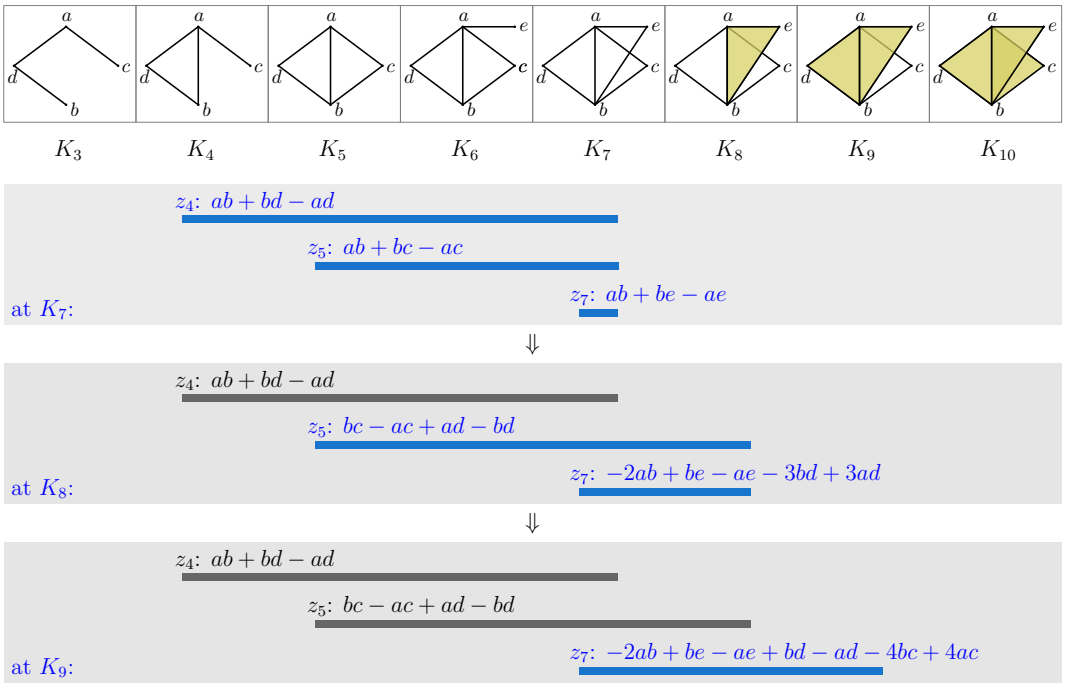


Figure 1 An example of the computation of harmonic chain barcode $\mathbb{H}_1(F)$ and representatives.

(a) Harmonic chain barcode (b) Ordinary persistence barcode

Figure 2 Harmonic chain barcode and ordinary persistence barcode for the filtration in Fig. 1. Deviating from conventions in ordinary persistence, bars are drawn as closed integer intervals, e.g., $[7, 7]$ in the ordinary barcode is killed by the addition of abe in K_8.

The following (rephrased) proposition from [24] helps prove the correctness of Algorithm 1:

▶ **Proposition 14** (Proposition 9, [24]). *If a pairing π of the birth and death indices in $\mathbb{H}(F)$ satisfies that each interval from π has a harmonic representative, then π induces the harmonic chain barcode $\mathsf{B}^{\mathbb{H}}(F)$.*

▶ **Theorem 15.** *Algorithm 1 correctly computes the harmonic chain barcode for filtration F.*

Proof. We first show by induction that before Algorithm 1 processes each σ_i, the p-cycle z maintained for any $b \in U^p$ is a partial p-representative for $[b, i]$. Suppose that the claim is true before processing σ_i. If σ_i is positive, index $i + 1$ is added to U^p and we assign $z \in \mathbb{H}_p(K_{i+1}) \setminus \mathbb{H}_p(K_i)$ to $i + 1$, which is clearly a partial representative for $[i + 1, i + 1]$. For an index b previously in U^p having a partial representative z, since $\mathbb{H}_p(K_i) \subseteq \mathbb{H}_p(K_{i+1})$ (Theorem 7), we have that z is still a partial representative for the interval $[b, i + 1]$.

In the case that σ_i is negative, whenever we update a representative z_j as $z_j - (\alpha_j/\alpha_*) \cdot z_{j^*}$, we have that $b_{j^*} < b_j$ and hence the updated z_j remains born in $\mathbb{H}_{p-1}(K_{b_j})$. Therefore, the new z_j is still a partial representative for $[b, i]$, which is also a partial representative for $[b, i + 1]$ because now $z_j \in \mathbb{H}_{p-1}(K_{i+1})$.

By Proposition 14, we only need to show that each interval output by the algorithm admits a harmonic representative. This follows from the algorithm. For example, whenever we output an interval $[b_{j^*}, i] \in \mathsf{B}^{\mathbb{H}}_{p-1}(F)$ when σ_i is negative, we have $z_{j^*} \notin \mathsf{B}^{\mathbb{H}}_{p-1}(K_{i+1})$, which ensures the death condition. ◀

Implementation. Due to space limitations, we only briefly describe the gist of the implementation; refer to the full version of the paper for details and missing proofs. The only step in Algorithm 1 that cannot be easily implemented is finding a new harmonic cycle born in $\mathbb{H}_p(K_{i+1})$ when σ_i is positive. In this case, the dimension of the harmonic chain group increases and we need to find a single harmonic cycle independent of the previous harmonic group. We can do this naively by computing a basis for the kernel of δ in K_{i+1}, and check all cocycles in the basis to see if they are independent of the existing set of harmonic representatives maintained for K_i. This, however, requires at least $\Theta(m^3)$ per simplex insertion.

For a more efficient implementation, before iteration i that processes the insertion $K_i \xhookrightarrow{\sigma_i} K_{i+1}$, we maintain the following matrices for each homology degree p:

1. $\mathtt{H}^p$: Columns in $\mathtt{H}^p$ are partial p-representatives for all indices in U^p.
2. $\mathtt{R}^p$: Columns in $\mathtt{R}^p$ form a boundary basis with distinct pivots for $B_p(K_i)$.
3. $\mathtt{Cob}^p$: Columns in $\mathtt{Cob}^p$ form a coboundary basis for $B^p(K_i)$.
4. $\mathtt{SC}^p$: Columns in $\mathtt{SC}^p$ encode "p-pseudo-cochains" (see full version for definition) in K_i based on the basis provided in $\mathtt{R}^p$, whose coboundaries are columns in $\mathtt{Cob}^{p+1}$.
5. $\mathtt{BoC}^p$: Columns in $\mathtt{BoC}^p$ are boundaries of the coboundaries in $\mathtt{Cob}^{p+1}$, with distinct pivots.

In summary, columns in $\mathtt{SC}^{p-1}$, $\mathtt{Cob}^p$, and $\mathtt{BoC}^{p-1}$ have one-to-one correspondence as follows:

$$\mathtt{SC}^{p-1}[j] \overset{\delta}{\longmapsto} \mathtt{Cob}^p[j] \overset{\partial}{\longmapsto} \mathtt{BoC}^{p-1}[j].$$

Assuming σ_i is a p-simplex, the rationale for maintaining the matrices is as follows:

- We use $\mathtt{BoC}^{p-1}$ to find a linear combination ϕ of $\hat{\sigma}_i$ and cocycles in $\mathtt{Cob}^p$ such that $\partial\phi = 0$ (see Algorithm 2). The cocycle ϕ is then a new harmonic p-cycle when σ_i is positive (see Proposition 16).
- $\mathtt{SC}^{p-1}$ helps maintain the coboundary basis in $\mathtt{Cob}^p$.
- $\mathtt{R}^{p-1}$ provides the boundary basis on which pseudo-cochains in $\mathtt{SC}^{p-1}$ can be defined (again, refer to the full version for details of pseudo-cochains).

Algorithm 2 Reduction based on $\mathtt{BoC}^{p-1}, \mathtt{Cob}^p$ for finding a new-born harmonic cycle.

```
φ ← σ̂_i
ζ ← ∂(σ̂_i)
while ζ ≠ 0 and ∃j s.t. pivot(BoC^{p-1}[j]) = pivot(ζ) do
    k ← pivot(ζ)
    α1 ← ζ[k]
    α2 ← BoC^{p-1}[j][k]
    ζ ← ζ − (α1/α2) · BoC^{p-1}[j]
    φ ← φ − (α1/α2) · Cob^p[j]
```

▶ **Proposition 16.** *If Algorithm 2 ends with $\zeta = 0$, then σ_i is positive and ϕ is a new harmonic p-cycle born in $\mathbb{H}_p(K_{i+1})$; otherwise, σ_i is negative.*

Besides finding a new harmonic cycle when σ_i is positive, our algorithm also needs to do the following: (1) update $\texttt{Cob}^p$ (and also $\texttt{BoC}^{p-1}$) because columns of $\texttt{Cob}^p$ may not be coboundaries anymore as we proceed; (2) update $\texttt{BoC}^{p-1}$ (and also $\texttt{SC}^{p-1}$, $\texttt{Cob}^p$) to make pivots in $\texttt{BoC}^{p-1}$ distinct again; (3) add a new coboundary to $\texttt{Cob}^p$ (and new columns to $\texttt{SC}^{p-1}$, $\texttt{BoC}^{p-1}$) to form a coboundary basis for $B_p(K_{i+1})$ when σ_i is negative. See the full version for details. We conclude the following:

▶ **Theorem 17.** *The harmonic chain barcode of F can be computed in $O(m^3)$ time.*

6 Sublevel set harmonic chain barcode and its stability

In this section, we briefly introduce the notion of *sublevel set harmonic chain barcodes* and present our stability results based on this notion; for details and proofs, refer to the full version of the paper. Our stability proof makes use of the work on the algebraic stability of block decomposable $\mathbb{R}^2$-modules [1, 6]. In brief, we "lift" the 1-parameter harmonic zigzag module to an $\mathbb{R}^2$-indexed 2-parameter persistence module which is block decomposable. We then show that in the typical setting of sublevel set filtrations, there exists an interleaving between the lifted modules. Our main work here is to construct a concrete extension to $\mathbb{R}^2$-modules and an interleaving between the extensions realizing the category-theoretic concepts employed in [1, 6]. From the Isometry Theorem [3, 6], we then deduce the stability of the sublevel set harmonic chain barcode. We also address the disparity between the natural harmonic chain barcode of a sublevel set filtration (which consists of closed-open intervals[1]) and common conventions of zigzag persistence (which work with closed-closed intervals in our setting).

We assume, up until now, that a filtration is simplex-wise. This assumption is not true for a sublevel set filtration, but we shall construct a simplex-wise filtration from a general filtration.

Let $\tilde{F}$ be an increasing filtration of K that is not necessarily simplex-wise, i.e., two consecutive complexes in $\tilde{F}$ can differ by more than one simplex. We fix an ordering, once and for all, for vertices of K, which also fixes an ordering for all simplices in K (using, say, the lexicographic ordering of the ordered sequence of vertices in a simplex). Given $\delta > 0$ small enough, we expand $\tilde{F}$ into a simplex-wise filtration $\tilde{F}(\delta)$ by expanding each inclusion $K_{t_{i-1}} \hookrightarrow K_{t_i}$ in $\tilde{F}$ starting with $i = 1$. To differentiate, denote each complex in $\tilde{F}(\delta)$ as K'_t where $t \in \mathbb{R}$ is a timestamp. In each iteration, we expand the inclusion $K_{t_{i-1}} \hookrightarrow K_{t_i}$ in $\tilde{F}$ into several simplex-wise inclusions in $\tilde{F}(\delta)$:

$$K_{t_{i-1}} = K'_s \hookrightarrow K'_{t_i-\delta} \hookrightarrow K'_{t_i} \hookrightarrow K'_{t_i+\delta} \hookrightarrow \cdots \hookrightarrow K'_{t_i+(k-1)\delta} = K_{t_i}.$$

In the above, $K'_{t_i-\delta}$ is a "dummy" complex equal to K'_s (so $K'_s \hookrightarrow K'_{t_i-\delta}$ is an identity map) and k is the number of simplices added from $K_{t_{i-1}}$ to K_{t_i} in $\tilde{F}$. We also have that the k number of simplices are added based on their simplex ordering described above.

To see the rationale of introducing $\tilde{F}(\delta)$, we first define the following:

▶ **Definition 18.** *Let $\tilde{F}$ be an increasing filtration of complex K that is not necessarily simplex-wise and let $z \in Z_p(K)$ be a p-cycle. Let $t_{b(z)}$ be the time when z is born, i.e., $z \in Z_p(K_{t_{b(z)}}) \setminus Z_p(K_{t_{b(z)-1}})$, and $t_{d(z)}$ be the time when z dies as a harmonic chain, i.e.,*

[1] The closed-open intervals over the real values we have here are different from the closed-open intervals of [1, 6] over the integers. Indeed, when considering only integer indices, we only have closed-closed intervals in our setting. The extension of our closed-open interval would be similar to a closed block of [6] but with the right vertical side open.

$\delta(z) \neq 0$ in $K_{t_{d(z)}}$, but $\delta(z) = 0$ in K_t for $t_{b(z)} \leq t < t_{d(z)}$. Define the span *of z as $span(z) := [t_{b(z)}, t_{d(z)})$. If z is not harmonic at birth, its span is the empty set (notice that a cycle continues to be non-harmonic once it becomes so). Moreover, define the* bar *of z as $bar(z) := [t_{b(z)}, t_{d(z)-1}]$.*

Based on the construction, we observe a relation between the bar of z in $\tilde{F}(\delta)$ and the span of z in $\tilde{F}$ for a cycle z. Moreover, adding the dummy complexes makes sure that the closed bars of the simplex-wise filtration $\tilde{F}(\delta)$ "induce" the closed-open time intervals in $\tilde{F}$ in which a cycle is harmonic (see full version for details). In particular, for any small δ, we obtain an obvious mapping $\Gamma : \tilde{F}(\delta) \to \tilde{F}$. We then have the following Definition 19.

▶ **Definition 19.** *Let F be a simplex-wise filtration (as in Sec. 3.2) and let $\breve{\mathsf{B}}^{\mathbb{H}}(F)$ be the* closed-open barcode of $\mathbb{H}(F)$*, i.e., $[t_b, t_d) \in \breve{\mathsf{B}}^{\mathbb{H}}(F)$ iff $[t_b, t_{d-1}] \in \mathsf{B}^{\mathbb{H}}(F)$. Moreover, let $\tilde{F}$ and $\tilde{F}(\delta)$ be as defined in this section. We define the* harmonic chain barcode of $\tilde{F}$ *as the set $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F}) := \Gamma(\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F}(\delta)))$ for δ small enough.*

▶ Remark 20. Note that $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F})$ is the limit of $\mathsf{B}^{\mathbb{H}}(\tilde{F}(\delta))$ as δ approaches 0.

We now state our stability result for sublevel set filtrations. See the full version for the proof of this theorem (and additional observations).

▶ **Theorem 21.** *Let $\hat{f}, \hat{g} : |K| \to \mathbb{R}$ be simplex-wise linear functions, and let $\tilde{F}$ and $\tilde{G}$ be the sublevel set filtrations of $\hat{f}$ and $\hat{g}$ respectively. Then,*

$$d_B(\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F}), \breve{\mathsf{B}}^{\mathbb{H}}(\tilde{G})) \leq ||\hat{f} - \hat{g}||_\infty.$$

Inducing the barcode $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F})$ from $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F}(\delta))$ allows us to use the off-the-shelf results of [6] for proving stability and does not affect the computation of $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F})$. In particular, we have the following.

▶ **Theorem 22.** *Given a sublevel-set filtration $\tilde{F}$ of the complex K with m simplices, the harmonic chain barcode $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F})$ can be computed in $O(m^3)$ time.*

Proof. The computation of $\breve{\mathsf{B}}^{\mathbb{H}}(\tilde{F})$ is equivalent to the computation of $\mathsf{B}^{\mathbb{H}}(\tilde{F}(\delta))$ for a single δ small enough, using the algorithm in Sec. 5 and mapping the resulting bars using Γ to the filtration $\tilde{F}$. This simply amounts to changing timestamps of the form $t_i \pm \ell\delta$ into t_i. ◀

References

1. Håvard Bakke Bjerkevik. On the stability of interval decomposable persistence modules. *Discrete & Computational Geometry*, 66(1):92–121, 2021. doi:10.1007/S00454-021-00298-0.
2. Saugata Basu and Nathanael Cox. Harmonic persistent homology. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1112–1123. IEEE, 2022.
3. Ulrich Bauer and Michael Lesnick. Induced matchings of barcodes and the algebraic stability of persistence. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 355–364, 2014.
4. Paul Bendich, Herbert Edelsbrunner, Dmitriy Morozov, and Amit Patel. Homology and robustness of level and interlevel sets. *Homology, homotopy, and applications*, 15(1):51–72, 2013.
5. Glencora Borradaile, William Maxwell, and Amir Nayyeri. Minimum bounded chains and minimum homologous chains in embedded simplicial complexes. In *Proceedings of the 36th International Symposium on Computational Geometry (SoCG 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

6 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & geometric topology*, 18(6):3133–3204, 2018.

7 Oleksiy Busaryev, Tamal K. Dey, and Yusu Wang. Tracking a generator by persistence. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics*, pages 278–287, 2010. doi:10.1007/978-3-642-14031-0_31.

8 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405, 2010. doi:10.1007/S10208-010-9066-0.

9 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 247–256, 2009. doi:10.1145/1542362.1542408.

10 Gunnar Carlsson, Afra J. Zomorodian, Anne Collins, and Leonidas J. Guibas. Persistence barcodes for shapes. *Proceedings Eurographs/ACM SIGGRAPH Symposium on Geometry Processing*, pages 124–135, 2004. doi:10.2312/SGP/SGP04/127-138.

11 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.

12 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th International Symposium on Computational Geometry (SoCG 2009)*. ACM Press, 2009. doi:10.1145/1542362.1542426.

13 Erin Wolf Chambers, Salman Parsa, and Hannah Schreiber. On complexity of computing bottleneck and lexicographic optimal cycles in a homology class. In *Proceedings of the 38th International Symposium on Computational Geometry (SoCG)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

14 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *25th Annual Symposium on Computational Geometry (SoCG 2009)*, pages 237–246, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1542362.1542407.

15 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.

16 Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Computational Geometry*, 43(2):169–181, 2010. Special Issue on the 24th European Workshop on Computational Geometry (EuroCG'08). doi:10.1016/j.comgeo.2009.06.004.

17 Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete & Computational Geometry*, 45(3):425–448, 2011. doi:10.1007/S00454-010-9322-8.

18 Jiahui Chen, Yuchi Qiu, Rui Wang, and Guo-Wei Wei. Persistent Laplacian projected Omicron BA. 4 and BA. 5 to become new dominating variants. *Computers in Biology and Medicine*, 151:106262, 2022. doi:10.1016/J.COMPBIOMED.2022.106262.

19 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Proceedings of the 21st Annual Symposium on Computational Geometry*, pages 263–271, 2005. doi:10.1145/1064092.1064133.

20 Alessandro De Gregorio, Marco Guerra, Sara Scaramuccia, and Francesco Vaccarino. Parallel decomposition of persistence modules through interval bases. *arXiv preprint*, 2021. arXiv:2106.11884.

21 Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, 45(4):737–759, 2011. doi:10.1007/S00454-011-9344-X.

22 Tamal K. Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 345–354, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2582112.2582165.

23 Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 221–230, 2010. doi:10.1145/1806689.1806721.

24 Tamal K. Dey and Tao Hou. Computing zigzag persistence on graphs in near-linear time. In *37th International Symposium on Computational Geometry*, 2021.

25 Tamal K Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2587–2606. SIAM, 2020. doi:10.1137/1.9781611975994.158.

26 Tamal K. Dey, Tao Hou, and Dmitriy Morozov. A fast algorithm for computing zigzag representatives. *ACM-SIAM Symposium on Discrete Algorithms (SODA) 2025,* to appear, 2025.

27 Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

28 Beno Eckmann. Harmonische Funktionen und Randwertaufgaben in einem Komplex. *Commentarii Mathematici Helvetici*, 17(1):240–255, 1944.

29 Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. doi:10.1007/S00454-002-2885-2.

30 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.

31 Robert Ghrist. Barcodes: The persistent topology of data. *Bullentin of the American Mathematical Society*, 45:61–75, 2008.

32 Joshua A. Grochow and Jamie Tucker-Foltz. Computational topology and the unique games conjecture. In *Proceedings of 34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

33 Nicola Guglielmi, Anton Savostianov, and Francesco Tudisco. Quantifying the structural stability of simplicial homology. *arXiv preprint*, 2023. doi:10.48550/arXiv.2301.03627.

34 Aziz Burak Gülen, Facundo Mémoli, and Zhengchao Wan. Orthogonal Möbius inversion and Grassmannian persistence diagrams. arXiv preprint, 2024. arXiv:2311.06870.

35 Davide Gurnari, Aldo Guzmán-Sáenz, Filippo Utro, Aritra Bose, Saugata Basu, and Laxmi Parida. Probing omics data via harmonic persistent homology. arXiv preprint, 2023. arXiv:2311.06357.

36 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.

37 Danijela Horak and Jürgen Jost. Spectra of combinatorial Laplace operators on simplicial complexes. *Advances in Mathematics*, 244:303–336, 2013.

38 Parvaneh Joharinad and Jürgen Jost. *Mathematical principles of topological and geometric data analysis*. Springer, 2023.

39 Alexandros Keros and Kartic Subr. Spectral coarsening with Hodge Laplacians. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

40 G. Kirchhoff. Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847. doi:10.1002/andp.18471481202.

41 Hyekyoung Lee, Moo K Chung, Hongyoon Choi, Hyejin Kang, Seunggyun Ha, Yu Kyeong Kim, and Dong Soo Lee. Harmonic holes as the submodules of brain network and network dissimilarity. In *Computational Topology in Image Context: 7th International Workshop, CTIC 2019, Málaga, Spain, January 24-25, 2019, Proceedings 7*, pages 110–122. Springer, 2019. doi:10.1007/978-3-030-10828-1_9.

42 Mao Li, Keith Duncan, Christopher N. Topp, and Daniel H. Chitwood. Persistent homology and the branching topologies of plants. *American Journal of Botany*, 104(3):349–353, 2017.

43 Andre Lieutier. Persistent harmonic forms. https://project.inria.fr/gudhi/files/2014/10/Persistent-Harmonic-Forms.pdf.

44 Jian Liu, Jingyan Li, and Jie Wu. The algebraic stability for persistent Laplacians. *arXiv preprint*, 2023. arXiv:2302.03902.

45 Jiajie Luo and Gregory Henselman-Petrusek. Interval decomposition for persistence modules freely generated over PIDs. arXiv preprint, 2023. arXiv:2310.07971.

46 Clément Maria and Steve Y. Oudot. Zigzag persistence via reflections and transpositions. 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 181–199. SIAM, 2015. doi:10.1137/1.9781611973730.14.

47 Facundo Mémoli, Zhengchao Wan, and Yusu Wang. Persistent Laplacians: Properties, algorithms and implications. *SIAM Journal on Mathematics of Data Science*, 4(2):858–884, 2022. doi:10.1137/21M1435471.

48 Russell Merris. Laplacian matrices of graphs: a survey. *Linear Algebra and its Applications*, 197-198:143–176, 1994. doi:10.1016/0024-3795(94)90486-3.

49 Joshua Mike and Jose Perea. Multiscale geometric data analysis via Laplacian eigenvector cascading. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1091–1098. IEEE, 2019. doi:10.1109/ICMLA.2019.00183.

50 Bojan Mohar, Y Alavi, G Chartrand, and OR Oellermann. The Laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991.

51 Ippei Obayashi. Volume-optimal cycle: Tightest representative cycle of a generator in persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 2(4):508–534, 2018. doi:10.1137/17M1159439.

52 Rui Wang, Duc Duy Nguyen, and Guo-Wei Wei. Persistent spectral graph. *International journal for numerical methods in biomedical engineering*, 36(9):e3376, 2020.

53 Afra J Zomorodian. *Topology for computing*, volume 16. Cambridge university press, 2005.

Incremental Planar Nearest Neighbor Queries with Optimal Query Time

John Iacono
Université libre de Bruxelles, Belgium

Yakov Nekrich
Michigan Technological University, Houghton, MI, USA

Abstract

In this paper we show that two-dimensional nearest neighbor queries can be answered in optimal $O(\log n)$ time while supporting insertions in $O(\log^{1+\varepsilon} n)$ time. No previous data structure was known that supports $O(\log n)$-time queries and polylog-time insertions. In order to achieve logarithmic queries our data structure uses a new technique related to fractional cascading that leverages the inherent geometry of this problem. Our method can be also used in other semi-dynamic scenarios.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Data Structures, Dynamic Data Structures, Nearest Neighbor Queries

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.59

Related Version *Full Version*: `https://arxiv.org/abs/2504.07366` [15]

Funding *John Iacono*: Research supported by the Fonds de la Recherche Scientifique – FNRS.
Yakov Nekrich: Supported by the National Science Foundation under NSF grant 2203278.

1 Introduction

In the nearest neighbor problem a set of points S is stored in a data structure so that for a query point q the point $p \in S$ that is closest to q can be found efficiently. The nearest neighbor problem and its variants are among the most fundamental and extensively studied problems in computational geometry; we refer to [19] for a survey. In this paper we study dynamic data structures for the Euclidean nearest neighbor problem in two dimensions. We show that the optimal $O(\log n)$ query time for this problem can be achieved while allowing insertions in time $O(\log^{1+\epsilon} n)$.

Previous Work

See Table 1. In the static scenario the planar nearest neighbor problem can be solved in $O(\log n)$ time by point location in Voronoi diagrams. However the dynamic variant of this problem is significantly harder because Voronoi diagrams cannot be dynamized efficiently: it was shown by Allen et al. [2] that a sequence of insertions can lead to $\Omega(\sqrt{n})$ amortized combinatorial changes per insertion in the Voronoi diagram. A static nearest-neighbor data structure can be easily transformed into an insertion-only data structure using the logarithmic method of Bentley and Saxe [3] at the cost of increasing the query time to $O(\log^2 n)$. Several researchers [9, 11, 20] studied the dynamic nearest neighbor problem in the situation when the sequence of updates is random in some sense (e.g. the deletion of any element in the data structure is equally likely). However their results cannot be extended to the case when the complexity of a specific sequence of updates must be analyzed.

Using a lifting transformation [10], 2-d nearest neighbor queries can be reduced to extreme point queries on a 3-d convex hulls. Hence data structures for the dynamic convex hull in 3-d can be used to answer 2-d nearest neighbor queries. The first such data structure (without

© John Iacono and Yakov Nekrich;
licensed under Creative Commons License CC-BY 4.0

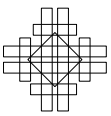


Table 1 Known results. Insertion and deletion times are amortized, † denotes in expectation.

	Query	Insert	Delete
Bentley and Saxe 1980 [3]	$O(\log^2 n)$	$O(\log^2 n)$	Not supported
Agarwal and Matoušek 1995 [1]	$O(\log n)$	$O(n^\varepsilon)$	$O(n^\varepsilon)$
"	$O(n^\varepsilon)$	$O(\log n)$	$O(\log n)$
Chan 2010 [4]	$O(\log^2 n)$	$O(\log^3 n)$ †	$O(\log^6 n)$ †
Chan and Tsakalidis 2016 [6]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^6 n)$
Kaplan et al. 2020 [16]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^5 n)$
Chan 2020 [5]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^4 n)$
Here	$O(\log n)$	$O(\log^{1+\varepsilon} n)$	Not supported

assumptions about the update sequence) was presented by Agarwal and Matoušek [1]. Their data structure supports queries in $O(\log n)$ time and updates in $O(n^\varepsilon)$ time; another variant of their data structure supports queries in $O(n^\varepsilon)$ time and updates in $O(\log n)$ time. A major improvement was achieved in a seminal paper by Chan [4] where he presented a structure that supports queries in $O(\log^2 n)$ time, insertions in $O(\log^3 n)$ expected time and deletions in $O(\log^6 n)$ expected time. The update procedure can be made deterministic using the result of Chan and Tsakalidis [6]. The deletion time was further reduced to $O(\log^5 n)$ [16] and to $O(\log^4 n)$ [5]. This sequence of papers makes use of shallow cuttings, a general powerful technique, but, alas, all uses of it for the point location problem in 2-d have resulted in $O(\log^2 n)$ query times.

Even in the case of insertion-only scenario, the direct application of the 45-year-old classic technique of Bentley and Saxe [3] remains the best insertion-only method with polylogarithmic update before this work; no data structure with $o(\log^2 n)$ query time and polylogarithmic update time was described previously.

Our Results

We demonstrate that optimal $O(\log n)$ query time and poly-logarithmic update time can be achieved in some dynamic settings. The following scenarios are considered in this paper:

1. We describe a *semi-dynamic insertion-only* data structure that uses $O(n)$ space, supports insertions in $O(\log^{1+\varepsilon} n)$ amortized time and answers queries in $O(\log n)$ time.
2. In the *semi-online* scenario, introduced by Dobkin and Suri [12], we know the deletion time of a point p when a point p is inserted, i.e., we know how long a point will remain in a data structure at its insertion time. We describe a semi-online fully-dynamic data structure that answers queries in $O(\log n)$ time and supports updates in $O(\log^{1+\varepsilon} n)$ amortized time. The same result is also valid in the *offline* scenario when the entire sequence of updates is known in advance.
3. In the *offline partially persistent scenario*, the sequence of updates is known and every update creates a new version of the data structure. Queries can be asked to any version of the data structure. We describe an offline partially persistent data structure that uses $O(n \log^{1+\varepsilon} n)$ space, can be constructed in $O(n \log^{1+\varepsilon} n)$ time and answers queries in $O(\log n)$ time.

All three problems considered in this paper can be reduced to answering point location queries in (static) Voronoi diagrams of $O(\log n)$ different point sets. For example, we can obtain an insertion-only data structure by using the logarithmic method of Bentley and Saxe [3], which we now briefly describe. The input set S is partitioned into a logarithmic number of subsets $S_1, \ldots, S_f$ of exponentially increasing sizes. In order to find the nearest neighbor of some query point q we locate q in the Voronoi diagram of each set S_i and report the point closest to q among these nearest neighbors. Since each point location query takes $O(\log n)$ time, answering a logarithmic number of queries takes $O(\log^2 n)$ time.

The fractional cascading technique [7] applied to this problem in one dimension decreases the query cost to logarithmic by sampling elements of each S_i and storing copies of the sampled elements in other sets S_j, $j < i$. Unfortunately, it was shown by Chazelle and Liu [8] that fractional cascading does not work well for two-dimensional non-orthogonal problems, such as point location: in order to answer $O(\log n)$ point location queries in $O(\log n)$ time, we would need $\tilde{\Omega}(n^2)$ space, even in the static scenario.

To summarize, the two obvious approaches to the insertion-only problem are to maintain a single search structure and update it with each insertion, the second is to maintain a collection of static Voronoi diagrams of exponentially-increasing size and to execute nearest neighbor queries by finding the closest point in all structures, perhaps aided by some kind of fractional cascading. The first approach cannot obtain polylogarithmic insertion time due to the lower bound on the complexity change in Voronoi diagrams caused by insertions [2], and the second approach cannot obtain $O(\log n)$ search time due to Chazelle and Liu's lower bound [8]. Our main intellectual contribution is showing that the lower bound of Chazelle and Liu [8] can be circumvented for the case of point location in Voronoi diagrams. Specifically, a strict fractional cascading approach requires finding the closest point to a query point in each of the subsets S_i; we loosen this requirement: in each S_i, we either find the closest point or provide a certificate that the closest point in S_i is not the closest point in S. This new, powerful and more flexible form of fractional cascading is done by using a number of novel observations about the geometry of the problem. We imagine our general technique may be applicable to speeding up search in other dynamic search problems. Our method employs planar separators to sample point sets and uses properties of Voronoi diagrams to speed up queries. We explain our method and show how it can be applied to the insertion-only nearest neighbor problem in Section 2.

A further modification of our method, that is described in the full version of this paper [15], improves the insertion time and the insertion time to $O(\log^{1+\varepsilon} n)$ and space usage to $O(n)$. The description of partially persistent and semi-online variants is also in the full version.

2 Basic Insertion-Only Structure

We present our basic insertion only structure in several parts. In the first part, the overview (§2.1), we present the structure where one needed function, jump, is presented as a black box. With the jump function abstracted, our structure is a combination of known techniques, notably the logarithmic method of Bentley and Saxe [3] and sampling. In §2.2 we present the implementation of the jump function and the needed geometric preliminaries. In contrast to combination of standard techniques presented in the overview which require little use of geometry, our implementation of the jump function is novel and requires thorough geometric arguments. We then fully describe the underlying data structures needed in §2.3. Note that all arguments presented here are done with the goal of obtaining $O(\log n)$ queries and polylogarithmic insertion. We opt here for clarity of presentation versus reducing the number of logarithmic factors in the insertion, as the arguments are already complex.

2.1 Overview

We let S denote the set of points currently stored in the structure, and use n to denote $|S|$. In this section we set a constant d to 2. Even though for this section d is fixed, we will express everything as a function of d.

Let $\mathcal{S} = \{S_1, S_2, \dots S_f\}$ denote a partition of S into sets of exponentially-increasing size where $f \coloneqq |\mathcal{S}| = \Theta(\log_d n)$ and $|S_i| = \Theta(d^i)$. Note that the partition of $\mathcal{S}$ into S_i is not unique. Let $NN(P, q)$ be the nearest neighbor of q in a point set P, which we assume to be unique. Given a point q, the computation of $NN(S, q)$ is the query that our data structure will support.

We now define a sequence of point sets $T_1, \dots T_f$. The intuition is that, as in classical fractional cascading [7], the set T_i contains all elements of S_i and a sample of elements from the sets T_j where $j > i$; this implies the last sets are equal: $T_f = S_f$. This sampling will be provided by the function $Sample_j(k)$ which returns a subset of T_j of size $O(|T_j|/d^{2k})$; while it will have other important properties, for now only the size matters.

We now can formally define T_i: $T_i \coloneqq S_i \cup \bigcup_{j=i+1}^{f} Sample_j(j-i)$. From this definition we have several observations which we group into a lemma, the proof can be found in the full version of this paper:

▶ **Lemma 1. Facts about T_i**

1. $T_f = S_f$
2. *T_i is a function of the S_j, for $j \geq i$.*
3. $S = \cup_{i=1}^{f} T_i$
4. $NN(S, q) \in \bigcup_{i=1}^{f} \{NN(T_i, q)\}$
5. $|T_i| = \Theta(d^i)$
6. *For any i* $\sum_{j=i+1}^{f} |Sample_j(j-i)| = \Theta(|T_i|)$

A note on notation

We assume the partition of S into the sets $\mathcal{S} = \{S_1, S_2, \dots, S_f\}$. Any further notation that includes a subscript, such as T_i, is a function of the S_j, for $j \geq i$. This compact notation makes explicit the dependence of $anything_i$ only on S_j, for $j \geq i$. This dependence in one direction only (i.e., on non-smaller indexes only) is crucial to the straightforward application of the standard Bentley and Saxe [3] rebuilding technique in the implementation of the data structure and the insertion operation described in section §2.3-2.4.

Voronoi and Delaunay

Let $Vor(P)$ be the Voronoi diagram of point set P, let $Cell(P, p)$ be the cell of a point p in $Vor(P)$, that is the locus of points in the plane whose closest element in P is p. Thus $q \in Cell(P, p)$ is equivalent to $NN(P, q) = p$. Let $|Cell(P, p)|$ be the complexity of the cell, that is, the number of edges on its boundary. Let $G(P)$ refer to the Delaunay graph of P, the dual graph of the Voronoi diagram of P; the degree of p in $G(P)$ is thus $|Cell(P, p)|$ and each point in P corresponds to a unique vertex in $G(P)$. Delaunay graphs are planar. To simplify the description, we will not distinguish between points in a planar set P and vertices of $G(P)$. For example, we will sometimes say that a point p has degree x or say that a point p' is a neighbor of p. We will find it useful to have a compact notation for expressing the union of Voronoi cells; thus for a set of points $P' \subseteq P$, let $Cells(P, P')$ denote $\bigcup_{p \in P'} Cell(P, p)$.

Pieces and Fringes

Given a graph, $G = (V, E)$, and a set of vertices $V' \subseteq V$, the *fringe* of V' (with respect to G) is the subset of V' incident to edges whose other endpoint is in $V \setminus V'$. Let $G = (V, E)$ be a planar graph. For any ρ, Frederickson [13] showed the vertices of G can be decomposed[1] into $\Theta(|V|/\rho)$ *pieces*, so that: (i) Each vertex is in at least one piece. (ii) Each piece has at most ρ vertices in total and only $O(\sqrt{\rho})$ vertices on its fringe. (iii) If a vertex is a non-fringe vertex of a piece (with respect to G), then it is not in any other pieces. (iv) The total size of all pieces is in $\Theta(|V|)$. Intuitively, the pieces are almost a partition of V where those vertices on the fringe of each piece may appear in multiple pieces. Such a decomposition of G can be computed in time $O(|V|)$ [14, 18]. We will apply this decomposition to T_i, which is both a point set and the vertex set of $G(T_i)$, for exponentially increasing sizes of ρ.

Given integers $1 \leq k < j < f$, let

$$Pieces_j(k) \coloneqq \{Piece_j^1(k), Piece_j^2(k), \ldots Piece_j^r(k)\}$$

be a decomposition of T_j into $r = \Theta(|T_j|/d^{4k})$ subsets such that each subset $Piece_j^l(k)$ has size $O(d^{4k})$ and a fringe of size $O(d^{2k})$ with respect to $G(T_i)$. We let

$$Seps_j(k) \coloneqq \{Sep_j^1(k), Sep_j^2(k), \ldots Sep_j^r(k)\}$$

be defined such that $Sep_j^\ell(k)$ denotes the fringe of $Piece_j^\ell(k)$, and let $\overline{Sep}_j^\ell(k)$ be $Piece_j^\ell(k) \setminus Sep_j^\ell(k)$. Thus each $Piece_j^\ell(k)$ is partitioned into its fringe vertices, $Sep_j^\ell(k)$, and its interior non-fringe vertices $\overline{Sep}_j^\ell(k)$; note that $\overline{Sep}_j^\ell(k)$ may be empty if all elements of $Piece_j^\ell(k)$ are on its fringe.

Finally, we define $Sample_j(k)$ to be the union of all the fringe vertices:

$$Sample_j(k) \coloneqq \bigcup_{Sep \in Seps_j(k)} Sep$$

thus $Seps_j(k)$ is a decomposition of $Sample_j(k)$.

For any $k \in [1..j-1]$, the decomposition of T_j into $Pieces_j(k)$, the partition of each $Piece_j^\ell(k)$ into $Sep_j^\ell(k)$ and $\overline{Sep}_j^\ell(k)$, and the set $Seps_j(k)$ can all be computed in time $O(|T_j|)$ using [18] if the Delaunay triangulation is available; if not it can be computed in time $O(|T_j| \log |T_j|)$. Thus computing these for all valid i takes time and space $O(|T_j| \log |T_j| \log_d |T_j|)$ as $k < j = O(\log_d |T_j|)$.

One property of this sampling technique is that points in T_j with Voronoi cells in $Vor(T_j)$ of complexity at least k are included in T_i if $j > i$ and $j - i = O(\log_d k)$. By complexity of a region, we mean the number of edges that bound this region.

▶ **Lemma 2.** *Given $i < j$, if $p \notin T_i$ and $p \in T_j$ then the complexity of $Cell(T_j, p)$ is $O(d^{4(j-i)})$.*

Proof. Suppose that $|Cell(T_j, p)| > cd^{4(j-i)}$, for a constant c chosen later, we will show this implies $p \in T_i$. Thus the degree of p in $G(T_j)$ is greater than $cd^{4(j-i)}$. Consider the piece $Piece_j^\ell(j-i)$ in $Pieces_j(j-i)$ that contains p. This piece has size at most $O(d^{4(j-i)})$, which is at most $cd^{4(j-i)}$ for some c (here we choose c). Thus in $G(T_j)$, p must have neighbors which are not in $Piece_j^\ell(j-i)$. By definition, p is thus in the fringe $Sep_j^\ell(j-i)$, which implies $p \in Sep_j^\ell(j-i)$. From the definition of T_i, $T_i \subset Sep_j^\ell(j-i)$ and which gives $p \in T_i$. ◀

[1] We use the word *decomposed* to mean a division of a set into into a collection of sets, the *decomposition*, whose union is the original set, but, unlike with a partition, elements may belong to multiple sets.

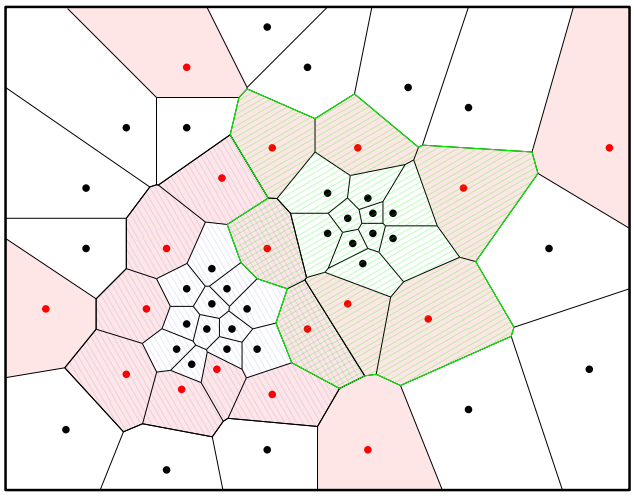


Figure 1 Part of a Voronoi diagram for a point set T_j. Two elements of $Pieces_j(k)$ have been highlighted, one in striped blue, call it $Piece_j^1(k)$, and one in striped green, call it $Piece_j^2(k)$. For each piece, the cells of fringe vertices are shaded red. Thus, the set $Sample_j(k)$ are the red verticies, and the region $Cells(T_j, Sample_j(k))$ is shaded red. The green-and-red shaded region is $Cells(T_j, Sep_j^2(k))$ and the green-but-not-red shaded region is $Cells(T_j, \overline{Sep}_j^2(k))$.

While we cannot bound the complexity of any Voronoi cell in a fringe, we can bound the complexity of $\overline{Sep}_j^\ell(j-i)$, the cells inside a fringe. Intuitively, each piece has the fringe cells on its exterior and non-fringe cells on its interior; imagining the fringe cells of a piece as an annulus gives two boundaries, the exterior boundary of the fringes, which is the boundary between the cells of this and other pieces, and the interior boundary of the fringes, which is the boundary between the fringe cells and the interior cells in this piece. Crucially, while the exterior boundary could have high complexity, the interior boundary does not, which we now formalize:

▶ **Lemma 3.** *The complexity of* $Cells(T_j, \overline{Sep}_j^\ell(j-i))$ *is* $O(d^{4(j-i)})$.

Proof. See Figure 2. Each cell in $Cells(T_j, \overline{Sep}_j^\ell(j-i))$ is adjacent to either other cells in $Cells(T_j, \overline{Sep}_j^\ell(j-i))$ or cells in $Cells(T_j, Sep_j^\ell(j-i))$. The adjacency graph of these Voronoi regions is planar, and as $\overline{Sep}_j^\ell(j-i) \cup Sep_j^\ell(j-i) = Piece_j^\ell(j-i)$, and recalling that $|Piece_j^\ell(j-i)| = O(d^{4(j-i)})$ gives the lemma. ◀

The Jump function: definition

At the core of our nearest neighbor algorithm is the function $Jump$, defined as follows. We will find it helpful to use $NN_R(q)$ for a range $R = [l, r]$ to denote $NN(\cup_{i\in[l,r]} T_i, q)$; for example $NN_{[1,k]}(q)$ is the nearest neighbor of q in $T_1, T_2, \ldots T_k$.

Intuitively, a call to $Jump(i, j, q, p_i, e_i)$ is used when trying to find the nearest neighbor of q, and assuming we know the nearest neighbor of q in $T_1, T_2 \ldots T_{(i+j)/2}$ determines whether there are any points that could be the nearest neighbor of q in $T_{(i+j)/2+1} \ldots T_j$. This

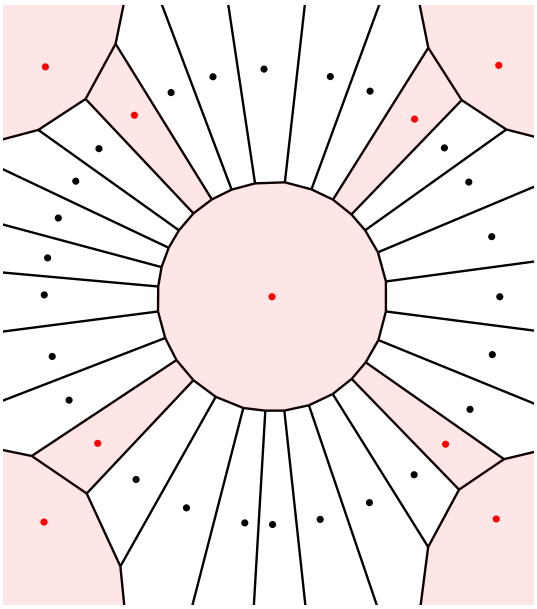


Figure 2 High complexity cells can occur in Voronoi diagrams. Such cells must be included in fringe verticies $Sample_j(k)$, illustrated in red for some point set T_j. This results in the complexity of the boundary of the interior sets $Cells(T_j, \overline{Sep}_j^{\ell}(k))$, the connected components of white Voronoi cells, are of complexity $O(d^{4(j-i)})$ by Lemma 3.

information could be either a simple *no*, or it could provide the nearest neighbor of q for some prefix of these sets. Additionally, the edge of an the Voronoi cell of the currently known nearest neighbor in the direction of the query point is always passed and returned to aide the search using the combinatorial bounds from Lemma 8, point 4.

- **Input to $Jump(i, j, q, p_i, e_i)$**:
  - Integers i and j, where $j - i$ is required to be a power of 2. We use m to refer to $(j + i)/2$, the midpoint.
  - Query point q.
  - Point p_i where $p_i = NN(T_i, q)$.
  - The edge e_i on the boundary of $Cell(T_i, p_i)$ that the ray $\overrightarrow{p_i q}$ intersects.
- **Output**: Either one of two results, *Failure* or a triple $(j', p_{j'}, e_{j'})$
  - If *Failure*, this is a certificate that $NN_{(m,\min(j,f)]}(q) \neq NN_{[1,\min(j,f)]}(q)$
  - If a triple $(j', p_{j'}, e_{j'})$ is returned, it has the following properties:
    - The integer j' is in the range $(m, j]$ and $NN_{(m,j')}(q) \neq NN_{[1,j')}(q)$.
    - The point $p_{j'}$ is $NN(T_{j'}, q)$.
    - The edge $e_{j'}$ is on the boundary of $Cell(T_{j'}, p_j)$ that the ray $\overrightarrow{p_{j'} q}$ intersects.

We will show later that $Jump$ runs in $O(j - i)$ time. Implementation details are deferred to Section 2.2.

The nearest neighbor procedure

A nearest neighbor query can be answered through a series of calls to the $Jump$ function:

- Initialize $i = 1, j = 2$, p_1 to be $NN(T_1, q)$, and e_1 to be the edge of $Cell(T_1, p_1)$ crossed by the ray $\overrightarrow{p_1 q}$; all of these can be found in constant time as $|T_1| = \Theta(1)$. Initialize $p_{nearest}$ to p_1.

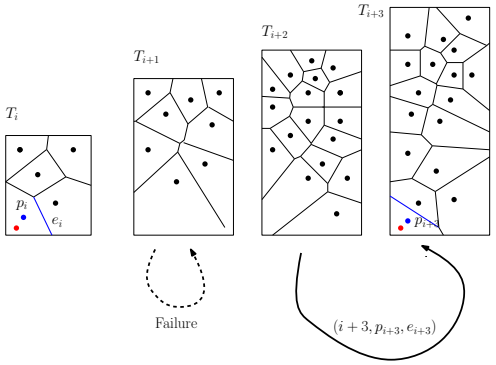


Figure 3 Two iterations of the Jump procedure. The query point q is shown in red. Points $p_i = NN(T_i, q)$, $p_{i+3} = NN(T_{i+3}, q)$, and edges e_i and e_{i+3} are shown in blue.

- Repeat the following while $\frac{i+j}{2} \leq f$:
  - Run $Jump(i, j, q, p_i, e_i)$. If the result is failure:
    - Set $j = j + (j - i)$
  - Else a triple $(j', p_{j'}, e_{j'})$ is returned:
    - If $d(p_{j'}, q) < d(p_{nearest}, q)$ set $p_{nearest} = p_{j'}$
    - Set $i = j'$ and set $j = j' + 1$
- Return $p_{nearest}$

We will show in the rest of this section that, given this jump function as a black box, we can correctly answer a nearest neighbor query in $O(\log n)$ time.

Correctness

The loop in the jump procedure will maintain the invariants that $p_{nearest}$ is $NN_{[1,\min(f,\frac{i+j}{2})]}(q)$, $i < j$ and $j - i$ is a power of two. The latter is because $j - i$ is set to either 1 or is doubled with each iteration of the loop. As to the first invariant, before the loop runs, the invariant requires $p_{nearest} = NN(T_1, q)$, which p_1 and $p_{nearest}$ are initialized to. During the loop we subscript i and j by new and old to indicate the value of variables at the beginning and at the end of the loop. Thus $p_{nearest} = NN_{[1,\min(f,\frac{i_{old}+j_{old}}{2})]}(q)$. Also if the loop runs, we know $\min(j_{old}, f) = j_{old}$. We distinguish between two cases depending on whether the jump function returns failure or a triple.

If the jump function returns failure, we know that

$$NN_{(\frac{i_{old}+j_{old}}{2},\min(j_{old},f)]}(q) \neq NN_{[1,\min(j_{old},f)]}(q).$$

Using this fact we can go from the invariant in terms of the old variables to the new ones:

$$\begin{aligned}
p_{nearest} &= NN_{[1,\min(f,\frac{i_{old}+j_{old}}{2})]}(q) && \text{Invariant}\\
&= NN_{[1,\frac{i_{old}+j_{old}}{2}]}(q) && \frac{i_{old}+j_{old}}{2} \leq f\\
&= NN_{[1,\min(j_{old},f)]}(q) && NN_{(\frac{i_{old}+j_{old}}{2},\min(j_{old},f)]}(q) \neq NN_{[1,\min(j_{old},f)]}(q)\\
&= NN_{[1,\min(f,\frac{i_{old}+j_{old}+(j_{old}-i_{old})}{2})]}(q) && \text{math}\\
&= NN_{[1,\min(f,\frac{i_{new}+j_{new}}{2})]}(q) && i_{new} = i_{old}; j_{new} = j_{old} + (j_{old} - i_{old})
\end{aligned}$$

Now we consider the case when a triple $(j', p_{j'}, e_{j'})$ is returned by the *Jump* function. We know that $NN_{(m,j']}(q) \neq NN_{[1,j']}(q)$, and using the same logic as from the failure case we can conclude that the old $p_{nearest}$ is $NN_{[1,j')}(q)$. The code sets $p_{nearest}$ to the point that is closer to q among the old $p_{nearest}$, equal to $NN_{[1,j')}(q)$, and $p_{j'}$, which is $NN(T_{j'}, q)$. Thus the new $p_{nearest}$ is equal to $NN_{[1,j']}(q)$ (note the closed interval) which we can rewrite to get the invariant as follows, using the fact that the subscript of NN is only dependent on the integers in the given range:

$$NN_{[1,j']}(q) = NN_{[1,i_{new}]}(q) = NN_{[1,\frac{i_{new}+i_{new}+1}{2}]}(q) = NN_{[1,\frac{i_{new}+j_{new}}{2}]}(q)$$

When the loop finishes, we have $j > f$ and thus

$$p_{nearest} = NN_{[1,\min(f,\frac{i+j}{2})]}(q) = NN_{[1,f]}(q) = \underset{p\in\{T_k | k\in[1,f] \text{ and } k\in\mathbb{Z}\}}{\arg\min} d(p,q) = NN(S,q)$$

Running time

▶ **Lemma 4.** *Given the Jump function, the running time of the nearest neighbor search function in $O(\log n)$.*

Proof. We use a potential argument. Let i_t and j_t denote the values of i and j after the loop has run t times, let a_t denote the runtime of the tth iteration of the loop, and let T denote the number of times the loop runs. The total runtime is thus $O(1) + \sum_{t=1}^{T} a_t$. Define $\Phi_t := c(2i_t + j_t)$, for some constant c chosen so that $a_t \leq \frac{c}{2}(j_j - i_t)$; as $i_0 = 1$ and $j_0 = 2$, $\Phi_0 = 4c$. (recall that the runtime of Jump is $O(j - i)$ for constant d).

We will argue that for all t, $a_t \leq \Phi_t - \Phi_{t-1} + O(1)$. Summing, this gives $\sum_{t=1}^{T} a_t \leq \Phi_T - \Phi_0 + O(T)$. Since i, j and T are in the range $[1, f]$, this bounds $\sum_{t=1}^{T} a_t$ by $3cf + O(f) = \Theta(\log n)$. Thus all that remains is to argue that $a_t \leq \Phi_t - \Phi_{t-1}$ for the two cases where the tth run of Jump ends in failure or returns a triple.

Case 1, Jump returns failure. In the case of failure, i remains the same, thus $i_t = i_{t-1}$ but j increases by $j - i$, thus $j_t = 2j_{t-1} - i_{t-1}$. Thus the potential increases by $c(j - i)$.

$$\begin{aligned}
\Phi_t - \Phi_{t-1} &= c(2i_t + j_t) - c(2i_{t-1} + j_{t-1}) \\
&= c(2i_{t-1} + 2j_{t-1} - i_{t-1}) - c(2i_{t-1} + j_{t-1}) \\
&= c(j_{t-1} - i_{t-1}) \\
&\geq a_t
\end{aligned}$$

Case 2, Jump returns a triple. i_t is set to j' and j_t changes to $j' + 1$. The key observation is that, due to the invariant in our correctness argument, $j_t \geq \frac{i_{t-1}+j_{t-1}}{2}$. Thus $i_t = j' \geq j_{t-1} \geq \frac{i_{t-1}+j_{t-1}}{2}$ and $j_t = j' + 1 \geq j_{t-1} + 1 \geq \frac{i_{t-1}+j_{t-1}}{2} + 1$; the potential change is

$$\begin{aligned}
\Phi_t - \Phi_{t-1} &= c(2i_t + j_t) - c(2i_{t-1} + j_{t-1}) \\
&\geq c(2 \cdot \frac{i_{t-1} + j_{t-1}}{2} + \frac{i_{t-1} + j_{t-1}}{2} + 1) - c(2i_{t-1} + j_{t-1}) \\
&= \frac{c}{2}(j_{t-1} - i_{t-i}) + c \\
&\geq a_t
\end{aligned}$$

◀

2.2 The jump function

2.2.1 Basic geometric facts

We begin with our most crucial geometric lemma, the one that we build upon to make our algorithm work. Informally, given point sets A and B which possibly have elements in common, and a query point q, if the closest point to q in B is also in A, then the closest point to q in $A \cup B$ is in A, and not in $B \setminus A$.

▶ **Lemma 5.** *Given i, j, $i < j$, suppose that there is a point q such that $q \in Cell(T_i, p_i) \cap Cell(T_j, p_j)$ for some $p_j \in Sample_j(j - i)$. Then $q \in Cell(T_i \cup T_j, p_i)$, or equivalently $NN(T_i \cup T_j, q) = p_i$.*

Proof. Since $p_j \in Sample_j(j - i)$, $p_j \in T_i$ by the definition of T_i. Since $q \in Cell(T_i, p_i)$, $dist(q, p_i) \leq dist(q, p_j)$. However, $q \in Cell(T_j, p_j)$ and p_j is the closest point to q in T_j: $dist(q, p_j) \leq dist(q, p'_j)$ for all p'_j in T_j. Combining $dist(q, p_i) \leq dist(q, p_j)$ with $dist(q, p_j) \leq dist(q, p'_j)$ for all p'_j in T_j gives the lemma. ◀

▶ **Lemma 6.** *If all elements of a point set P are in $Cell(S, p)$ for some point set S and $p \in S$, then all elements of the convex hull of P are in $Cell(S, p)$ as well.*

Proof. Immediate as $Cell(S, p)$ is convex. ◀

Lemmata 5 and 6 give us a tool to determine which parts of the Voronoi cell of some p_i in $\mathrm{Vor}(T_i)$ must also be part of the Voronoi cell of p_i in $\mathrm{Vor}(T_i \cup T_j)$. We define this region as $Hull_i(j, p_i)$, and then prove its properties.

Let $Hull_i(j, p_i)$, for some $p_i \in T_i$, denote the convex hull of

$$\{p_i\} \cup (Cell(T_i, p_i) \cap Cells(T_j, Sample_j(j - i))).$$

See Figure 4 for an example.

▶ **Lemma 7.** *$Hull_i(j, p_i) \subseteq Cell(T_i \cup T_j, p_i)$ and thus if $q \in Hull_i(j, p_i)$, $NN(q, S) \notin T_j \setminus T_i$*

Proof. The point p_i is in its own cell, $Cell(T_i \cup T_j, p_i)$, and by Lemma 5, all elements of $Cell(T_i, p_i) \cap Cells(T_j, Sample_j(j - i))$ are also in $Cell(T_i \cup T_j, p_i)$. Thus the convex hull of these points is a subset of $Cell(T_i \cup T_j, p_i)$ by Lemma 6. ◀

Thus, if a query point q is in $Hull_i(j, p_i)$, then, by Lemma 7, the set T_j can be ignored because $dist(q, p_i) \leq dist(q, p')$ for any $p' \in T_j \setminus T_i$.

Geometrically, determining whether a point in $Cell(T_i, p_i)$ is in a $Hull_i(j, p_i)$ and thus a full search in $\mathrm{Vor}(T_j)$ can be skipped is what our jump function does. We now need to turn to examining the combinatorial issues surrounding $Hull_i(j, p_i)$ and its interaction with $Cell(T_i, p_i)$ as we need the complexity of the regions examined to be bounded in such a way to allow efficient searching to see if a point is in $Cell(T_i, p_i)$. We begin by defining the part of $Cell(T_i, p_i)$ that is not in $Hull_i(j, p_i)$ as $\overline{Hull}_i(j, p_i)$ and proving a number of properties of this possibly disconnected region.

▶ **Lemma 8.** *Consider the region*

$$\overline{Hull}_i(j, p_i) \coloneqq Cell(T_i, p_i) \setminus Hull_i(j, p_i)$$

1. *Each connected component of $\overline{Hull}_i(j, p_i)$ is a subset of the union of Voronoi cells in one element of $Pieces_j(j - i)$; that is, each connected component of $\overline{Hull}_i(j, p_i)$ is a subset of $Cells(T_j, Piece^{\ell}_j(j - i))$ for some $Piece^{\ell}_j(j - i) \in Pieces_j(j - i)$.*

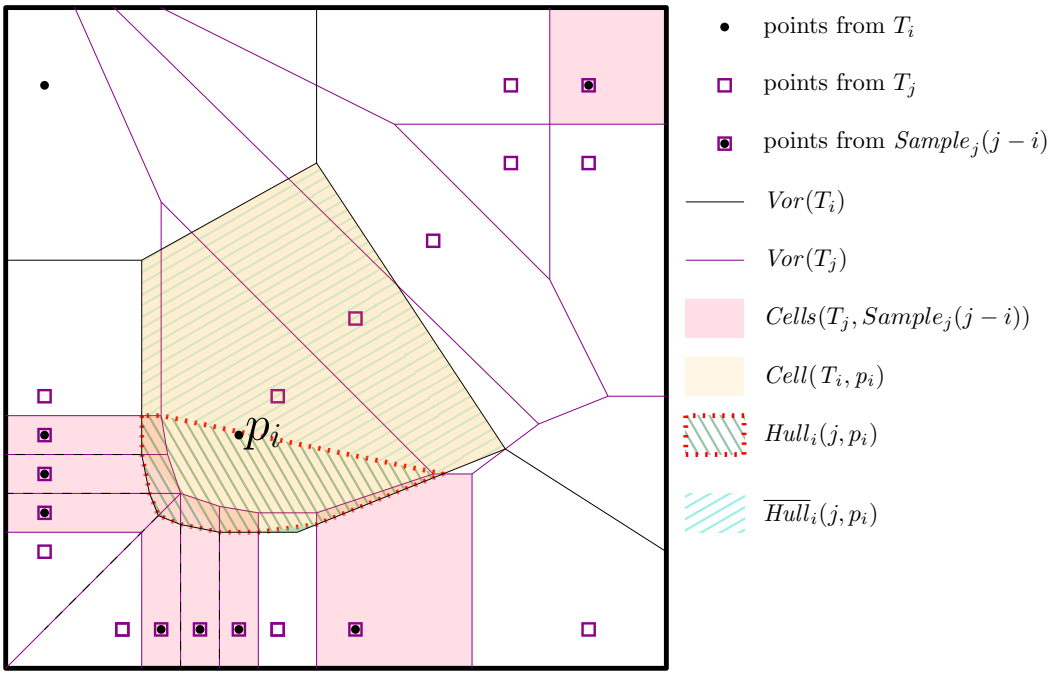


Figure 4 Illustration of the computation of $Hull_i(j, p_i)$ and $\overline{Hull}_i(j, p_i)$. Observe that $Hull_i(j, p_i)$ is the convex hull of those parts of $Cells(T_j, Sample_j(j-i))$ (shaded pink) that are inside $Cell(T_i, p_i)$ (shaded tan). $\overline{Hull}_i(j, p_i)$ is simply the remainder of $Cell(T_i, p_i)$, and has two connected components, including a small one at the bottom. By Lemma 7, the closest point in $T_i \cup T_j$ to all points in $Hull_i(j, p_i)$ is p_i.

2. *$\overline{Hull}_i(j, p_i)$ intersects each bounding edge of $Cell(T_i, p_i)$ in at most two connected components, each of which includes a vertex of $Cell(T_i, p_i)$.*
3. *Any line segment $\overline{p_i q}$, where q is on the boundary of $Cell(S_i, p_i)$ intersects $\overline{Hull}_i(j, p_i)$ in at most one connected component that, if it exists, includes q.*
4. *Let $\overline{qr}$ be a boundary edge of $Cell(T_i, p_i)$. The solid triangle $\triangle p_i qr$ intersects at most $d^{O(j-i)}$ edges on the boundary separating $Hull_i(j, p_i)$ from $\overline{Hull}_i(j, p_i)$.*

Proof.

1. Suppose one connected component $\overline{Hull}_i(j, p_i)$ contains points p, p' in both $Cells(T_j, Piece_j^{\ell}(j-i))$ and $Cells(T_j, Piece_j^{\ell'}(j-i))$ where $Piece_j^{\ell}(j-i)$ and $Piece_j^{\ell'}(j-i)$ are different elements of $Pieces(S_j, j-i)$. Consider a polyline that connects p and p' while remaining in the same connected component of $\overline{Hull}_i(j, p_i)$. Such a polyline must cross, at some point, some cell $Cell(T_j, p_j)$ of some $p_j \in T_j$ where $p_j \in Piece_j^{\ell}(j-i)$ but where the cell $Cell(T_j, p_j)$ is adjacent to at least one other cell in $Vor(T_j)$ that is not in $Piece_j^{\ell}(j-i)$. Thus p_j is by definition in $Sep_j^{\ell}(j-i)$; thus p_j is in $Hull_i(j, p_i)$ by its definition in Lemma 7. But this contradicts p_j is in $\overline{Hull}_i(j, p_i)$.
2. If this does not hold, there are points q_2 and q_3 on $\overline{Hull}_i(j, p_i)$, with q_2 closer to j than q_3, such that $q_2 \notin \overline{Hull}_i(j, p_i)$ and $q_3 \in \overline{Hull}_i(j, p_i)$. But this cannot happen: We know $p \in Hull_i(j, p_i)$ by construction and if p and q_2 are in $Hull_i(j, p_i)$, q_2 must be as well because the hull is a convex set.
3. By a similar argument as the last point, if this did not hold, there would be points p_1, q_1, q_2, q_3 in order on $\overline{p_i q}$, where there are some points q_1, q_2, q_3, q_4, q_5, in order, such that $q_1, q_3, q_5 \in \overline{Hull}_i(j, p_i)$ and $q_2, q_4 \in Hull_i(j, p_i)$. But this cannot happen: if q_2 and q_4 are in $Hull_i(j, p_i)$, q_3 must be as well because the hull is a convex set.

4. The complexity of $Hull_i(j, p_i) \cap \triangle p_i q r$ is at most the complexity of $Cells(T_j, \overline{Seps}_j^{\ell}(j-i))$ and $Cells(T, \overline{Seps}_j^{\ell'}(j-i))$, where $q \in Cells(T_j, \overline{Seps}_j^{\ell}(j-i))$ and $r \in Cells(T_j, \overline{Seps}_j^{\ell'}(j-i))$. By Lemma 3, the complexity of these regions (within the triangle $\triangle p_i q r$) is at most $d^{O(j-i)}$. Taking the convex hull only decreases the complexity of the objects on which a hull is defined. ◀

We can use these geometric facts to present the following corollary, which shows how we can use inclusion in $Hull_i(j, p_i)$ to determine where to find the nearest neighbor of q in T_j or to determine that $NN(S, q)$, q's nearest neighbor in S, is not in T_j *without needing to find the nearest neighbor of* q *in* T_j. This is our main novel idea, since, as discussed in the introduction, finding the nearest neighbors in every T_j in logarithmic time is not possible.

▶ **Corollary 9.** *Given:*

- i *and* j, $i < j$
- *a query point* q
- *a point* p_i *where* $p_i = NN(T_i, q)$
- *the edge* $\overline{v_1 v_2}$ *on the boundary of* $Cell(T_i, p_i)$ *that the ray* $\overrightarrow{p_i q}$ *intersects.*

then, by testing q *against the part of* $Hull_i(j, p_i)$ *that is inside* $\triangle p_i v_1 v_2$ *and has complexity* $d^{O(j-i)}$ *one of the following is true:*

- *If* q *is inside* $Hull_i(j, p_i)$*, then* q *is in* $Cell(T_i \cup T_j, p_i)$ *and* $NN(S, q)$ *is not in* T_j*.*
- *If* q *is outside* $Hull_i(j, p_i)$ *(and thus inside* $\overline{Hull}_i(j, p_i)$*): then* $NN(T_j, q)$ *is in the same element of* $\overline{Seps}_j(j-i)$ *as either* v_1 *or* v_2*.*

2.2.2 Implementing the jump function

We now use one additional idea to speed up the jump function: While testing if q is inside or outside of the part of $Hull_i(j, p_i)$ that intersects $\triangle p_i v_1 v_2$ can be done in time $O((j-i)\log d)$ (since the complexity of this part of the hull is $d^{O(j-i)}$ by Lemma 8, point 4), we can in fact do something stronger. We can test if q is inside or outside of the part of $Hull_i(j', p_i)$ that intersects $\triangle p_i v_1 v_2$, **for all** $\frac{i+j}{2} < j' < j$, in the same time $O((j-i)\log d)$. This is because the complexity of the subdivision of the plane induced by $\frac{j-i}{2}$ hulls of size $d^{O(j-i)}$ has complexity $d^{O(j-i)} \cdot O((j-i)^2)$ and thus we can determine in time $O((j-i)\log d)$ which is the smallest j' in the range $i < j' \leq j$ where q is not inside $Hull_i(j', p_i)$, or determine that for all j' in the range $i < j' \leq j$.

Thus we obtain the final jump procedure:

Method to compute $Jump(i, j, q, p_i, e_i)$:

1. Test to see what is the smallest j', $\frac{i+j}{2} < j' \leq j$ such that q is outside of the part of $Hull_i(j', p_i)$ that intersects $\triangle p_i v_1 v_2$. This will be done in time $O(\log d(j-i))$ with the convex hull search structure described in Section 2.3. If it is inside all such hulls, then $NN(S, q) \notin T_{j'}$ for all j' such that $\frac{i+j}{2} < j' \leq j$ and failure is returned. Otherwise:
2. Let $Piece_{v_1}$ and $Piece_{v_2}$ denote the elements of $Pieces_j(j'-i)$ that contain v_1 and v_2. These will be precomputed and accessible in constant time with the piece lookup structure described in Section 2.3.
3. Search in $Vor(T_{j'}, Piece_{v_1})$ and $Vor(T_{j'}, Piece_{v_2})$ to find $NN(Piece_{v_1} \cup Piece_{v_2}, q)$, call it $p_{j'}$. Note that, $p_{j'}$ is $NN(T_j, q)$. As both $Vor(T_{j'}, Piece_{v_1})$ and $Vor(T_{j'}, Piece_{v_2})$ have complexity $O(d^{j'-i})$, this can be done in time $O((j-i)\log d)$ with the piece interior search structure described in Section 2.3.
4. Find the edge $e_{j'}$ bounding $Cell(T_{j'}, p_{j'})$ that the ray $\overrightarrow{p_{j'} q}$ intersects. As $Cell(T_{j'}, p_{j'})$ has complexity $d^{O(j'-i)}$, this can be done in time $O((j-i)\log d)$ with binary search.

2.3 Data structures needed

The data structure is split into levels, where level i consists of:

I. S_i,
II. T_i,
III. The Voronoi diagram of T_i, $Vor(T_i)$, and a point location search structure for the cells of $Vor(T_i)$,
IV. The Delaunay triangulation of T_i, $G(T)$.
V. Additionally, we keep for each j, $1 \leq j < i$:
   i. The partition of T_i into $Pieces_j(k) \coloneqq \{Piece_j^1(k), Piece_j^2(k), \ldots Piece_j^{|Pieces_j(k)|}(k)\}$
   ii. The partition of each $Piece_j^\ell(k)$ into $Sep_j^\ell(k)$ and $\overline{Sep}_j^\ell(k)$
   iii. The set $Sample_j(k) \coloneqq \bigcup_{Sep \in Seps_j(k)} Sep$

For any level i, this information can be computed from T_i in time $O(|T_i| \log |T_i|)$ using [18][Theorem 3] to compute the partition into pieces, and standard results on Delaunay/Voronoi construction.

Additionally, less elementary data structures are needed for each level, which we describe separately: the convex hull search structure, the piece lookup structure, and the piece interior search structure.

Convex hull search structure

For level i, a convex hull structure is built for every combination of:

- A point p_i in T_i
- An edge e_i of $Cell(T_i, p_i)$
- An index j where $i < j$, $\frac{i+j}{2} \leq f$ and $j - i$ is a power of 2.

A convex hull structure answers queries of the form: given a point q in $Cell(T_i, p_i)$, return the smallest j', $\frac{i+j}{2} < j' \leq j$ such that q is outside of the part of $Hull_i(j', p_i)$ that intersects $\triangle p_i v_1 v_2$, where v_1 and v_2 are endpoints of e_i. There are $O(|T_i| \log \log_d |T_i|)$ such structures as j is at most $f = \Theta(\log_d n)$, and the complexity of a Voronoi diagram $Vor(T_i)$ is linear in the number of points it is defined on.

The method used is to simply store a point location structure which contains subdivision within $\triangle p_i v_1 v_2$ formed by the overlay of all boundaries of $Hull_i(j', p_i)$, for $\frac{i+j}{2} < j' \leq j$. As previously mentioned the complexity of this overlay is $O(d^{j-i} \cdot (j-i)^2)$ and thus point location can be done in the logarithm of this, which is $O((j-i) \log d)$.

As noted earlier there are $O(|T_i| \log \log_d n)$ structures, each of which takes space at most $O(j - i) = O(\log_d n)$ plus the number of intersections in the point location structure within the triangle. The $O(j-i)$ comes from the at most 2 edges from each hull that can pass through the triangle without intersection. For a given j, the sum of the complexities of $Hull_i(j, p_i)$ over all $p_i \in T_i$ is $O(T_i)$. As each hull edge can intersect at most $O(j - i) = O(\log_d n)$ other hull edges, that bounds the total space needed over one j to be $O(n \log_d n)$. The overall space usage is $O(n \log_d^2 n)$.

Piece lookup structure

Level i of the piece lookup structure contains for j, $i < j \leq \min(2i, f)$ and for each vertex v_i of $Vor(T_i)$ the index of which piece $Piece_j^\ell(j - i) \in Pieces_j^\ell(j - i)$ has q in $Cells(T_j, Piece_j^\ell(j - i))$. This can be precomputed using the point location search structure for $Vor(T_j)$ in time $O(\log d^j) = O(j \log d) = O(i) = O(\log n)$ for fixed d for each of the $O(|T_i|)$ vertices of $Vor(T_i)$. Summing over the choices for j gives a total runtime of $O(|T_i| \log^2 n)$ to pre-compute all answers. The space usage is $O(|T_i| \log n)$.

Piece interior search structure

For each $1 \leq i < j \leq f$ we store a point location structure that supports point location in time $O(j-i)$ in the Voronoi diagram for each set of points in $Pieces_j(j-i)$. Any standard linear-sized point location structure, such as that of Kirkpatrick [17], suffices since each element of $Pieces_j(j-i)$ has $O(4^{j-i})$ elements. For any fixed j, there are $j-1$ choices for i, and the sets in $Pieces_j(j-i)$ partition T_i. Thus the total size of all these structures is $O(|T_j| \log n)$. The construction cost, given the $Pieces_j(j-i)$, incurs another logarithmic factor due to the need to construct the Voronoi diagram and the point location structure (we do not assume each piece is connected). Thus the piece interior search structure for level j is constructed in $O(|T_j| \log^2 n)$ time.

2.4 Insertion time

Our description of the data structures needed can be summarized as follows:

▶ **Lemma 10.** *Level i of the data structure can be built in time and space $O(|T_i| \log^2 n)$ given all levels $j > i$.*

Insertion is thus handled by the classic logarithmic method of Bentley and Saxe [3] which transforms a static construction into a dynamic structure, and which we briefly summarize. To insert, we put the new point into S_1 and rebuild level 1. Every time a set S_i exceeds the upper limit of $\Theta(d^i)$, half of the items are moved from S_i to S_{i+1} and all levels from $i+1$ down to S_1 are rebuilt. So long as the upper and lower constants in the big Theta are at least a constant factor apart, the amortized insertion cost is $O(\log n)$ times the cost per item to rebuild a level, thus obtaining:

▶ **Lemma 11.** *Insertion can be performed with an amortized running time of $O(\log^3 n)$.*

The performance of our data structure can be summarized as follows:

▶ **Theorem 12.** *There exists a semi-dynamic insertion-only data structure that answers two-dimensional nearest neighbor queries in $O(\log n)$ time and supports insertions in $O(\log^3 n)$ amortized time. The data structure uses $O(n \log^2 n)$ space.*

References

1. Pankaj K. Agarwal and Jirí Matousek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.
2. Sarah R. Allen, Luis Barba, John Iacono, and Stefan Langerman. Incremental Voronoi diagrams. *Discrete and Computational Geometry*, 58(4):822–848, 2017. doi:10.1007/s00454-017-9943-2.
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. Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.
5. Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discrete and Computational Geometry*, 64(4):1235–1252, 2020. doi:10.1007/s00454-020-00229-5.
6. Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete and Computational Geometry*, 56(4):866–881, 2016. doi:10.1007/s00454-016-9784-4.

7 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. doi:10.1007/BF01840440.

8 Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *Journal of Computing & Systems Sciences*, 68(2):269–284, 2004. doi:10.1016/j.jcss.2003.07.003.

9 Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. *Computational Geometry*, 3:185–212, 1993. doi:10.1016/0925-7721(93)90009-U.

10 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

11 Olivier Devillers, Stefan Meiser, and Monique Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Computational Geometry*, 2:55–80, 1992. doi:10.1016/0925-7721(92)90025-N.

12 David P. Dobkin and Subhash Suri. Maintenance of geometric extrema. *Journal of the ACM*, 38(2):275–298, 1991. doi:10.1145/103516.103518.

13 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. doi:10.1137/0216064.

14 Michael T. Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and Systems Sciences*, 51(3):374–389, 1995. doi:10.1006/jcss.1995.1076.

15 John Iacono and Yakov Nekrich. Incremental planar nearest neighbor queries with optimal query time, 2025. arXiv:2504.07366.

16 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(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

17 David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. doi:10.1137/0212002.

18 Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Symposium on Theory of Computing (STOC)*, pages 505–514, 2013. doi:10.1145/2488608.2488672.

19 Joseph S.B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 849–874. CRC Press, 2017.

20 Ketan Mulmuley. *Computational geometry – An introduction through randomized algorithms*. Prentice Hall, 1994.

Improved Approximation Algorithms for Three-Dimensional Knapsack

Klaus Jansen
Kiel University, Germany

Debajyoti Kar
Indian Institute of Science, Bengaluru, India

Arindam Khan
Indian Institute of Science, Bengaluru, India

K. V. N. Sreenivas
Indian Institute of Science, Bengaluru, India

Malte Tutas
Kiel University, Germany

Abstract

We study the three-dimensional Knapsack (3DK) problem, in which we are given a set of axis-aligned cuboids with associated profits and an axis-aligned cube knapsack. The objective is to find a non-overlapping axis-aligned packing (by translation) of the maximum profit subset of cuboids into the cube. The previous best approximation algorithm is due to Diedrich, Harren, Jansen, Thöle, and Thomas (2008), who gave a $(7+\varepsilon)$-approximation algorithm for 3DK and a $(5+\varepsilon)$-approximation algorithm for the variant when the items can be rotated by 90 degrees around any axis, for any constant $\varepsilon > 0$. Chlebík and Chlebíková (2009) showed that the problem does not admit an asymptotic polynomial-time approximation scheme.

We provide an improved polynomial-time $(139/29+\varepsilon) \approx 4.794$-approximation algorithm for 3DK and $(30/7+\varepsilon) \approx 4.286$-approximation when rotations by 90 degrees are allowed. We also provide improved approximation algorithms for several variants such as the cardinality case (when all items have the same profit) and uniform profit-density case (when the profit of an item is equal to its volume). Our key technical contribution is *container packing* – a structured packing in 3D such that all items are assigned into a constant number of containers, and each container is packed using a specific strategy based on its type. We first show the existence of highly profitable container packings. Thereafter, we show that one can find near-optimal container packing efficiently using a variant of the Generalized Assignment Problem (GAP).

2012 ACM Subject Classification Theory of computation → Packing and covering problems

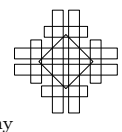
Keywords and phrases Approximation Algorithms, Hyperrectangle Packing, Multidimensional Knapsack, Three-dimensional Packing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.60

Related Version *Full Version*: `https://arxiv.org/abs/2503.19365` [16]

Funding *Klaus Jansen*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 453769249.
Debajyoti Kar: Google PhD Fellowship.
Arindam Khan: Google India Research Award, SERB Core Research Grant (CRG/2022/001176) on "Optimization under Intractability and Uncertainty", Ittiam Systems CSR grant, and the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-23-0001).
K. V. N. Sreenivas: Google PhD Fellowship.
Malte Tutas: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 453769249.

© Klaus Jansen, Debajyoti Kar, Arindam Khan, K. V. N. Sreenivas, and Malte Tutas;
licensed under Creative Commons License CC-BY 4.0



Acknowledgements We are grateful to Lars Prädel for helping us with some details of the 3D Strip Packing result. We are also thankful to Marvin Lira for initial discussions. We also thank the anonymous reviewers for their valuable comments.

1 Introduction

Three-dimensional Knapsack (3DK) is a fundamental problem in computational geometry, operations research, and approximation algorithms. In 3DK, we are given a set of items which are axis-aligned cuboids (defined by their width, depth, and height) with associated profits, and a three-dimensional (3D) knapsack which is an axis-aligned unit cube. The goal is to find a nonoverlapping axis-aligned packing (by translation) of a subset of items into the knapsack such that the profit of the packed items is maximized.

In 1965, Gilmore and Gomory [13] introduced the 3D Knapsack problem – motivated by cutting stock problems (e.g., cutting up of graphite blocks for anodes). They provided heuristics for the special case when the items need to be cut along axis-aligned planes. Since then, the problem and its several variants (under practical constraints) have been well-studied in operations research – typically from the viewpoints of meta-heuristics, MILP, tree search, machine learning methods, and greedy algorithms [5, 25]. In the last decade, 3DK has become a central problem in logistics to increase operational efficiency by efficiently utilizing the space in various domains, such as airline cargo management, warehouse management, and robotic container loading [1]. The prominence of this problem in the industry is evidenced by the comprehensive survey on 3D packing by Ali, Ramos, Carravilla, and Oliveira [1], who have mentioned hundreds of papers in recent years.

Packing problems are fundamental and popular among mathematicians (e.g., see Kepler's sphere packing conjecture [23] and *Soma cube* puzzle [14]). Surprisingly, 3DK is relatively less studied in computational geometry and approximation algorithms. Only in 2008, Diedrich, Harren, Jansen, Thöle, and Thomas [10] provided the first provable approximation guarantees for the problem. They first gave a simple polynomial-time $(9+\varepsilon)$-approximation algorithms for any constant $\varepsilon > 0$. Thereafter, they gave a more sophisticated $(7+\varepsilon)$-approximation algorithm for 3DK. For the case when the items can be rotated by 90 degrees around any axis, they provided a $(5+\varepsilon)$-approximation algorithm. Afterward, Chlebík and Chlebíková [7] showed that the problem admits no asymptotic polynomial-time approximation schemes (APTAS) even for the cardinality case (when each item has the same profit). Lu, Chen, and Cha [24] showed that even packing cubes into a cube is strongly NP-hard.

There has been progress in some special cases. When all items are d-dimensional hypercubes, Harren [15] gave a $(1+2^{-d}+\varepsilon)$-approximation algorithm. Recently, Jansen, Khan, Lira, and Sreenivas [17] gave a PTAS for this special case. Afterwards, Buchem, Deuker, and Wiese [4] obtained an EPTAS. Furthermore, they gave dynamic algorithms with polylogarithmic query and update times, matching the same approximation guarantees.

For the 2D variant when all items are rectangles (2DK), Jansen and Zhang [20] gave a $(2+\varepsilon)$-approximation algorithm. Gálvez, Grandoni, Ingala, Heydrich, Khan, and Wiese [11] gave an improved 1.89-approximation algorithm. Later, Gálvez, Grandoni, Khan, Ramírez-Romero, and Wiese [12] gave a pseudopolynomial-time 4/3-approximation algorithm. For the variant of *guillotine packing* (where items need to be packed such that they can be separated by a series of end-to-end cuts), Khan, Maiti, Sharma, and Wiese [21] gave a pseudopolynomial-time approximation scheme (PPTAS) for the problem.

Compared to the classical (1D) Knapsack, multidimensional Knapsack is much harder as there are different types of items and they can interact in a complicated way. For example, 2D items can be big (large in both dimensions), small (tiny in both dimensions), vertical (large in

height, small in width), or horizontal (large in width, small in height). Most approximation algorithms for 2DK show the existence of a highly profitable structured packing where the knapsack is partitioned into a constant number of boxes such that each box contains only items of one type and can be packed easily by simple greedy algorithms. Large boxes contain a single large item, vertical (resp. horizontal) boxes contain only vertical (resp. horizontal) items (thus, one can essentially ignore the long dimension of the item, and the box becomes a 1D knapsack), and small boxes contain small items (one can consider the area of an item as a proxy and use 1D knapsack solution). However, this is difficult for 3D packing. As pointed out by Bansal, Correa, Kenyon, and Sviridenko [3], one of the obstacles to generalize methods from rectangle packing to cuboid packing is that "*the interaction of say 3-dimensional rectangles which are long in one direction but short in the other two seems much more complicated than in the two-dimensional case.*"

In 2008, the authors in [10] mentioned that "*it is of interest whether here (for 3DK) an algorithm with ratio* $(6+\varepsilon)$ *or less exists.*" However, despite significant progress in the 2D case, there has been no improved approximation algorithm for 3DK in almost two decades.

1.1 Our results

In this paper, we introduce *container packing* for 3D Knapsack. We show that we can divide the 3D knapsack into $O(1)$ number of regions (called *containers*) such that the dimensions of these containers come from a polynomial-sized set. For each container C we define its capacity $\texttt{cap}(C) \in \mathbb{Q}_{\geq 0}$ and for each item i we define its size in C to be $f_C(i) \in \mathbb{Q}_{\geq 0}$. We also associate an algorithm $\mathcal{A}_C$ for the container. We require $\mathcal{A}_C, f_C$ such that for any C and any set of items T satisfying the packing constraint $\sum_{i \in T} f_C(i) \leq \texttt{cap}(C)$, almost all items in T (barring a small profit subset) can be packed into C by algorithm $\mathcal{A}_C$. The assignment of items into containers thus becomes a variant of the Multiple Knapsack problem (however, the knapsacks can have different capacities and items can have sizes specific to the knapsacks) – also a special case of Generalized Assignment Problem (GAP). For $O(1)$ knapsacks, there exists a PTAS for GAP [11]. After the assignment of items to the container C, the corresponding algorithm $\mathcal{A}_C$ ensures that the items admit a feasible profitable packing.

Container packing is a powerful and general technique. For example, by choosing appropriate functions f_C, our containers can handle complicated packings, beyond simple greedy algorithms. For example, we use six different types of containers in this work (see Figures 4 and 8). Apart from the Stack containers and Volume containers (similar to 2D packing), we introduce Area containers (packed using a variant of Next-Fit-Decreasing-Height (`NFDH`) algorithm [9]), Steinberg containers (where different types of items are packed together using our new volume-based packing algorithm `3D-Vol-Pack` – it packs items in layers where each layer is packed using either a greedy algorithm or a 2D packing algorithm called Steinberg's algorithm [26]), and L-Containers (where two different types of stacks are packed together when rotations are allowed). Note that Steinberg container or L-Container goes beyond simple stack or shelf-based algorithms and equips us to handle more complicated packings of different types of items. Finally, we show that given an optimal packing, we can modify it to some *container packing* having significant profit. The functions f_C can be considered analogous to the *weighting functions* in bin packing [9], and we expect that more complicated packing algorithms can be incorporated into this framework by suitably defining f_C.

Using this, we obtain improved approximation guarantees for 3DK (see Table 1). In particular, we obtain $(139/29+\varepsilon)$-approximation for 3DK and $(30/7+\varepsilon)$-approximation for 3DKR (when rotations by 90 degrees are allowed around all axes). For the cardinality case

Table 1 Summary of our results.

	General case	**Cardinality case**	**Profit = Volume**
Without rotations	$\frac{139}{29} + \varepsilon < 4.794$	$\frac{17}{4} + \varepsilon = 4.25 + \varepsilon$	$4 + \varepsilon$
With rotations	$\frac{30}{7} + \varepsilon < 4.286$	$\frac{24}{7} + \varepsilon < 3.429$	$3 + \varepsilon$

(when all items have the same profit), we obtain $17/4 + \varepsilon$ and $24/7 + \varepsilon$, respectively, for the cases without and with rotations. Finally, in the case when the profit of an item is equal to its volume (also called uniform profit-density), we obtain approximation guarantees of $4 + \varepsilon$ and $3 + \varepsilon$, without and with rotations, respectively.

Due to space limitations, some parts of the proofs are omitted in this extended abstract. For missing details, we refer the reader to the full version [16].

1.2 Related work

Two other related problems are 3D Bin Packing (BP) and Strip Packing (SP). In 3D BP, given a set of cuboids and unit cubes as bins, the goal is to find a nonoverlapping axis-aligned packing of all items into the minimum number of bins. In 3D SP, given a set of cuboids, the goal is to pack them into a single 3D rectangular box of unlimited height such that the height of the packing is minimized. For 3D BP and 3D SP, the best-known (asymptotic) approximation ratios are $T_\infty^2 \approx 2.86$ [6] and $3/2 + \varepsilon$ [19], respectively. For more details on related problems, we refer the readers to surveys [2, 8] on approximation and online algorithms for multidimensional packing.

2 Notation and Preliminaries

We are given a set of n items $I = \{1, 2, \ldots, n\}$. Each item $i \in I$ is a cuboid with width $w_i \in (0, 1]$, depth $d_i \in (0, 1]$, height $h_i \in (0, 1]$, and an associated profit $p_i \in \mathbb{Q}$. The volume of an item i is $v(i) = w_i \times d_i \times h_i$, and its base area is defined as $w_i d_i$. We are also given a 3D unit cube knapsack $K := [0, 1] \times [0, 1] \times [0, 1]$. If an item i is placed (by translation) in an axis-aligned way at (i_x, i_y, i_z) then it occupies the region: $[i_x, i_x + w_i] \times [i_y, i_y + d_i] \times [i_z, i_z + h_i]$. For a feasible packing we need $i_x \in [0, 1 - w_i], i_y \in [0, 1 - d_i], i_z \in [0, 1 - h_i]$. In a packing, two items are nonoverlapping if their interiors are disjoint. Our goal is to find a nonoverlapping packing of a subset of these items that maximizes the total profit. In 3DK, we allow only translation, whereas 3DKR allows translation and rotation by 90 degrees around all axes. For any set of items T, we shall let $p(T), h(T)$ and $v(T)$ denote the total profit, sum of heights, and volume of T, respectively. Let OPT denote an optimal packing and $\texttt{opt} := p(\text{OPT})$.

2.1 Packing subroutines

We now describe a few procedures that we repeatedly use throughout the paper to pack items. For a cuboidal box B, let w_B, d_B, h_B denote the width, depth, and height of B, respectively. The volume of the box is then $v(B) = w_B \times d_B \times h_B$.

Next-Fit-Decreasing-Height (`NFDH`) Algorithm [9]. `NFDH` is a shelf-based packing algorithm for packing rectangles. `3D-NFDH` is its generalization in three dimensions (see Figure 1). The following lemmas summarize the guarantees of these algorithms.

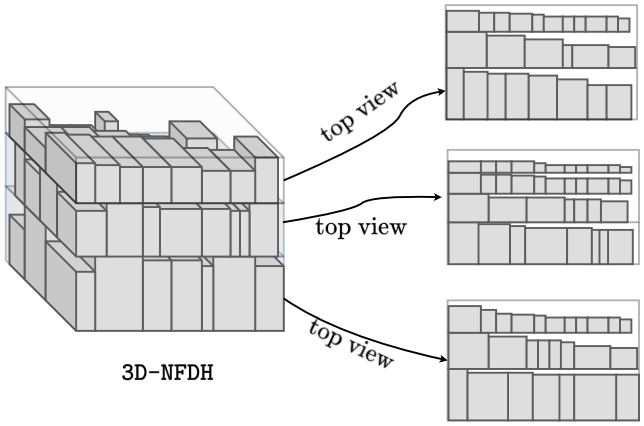


Figure 1 `3D-NFDH` with three layers. Each layer is packed using (two-dimensional) `NFDH`.

▶ **Lemma 1** ([11]). *We are given a rectangular box of length ℓ (horizontal dimension) and breadth b (vertical dimension), and a set S of n (2D) rectangles, where each $i \in S$ has length $\ell_i \leq \varepsilon\ell$ and breadth $b_i \leq \varepsilon b$. If $\sum_{i\in S} \ell_i b_i \leq (1-2\varepsilon)\ell b$, then the whole set S can be packed into the box using NFDH in $O(n \log n)$ time.*

▶ **Lemma 2.** *We are given a cuboidal box B, and a set T of n items where each $i \in T$ satisfies $w_i \leq \varepsilon w_B$, $d_i \leq \varepsilon d_B$ and $h_i \leq \varepsilon h_B$. If $v(T) \leq (1-3\varepsilon)v(B)$, then the whole set T can be packed into B using 3D-NFDH in $O(n \log^2 n)$ time.*

Steinberg's Algorithm [26]. Steinberg's Algorithm is a commonly used 2D packing algorithm with the following performance guarantee.

▶ **Lemma 3** ([26]). *We are given a set S of n rectangles and a rectangular box of size $\ell \times b$. Let $\ell_{\max} \leq \ell$ and $b_{\max} \leq b$ be the maximum length and maximum breadth among the rectangles in S, respectively, and let $a(S)$ be the total area of the rectangles in S. Also, we denote $x_+ := \max(x, 0)$. If $2a(S) \leq \ell b - (2\ell_{\max} - \ell)_+(2b_{\max} - b)_+$, then there is an algorithm that can pack the whole set S into the rectangular box in $O\left(\frac{n \log^2 n}{\log \log n}\right)$ time.*

We utilize the above algorithm to pack items in layers inside a box. Below, we present two algorithms `3D-Vol-Pack` and `3DR-Vol-Pack` for packing 3D items for the cases without and with rotations, respectively. As Steinberg's algorithm is used as a black box in many 2D packing problems, we believe our volume-based bounds and algorithms should find usage in other 3D packing problems. In both cases, we are given a box B with height h_B, width w_B, and depth d_B, and a set of items T.

`3D-Vol-Pack`. We assume that each $i \in T$ satisfies either $w_i \leq w_B/2$ or $d_i \leq d_B/2$. Let T_w be the items of T having width at most $w_B/2$, and let $T_d = T \setminus T_w$. Then, similar to [10], we further classify T_w as follows: let $T_{w\ell} \subseteq T_w$ be the items whose base area (area of the bottom face) exceeds $\frac{1}{6}w_B d_B$, and let $T_{ws} = T_w \setminus T_{w\ell}$. Next, we sort the sets $T_{w\ell}$ and T_{ws} in non-increasing order of heights. We group the items of $T_{w\ell}$ in pairs (except possibly the last item), and pack each pair in a single layer. See Figure 2. For packing items in T_{ws}, we group them into maximal collections of total base area not exceeding $\frac{1}{2}w_B d_B$, and pack each collection in a single layer using Steinberg's algorithm (Lemma 3). Analogously, we obtain a

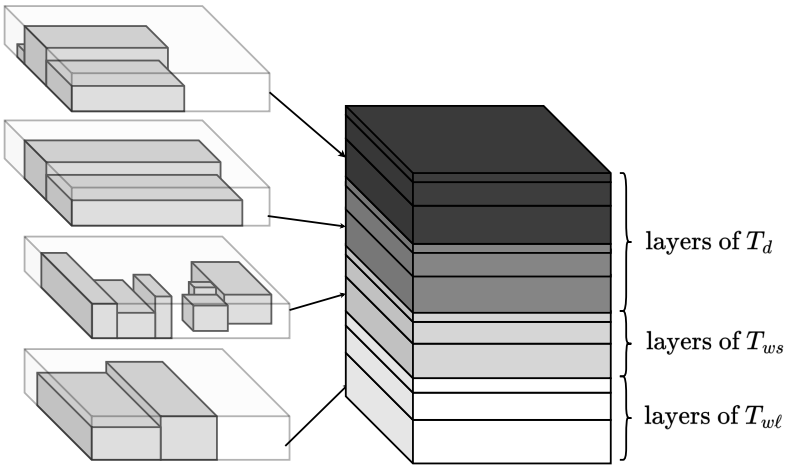


Figure 2 An example packing by Algorithm `3D-Vol-Pack`.

packing of the items in T_d in layers. Finally, we stack the layers one above the other inside the box B as long as the height of the box is not exceeded. The following lemma provides a guarantee when the item heights are small compared to the box height.

▶ **Lemma 4.** *Given a box B, and a set T of n items where each $i \in T$ satisfies $w_i \leq w_B$, $d_i \leq d_B$ and $h_i \leq \varepsilon h_B$. Further, either $w_i \leq w_B/2$ or $d_i \leq d_B/2$ holds for every $i \in T$. If $v(T) \leq \left(\frac{1}{3} - 2\varepsilon\right) v(B)$, then `3D-Vol-Pack` packs the whole set T into B in $O\left(\frac{n \log^2 n}{\log \log n}\right)$ time.*

Next, we give an algorithm when the items can be rotated by 90 degrees about any axis.

`3DR-Vol-Pack`. Let $T_\ell = \{i \in T \mid w_i > w_B/2 \text{ and } d_i > d_B/2\}$, and let $T_s = T \setminus T_\ell$. First, we pack T_s in layers using `3D-Vol-Pack`. Next, we sort the items of T_ℓ in non-increasing order of their widths and pack them in layers, one in each layer, touching the left face of the box, as long as the height of the box is not exceeded. Following this, we rotate the remaining items of T_ℓ about the depth dimension so that their widths and heights are interchanged, and as long as possible, pack them next to each other starting from the right face of the box, such that each item touches the top face of the box (see Figure 3).

The next lemma states that when B is a cube and when the height of each item is very small compared to box B, we can pack a significant volume.

▶ **Lemma 5.** *Given a cubical box B of side length w, and a set T of n items where for each $i \in T$, there exists an orientation so that $w_i \leq w$, $d_i \leq w$ and $h_i \leq \varepsilon^2 w$ holds. If $v(T) \leq \left(\frac{7}{24} - 5\varepsilon\right) v(B)$, then T can be packed into B by `3DR-Vol-Pack` in $O\left(\frac{n \log^2 n}{\log \log n}\right)$ time.*

3 Container Packing

In this section, we first define container packing and describe the containers used in our work. Then, we state our main structural result which shows that there exist container packings with high profit that can be found in polynomial time.

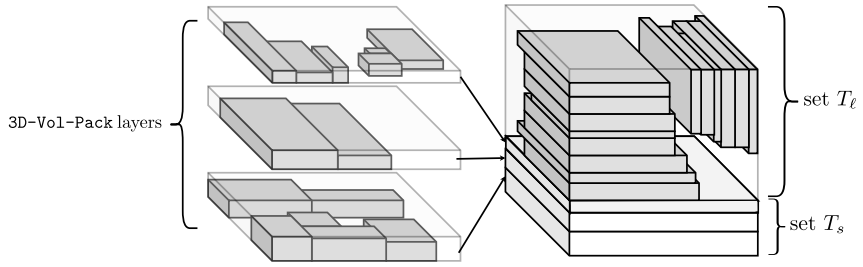


Figure 3 An example packing by Algorithm `3DR-Vol-Pack`.

▶ **Definition 6** (Container Packing). *A packing of a set of items $I' \subseteq I$ is said to be a Container Packing if*

- *the knapsack can be partitioned into a collection $\mathcal{C}$ of non-overlapping regions called containers and some empty spaces. Each container $C \in \mathcal{C}$ has an associated capacity $\mathtt{cap}(C)$, a function $f_C \colon I \to \mathbb{Q}_{\geq 0}$, and a polynomial-time algorithm $\mathcal{A}_C$,*
- *there is a function $g \colon I' \to \mathcal{C}$, such that for each $C \in \mathcal{C}$, items in $g^{-1}(C)$ are packed into C using algorithm $\mathcal{A}_C$,*
- *for any $C \in \mathcal{C}$ and $T \subseteq I$ such that $\sum_{i \in T} f_C(i) \leq \mathtt{cap}(C)$, for any constant $\varepsilon > 0$, there exists a polynomial-time computable set $T' \subseteq T$ with $p(T') \geq (1 - O(\varepsilon))p(T)$ such that T' can be packed into C by the algorithm $\mathcal{A}_C$.*

A Container Packing is said to be *guessable* if

- the number of containers inside the knapsack is bounded by a constant $O_\varepsilon(1)$,[1]
- the number of distinct possible types of containers (defined by their sizes along each dimension) is bounded by $n^{O_\varepsilon(1)}$, a polynomial in n,
- whether a given collection of $O_\varepsilon(1)$ containers can be placed non-overlappingly inside the knapsack, can be checked in $n^{O_\varepsilon(1)}$ (polynomial) time.

We shall call a container *guessable* if it comes from some polynomially-sized set of distinct container types. Though, in this paper we only use cuboids as containers, the idea of container packing can be generalized to other shapes in 3D or higher dimensions.

▶ **Theorem 7.** *There is a PTAS for maximum profit guessable container packing.*

Proof. We consider all feasible choices of containers that fit into the 3D knapsack, and for each such choice, we create an instance of GAP with one (1D) knapsack per container, where the (1D) knapsack corresponding to a container C has capacity $\mathtt{cap}(C)$ and the size of an item i for the (1D) knapsack is given by $f_C(i)$. The profit of an item for any (1D) knapsack is set to be the same as the actual profit of the item. Using the PTAS for GAP with $O(1)$ knapsacks [11], we obtain a feasible assignment of items into the containers, and finally for each container C, we use the algorithm $\mathcal{A}_C$ to pack a subset of the items assigned to it, by losing only an $O(\varepsilon)$-fraction of profit. ◀

3.1 Classification of containers

We now describe the various types of containers that we use to pack items. For a container C, its width, depth, and height are denoted by w_C, d_C, and h_C, respectively, and its volume

[1] The notation $O_\varepsilon(f(n))$ means that the implicit constant hidden in big-O notation can depend on ε.

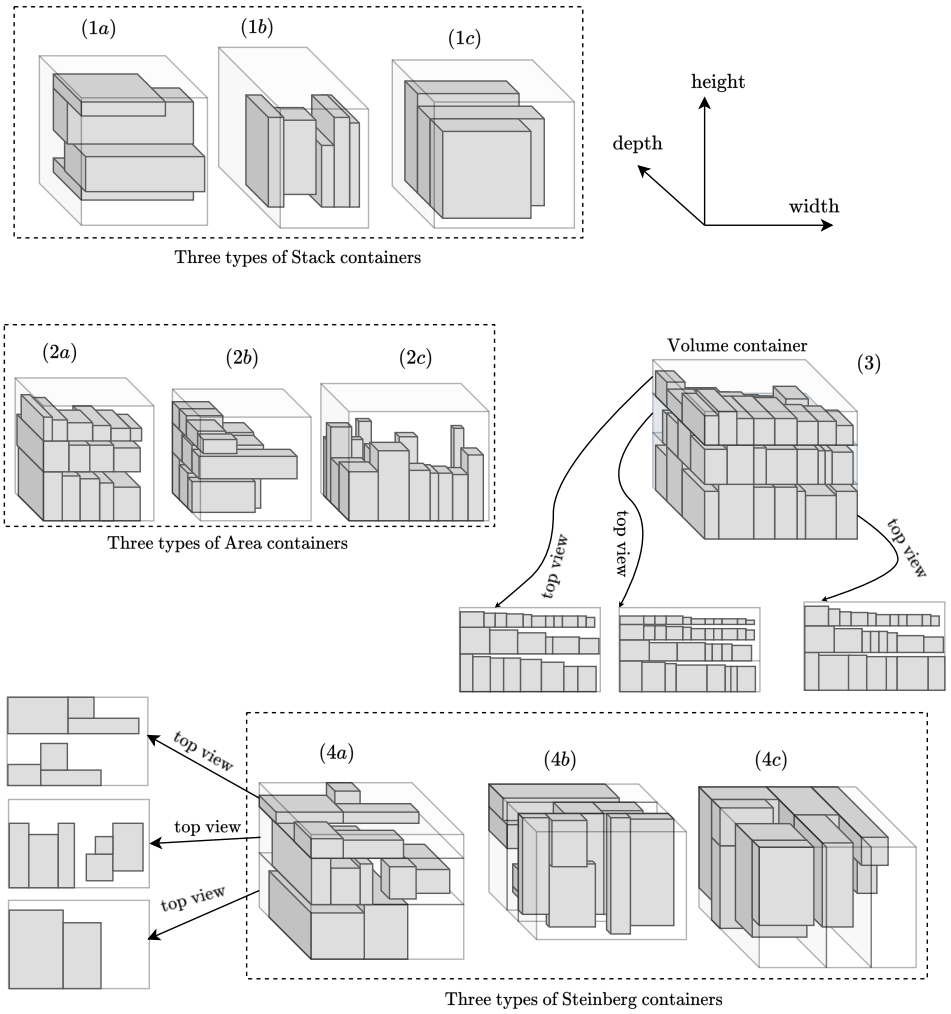


Figure 4 Different kinds of containers that we use. We have one type of Volume container and three types of Stack, Area, and Steinberg containers (one for each dimension).

$v(C) := w_C \times d_C \times h_C$. For each container C, we specify its capacity $\texttt{cap}(C)$, the function f_C, and the packing algorithm $\mathcal{A}_C$. Let T denote the set of items assigned to C such that $\sum_{i \in T} f_C(i) \leq \texttt{cap}(C)$. We suggest the reader to follow the descriptions alongside Figure 4.

- **Stack containers:** These containers pack items in layers, with one item in each layer. We consider a container C that stacks items along the height. See container $(1a)$ in Figure 4. The capacity $\texttt{cap}(C)$ of such a container is set to its height h_C, and for any item i, $f_C(i) := h_i$ if $w_i \leq w_C$ and $d_i \leq d_C$, and $f_C(i) := \infty$ otherwise.

  $\mathcal{A}_C$ just stacks the items of set T in the container C one above the other. Analogously containers $(1b)$ and $(1c)$ show containers that stack along width and depth, respectively.
- **Area containers:** Inside these containers, the items are packed using the `NFDH` algorithm, by ignoring one of the dimensions. We describe Area containers that use `NFDH` along the front face of the container (see container $(2a)$). For such a container C, we set $\texttt{cap}(C) := w_C h_C$, and for any item i, we set $f_C(i) := w_i h_i$ if $w_i \leq \varepsilon w_C$, $h_i \leq \varepsilon h_C$ and $d_i \leq d_C$, and $f_C(i) := \infty$ otherwise.

$\mathcal{A}_C$ first sorts the items of set T in non-increasing order of profit/front area ratio. Then, we select the largest prefix whose total front area does not exceed $(1-2\varepsilon)w_C h_C$. We pack this prefix using `NFDH` on the front face. Analogously, container $(2b)$ and $(2c)$ show packing when `NFDH` is used on the left and bottom face of C, respectively.

- **Volume containers:** The items are packed using `3D-NFDH` inside these containers (see container (3)). The capacity of such a container C is set to be its volume $v(C)$, and $f_C(i) := v(i)$ if $w_i \leq \varepsilon w_C$, $d_i \leq \varepsilon d_C$ and $h_i \leq \varepsilon h_C$, and $f_C(i) := \infty$ otherwise.
  $\mathcal{A}_C$ sorts the items of T in non-decreasing order of profit/volume and selects the largest prefix with volume not exceeding $(1-3\varepsilon)w_C h_C d_C$. We pack this prefix using `3D-NFDH`.
- **Steinberg containers:** These containers pack items in layers using `3D-Vol-Pack`. We describe Steinberg containers that pack items in layers along the height (analogous for other cases). We set $\texttt{cap}(C) := v(C)/3$. For any item i, we set $f_C(i) := v(i)$ if $h_i \leq \varepsilon h_C$, and either $w_i \leq w_C/2$ or $d_i \leq d_C/2$ holds. For all other items, we set $f_C(i) := \infty$.
  $\mathcal{A}_C$ first sorts the items in T in non-increasing order of profit/volume ratio. Then, we select the largest prefix whose volume does not exceed $\left(\frac{1}{3} - 2\varepsilon\right) v(C)$, and pack this prefix using `3D-Vol-Pack`. Analogously, container $(4b)$ and $(4c)$ show packing when layers are stacked along the depth and width, respectively.

3.2 Our Structural Lemma

Let $\texttt{opt}_{\text{gs}}$ be the maximum profit of a guessable container packing. The main structural result of our paper is the following.

▶ **Lemma 8.** *For any $\varepsilon > 0$, the optimal profit* `opt` *and the optimal profit of a guessable container packing* $\texttt{opt}_{\text{gs}}$ *satisfies* $\texttt{opt} \leq \left(\frac{139}{29} + \varepsilon\right) \texttt{opt}_{\text{gs}}$.

Together with Theorem 7, this gives us the following theorem.

▶ **Theorem 9.** *For any constant $\varepsilon > 0$, there is a polynomial-time $\left(\frac{139}{29} + \varepsilon\right)$-approximation algorithm for 3DK.*

We prove Lemma 8 in Section 4 by establishing several lower bounds on $\texttt{opt}_{\text{gs}}$. Specifically, we present five different ways to restructure OPT into a guessable container packing. Our first lower bound is based on a structural result of the 3D Strip Packing problem due to [19]. The second lower bound is obtained by restructuring the packing of items in $\text{OPT} \cap (I_{1\ell} \cup L)$ (or similarly $\text{OPT} \cap (I_{2\ell} \cup L)$ or $\text{OPT} \cap (I_{3\ell} \cup L)$). The third and fourth lower bounds are based on volume-based arguments, using our simple packing algorithm `3D-Vol-Pack`. Our final lower bound is based on a structural result from the 2D Knapsack problem [11].

4 Proof of Structural Lemma

Now we prove the structural lemma. First, we classify the items.

Classification of items. Let $\varepsilon > 0$ be an accuracy parameter. Let μ be a sufficiently small constant depending on ε (setting $\mu := \varepsilon^{2^{1/\varepsilon^2}}$ suffices). We now classify the input items based on their dimensions. Let $L \subseteq I$ be the set of items whose width, depth and height all exceed μ. Let $I_1 \subseteq I$ be the set of items having height at most μ. Similarly, let $I_2 \subseteq I \setminus I_1$ be those items whose widths are at most μ, and $I_3 := I \setminus (L \cup I_1 \cup I_2)$ be the remaining items. Thus each item in I_3 has depth at most μ. Let $I_{1\ell}$ consist of the items in I_1 whose width and depth are both more than $1/2$, and let $I_{1s} = I_1 \setminus I_{1\ell}$. See Figure 5. The sets $I_{2\ell}, I_{2s}, I_{3\ell}, I_{3s}$ are defined analogously.

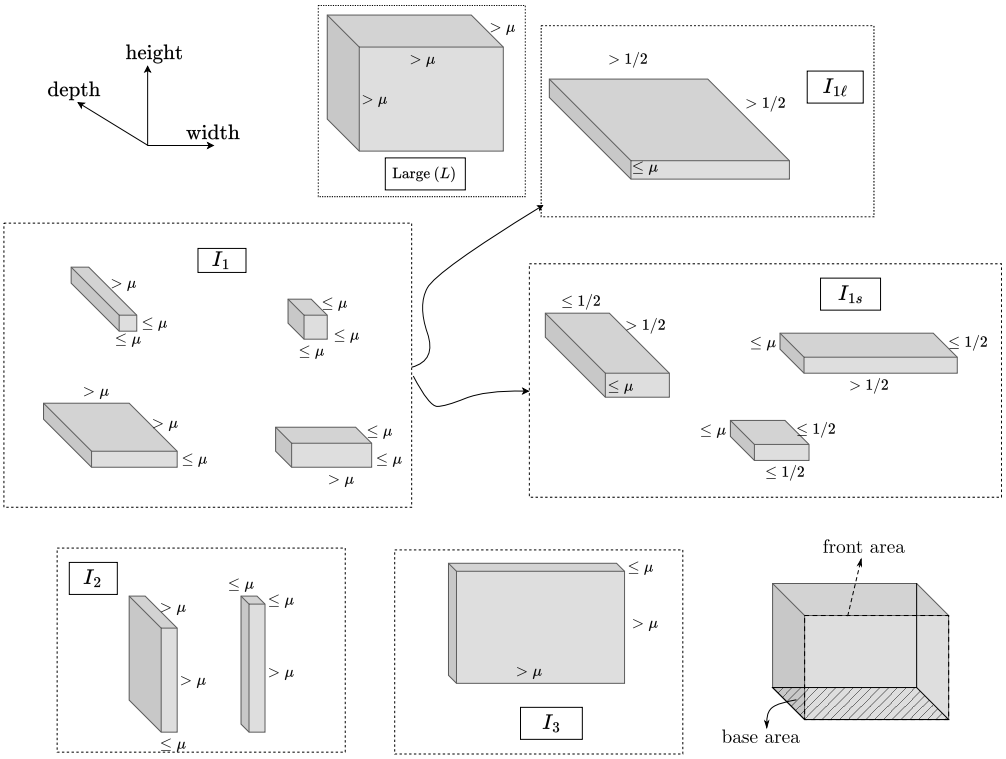


Figure 5 Classification of input items. Front area and base area are shown at the bottom right.

Let $\mathtt{opt}_i := p(\mathrm{OPT} \cap I_i)$ for $i \in [3]$, and $\mathtt{opt}_L := p(\mathrm{OPT} \cap L)$, so that $\mathtt{opt} = \mathtt{opt}_1 + \mathtt{opt}_2 + \mathtt{opt}_3 + \mathtt{opt}_L$. Let $\mathtt{opt}_{1\ell} := p(\mathrm{OPT} \cap I_{1\ell})$ and $\mathtt{opt}_{1s} := p(\mathrm{OPT} \cap I_{1s})$, so that $\mathtt{opt}_{1\ell} + \mathtt{opt}_{1s} = \mathtt{opt}_1$. The quantities $\mathtt{opt}_{2\ell}, \mathtt{opt}_{2s}$ and $\mathtt{opt}_{3\ell}, \mathtt{opt}_{3s}$ are defined analogously for the sets I_2 and I_3, respectively.

Finally, let $v_1 := v(\mathrm{OPT} \cap I_1)$, and define v_2, v_3 analogously. Clearly, v_1, v_2, v_3 sum up to at most the volume of the knapsack which is 1. We define $h_{1\ell} := h(\mathrm{OPT} \cap I_{1\ell})$ and $v_{1s} := v(\mathrm{OPT} \cap I_{1s})$. The quantities v_{2s}, v_{3s} are defined analogously.

4.1 First lower bound on $\mathtt{opt}_{\mathbf{gs}}$

We consider the packing of the items in $\mathrm{OPT} \cap I_1$ (i.e., items with height at most μ) whose total profit is $\mathtt{opt}_1$. We build on the structure derived from the 3D Strip Packing algorithm of Jansen and Prädel [19]. However, we need several refinements to transform the packing of [19] into a guessable container packing.

First, using standard arguments, we discard a set of (medium) items (items whose widths or depths lie in the range $[\delta^{10}, \delta)$ for a suitably chosen δ) at a $(1+\varepsilon)$-factor loss in the approximation ratio. Then the items are divided into four classes depending on their dimensions. An item i is big if both $w_i \geq \delta$ and $d_i \geq \delta$, wide if $w_i \geq \delta$ and $d_i < \delta^{10}$, long if $w_i < \delta^{10}$ and $d_i \geq \delta$, and small if both $w_i < \delta^{10}$ and $d_i < \delta^{10}$. The following lemma is obtained by an adaptation of the result of [19] (see Figure 6).

▶ **Lemma 10.** *There exist disjoint sets $T_1, T_2 \subseteq OPT \cap I_1$ having a profit of at least $\left(\frac{2}{3} - O(\varepsilon)\right) \mathtt{opt}_1$, such that*

- *Items of T_1 are packed into a collection of $O(1/\delta^6)$ configurations, where each configuration is a box having a 1×1 base, that is partitioned into $O(1/\delta^2)$ (cuboidal) slots for packing big items, $O(1/\delta^6)$ slots for packing wide and long items (using NFDH), and $O(1/\delta^5)$ slots for packing small items (using 3D-NFDH). The height of a slot is the same as the height of the corresponding configuration,*
- *The configurations are stacked one on top of the other inside the knapsack and their heights sum up to at most $1 - 2\varepsilon$,*
- *The set T_2 contains only wide, long and small items and has a total volume of $O(\delta^4)$.*

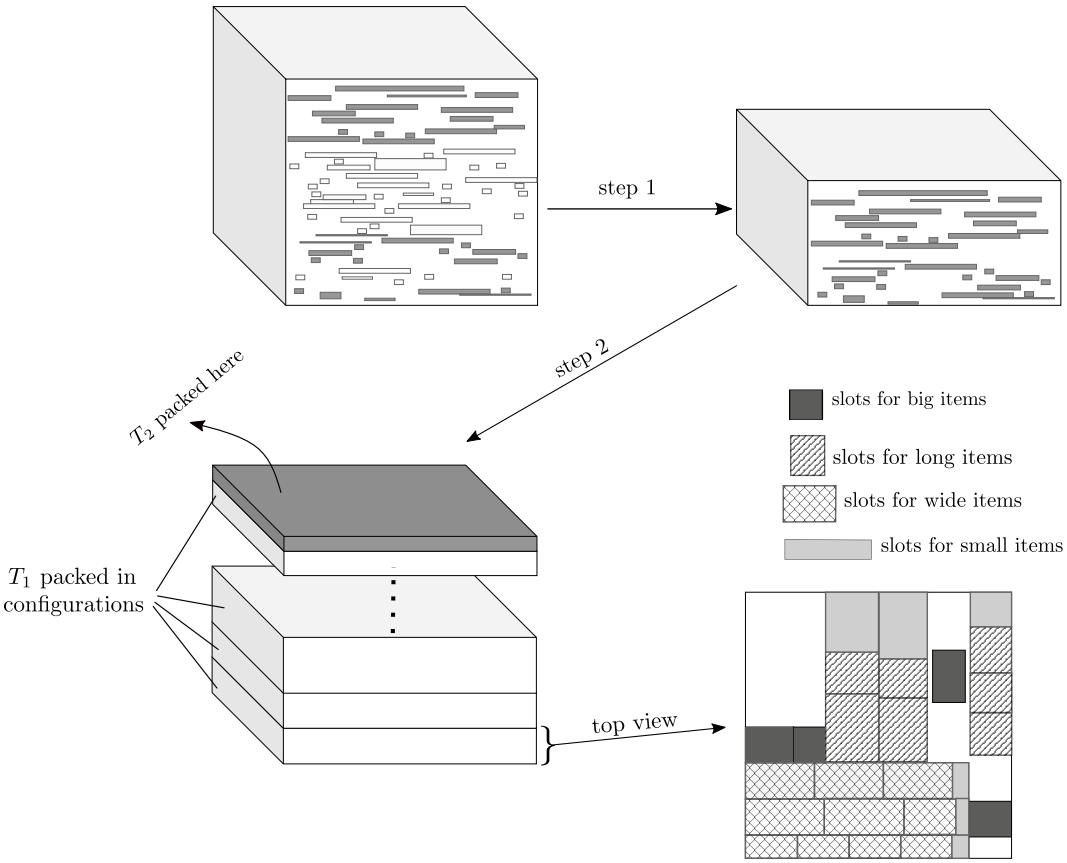


Figure 6 In Step 1, we remove at most $(1/3 + O(\varepsilon))$ fraction of the profit. In Step 2, we obtain a structured packing by [19].

Our objective is to build Area and Volume containers out of the slots for packing the wide/long and small items, respectively. The trouble is that the slot dimensions might not be large enough compared to the dimensions of the packed items, which is a necessary condition for the construction of such containers (see Section 3.1). We obtain a guessable container packing by the following lemma.

▶ **Lemma 11.** *By losing a profit of at most $\delta \mathtt{opt}_1$, and rounding the height of each configuration to the next integer multiple of μ/ε, each configuration for T_1 can be further partitioned into $O(1/\delta^6)$ guessable containers. Moreover, the items of T_2 can be packed inside a Steinberg container of height ε.*

After applying the above lemma, the rounded heights of all the configurations for T_1 sum up to at most $(1-2\varepsilon) + \frac{\mu}{\varepsilon} \cdot O(\frac{1}{\delta^6}) \leq 1-\varepsilon$, for sufficiently small μ. Since items of T_2 are packed inside a Steinberg container of height ε, all the containers fit inside the knapsack and we obtain the following theorem overall.

▶ **Theorem 12.** *We can repack a subset of $OPT \cap I_1$ with profit at least $\left(\frac{2}{3} - O(\varepsilon)\right) \mathtt{opt}_1$ into $O(1/\delta^{12})$ guessable Stack, Area, and Volume containers.*

An equivalent restructuring of $\mathrm{OPT} \cap I_2$ and $\mathrm{OPT} \cap I_3$ gives the following corollary.

▶ **Corollary 13.** $\mathtt{opt}_{\mathrm{gs}} \geq \left(\frac{2}{3} - O(\varepsilon)\right) \max\{\mathtt{opt}_1, \mathtt{opt}_2, \mathtt{opt}_3\}$.

4.2 A simple $(5+\varepsilon)$-approximation

We now present a simple $(5+\varepsilon)$-approximation for 3DK, which already improves upon the $(7+\varepsilon)$-approximation [10]. For this, first, we provide the following packing guarantee.

▶ **Lemma 14.** *Let T be a set of items such that each $i \in T$ satisfies $h_i \leq \varepsilon^4$. If $v(T) \leq 1/4$, then there exists $T' \subseteq T$ with $p(T') \geq (1-\varepsilon)p(T)$ such that there is a guessable container packing of T' into a Stack and a Steinberg container.*

Then the following lemma, together with Theorem 7, gives a $(5+\varepsilon)$-approximation.

▶ **Lemma 15.** $\mathtt{opt} \leq (5+\varepsilon)\mathtt{opt}_{\mathrm{gs}}$.

Proof. Trivially we have $\mathtt{opt}_{\mathrm{gs}} \geq \mathtt{opt}_L$, since we can create a Stack container for every item in $\mathrm{OPT} \cap L$ (at most $1/\mu^3$ such items). Hence if $\mathtt{opt}_L > \mathtt{opt}/5$, we are already done. Otherwise we have $\mathtt{opt}_1 + \mathtt{opt}_2 + \mathtt{opt}_3 \geq \frac{4}{5}\mathtt{opt}$. Now since $v_1 + v_2 + v_3 \leq 1$, there must exist an $i \in [3]$ such that $\frac{\mathtt{opt}_i}{v_i} \geq \frac{4}{5}\mathtt{opt}$ – w.l.o.g. assume $i = 1$. We have two cases.

Case 1: $v_1 \leq \frac{1}{4}$**.** In this case, Lemma 14 implies that $\mathtt{opt}_{\mathrm{gs}} \geq (1-\varepsilon)\mathtt{opt}_1$ (since each item in I_1 has height at most $\mu < \varepsilon^4$). Hence if $\mathtt{opt}_1 > \mathtt{opt}/5$, we are done. Otherwise we have $\mathtt{opt}_2 + \mathtt{opt}_3 \geq \frac{3}{5}\mathtt{opt}$. W.l.o.g. assume that $\mathtt{opt}_2 \geq \frac{3}{10}\mathtt{opt}$, Corollary 13 gives us $\mathtt{opt}_{\mathrm{gs}} \geq \left(\frac{2}{3} - O(\varepsilon)\right) \cdot \frac{3}{10}\mathtt{opt} = \left(\frac{1}{5} - O(\varepsilon)\right)\mathtt{opt}$.

Case 2: $v_1 > \frac{1}{4}$**.** In this case, we sort the items of $\mathrm{OPT} \cap I_1$ in non-increasing order of their profit to volume ratio, and pick the maximal prefix T whose volume does not exceed $\frac{1}{4}$. Then we have $v(T) \geq \frac{1}{4} - \mu$. Applying Lemma 14 to T, we obtain $\mathtt{opt}_{\mathrm{gs}} \geq (1-\varepsilon)\left(\frac{1/4-\mu}{v_1}\right)\mathtt{opt}_1 \geq \left(\frac{1}{5} - O(\varepsilon)\right)\mathtt{opt}$, thus completing the proof. ◀

The ratio of $(5+\varepsilon)$ is tight. One such instance is given when $v_1 = \frac{1}{8}, v_2 = v_3 = \frac{3}{8}$ and $\mathtt{opt}_1 = \mathtt{opt}_L = \frac{1}{5}\mathtt{opt}$ and $\mathtt{opt}_2 = \mathtt{opt}_3 = \frac{3}{10}\mathtt{opt}$.

We will obtain further improvements by using the lower bounds in the following sections.

4.3 Second lower bound on $\mathtt{opt}_{\mathrm{gs}}$

In this subsection, we consider the packing of $\mathrm{OPT} \cap (I_{1\ell} \cup L)$, and restructure it to obtain a guessable container packing, by losing only an $O(\varepsilon)$-fraction of the profit. Let $\mathrm{OPT}_{1\ell} := \mathrm{OPT} \cap I_{1\ell}$ and $\mathrm{OPT}_L := \mathrm{OPT} \cap L$. Since each item in L has all side lengths at least μ, the number of such items in the packing is $|\mathrm{OPT}_L| \leq 1/\mu^3$. Also, since the width and depth of each item in $I_{1\ell}$ is at least $1/2$, the items of $\mathrm{OPT}_{1\ell}$ must be packed in layers, i.e., the top and bottom faces of any item in $\mathrm{OPT}_{1\ell}$ when extended parallel to the bottom face of the knapsack will not intersect another item from $\mathrm{OPT}_{1\ell}$. We now discretize the positions of the items in the packing.

▶ **Lemma 16.** *The distance of the left (resp. front) face of any item from the left (resp. front) face of the knapsack can be assumed to lie in a polynomial-sized set.*

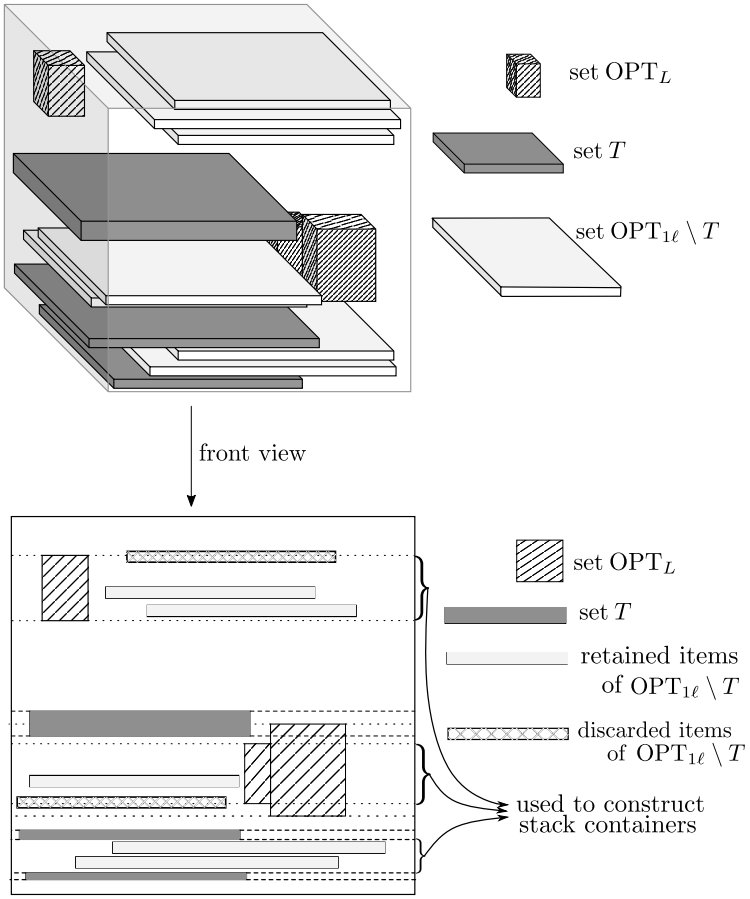


Figure 7 Packing of items in $OPT \cap (L \cup I_{1\ell})$: The dotted (resp. dashed) lines show (front view of) horizontal planes passing through the top and bottom faces of items in OPT_L (resp. T). We discard the items in $\mathrm{OPT}_{1\ell} \setminus T$ that are cut by these planes.

Let $T \subseteq \mathrm{OPT}_{1\ell}$ be the k items of largest profit in $\mathrm{OPT}_{1\ell}$, for some k to be chosen later. Our goal is to obtain a container packing of all items in OPT_L, together with a subset of $\mathrm{OPT}_{1\ell}$ having a profit of at least $(1-\varepsilon)\mathtt{opt}_{1\ell}$. To this end, we shall pack all items in T and ensure that the number of discarded items of $\mathrm{OPT}_{1\ell}$ is bounded by ck, for some constant c. By a result of [18], the profit of the discarded items can be bounded by $\varepsilon\mathtt{opt}_{1\ell}$, for appropriately chosen k. We now outline the details of our restructuring procedure. See Figure 7.

We draw horizontal planes passing through the top and bottom faces of the items in $T \cup \mathrm{OPT}_L$, and discard the items of $\mathrm{OPT}_{1\ell} \setminus T$ intersecting these planes (see Figure 7). The number of discarded items is bounded by $2|\mathrm{OPT}_L| = O(1/\mu^3)$. The remaining items of $\mathrm{OPT}_{1\ell} \setminus T$ are now completely packed inside at most $k + 2|\mathrm{OPT}_L| + 1$ horizontal strips demarcated by the planes. Each such strip is a cuboidal box that is pierced from top to bottom by at most $1/\mu^2$ Large items. Further, there are only polynomially-many positions of these Large items inside the strip owing to Lemma 16. We focus on one such strip $\mathcal{S}$.

▶ **Lemma 17.** *The items of $OPT_{1\ell} \setminus T$ present inside the strip $\mathcal{S}$ can be repacked into at most $O(1/\mu^8)$ Stack containers, whose widths and depths lie in a polynomial-sized set.*

For each item in $T \cup \mathrm{OPT}_L$, we create a Stack container containing the single item itself. Together with Lemma 17, we get $O_\mu(1)$ number of Stack containers inside the knapsack. We now discretize the heights of the Stack containers, so that they lie in a polynomial-sized set.

▶ **Lemma 18.** *By discarding $O(1/\mu^{11})k$ items from $OPT_{1\ell} \setminus T$, the height of each Stack container can be made of the form $j\bar{h}$, where $j \in [n]$ and $\bar{h}$ is the height of some item in $I_{1\ell} \cup L$.*

Together with the $O(1/\mu^3)$ items that were discarded previously, Lemma 18 gives us that the total number of discarded items so far is at most $O(1/\mu^{11})k$. We now use the following lemma from [18].

▶ **Lemma 19** ([18])**.** *Let $p_1 \geq p_2 \geq \ldots \geq p_n > 0$ be a sequence of real numbers and $P = \sum_{i=1}^{n} p_i$. Let c be a nonnegative integer and $\varepsilon > 0$. If $n = c^{\Omega(1/\varepsilon)}$, then there is an integer $k \leq c^{O(1/\varepsilon)}$ such that $p_{k+1} + \ldots p_{k+ck} \leq \varepsilon P$.*

The above lemma invoked with $c = O(1/\mu^{11})$ and with the correct guess of k gives us that the total profit of the discarded items is at most $\varepsilon \mathtt{opt}_{1\ell}$, assuming that $|\mathrm{OPT}_{1\ell}|$ was sufficiently large (at least $n_0 := (1/\mu)^{\Omega(1/\varepsilon)}$) to begin with (otherwise, it is trivial). The following theorem summarizes the arguments of this section.

▶ **Theorem 20.** *Consider the packing of $\mathrm{OPT} \cap (I_{1\ell} \cup L)$. Then by discarding items of $\mathrm{OPT} \cap I_{1\ell}$ having a total profit of at most $\varepsilon \mathtt{opt}_{1\ell}$, the remaining items can be repacked into a guessable container packing, wherein items of L are placed inside at most $1/\mu^3$ Stack containers, and items of $I_{1\ell}$ are packed into at most $(1/\mu)^{O(1/\varepsilon)}$ Stack containers.*

4.4 Third lower bound on $\mathtt{opt}_{\mathbf{gs}}$

In this subsection, we restructure the packing of items in $\mathrm{OPT} \cap I_1$. Recall that $\mathtt{opt}_{1\ell} = p(\mathrm{OPT} \cap I_{1\ell})$ and $\mathtt{opt}_{1s} = p(\mathrm{OPT} \cap I_{1s})$, so that $\mathtt{opt}_{1\ell} + \mathtt{opt}_{1s} = \mathtt{opt}_1$. Also, $v_1 := v(\mathrm{OPT} \cap I_1)$ and $v_{1s} = v(\mathrm{OPT} \cap I_{1s})$.

▶ **Theorem 21.** *Consider the packing of $OPT \cap I_1$. It is possible to repack subsets of $OPT \cap I_{1\ell}$ and $OPT \cap I_{1s}$ into a guessable Stack and Steinberg container, respectively, whose profit, say P, satisfies the following properties:*

- $P \geq (1 - O(\varepsilon)) \dfrac{\mathtt{opt}_{1s}}{\max\{3v_{1s}, 1\}}$,
- *If $v_1 \leq 1/3$, then* $P \geq (1 - O(\varepsilon)) \left(\dfrac{3}{4} \mathtt{opt}_{1\ell} + \mathtt{opt}_{1s} \right)$,
- *If $v_1 \leq 1/4$, then* $P \geq (1 - O(\varepsilon)) \mathtt{opt}_1$.

4.5 Fourth lower bound on $\mathtt{opt}_{\mathbf{gs}}$

In the previous sections, we repacked items from exactly one set out of I_1, I_2, I_3. In this subsection, we restructure the packing of a subset of items from I_1, together with items from $I_2 \cup I_3 \cup L$. To this end, we introduce a classification of the items in $I_2 \cup I_3 \cup L$. Let S_2 (resp. S_3, S) be the set of items in I_2 (resp. I_3, L) with height at most $1/2$. Define $\mathtt{opt}_{2t} := p(\mathrm{OPT} \cap S_2)$ and $\mathtt{opt}_{2h} := p(\mathrm{OPT} \cap (I_2 \setminus S_2))$, so that $\mathtt{opt}_{2t} + \mathtt{opt}_{2h} = \mathtt{opt}_2$. The quantities $\mathtt{opt}_{3t}$ and $\mathtt{opt}_{3h}$ are defined analogously. Let $\mathtt{opt}_{Lt} := p(\mathrm{OPT} \cap S)$ and $\mathtt{opt}_{Lh} := p(\mathrm{OPT} \cap (L \setminus S))$ so that $\mathtt{opt}_{Lt} + \mathtt{opt}_{Lh} = \mathtt{opt}_L$. The following theorem repacks items from $I_1 \cup S_2 \cup S_3 \cup S$.

▶ **Theorem 22.** *Consider the packing of* $OPT \cap (I_1 \cup S_2 \cup S_3 \cup S)$ *and assume that* $v_1 \leq 1/6$ *and* $v_2, v_3 > 1/4$*. Then there is a guessable container packing of a subset* T *of these items into* $O(1/\mu^3)$ *Stack and Steinberg containers having a profit of at least*

$$(1 - O(\varepsilon)) \left(\frac{3}{4}\mathtt{opt}_{1\ell} + \mathtt{opt}_{1s} + \max\left\{ \frac{3}{20}(\mathtt{opt}_{2t} + \mathtt{opt}_{3t}), \frac{1}{3}\mathtt{opt}_{Lt} \right\} \right).$$

4.6 Fifth lower bound on $\mathtt{opt}_{\mathtt{gs}}$

In the previous subsection, we repacked items from S_2, S_3 or S together with items in I_1. Now we consider items in $(I_2 \setminus S_2) \cup (I_3 \setminus S_3) \cup (L \setminus S)$ in OPT. As all these items have heights of more than 1/2, they can not be stacked on top of each other. So, we can ignore their heights, and effectively obtain a 2D Knapsack instance. Now we use a container-based algorithm for 2DK from [22] to obtain a container packing of these items.

▶ **Theorem 23.** *Consider the packing of* $OPT \cap ((I_2 \cup I_3 \cup L) \setminus (S_2 \cup S_3 \cup S))$*. Then there is a guessable container packing into* $O_\mu(1)$ *Stack and Area containers each of height 1, having profit at least* $\left(\frac{1}{2} - O(\varepsilon)\right)(\mathtt{opt}_{2h} + \mathtt{opt}_{3h} + \mathtt{opt}_{Lh})$*.*

4.7 Bounding $\mathtt{opt}_{\mathtt{gs}}$

Equipped with all the lower bounds of the preceding subsections, we establish Lemma 8 by an intricate case analysis depending on the volumes of various sets of items in the optimal packing. Since v_1, v_2, v_3 sum up to at most the volume of the knapsack which is 1, at least one among them must not exceed 1/3. If at most one of them exceeds 1/3, we show a stronger bound of $\left(\frac{9}{2} + \varepsilon\right)$ on the approximation ratio. Otherwise, we assume w.l.o.g. that $v_1 \leq 1/3$ and $v_2, v_3 > 1/3$. We further divide into subcases depending on the values of v_{2s} and v_{3s}. It turns out that the worst bound on $\mathtt{opt}_{\mathrm{gs}}$ occurs when v_{2s}, v_{3s} both exceed 1/3 and $v_1 \leq 1/6$. However for this case, Theorem 22 and Theorem 23 give us highly profitable packings, and we achieve an approximation ratio of $\left(\frac{139}{29} + \varepsilon\right)$. The details of the analysis are deferred to the full version [16].

5 Other variants

Using our techniques, we obtain further improvements for other variants: (1) cardinality case (when all items have the same profit), and (2) uniform profit-density case (when the profit of an item is equal to its volume).

▶ **Theorem 24.** *For any constant* $\varepsilon > 0$*, there is a polynomial-time* $(17/4 + \varepsilon)$*-approximation algorithm for 3DK when all items have the same profit.*

▶ **Theorem 25.** *For any constant* $\varepsilon > 0$*, there is a polynomial-time* $(4 + \varepsilon)$*-approximation algorithm for 3DK when the profit of an item is equal to its volume.*

5.1 Rotational case

When the items can be rotated by 90 degrees around any axis, we show the existence of improved container packing. Specifically, we introduce a new type of container called L-Container and obtain improved guarantees based on the packed volume.

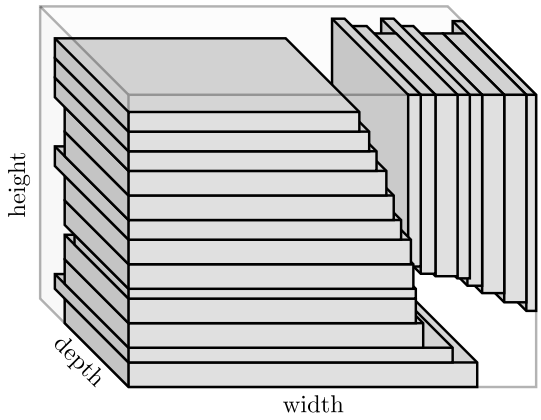


Figure 8 An example packing of items inside an L-Container.

L-Container. We describe L-Containers which ignore depth while packing the items (see Figure 8; two other types of L-Containers exist which ignore width, height, respectively). We first require that C must satisfy the condition $w_C \geq h_C$. The capacity of C, $\texttt{cap}(C)$, is given by $w_C h_C - w_C^2/4$. For each item i packed inside C, there must be some orientation with width at least $w_C/2$, depth at least $d_C/2$, and height at most εh_C. If these conditions are satisfied, then we set $f_C(i) \coloneqq w_i h_i$ and ∞ otherwise. Algorithm $\mathcal{A}_C$ is as follows. Given a set of items T satisfying $f_C(T) \leq \texttt{cap}(C)$, we arrange them in non-increasing order of profit/(front area) ratio. Then, we select the largest prefix T' whose total front area does not exceed $w_C h_C - w_C^2/4 - 3\varepsilon h_C^2$. We now sort the items in T' in non-increasing order of widths, start from the bottom left corner, and keep stacking them one upon each other to the maximum extent possible. Then, we rotate the remaining items by 90 degrees around the depth axis, start from the top right corner and keep packing them side by side.

The main ingredient to obtain improved approximation ratios in the rotational case is our algorithm `3DR-Vol-Pack` (see Section 2.1). Although, as stated the algorithm does not pack into (guessable) containers, it can be modified to return a guessable container packing into Stack, Steinberg, and L-Containers. Using this, we obtain the following theorems for the general case, cardinality case, and uniform profit-density case, respectively.

▶ **Theorem 26.** *For any constant $\varepsilon > 0$, there is a polynomial-time $(30/7 + \varepsilon)$-approximation algorithm for 3DKR.*

▶ **Theorem 27.** *For any constant $\varepsilon > 0$, there is a polynomial-time $(24/7 + \varepsilon)$-approximation algorithm for 3DKR when all items have the same profit.*

▶ **Theorem 28.** *For any constant $\varepsilon > 0$, there is a polynomial-time $(3 + \varepsilon)$-approximation algorithm for 3DKR when the profit of an item is equal to its volume.*

6 Conclusion

In this work, our primary focus has been to obtain improved polynomial-time approximation guarantees. Our current runtime of $n^{f(1/\varepsilon)}$ is mainly because of the step where we guess the containers and assign the items. Once this is done, every $\mathcal{A}_C$ runs in time $O(n \log^2 n)$. Hence, one interesting direction is whether the time to guess the containers and assign the items can be brought down to $(n \cdot (\log n)^{f(1/\varepsilon)})$ using the recent techniques of [4]. Another interesting open problem is to provide a tight approximation for algorithms based on container packing.

References

1 Sara Ali, António Galrão Ramos, Maria Antónia Carravilla, and José Fernando Oliveira. Online three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, 168:108122, 2022. doi:10.1016/j.cie.2022.108122.

2 Helmut Alt. Computational aspects of packing problems. *Bulletin of EATCS*, 1(118), 2016.

3 Nikhil Bansal, José R. Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006. doi:10.1287/moor.1050.0168.

4 Moritz Buchem, Paul Deuker, and Andreas Wiese. Approximating the geometric knapsack problem in near-linear time and dynamically. In *Symposium on Computational Geometry (SoCG)*, volume 293, pages 26:1–26:14, 2024. doi:10.4230/LIPIcs.SoCG.2024.26.

5 Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. Knapsack problems – An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143:105693, 2022. doi:10.1016/j.cor.2021.105693.

6 Alberto Caprara. Packing d-dimensional bins in d stages. *Mathematics of Operations Research*, 33(1):203–215, 2008. doi:10.1287/moor.1070.0289.

7 Miroslav Chlebík and Janka Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. *Journal of Discrete Algorithms*, 7(3):291–305, 2009. doi:10.1016/j.jda.2009.02.002.

8 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.

9 Edward G. Coffman, Jr, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980. doi:10.1137/0209062.

10 Florian Diedrich, Rolf Harren, Klaus Jansen, Ralf Thöle, and Henning Thomas. Approximation algorithms for 3D orthogonal knapsack. *Journal of Computer Science and Technology*, 23(5):749–762, 2008. doi:10.1007/s11390-008-9170-7.

11 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, Sandy Heydrich, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. *ACM Transactions on Algorithms*, 17(4):33:1–33:67, 2021. doi:10.1145/3473713.

12 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramírez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple L-shapes, spirals, and more. In *International Symposium on Computational Geometry (SoCG)*, pages 39:1–39:17, 2021. doi:10.1007/s00453-023-01130-2.

13 P. C. Gilmore and Ralph E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965.

14 David Hillel Goodman and Ilan Garibi. *Soma Puzzle Book, The: A New Approach To The Classic Pieces*. World Scientific, 2019.

15 Rolf Harren. Approximation algorithms for orthogonal packing problems for hypercubes. *Theoretical Computer Science*, 410(44):4504–4532, 2009. doi:10.1016/J.TCS.2009.07.030.

16 Klaus Jansen, Debajyoti Kar, Arindam Khan, KVN Sreenivas, and Malte Tutas. Improved approximation algorithms for three-dimensional knapsack. *arXiv preprint arXiv:2503.19365*, 2025.

17 Klaus Jansen, Arindam Khan, Marvin Lira, and K. V. N. Sreenivas. A ptas for packing hypercubes into a knapsack. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 229, pages 78:1–78:20, 2022. doi:10.4230/LIPIcs.ICALP.2022.78.

18 Klaus Jansen and Lorant Porkolab. Improved approximation schemes for scheduling unrelated parallel machines. In *ACM Symposium on Theory of Computing (STOC)*, pages 408–417, 1999. doi:10.1145/301250.301361.

19 Klaus Jansen and Lars Prädel. A new asymptotic approximation algorithm for 3-dimensional strip packing. In *SOFSEM: Theory and Practice of Computer Science*, pages 327–338, 2014. doi:10.1007/978-3-319-04298-5_29.

20 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007. doi:10.1007/S00453-006-0194-5.

21 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *International Symposium on Computational Geometry (SoCG)*, volume 189, pages 48:1–48:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.48.

22 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *arXiv preprint*, 2021. arXiv:2102.05854.

23 Jeffrey C. Lagarias. *The Kepler Conjecture: The Hales-Ferguson Proof*. Springer Science & Business Media, 2011.

24 Yiping Lu, Danny Z. Chen, and Jianzhong Cha. Packing cubes into a cube is NP-complete in the strong sense. *Journal of Combinatorial Optimization*, 29(1):197–215, 2015. doi:10.1007/s10878-013-9701-1.

25 Everton Fernandes Silva, Túlio Angelo Machado Toffolo, and Tony Wauters. Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. *Computers & Operations Research*, 109:12–27, 2019. doi:10.1016/j.cor.2019.04.020.

26 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

k-Dimensional Transversals for Fat Convex Sets

Attila Jung
ELTE Eötvös Loránd University, Budapest, Hungary
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Dömötör Pálvölgyi
ELTE Eötvös Loránd University, Budapest, Hungary
HUN-REN Alfréd Rényi Institute of Mathematics, Budapest, Hungary

Abstract

We prove a fractional Helly theorem for k-flats intersecting fat convex sets. A family $\mathcal{F}$ of sets is said to be ρ-fat if every set in the family contains a ball and is contained in a ball such that the ratio of the radii of these balls is bounded by ρ. We prove that for every dimension d and positive reals ρ and α there exists a positive $\beta = \beta(d, \rho, \alpha)$ such that if $\mathcal{F}$ is a finite family of ρ-fat convex sets in $\mathbb{R}^d$ and an α-fraction of the $(k+2)$-size subfamilies from $\mathcal{F}$ can be hit by a k-flat, then there is a k-flat that intersects at least a β-fraction of the sets of $\mathcal{F}$. We prove spherical and colorful variants of the above results and prove a $(p, k+2)$-theorem for k-flats intersecting balls.

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

Keywords and phrases discrete geometry, transversals, Helly, hypergraphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.61

Related Version *Previous Version*: https://arxiv.org/abs/2311.15646

Funding *Attila Jung*: Supported by the ERC Advanced Grant "ERMiD", by the EXCELLENCE-24 project no. 151504 Combinatorics and Geometry of the NRDI Fund, and by the NKFIH grants TKP2021-NKTA-62, FK132060 and SNN135643.
Dömötör Pálvölgyi: Supported by the ERC Advanced Grant "ERMiD", by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the grants TKP2021-NKTA-62, ÚNKP-23-5 and EXCELLENCE-24 project no. 151504 Combinatorics and Geometry of the NRDI Fund.

Acknowledgements We would like to thank Andreas Holmsen for useful discussions, and for bringing [16] to our attention.

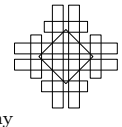
1 Introduction

We say that a family $\mathcal{T}$ of geometric objects *hits* or *pierces* another family $\mathcal{F}$ if for every $F \in \mathcal{F}$ there is a $T \in \mathcal{T}$ such that $T \cap F \neq \emptyset$. If $\mathcal{T}$ consists of a single k-flat (k-dimensional affine subspace), then we say that $\mathcal{F}$ has a k-transversal. In this language, Helly's theorem [10] states, that for a finite family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$, if every subfamily of size $d+1$ has a 0-transversal, i.e., a point contained in all $d+1$ sets, then the whole family has a 0-transversal, i.e., a point contained in all the sets. Vicensini [21] asked whether one can generalize Helly's theorem to k-transversals for $0 < k < d$, but this was shown to be false by Santaló [19], who constructed arbitrarily large families of convex sets in the plane without a 1-transversal whose every proper subfamily has a 1-transversal.

A natural next question in this direction is whether the following fractional Helly theorem of Katchalski and Liu has any analog for k-transversals.

▶ **Theorem 1** (Katchalski and Liu [14]). *For every dimension d there exists a function $\beta\colon (0,1] \to (0,1]$ with the following property. Let $\mathcal{F}$ be a finite family of convex sets in $\mathbb{R}^d$ and $\alpha \in (0,1]$. If at least $\alpha\binom{|\mathcal{F}|}{d+1}$ of the $(d+1)$-size subfamilies of $\mathcal{F}$ have a 0-transversal, then there exists a subfamily of $\mathcal{F}$ of size at least $\beta(\alpha)|\mathcal{F}|$ which has a 0-transversal.*

© Attila Jung and Dömötör Pálvölgyi;
licensed under Creative Commons License CC-BY 4.0



Alon and Kalai showed that it can indeed be generalized to $(d-1)$-transversals (hyperplanes) intersecting convex sets.

▶ **Theorem 2** (Alon and Kalai [1]). *For every dimension d there exists a function $\beta\colon (0,1] \to (0,1]$ with the following property. Let $\mathcal{F}$ be a finite family of convex sets from $\mathbb{R}^d$ with $|\mathcal{F}| = n$. If for some positive α at least $\alpha\binom{n}{d+1}$ of the $(d+1)$-size subfamilies of $\mathcal{F}$ have a $(d-1)$-transversal, then there exists a subfamily of $\mathcal{F}$ of size at least $\beta(\alpha)n$ which has a $(d-1)$-transversal.*

Later, Alon, Kalai, Matoušek and Meshulam showed that the above statement does not generalize to 1-flats (lines) intersecting convex sets in $\mathbb{R}^3$.

▶ **Theorem 3** (Alon, Kalai, Matoušek and Meshulam [2]). *For every integers n and m there exists a family of at least n convex sets in $\mathbb{R}^3$ such that any m of them have a 1-transversal, but no $m+4$ of them have a 1-transversal.*

By a suitable lifting argument this implies that there are no fractional Helly theorems for k-flats intersecting convex sets if $0 < k < d-1$. Following this example it is natural to look for fractional Helly-type theorems for special kinds of sets and k-flats with $0 < k < d-1$. For example, Matoušek proved, that k-flats intersecting a family of sets with bounded description complexity admit a fractional Helly theorem [16]. A family of sets is said to have bounded description complexity, if each of the sets can be described using a bounded number of bounded degree polynomial inequalities.

▶ **Theorem 4** (Matoušek [16]). *For every dimension d there exists a function $\beta\colon (0,1] \to (0,1)$ with the following property. Let $\mathcal{F}$ be a finite family of sets of bounded description complexity in $\mathbb{R}^d$ and $\alpha \in (0,1]$. If at least $\alpha\binom{|\mathcal{F}|}{(k+1)(d-k)+1}$ of the $((k+1)(d-k)+1)$-size subfamilies of $\mathcal{F}$ have a k-transversal, then there exists a subfamily of $\mathcal{F}$ of size at least $\beta(\alpha)|\mathcal{F}|$ which has a k-transversal.*

We prove a fractional Helly theorem for ρ-fat convex sets. A family $\mathcal{F}$ of sets is ρ-fat, if for every $K \in \mathcal{F}$ there exist $x_K \in \mathbb{R}^d$ and $r_K, R_K \in \mathbb{R}$ such that $R_K \leq \rho r_K$ and K contains a ball of radius r_K centered at x_k and is contained in a ball of radius R_K centered at x_k.[1]

▶ **Theorem 5.** *For every d and ρ there exists a function $\beta\colon (0,1] \to (0,1)$ with the following property. Let $\mathcal{F}$ be a finite family of ρ-fat convex sets in $\mathbb{R}^d$, $0 \leq k \leq d$ an integer, and $\alpha \in (0,1]$. If at least $\alpha\binom{|\mathcal{F}|}{k+2}$ of the $(k+2)$-tuples of $\mathcal{F}$ have a k-transversal, then $\mathcal{F}$ has a subfamily of size at least $\beta(\alpha)|\mathcal{F}|$ which has a k-transversal.*

As discussed above, the $k = d-1$ case is known to hold even without the fatness assumption [1], but the $0 < k < d-1$ case is known to be false without fatness, even if we replace the $(k+2)$-tuples in the assumption with m-tuples for any finite m [2].

Note that as any convex body in $\mathbb{R}^d$ can be made d-fat with an affine transformation by John's theorem [12], our result also holds for the family of homothets of any fixed convex body. For these, Matoušek's result would only work if the convex body has bounded union complexity, like a ball or a polyhedron, and even in this case would only give $((k+1)(d-k)+1)$-size subfamilies, which is improved by our $(k+2)$-size subfamilies for all $k < d-1$. (This is an improvement because by averaging if some α fraction of the $((k+1)(d-k)+1)$-tuples intersect, then also some α fraction of the $(k+2)$-tuples intersect.)

[1] Note that requiring these balls to be concentric is not a significant restriction; if K is contained in some ball of radius R, then it is contained in a ball of radius $2R$ centered at any point of K.

The proof of our main Theorem 5 can be found in Section 2. Along the way we prove spherical and colorful variants of the result as well.

Now we turn to the history of our other main result. A celebrated generalization of Helly's theorem is the following (p, q)-theorem of Alon and Kleitman.

▶ **Theorem 6** (Alon and Kleitman [3]). *For every $p \geq d+1$ there exists $C(p, d)$ such that the following holds. If $\mathcal{F}$ is a family of compact convex sets in $\mathbb{R}^d$ such that among any p members of $\mathcal{F}$ there are $d+1$ with a 0-transversal, then there exist at most C points hitting all of $\mathcal{F}$.*

The fractional Helly theorem of Katchalski and Liu plays a crucial role in the proof of Theorem 6. It would be very interesting to know whether our fractional Helly theorem for k-flats intersecting ρ-fat convex sets has such a corollary.

▶ **Conjecture 7.** *For every p, d, k and ρ there exists a C such that the following holds. If $\mathcal{F}$ is a family of ρ-fat compact convex sets in $\mathbb{R}^d$ such that among any p members of $\mathcal{F}$ there are $k+2$ with a k-transversal, then there exist at most C k-flats hitting all of $\mathcal{F}$.*

We do not even know whether the statement holds if we replace $k+2$ by any $m > k+2$ which does not depend on $\mathcal{F}$. The missing ingredient is a weak ε-net theorem for k-flats intersecting ρ-fat convex bodies. In Section 4, we discuss how two of the main approaches used to prove the existence of (weak) ε-nets fails in our case. As one of the standard approaches can be used if we replace ρ-fat convex sets with balls (1-fat convex sets), we obtain the following corollary.

▶ **Theorem 8.** *For every three positive integers d, $k < d$ and $p \geq k+2$ there exists a $C = C(d, k, p)$ such that the following holds. If $\mathcal{F}$ is a (possibly infinite) family of closed balls in $\mathbb{R}^d$ such that among any p members of $\mathcal{F}$ there are $k+2$ that can be hit by a single k-flat, then there exist at most C k-flats hitting all of $\mathcal{F}$.*

Theorem 8 follows from results in Section 4. We show spherical and colorful variants of this result as well. Note that a weaker version of Theorem 8, where we replace $k+2$ by $(k+1)(d-k)+1$ follows from the more general results of Matoušek [16].

Keller and Perles showed that if $\mathcal{F}$ is an infinite family of d-dimensional balls, and among any $\aleph_0$ balls some $k+2$ balls have a k-transversal, then the whole family can be pierced with finitely many k-dimensional affine subspaces [15]. In a forthcoming paper [13] it is shown that this infinite variant follows from our Theorems 5 and 8 with a purely combinatorial proof.

Our Theorem 8 generalizes to certain uniformly fat families as follows. A family $\mathcal{F}$ of sets in $\mathbb{R}^d$ is called (r, R)-fat if for every $K \in \mathcal{F}$ there is an $x_K \in \mathbb{R}^d$ such that $B(x_K, r) \subseteq K \subseteq B(x_K, R)$. Note that an (r, R)-fat family is ρ-fat with $\rho = \frac{R}{r}$ but a ρ-fat family might not be (r, R)-fat for any fixed r, R as it can contain arbitrarily small and large sets. Because of this, the argument described in the next paragraph, does not apply directly to ρ-fat sets in $\mathbb{R}^d$.

Fractional Helly and (p, q)-type results about k-transversals of (congruent) balls generalize easily to (not necessarily convex) (r, R)-fat sets as were described in [6, 15].

▶ **Corollary 9.** *For every three positive integers d, $k < d$ and $p \geq k+2$, and positive reals $r \leq R$ there exists a $C = C(d, k, p, R/r)$ such that the following holds. If $\mathcal{F}$ is a (possibly infinite) family of (r, R)-fat sets in $\mathbb{R}^d$ such that among any p members of $\mathcal{F}$ there are $k+2$ that can be hit by a single k-flat, then there exist at most C k-flats hitting all of $\mathcal{F}$.*

We give a short summary of the argument here. If among every p members of an (r, R)-fat family $\mathcal{F}$, there are $k+2$ which have a k-transversal, then so do the $(k+2)$-tuples of balls of radius R containing them. We can apply Theorem 8 to the family of R-balls to obtain a bounded size family $\mathcal{T}$ of k-flats hitting all the balls. By considering parallel k-flats close to members of $\mathcal{T}$, we can conclude that a bounded number of them stab the r-balls contained in the members of $\mathcal{F}$. Therefore, there is a bounded size family of k-flats hitting every member of $\mathcal{F}$. This argument works even in the case when the members of $\mathcal{F}$ are not convex. Not even connectedness is required, only (r, R)-fatness.

Holmsen and Matoušek asked whether a fractional Helly theorem exists for disjoint translates of a convex set in $\mathbb{R}^3$ with respect to lines [11]. As any convex body in $\mathbb{R}^3$ can be made 3-fat with an affine transformation, our Theorem 5 answers their question. But we cannot really take credit for this, as a positive answer already follows from a result of Matoušek [16] combined with the trick of making the set $(1, 3)$-fat with an affine transformation, and then using the reduction to balls that we sketched earlier. This method would only give a fractional Helly theorem where the assumption is about lines intersecting 5-tuples of sets (without using our Theorem 5); it was observed by Dobbins and Holmsen a few years ago (personal communication), independently of us, that this can be improved to the optimal 3, and they have also obtained fractional Helly theorems for (r, R)-fat sets with respect to k-flats (unpublished).

As another direction generalizing Helly-type theorems to k-transversals, we mention a result of Hadwiger which shows that although Helly's theorem does not generalize directly to 1-transversals, a finite family of pairwise disjoint convex sets in the plane has a 1-transversal if and only if the family has a linear ordering such that any 3-tuple of convex sets has a 1-transversal which is consistent with the ordering [8] (see the definition there). Hadwiger's result has been generalized to $(d-1)$-transversals by Goodman, Pollack and Wenger [7], and very recently to k-transversals for any $0 \leq k < d$ by McGinnis and Sadovek [17].

2 Fractional Helly Theorems

In this section we prove Theorem 5 in the below more general, colorful form, an analog of a colorful version of the fractional Helly theorem due to Bárány, Fodor, Montejano, Oliveros, and Pór [5]. Note that our Theorem 5 is a special case of Theorem 10 with $\mathcal{F}_i = \mathcal{F}$ for all i.

▶ **Theorem 10.** *For every $\rho \geq 1$ and every dimension d there exists a function $\beta : (0, 1] \to (0, 1)$ with the following property. Let $\mathcal{F}_1, \ldots, \mathcal{F}_{k+2}$ be families of ρ-fat convex sets in $\mathbb{R}^d$. If at least $\alpha|\mathcal{F}_1| \cdots |\mathcal{F}_{k+2}|$ of the colorful selections have a k-transversal, then there exists an i with $\mathcal{F}_i$ having a subfamily of size at least $\beta(\alpha)|\mathcal{F}_i|$ which has a k-transversal.*

Our inductive proof of Theorem 10 also requires a spherical analog, for which we need the following definitions. A cap of the sphere $\mathbb{S}^d$ is the intersection of $\mathbb{S}^d$ with a ball in $\mathbb{R}^{d+1}$ (we can assume that the center of the ball is on $\mathbb{S}^d$), a great k-sphere is the intersection of $\mathbb{S}^d$ with a $(k+1)$-dimensional linear subspace of $\mathbb{R}^{d+1}$ (assuming that the origin is the center of $\mathbb{S}^d$), and a spherical k-transversal for a spherical family is a great k-sphere intersecting all members of the family. We can describe caps as subsets of $\mathbb{S}^d$ of the form $B(x, \varepsilon) = \{y \in \mathbb{S}^d : \angle(x, y) = \cos^{-1}(xy) \leq \varepsilon\}$ where $x \in \mathbb{S}^d$. Beware that the definition of $B(x, r)$ implicitly depends on the space we are working in! We intentionally do not use a different notation so that our argument can be described more generally, but it will be always clear from the context in which space we are in. A ρ-fat family of spherical sets can be defined analogously to the Euclidean case, with the existence of caps $B(x_K, r_K)$ and $B(x_K, R_K)$ for

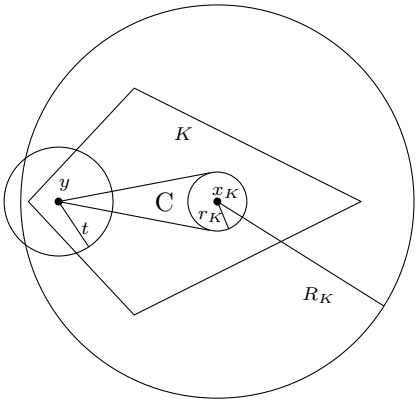


Figure 1 A property of fat convex sets.

every $K \in \mathcal{F}$ with the property that $B(x_K, r_K) \subset K \subset B(x_K, R_K)$ and $R_K \leq \rho r_K$. A set $K \subset \mathbb{S}^d$ is (spherically) convex, if either $K = \mathbb{S}^d$, or for no $v \in \mathbb{S}^d$ we have both $v, -v \in K$, and K contains the shorter great circle arc connecting any two points from K.

▶ **Theorem 11.** *For every $\rho \geq 1$ and every dimension d there exists a function $\beta : (0, 1] \to (0, 1)$ with the following property. Let $\mathcal{F}_1, \ldots, \mathcal{F}_{k+2}$ be families of ρ-fat convex sets in $\mathbb{S}^d$. If at least $\alpha|\mathcal{F}_1| \cdots |\mathcal{F}_{k+2}|$ of the colorful selections have a spherical k-transversal, then there exists an i with $\mathcal{F}_i$ having a subfamily of size at least $\beta(\alpha)|\mathcal{F}_i|$ which has a spherical k-transversal.*

Notice that in fact Theorem 11 implies Theorem 10 as any counterexample in $\mathbb{R}^d$ would also give a counterexample on the surface of a large enough sphere $\mathbb{S}^d$ with small (compared to the radius of the sphere) ρ-fat sets on it. However, we prove both Theorems 10 and 11 as we believe that our argument is easier to understand in the more natural setting of $\mathbb{R}^d$, and then think it through that it also works in $\mathbb{S}^d$ without much change. We mark the differences in our simultaneous proof for both theorems with brackets [ ].

The following is a key property of ρ-fat convex sets.

▷ Claim 12. For all d and ρ we have a $\gamma > 0$ such that the following holds. If K is a ρ-fat convex set in $\mathbb{R}^d$ [or in $\mathbb{S}^d$], $y \in K$ and $t \leq R_K$, then $\mathrm{vol}(K \cap B(y, t)) \geq \gamma \,\mathrm{vol}(B(y, t))$, where $B(y, t)$ denotes the ball [or spherical cap] of radius t centered at y. (See Figure 1.)

Proof. Let $B(x_K, r_K) \subset K \subset B(x_K, R_K)$ with $R_K \leq \rho r_K$, and define $r'_K = \min(r_K, t) \geq t/\rho$ and $C = \mathrm{conv}(\{y\} \cup B(x_K, r_K)) \subset K$. C contains all the points u of $B(y, r'_K)$ such that $\angle(u - y, x_K - y)$ is at most some number depending only on ρ, so C contains a constant fraction of $B(y, t)$. ◁

Proof of Theorems 10 and 11. The proof is by induction on k and d. Let $n_i = |\mathcal{F}_i|$. Let $\mathcal{H}$ be the $(k + 2)$-uniform hypergraph with vertex set $\mathcal{F} = \cup \mathcal{F}_i$, where the edges are those colorful $(k + 2)$-tuples which have a [spherical] k-transversal. Charge every hyperedge to its member K with the smallest bounding radius R_K. In case of ties, charge arbitrarily to one of the members K with smallest R_K.

Since in total there are $\alpha n_1 \cdots n_{k+2}$ many charges, there is a color class $\mathcal{F}_i$ whose members are charged $\frac{\alpha}{k+2} n_1 \cdots n_{k+2}$ times. Without loss of generality, we may assume that this color class is $\mathcal{F}_{k+2}$. Pick the set K_0 from $\mathcal{F}_{k+2}$ with the most charge. By averaging, K_0 was charged at least $\frac{\alpha}{k+2} n_1 \cdots n_{k+1}$ times.

If $k = 0$, then we have $\frac{\alpha}{2}n_1$ sets from $\mathcal{F}_1$ intersecting $K_0 \in \mathcal{F}_2$. We have a point $y_K \in B(x_{K_0}, R_{K_0}) \cap K$ for all such $K \in \mathcal{F}_1$. Since $R_{K_0} \leq R_K$, the set K contains a positive fraction of the ball $B(y_K, R_{K_0})$ by Claim 12. Moreover, $B(y_K, R_{K_0})$ occupies a positive fraction of $B(x_{K_0}, 2R_{K_0})$, so it follows that every K contains a positive fraction of $B(x_{K_0}, 2R_{K_0})$. In this case, a constant fraction of the $\frac{\alpha}{2}n_1$ sets can be hit by a single point by the volumetric pigeonhole principle (where the constant depends only on the dimension and ρ).

If $k > 0$, we can reduce finding a [spherical] k-transversal in $\mathbb{R}^d$ [in $\mathbb{S}^d$] to finding a spherical $(k-1)$-transversal in $\mathbb{S}^{d-1}$ as follows.

Euclidean case. If we are in the Euclidean space $\mathbb{R}^d$, we may assume that $B_0 = B(x_{K_0}, R_{K_0})$ is the unit ball centered at the origin. Let $K' = K + B_0$ (where $+$ denotes the Minkowski sum) if $K \in \cup\mathcal{F}_i \setminus \{K_0\}$, and let K'_0 be the degenerate ball containing only the origin. Denote the new families by $\mathcal{F}' = \cup\mathcal{F}'_i$.

▷ Claim 13. At least $\frac{\alpha}{k+2}n_1 \cdots n_{k+1}$ of the colorful selections of $\mathcal{F}'_1, \ldots, \mathcal{F}'_{k+1}$ have a k-dimensional *linear* subspace hitting them.

Proof. The set K_0 was charged $\frac{\alpha}{k+2}n_1 \cdots n_{k+1}$ times. Take $K_1 \in \mathcal{F}_1, \ldots, K_{k+1} \in \mathcal{F}_{k+1}$ such that $\{K_0, K_1, \ldots, K_{k+1}\}$ was charged to K_0 and translate their hitting k-flat to the origin. As the translation is by at most R_{K_0}, if any K_i was hit by the original k-flat, then K'_i is hit by the translated (linear) k-flat. ◁

Now, centrally project every set in $\mathcal{F}' \setminus \{K'_0\}$ to the surface of the unit sphere from the origin. (If a set contains the origin, its projection is the entire sphere.)

▷ Claim 14. A projection of a ρ-fat convex set is $\frac{\pi}{2}\rho$-fat.

Proof. The projection of a ball $B(x, r)$ to the unit sphere centered at the origin is a cap of the unit sphere with radius $\sin^{-1}(r/t)$ if x is at distance $t \geq r$ of the origin. Thus, the projection of a ρ-convex set K is $\frac{\sin^{-1}(R_K/t)}{\sin^{-1}(r_K/t)}$-fat if x_K is at distance $t > R_K$ from the origin. But $t \leq \sin^{-1}(t) \leq \frac{\pi}{2}t$, thus $\frac{\sin^{-1}(R_K/t)}{\sin^{-1}(r_K/t)} \leq \frac{\pi}{2}\frac{R_k}{r_k} = \frac{\pi}{2}\rho$ and the image of K is $\frac{\pi}{2}\rho$-fat.

If $t \leq R_K$, then the projection of $B(x_K, r_K)$ is a cap with radius $\sin^{-1}(r_K/t) \geq \frac{r_K}{t} \geq \frac{1}{\rho}$. Either the projection of K is contained in a cap of radius $\pi/2$ and contains a cap of radius $\frac{1}{\rho}$, or the image of K is the complete unit sphere. In both cases the projection is $\frac{\pi}{2}\rho$-fat. ◁

Denote the obtained family of $\rho' = \frac{\pi}{2}\rho$-fat spherically convex sets by $\mathcal{F}^*$. As $\frac{\alpha}{k+2}n_1 \cdots n_{k+1}$ of the colorful selections of $\mathcal{F}'_1, \ldots, \mathcal{F}'_{k+1}$ have a k-dimensional *linear* subspace hitting them by Claim 13, we have that $\frac{\alpha}{k+2}n_1 \cdots n_{k+1}$ of the colorful selections of $\mathcal{F}^*_1, \ldots, \mathcal{F}^*_{k+1}$ have a great $(k-1)$-sphere intersecting them. The assumptions of the spherical colorful fractional Helly Theorem 11 on $\mathbb{S}^{d-1}$ with great $(k-1)$-spheres are satisfied, thus we have an $i \in [k+1]$ and a subfamily $\mathcal{G}^* \subset \mathcal{F}^*_i$ of size at least $\beta^*|\mathcal{F}^*_i|$ which can be hit by a great $(k-1)$-sphere, where β^* is the value we obtain from Theorem 11 with parameters $d-1, k-1$ and $\frac{\alpha}{k+2}$. This means that the corresponding family of preimages $\mathcal{G}' = \{K' \in \mathcal{F}' : \mathrm{proj}_{\mathbb{S}^{d-1}} K' \in \mathcal{G}^*\}$ can be hit by a k-dimensional linear subspace F. Let $\mathcal{G} = \{K \in \mathcal{F} : K' \in \mathcal{G}'\}$. If we project K_0 and every set $K \in \mathcal{G}$ to the orthogonal complement $F^\perp$ of F, every projected set $\mathrm{proj}_{F^\perp} K$ will intersect the set $\mathrm{proj}_{F^\perp} K_0$.

Since for all K there exists a $y_K \in B(0,1) \cap \mathrm{proj}_{F^\perp} K$, and $\mathrm{proj}_{F^\perp} K$ contains a ball of $F^\perp$ of radius ρ, we know that K occupies a constant fraction of $B(y_K, 1) \cap F^\perp \subset B(0,2) \cap F^\perp$ by Claim 12 and thus a positive fraction of $B(0,2) \cap F^\perp$. By the pigeonhole principle

we can find a point $x \in F^\perp$ in the intersection of a constant fraction of the projections $\{\text{proj}_{F^\perp} K : K \in \mathcal{G}'\}$. $F + x$ (the translation of F by x) is a k-flat intersecting a constant fraction of the fat convex sets in $\mathcal{G}'$.

Spherical case. For this case, we need some claims about the geometry of spherical caps and great k-spheres which are described in Section 3, but otherwise the main argument is the same as in the Euclidean case. First, we deal with the case where there are many large sets in $\cup_i \mathcal{F}_i$. Note that we may assume that all the n_is are large enough, otherwise a β-fraction of a color class will consists of a single set K (if β is chosen to be small enough). Let $c < \alpha/(k+2)$ be a small constant. If there is an $\mathcal{F}_i$ such that for at least cn_i of the sets $K \in \mathcal{F}_i$ we have $R_K > \pi/8$, then a constant fraction of them can be hit with a single point by the volumetric pigeonhole principle, as all of them contain caps with radius at least $\rho\pi/8$. This constant fraction depends only on d and ρ. If we do not have this many large sets in any of the $\mathcal{F}_i$, we can simply delete the large ones, as this way we loose at most $c(k+2)n_1 \dots n_{k+2}$ out of the original $\alpha n_1 \dots n_{k+2}$ colorful $(k+2)$-tuples with a spherical k-transversal. Thus, we can have families of size at least $(1-c)n_i$ with at least an $(\alpha - c(k+2))n_1 \dots n_{k+2}$ colorful $(k+2)$-tuples having a spherical k-transversal and no set K in the families with $R_K > \pi/8$.

Let K_0 be the most charged set as defined above, let $\varepsilon = R_{K_0}$ and $B_0 = B(x_{K_0}, \varepsilon)$. For a set $K \subset \mathbb{S}^d$, its ε-neighborhood is $K^\varepsilon = \{u \in \mathbb{S}^d : \exists v \in K \text{ with } \angle(u,v) \le \varepsilon\}$, and $\text{conv}(K)$ is the intersection of all the spherically convex sets containing K. If K is a ρ-fat convex set, then $\text{conv}(K^\varepsilon)$ is a ρ-fat convex set by Claim 15. For every $K \in \cup\mathcal{F}_i \setminus \{K_0\}$, let $K' = \text{conv}(K^\varepsilon)$ and let $K_0' = \{x_{K_0}\}$. Denote the new families of spherically convex ρ-fat sets by $\mathcal{F}' = \cup\mathcal{F}_i'$. By Claim 16, if some sets in $\mathcal{F}$ can be hit by a great k-sphere, then the corresponding sets in $\mathcal{F}'$ can be hit by a great k-sphere as well. Now let $\mathcal{F}^* = \text{proj}_{B_0} \mathcal{F}'$, where proj_{B_0} denotes the projection into the boundary ∂B_0. By Claim 19, $\mathcal{F}^*$ is a family of spherically convex $\rho' = \frac{\pi^3}{4}\rho^2$-fat sets of the $(d-1)$-dimensional sphere ∂B_0. Without loss of generality, assume that $K_0 \in \mathcal{F}_{k+2}$. As K_0 is the most charged set, at least $\frac{\alpha}{k+2}$-fraction of the colorful selections of $\mathcal{F}_1', \dots, \mathcal{F}_{k+1}'$ have a spherical k-transversal containing x_0. These great k-spheres of $\mathbb{S}^d$ become great $(k-1)$-spheres of ∂B_0 after the projection. Thus, at least $\frac{\alpha}{k+2}$-fraction of the colorful selections of $\mathcal{F}_1^*, \dots, \mathcal{F}_{k+1}^*$ have a $(k-1)$-transversal. The assumptions of the spherical colorful fractional Helly Theorem 11 on ∂B_0 with spherical $(k-1)$-transversals are satisfied, thus we have an i and a subfamily $\mathcal{G}^* \subset \mathcal{F}_i^*$ of size at least $\beta^*|\mathcal{F}_i^*|$ which has a spherical $(k-1)$-transversal. This means that the corresponding family of preimage sets $\mathcal{G}'$ has a spherical k-transversal. By Claim 20, a large subfamily of $\mathcal{G}$ has a spherical k-transversal as well. ◀

3 Caps and projections on the sphere

In this section, we prove the claims about spherical sets that were used in the proof of Theorem 11. All the arguments are fairly straight-forward modifications of the simple proofs used in the Euclidean case, but more tedious calculations are required.

Recall that $B(v,\varepsilon)$ is the spherical cap centered at v with angle ε, and a great k-sphere of $\mathbb{S}^d$ is the intersection of $\mathbb{S}^d$ with a $(k+1)$-dimensional linear subspace of $\mathbb{R}^{d+1}$ [4]. A great 1-sphere is called a great circle.

For $u \in \mathbb{S}^d \setminus \{v, -v\}$, let $\text{rot}(u,v)$ be the rotation r with $r(u) = v$ which is constant on $\text{span}(\{u,v\})^\perp$, so we rotate with the uov angle $\angle(u,v)$ around the center o, keeping $\text{span}(\{u,v\})^\perp$ fixed. If $u \notin \{v,-v\}$, the projection of $u \in \mathbb{S}^d$ onto the boundary $\partial B(v,\varepsilon)$ of the cap $B(v,\varepsilon)$ is the intersection of $\partial B(v,\varepsilon)$ with the halfplane embedded in $\mathbb{R}^{d+1}$ containing

o and v on its boundary, and u in its interior. In other words, if $u \notin B(v,\varepsilon)$, then the projection is the point of $\partial B(v,\varepsilon)$ that is hit when we rotate u to v. It is denoted by $\mathrm{proj}_B(u)$, or simply $\mathrm{proj}(u)$ if B is clear from the context. For any set X and cap $B = B(v,\varepsilon)$, let $\mathrm{proj}_B X = \{\mathrm{proj}_B(u) : u \in X\}$ if $X \cap \{v,-v\} = \emptyset$, and let $\mathrm{proj}_B X = B$ if $X \cap \{v,-v\} \neq \emptyset$.

▷ Claim 15. If $K \subseteq \mathbb{S}^d$ is ρ-fat, then $\mathrm{conv}(K^\varepsilon)$ is ρ-fat as well.

Proof. If $\mathrm{conv}(K^\varepsilon) = \mathbb{S}^d$, then it is ρ-fat with $\rho = 1$. Otherwise, as $B(x_K, r_K+\varepsilon) \subseteq \mathrm{conv}(K^\varepsilon) \subseteq B(x_K, R_K+\varepsilon)$, the ρ-fatness of K^ε follows from $\frac{R_K+\varepsilon}{r_K+\varepsilon} \leq \frac{R_K}{r_K} \leq \rho$. ◁

▷ Claim 16. If the spherical sets $K_1,\dots,K_n \subset \mathbb{S}^d$ can be pierced with a great k-sphere, and $K_1 \subset B(v,\varepsilon)$, then $B(v,0), K_2^\varepsilon, \dots, K_n^\varepsilon$ can also be pierced with a great k-sphere.

Proof. Take a great k-sphere F intersecting all of $K_1,\dots,K_n \subset \mathbb{S}^d$, let $v' = \mathrm{proj}_F(v)$ and $\mathrm{rot} = \mathrm{rot}(v',v)$. For any point $u \in \mathbb{S}^d$, the spherical distance between u and $\mathrm{rot}(u)$ is at most ε. Hence, if $u \in K_i \cap F$, then $\mathrm{rot}(u) \in K_i^\varepsilon \cap \mathrm{rot}(F)$, thus the great k-sphere $\mathrm{rot}(F)$ intersects all of $B(v,0), K_2^\varepsilon, \dots, K_n^\varepsilon$. ◁

We want to show that during a gnomonic projection the distance of two points close to the center of the projected image cannot be distorted too much. This is probably a well-known statement, but we could not find it anywhere, so we include the simple calculation below. If we embed $\mathbb{S}^d$ into $\mathbb{R}^{d+1}$ as the unit sphere, the spherical distance of x and y becomes $\angle(x,y)$, while their Euclidean distance is $|x-y|$.

▷ Claim 17. Embed $\mathbb{S}^d$ into $\mathbb{R}^{d+1}$ as the unit sphere. If $u,v \in B(w,\pi/4) \subset \mathbb{S}^d$ and p is the central projection from the origin to the supporting hyperplane of $\mathbb{S}^d$ at w, then we have $\angle(u,v) \leq |p(u)-p(v)| \leq 2\angle(u,v)$.

Proof. As u,v,w are contained in a 2-dimensional subsphere, it is enough to show the claim for $d=2$. First we calculate the distortion of length if u and w have the same latitude of longitude.

Case 1: u,v,w are on the same great circle. As we have $|p(u)-w| = \tan(\angle(u,w))$ and $1 \leq \tan'(\varphi) \leq 2$ if $\varphi \in (0,\pi/4)$, we have $\angle(u,v) \leq |u-v| \leq 2\angle(u,v)$.

Case 2: $\angle(u,w) = \angle(v,w) = \varphi$. In this case p acts on them as a homothety with ratio $1/\cos(\varphi)$, which is between 1 and $\sqrt{2}$ if $\varphi \in (0,\pi/4)$.

We can argue that these are the extreme cases and the distortion of length is always between 1 and 2. We show that for close enough points the distortion of distance is between 1 and 2. But then the global distortion has to be in the same interval.

For every $u \in B(w,\pi/4)$ the points of $\mathbb{S}^d$ with a small enough given distance from u form a circle inside $B(w,\pi/4)$. Its image under p is an ellipse. Due to symmetry the axes of the ellipse correspond to point pairs with the same latitude or longitude. We have seen in Cases 1 and 2 that the ratios of the lengths of the axes and the radius of the circle are between 1 and 2. But then for every point v in the circle we have $\angle(u,v) \leq |p(u)-p(v)| \leq 2\angle(u,v)$. ◁

▷ Claim 18. If $K \subseteq B(w,\pi/8) \subset \mathbb{S}^d$ is convex and $\varepsilon \leq \pi/8$, then $\mathrm{conv}(K^\varepsilon) \subset K^{2\varepsilon}$.

Proof. If $u \in \mathrm{conv}(K^\varepsilon)$, then there are $u',u'' \in K^\varepsilon$ such that $u \in \mathrm{conv}(\{u',u''\})$. Let $v',v'' \in K$ be such that $\angle(u',v'), \angle(u'',v'') \leq \varepsilon$. Let p be the central projection of $B(w,\pi/4)$ from the origin to the supporting hyperplane of $\mathbb{S}^d$ at w. As $u',u'',v',v'' \in B(w,\pi/4)$, we have $|p(u')-p(v')|, |p(u'')-p(v'')| \leq 2\varepsilon$ by Claim 18. As the projections of $\mathrm{conv}(\{u',u''\})$ and $\mathrm{conv}(\{v',v''\})$ are segments of a Euclidean space whose endpoints have distance at most 2ε, there is a point $v \in \mathrm{conv}(\{v',v''\})$ such that $|p(v)-p(u)| \leq 2\varepsilon$. We have $\angle(u,v) \leq 2\varepsilon$ by Claim 18. As $v \in K$, this proves $u \in K^{2\varepsilon}$. ◁

▷ Claim 19. Let $B = B(v, \varepsilon)$ be a cap, and K be a ρ-fat convex set of $\mathbb{S}^d$. The projection $\mathrm{proj}_B K$ is a $\frac{\pi^3}{4}\rho^2$-fat convex set of the $(d-1)$-dimensional sphere ∂B.

Proof. If $\mathrm{proj}_B K = B$, then it is convex by definition. Otherwise, as proj_B maps (shorter) great circle arcs to (shorter) great circle arcs or points, the projection is convex. Let $\rho' = \frac{\pi^3}{4}\rho^2$. To show ρ'-fatness, we need to examine how the radius of $B(x_K, r_K) \subset K$ and $B(x_K, R_K) \supset K$ change after the projection.

Assume that $B(x, r)$ is any cap of $\mathbb{S}^d$ and $v \notin B(x, r)$. Embed $\mathbb{S}^d$ in $\mathbb{R}^{d+1}$ as the unit sphere and look at the simplex spanned by $\underline{0}, x, v$ and b where $b \in \partial B(x, r)$. The radius φ of $\mathrm{proj}_{B(v,\varepsilon)} B(x, r)$ will be $\max_b \angle(b - v, x - v)$.

As $\frac{\sin\angle(b-v,x-v)}{|x-b|} = \frac{\sin\angle(x-b,v-b)}{|x-v|}$ by the law of sines, $\angle(b - v, x - v)$ is maximal if $\angle(x - b, v - b) = \pi/2$. Fix such a b. We have $\sin\varphi = \frac{|x-b|}{|x-v|}$.

Let $\angle(v, x) = t$. As $|x| = |v| = |b| = 1$, we have $|x-v| = 2-2\cos t$ and $|x-b| = 2-2\cos r$ by the law of cosines. As $1 - \cos t = \sin^2(t/2)$ and $\sin t \le t$, we have $\varphi \ge \sin\varphi = \frac{|x-b|}{|x-v|} = \frac{\sin^2(r/2)}{\sin^2(t/2)}$. If $\varphi \le \pi/2$, then we also have $\varphi \le \frac{\pi}{2}\sin\varphi = \frac{\pi}{2}\frac{\sin^2(r/2)}{\sin^2(t/2)}$.

Now returning to the proof, for simplicity we will write proj for proj_B. Let φ_r and φ_R be such that $\mathrm{proj}\, B(x_K, r_K) = B(\mathrm{proj}\, x_K, \varphi_r)$ and $\mathrm{proj}\, B(x_K, R_K) = B(\mathrm{proj}\, x_K, \varphi_R)$.

If $\varphi_R \ge \pi/2$, then $R \ge t$. In this case we have either $\varphi_r \ge \pi/2$ or $\varphi_r \ge \sin\varphi_r = \frac{\sin^2(r/2)}{\sin^2(t/2)} \ge \frac{4}{\pi^2}\frac{r^2}{t^2} \ge \frac{4}{\pi^2}\frac{r^2}{t^2} \ge \frac{4}{\pi^2\rho^2}$ and so $\varphi_R \le \pi \le \frac{\pi^3}{4}\rho^2\varphi_r$.

Otherwise $\varphi_R < \varphi/2$ and we have $\frac{\varphi_R}{\varphi_r} \le \frac{\pi}{2}\frac{\sin\varphi_R}{\sin\varphi_r} = \frac{\pi}{2}\frac{\sin^2(R/2)}{\sin^2(r/2)} \le \frac{\pi^3}{8}\frac{R^2}{r^2} \le \frac{\pi^3}{8}\rho^2$. As $\mathrm{proj}\, B(x_K, r_K) \subset \mathrm{proj}\, K \subset \mathrm{proj}\, B(x_K, R_K)$, in this case K is $\frac{\pi^3}{8}\rho^2$-fat. ◁

▷ Claim 20. For every d, ρ and k there exists a constant $c > 0$ such that the following holds for every $\varepsilon > 0$. Let $K_2, \dots, K_n$ be ρ-fat convex sets of $\mathbb{S}^d$ with $\varepsilon \le R_{K_i}$ for all $i \in \{2, \dots, n\}$. If $\mathrm{conv}(K_2^\varepsilon), \dots, \mathrm{conv}(K_n^\varepsilon)$ can be hit with a great k-sphere, then cn sets from $K_2, \dots, K_n$ can be hit with a great k-sphere as well.

Proof. If $R_{K_i} \ge \pi/8$ for at least half of the sets, then, as those sets all contain caps of radius at least $\pi/(8\rho)$, a constant fraction of them can be hit even with a single point by the volumetric pigeonhole principle. Otherwise delete all the sets K_i with $R_{K_i} \ge \pi/8$ and at least half of the sets remain. Let the family of the remaining at least $n/2$ sets be $\mathcal{H}$.

Let F be the great k-sphere hitting all the sets of $\{\mathrm{conv}(K^\varepsilon) : K \in \mathcal{H}\}$. In this case, all $K \in \mathcal{H}$ are at distance at most 2ε from F by Claim 18.

If $k = 0$, let $F = \{\pm v\}$. For all $K \in \mathcal{H}$ there exists a $y_K \in K$ which is at distance at most 2ε from one of the two antipodal points of F. As K occupies a constant fraction of $B(y_K, \varepsilon)$ by Claim 12 and $B(y_K, \varepsilon)$ contains a constant fraction of $B(v, 2\varepsilon) \cup B(-v, 2\varepsilon)$, the volumetric pigeonhole principle yields a point intersecting cn of the sets of $\mathcal{H}$.

We can reduce the $k > 0$ case to the $k = 0$ case with a suitable projection. For a $K \in \mathcal{H}$, let $f(K)$ be a closest point of F to K. Let $\delta = \delta(d, k) > 0$ be a small enough number, and let N be a metric δ-net of F (its size depends only on k and d). By the pigeonhole principle, there exists a point $u \in N$ and a subfamily $\mathcal{H}' \subset \mathcal{H}$ with $|\mathcal{H}'| \ge c'|\mathcal{H}|$ such that for all $K \in \mathcal{H}'$, the point $f(K)$ has distance at most δ from u.

Embed $\mathbb{S}^d$ into $\mathbb{R}^{d+1}$ as the unit sphere and let V be the $(k+1)$-dimensional linear subspace of $\mathbb{R}^{d+1}$ with $V \cap \mathbb{S}^d = F$. Let $u, u_1, \dots, u_k$ be an orthonormal basis of V. Let $B_i = B(u_i, \pi/2)$ be the halfsphere centered at u_i, let $\mathrm{proj}_i = \mathrm{proj}_{B_i}$ be the projection from $\mathbb{S}^d$ to ∂B_i as defined above, and let $\mathrm{proj}_{F,u} = \mathrm{proj}_1 \circ \mathrm{proj}_2 \circ \dots \circ \mathrm{proj}_k$. The repeated application of Claim 19 gives us that if $K \subset \mathbb{S}^d$ is a ρ-fat convex set, then $\mathrm{proj}_{F_u} K$ is a $\rho' = c^k\rho^{2k}$-fat convex set where $c \ge 1$ is a universal constant.

Apply $\mathrm{proj}_{F,u}$ to F and the sets of $\mathcal{H}'$ to get projections in the great $(d-k)$-sphere $\partial B_2 \cap \ldots \cap \partial B_k$. This way F becomes a pair of antipodal points $\{\pm u\}$, the projections of members of $\mathcal{H}'$ are ρ'-fat, and if $\delta(d,k)$ is sufficiently small, they are at distance at most 3ε from u. By the $k=0$ case we can find a point intersecting a constant fraction of the sets in $\{\mathrm{proj}_{F,u} K : K \in \mathcal{H}'\}$. The preimage of this point under $\mathrm{proj}_{F,u}$ is part of a great k-sphere of $\mathbb{S}^d$, and hits a constant fraction of $\mathcal{H}'$, thus a constant fraction of $\mathcal{H}$ and of $\{K_2, \ldots, K_n\}$. ◁

4 $(p, k+2)$-theorems

The specific $\rho = 1$ case of the fractional Helly results of Section 2 implies (p,q)-type results for k-flats intersecting balls. In this section we summarize arguments showing that Theorem 10 implies Theorem 8. The following colorful generalization can also be proved with a variant of this method. To keep the presentation simple, we only describe the arguments in a non-colorful setting. The interested reader can find a version of the colorful arguments in [5].

▶ **Theorem 21.** *For every three positive integers d, $k < d$ and $p \geq k+2$ there exists a C such that the following holds. If $\mathcal{F}_1, \ldots, \mathcal{F}_p$ are families of closed balls in $\mathbb{R}^d$ such that for every $B_1 \in \mathcal{F}_1, \ldots, B_p \in \mathcal{F}_p$ there are $k+2$ balls that can be hit by a single k-flat, then there exist an $i \in [p]$ and at most C k-flats hitting all balls of $\mathcal{F}_i$.*

By the same method, the following is a corollary of Theorem 11.

▶ **Theorem 22.** *For every three positive integers d, $k < d$ and $p \geq k+2$ there exists a C such that the following holds. If $\mathcal{F}_1, \ldots, \mathcal{F}_p$ are finite families of closed caps in $\mathbb{S}^d$ such that for every $B_1 \in \mathcal{F}_1, \ldots, B_p \in \mathcal{F}_p$ there are $k+2$ caps that can be hit by a great k-sphere, then there exist an $i \in [p]$ and at most C great k-spheres hitting all caps of $\mathcal{F}_i$.*

Note that the case where the families of balls (caps) can be infinite follows from the finite case by a standard compactness argument. The proofs of the finite versions are simple applications of the Alon-Kleitman method established in [3], later described in a more general setting in [2], and adapted to colorful variants in [5]. The abstract (purely combinatorial) proof described in [2] has two main steps. To state them, we phrase the fractional Helly and (p,q)-theorems in an abstract, purely combinatorial language as follows.

Let $\mathcal{H}$ be a set system over the base set V. For an integer q and a function $\beta\colon (0,1] \to (0,1]$, the set system $\mathcal{H}$ satisfies the fractional Helly property $\mathrm{FH}(q,\beta)$ if the following holds. For every $\alpha > 0$ and every finite family $\mathcal{G} \subset \mathcal{H}$, if $\alpha\binom{|\mathcal{G}|}{q}$ of the q-tuples of $\mathcal{G}$ has a nonempty intersection, then there exists a subfamily $\mathcal{G}' \subset \mathcal{G}$ of size $\beta(\alpha)|\mathcal{G}|$ such that all the members of $\mathcal{G}'$ have an element of V in common. The fractional Helly theorem of Katchalski and Liu (Theorem 1) states that if $V = \mathbb{R}^d$, and $\mathcal{H}$ consists of all the convex sets, then there exists β such that $\mathcal{H}$ satisfies $\mathrm{FH}(d+1,\beta)$. Our Theorem 5 states that if $V = \{k\text{-flats of } \mathbb{R}^d\}$ and $\mathcal{H}$ consists of families of k-flats intersecting a common ρ-fat convex set, then $\mathcal{H}$ satisfies $\mathrm{FH}(k+2,\beta)$ with some β.

A set system $\mathcal{H}$ satisfies the (p,q)-condition if among any p members of $\mathcal{H}$, some q intersect. Note that for $p \geq q \geq r$, the (p,q)-condition implies the (p,r)-condition. The transversal number $\tau(\mathcal{H})$ denotes the minimum size of a set $T \subset V$ with $T \cap H \neq \emptyset$ for all $H \in \mathcal{H}$. The fractional transversal number $\tau^*(\mathcal{H})$ is the minimum $\sum_{v \in V} t(v)$ over functions $t\colon V \to [0,1]$ with $\sum_{v \in H} t(v) \geq 1$ for all $H \in \mathcal{H}$. Alon, Kalai, Matoušek and Meshulam proved the following two theorems.

▶ **Theorem 23** (Theorem 8 in Alon, Kalai, Matoušek and Meshulam [2])**.** *If $\mathcal{H}$ satisfies $FH(q,\beta)$ and the (p,q)-condition with some $p \geq q$, then $\tau^*(\mathcal{H}) \leq O_{p,q,\beta}(1)$.*

Let $\mathcal{H}^\cap$ be the family of all sets that are the intersections of some members of $\mathcal{H}$.

▶ **Theorem 24** (Theorem 9 in Alon, Kalai, Matoušek and Meshulam [2]). *For every q and β there exists a function $f_{q,\beta}$ such that if $\mathcal{H}^\cap$ satisfies $FH(q,\beta)$, then $\tau(\mathcal{H}) \leq f_{q,\beta}(\tau^*(\mathcal{H}))$.*

Note that if $\mathcal{H}^\cap$ satisfies $F(q,\beta)$ and $\mathcal{H}$ satisfies the (p,q) condition, then its transversal number is bounded by a function of p, q and β as a consequence of the above two theorems. As the family of convex sets of $\mathbb{R}^d$ is closed under intersection, and satisfies $FH(d+1,\beta)$, this provides an alternative proof of Theorem 6 of Alon and Kleitman. But the same reasoning is usually not applicable to questions about k-transversals with $k > 0$, as the corresponding $\mathcal{H}^\cap \neq \mathcal{H}$ is hard to analyze, for example, for balls the intersections are no longer fat. However, in this special case, another proof method works.

If we restrict the question to k-transversals of balls [caps], as in Theorem 21 [Theorem 22], then we can bound τ as a function of τ^* and thus prove our $(p, k+2)$-theorems. The bound on τ follows from a classical result of Haussler and Welzl [9] . The VC-dimension of $\mathcal{H}$ is the supremum of the sizes of subsets $S \subseteq V$ such that for every $X \subseteq S$ there is an $H \in \mathcal{H}$ satisfying $H \cap S = X$.

▶ **Theorem 25** (Haussler and Welzl [9]). *For every d there exists a function f_d such that if $\mathcal{H}$ has VC-dimension at most d, then $\tau(\mathcal{H}) \leq f_d(\tau^*(\mathcal{H}))$.*

The VC-dimension of k-flats intersecting balls [caps] is bounded by the Milnor-Thom theorem on sign patterns of polynomials [18, 20], because a family of k-flats intersecting a given ball [cap] can be described by a bounded number of polynomial inequalities of bounded degree.

As for k-flats intersecting ρ-fat convex sets neither Theorem 24, nor Theorem 25 is applicable, the missing ingredient is proving a (p,q)-theorem for k-flats intersecting ρ-fat convex sets.

▶ **Conjecture 26.** *There is a function f such that if $\mathcal{H}$ consists of families of k-flats intersecting members of a family of ρ-fat convex sets in $\mathbb{R}^d$, then $\tau(\mathcal{H}) \leq f(\tau^*(\mathcal{H}))$.*

References

1. Noga Alon and Gil Kalai. Bounding the piercing number. *Discrete & Computational Geometry*, 13:245–256, 1995. doi:10.1007/BF02574042.
2. Noga Alon, Gil Kalai, Jiří Matoušek, and Roy Meshulam. Transversal numbers for hypergraphs arising in geometry. *Advances in Applied Mathematics*, 29(1):79–101, 2002. doi:10.1016/S0196-8858(02)00003-9.
3. Noga Alon and Daniel J Kleitman. Piercing convex sets and the Hadwiger-Debrunner (p,q)-problem. *Advances in Mathematics*, 96(1):103–112, 1992.
4. Boris Aronov, Jacob E Goodman, and Richard Pollack. A Helly-type theorem for higher-dimensional transversals. *Computational Geometry*, 21(3):177–183, 2002. doi:10.1016/S0925-7721(01)00025-6.
5. Imre Bárány, Ferenc Fodor, Luis Montejano, Deborah Oliveros, and Attila Pór. Colourful and fractional (p,q)-theorems. *Discrete & Computational Geometry*, 51(3):628–642, 2014. doi:10.1007/S00454-013-9559-0.
6. Arijit Ghosh and Soumi Nandi. Heterochromatic higher order transversals for convex sets. *arXiv preprint arXiv:2212.14091 version 1*, 2022. doi:10.48550/arXiv.2212.14091.
7. Jacob E Goodman, Richard Pollack, and Rephael Wenger. Geometric transversal theory. In *New trends in discrete and computational geometry*, pages 163–198. Springer, 1993.
8. Hugo Hadwiger and Hans Debrunner. Über eine Variante zum Hellyschen Satz. *Archiv der Mathematik*, 8:309–313, 1957. doi:10.1007/BF01898794.

9 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.

10 Eduard Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923.

11 Andreas Holmsen and Jiří Matoušek. No Helly theorem for stabbing translates by lines in $\mathbb{R}^3$. *Discrete & Computational Geometry*, 31(3):405–410, 2004. doi:10.1007/s00454-003-0796-5.

12 Fritz John. Extremum problems with inequalities as subsidiary conditions. studies and essays presented to R. Courant on his 60th birthday, 1948.

13 Attila Jung and Dömötör Pálvölgyi. A note on infinite versions of (p, q)-theorems. *arXiv preprint*, 2024. arXiv:2412.04066.

14 Meir Katchalski and Andy Liu. A problem of geometry in $\mathbf{R}^n$. *Proceedings of the American Mathematical Society*, 75(2):284–288, 1979. doi:10.2307/2042758.

15 Chaya Keller and Micha A. Perles. An $(\aleph_0, k+2)$-theorem for k-transversals. *38th International Symposium on Computational Geometry (SoCG 2022) and arXiv preprint arXiv:2306.02181*, 2022–2023. arXiv:2306.02181.

16 Jiří Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discrete & Computational Geometry*, 31:251–255, 2004. doi:10.1007/S00454-003-2859-Z.

17 Daniel McGinnis and Nikola Sadovek. A necessary and sufficient condition for k-transversals. *arXiv preprint*, 2024. arXiv:2411.07241.

18 John Milnor. On the Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964.

19 Luis A. Santaló. A theorem on sets of parallelepipeds with parallel edges. *Publicaciones del Instituto de Matemática de la Universidad Nacional del Litoral*, 2:49–60, 1940.

20 René Thom. Sur l'homologie des variétés algébriques réelles. *Differential and combinatorial topology*, pages 255–265, 1965.

21 Paul Vincensini. Figures convexes et variétés linéaires de l'espace euclidiena n dimensions. *Bulletin des Sciences Mathématiques*, 59:163–174, 1935.

On Approximability of ℓ_2^2 Min-Sum Clustering

Karthik C. S.
Rutgers University, Piscataway, NJ, USA

Euiwoong Lee
University of Michigan, Ann Arbor, MI, USA

Yuval Rabani
The Hebrew University of Jerusalem, Israel

Chris Schwiegelshohn
Aarhus University, Denmark

Samson Zhou
Texas A&M University, College Station, TX, USA

Abstract

The ℓ_2^2 min-sum k-clustering problem is to partition an input set into clusters $C_1, \ldots, C_k$ to minimize $\sum_{i=1}^{k} \sum_{p,q \in C_i} \|p-q\|_2^2$. Although ℓ_2^2 min-sum k-clustering is NP-hard, it is not known whether it is NP-hard to approximate ℓ_2^2 min-sum k-clustering beyond a certain factor.

In this paper, we give the first hardness-of-approximation result for the ℓ_2^2 min-sum k-clustering problem. We show that it is NP-hard to approximate the objective to a factor better than 1.056 and moreover, assuming a balanced variant of the Johnson Coverage Hypothesis, it is NP-hard to approximate the objective to a factor better than 1.327.

We then complement our hardness result by giving a fast PTAS for ℓ_2^2 min-sum k-clustering. Specifically, our algorithm runs in time $O(n^{1+o(1)}d \cdot 2^{(k/\varepsilon)^{O(1)}})$, which is the first nearly linear time algorithm for this problem. We also consider a learning-augmented setting, where the algorithm has access to an oracle that outputs a label $i \in [k]$ for input point, thereby implicitly partitioning the input dataset into k clusters that induce an approximately optimal solution, up to some amount of adversarial error $\alpha \in \left[0, \frac{1}{2}\right)$. We give a polynomial-time algorithm that outputs a $\frac{1+\gamma\alpha}{(1-\alpha)^2}$-approximation to ℓ_2^2 min-sum k-clustering, for a fixed constant $\gamma > 0$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Facility location and clustering

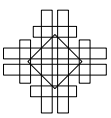
Keywords and phrases Clustering, hardness of approximation, polynomial-time approximation schemes, learning-augmented algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.62

Related Version *Full Version*: https://arxiv.org/abs/2412.03332

Funding *Karthik C. S.*: Supported by NSF 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.
Euiwoong Lee: Supported in part by NSF grant CCF-2236669 and Google.
Yuval Rabani: Supported in part by ISF grants 3565-21 and 389-22, and by BSF grant 2023607.
Chris Schwiegelshohn: Supported in part by the Independent Research Fund Denmark (DFF) under a Sapere Aude Research Leader grant No 1051-00106B and Google.
Samson Zhou: Supported in part by NSF CCF-2335411. The work was conducted in part while the author was visiting the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program.

© Karthik C. S., Euiwoong Lee, Yuval Rabani, Chris Schwiegelshohn, and Samson Zhou;
licensed under Creative Commons License CC-BY 4.0



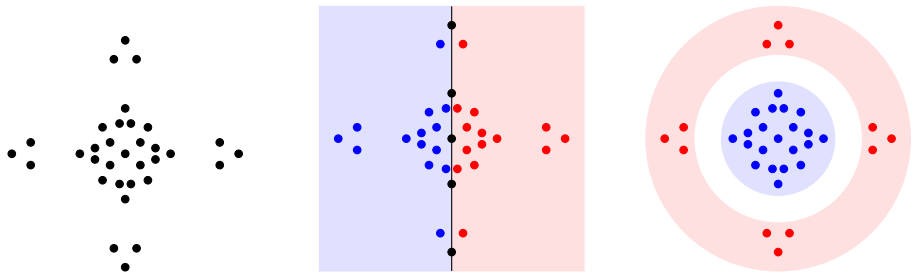
1 Introduction

Clustering is a fundamental technique that partitions an input dataset into distinct groups called clusters, which facilitate the identification and subsequent utilization of latent structural properties underlying the dataset. Consequently, various formulations of clustering are used across a wide range of applications, such as computational biology, computer vision, data mining, and machine learning [39, 70]. Ideally, the elements of each cluster are more similar to each other than to elements in other clusters. To formally capture this notion, a dissimilarity metric is often defined on the set of input elements, so that more closer objects in the metric correspond to more similar objects. Perhaps the most natural goal would be to minimize the intra-cluster dissimilarity in a partitioning of the input dataset. This objective is called the *min-sum k-clustering* problem and has received significant attention due to its intuitive clustering objective [33, 37, 58, 63, 11, 24, 22, 2, 12, 10, 18].

In this paper, we largely focus on the ℓ_2^2 min-sum k-clustering formulation. Formally, the input is a set X of n points in $\mathbb{R}^d$ and the goal is to partition $X = C_1 \dot\cup \cdots \dot\cup C_k$ into k clusters to minimize the quantity $\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \sum_{p,q\in C_i} \|p-q\|_2^2$, where $\|\cdot\|_2$ denotes the standard Euclidean ℓ_2 norm.

Whereas classical centroid-based clustering problems such as k-means and k-median leverage distances between data points and cluster centroids to identify convex shapes that partition the dataset, min-sum k-clustering is a density-based clustering that can handle complex structures in data that may not be linearly separable. In particular, min-sum k-clustering can be more effective than traditional centroid-based clustering in scenarios where clusters have irregular, non-convex shapes or overlapping clusters. A simple example of the ability of min-sum clustering to capture more natural structure is an input that consists of two concentric dense rings of points in the plane. Whereas min-sum clustering can partition the points into the separate rings, centroid-based clustering will instead create a separating hyperplane between these points, thereby "incorrectly" grouping together points of different rings. See Figure 1 for an example of the ability of min-sum clustering to capture natural structure in cases where centroid-based clustering fails.

Moreover, min-sum clustering satisfies Kleinberg's consistency axiom [47], which informally demands that the optimal clustering for a particular objective should be preserved when distances between points inside a cluster are shrunk and distances between points in different clusters are expanded. By contrast, many centroid-based clustering objectives, including k-means and k-median, do not satisfy Kleinberg's consistency axiom [57].



(a) Input dataset. **(b)** Centroid-based clustering. **(c)** Density-based clustering.

Figure 1 Clustering of input dataset in Figure 1a with $k = 2$. Figure 1b is an optimal centroid-based clustering, e.g., k-median or k-means, while the more natural clustering in Figure 1c is an optimal density-based clustering, e.g., ℓ_2 min-sum k-clustering.

On the other hand, theoretical understanding of density-based clustering objectives such as min-sum k-clustering is far less developed than that of their centroid-based counterparts. It can be shown that min-sum k-clustering with the ℓ_2^2 cost function is NP-hard, using arguments from [2]. The problem is NP-hard even for $k = 2$ [25] in the metric case, where the only available information about the points is their pairwise dissimilarity. In fact, for general k in the metric case, it is NP-hard to approximate the problem within a 1.415-multiplicative factor [32, 18]. However, no such hardness of approximation is known for the Euclidean case, i.e., ℓ_2^2 min-sum, where the selected cost function is based on the geometry of the underlying space; the only known lower bound is the NP-hardness of the problem [2, 10, 3]. Thus a fundamental open question is:

▶ **Question 1.** *Is ℓ_2^2 min-sum k-clustering APX-hard? That is, does there exist a natural hardness-of-approximation barrier for polynomial time algorithms?*

Due to existing APX-hardness results for centroid-based clustering such as k-means and k-median [50, 17, 19], it is widely believed that ℓ_2^2 min-sum clustering is indeed APX-hard. Thus, there has been a line of work preemptively seeking to overcome such limitations. Indeed, on the positive side, [36] first showed that min-sum k-clustering in the d-dimensional ℓ_2^2 case can be solved in polynomial time if both d and k are constants. For general graphs and fixed constant k, [33] gave a 2-approximation algorithm using runtime $n^{\mathcal{O}(k)}$. The approximation guarantees were improved by a line of work [37, 58, 63], culminating in polynomial-time approximation schemes by [24] for both the ℓ_2^2 case and the metric case. Without any assumptions on d and k, [11] introduced a polynomial algorithm that achieves an $\mathcal{O}\left(\frac{1}{\varepsilon}\log^{1+\varepsilon} n\right)$-multiplicative approximation. Therefore, a long-standing direction in the study of ℓ_2^2 min-sum clustering is:

▶ **Question 2.** *How can we algorithmically bridge the gap between the NP-hardness of solving the ℓ_2^2 min-sum clustering and the large multiplicative guarantees of existing approximation algorithms?*

A standard approach to circumvent poor dependencies on the size of the input dataset is to sparsify the problem. Informally, we would like to reduce the search space by considering fewer candidate solutions and reduce the dependency on the number of input points by aggregating them. For min-sum clustering this is a particular challenge, as a candidate solution is a partition and the cost of that partition depends on all pairwise distances between all the points. While sparsification algorithms exist for graph clustering [40, 51] and k-means clustering [21, 20], where the output is typically called a coreset, similar constructions are not known to exist for min-sum clustering.

Another standard approach to overcome limitations inherent in worst-case impossibility barriers is to consider beyond worst case analysis. To that end, recent works have observed that in many applications, auxiliary information is often available and can potentially form the foundation upon which machine learning models are built. For example, previous datasets with potentially similar behavior can be used as training data for models to label future datasets. However, these heuristics lack provable guarantees and can produce embarrassingly inaccurate predictions when generalizing to unfamiliar inputs [65]. Nevertheless, *learning-augmented algorithms* [60] have been shown to achieved both good algorithmic performance when the oracle is accurate, i.e., consistency, and standard algorithmic performance when the oracle is inaccurate, i.e., robustness for a wide range of settings, such as data structure design [48, 59, 55], algorithms with faster runtime [26, 15, 23], online algorithms with better

competitive ratio [62, 30, 49, 68, 69, 9, 35, 56, 1, 4, 8, 31, 45, 41, 5, 64], and streaming algorithms that are more space-efficient [34, 38, 43, 14, 13, 54]. In particular, [28, 61] introduce algorithms for k-means and k-median clustering that can achieve approximation guarantees beyond the known APX-hardness limits.

1.1 Our Contributions

In this paper, we perform a comprehensive study on the approximability of the ℓ_2^2 min-sum k-clustering by answering Question 1 and Question 2.

Hardness-of-approximation of min-sum k-clustering. We first answer Question 1 in the affirmative, by not only showing that the ℓ_2^2 min-sum k-clustering is APX-hard but further giving an explicit constant NP-hardness of approximation result for the problem.

▶ **Theorem 3** (Hardness of approximation of ℓ_2^2 min-sum k-clustering)**.** *It is NP-hard to approximate ℓ_2^2 min-sum k-clustering to a factor better than* 1.056*. Moreover, assuming the Dense and Balanced Johnson Coverage Hypothesis (*$\mathsf{Balanced} - \mathsf{JCH}^*$*), we have that the ℓ_2^2 min-sum k-clustering is NP-hard to approximate to a factor better than* 1.327*.*

We remark that $\mathsf{Balanced} - \mathsf{JCH}^*$ in the theorem statement above is simply a balanced formulation of the recently introduced Johnson Coverage Hypothesis [19].

Fast polynomial-time approximation scheme. In light of Theorem 3, a natural question would be to closely examine alternative conditions in which we can achieve a $(1+\varepsilon)$-approximation to min-sum k-clustering, i.e., Question 2. To that end, there are a number of existing polynomial-time approximation schemes (PTAS) [37, 58, 63, 24], the best of which uses runtime $n^{\mathcal{O}\left(k/\varepsilon^2\right)}$ for the ℓ_2^2 case. However, as noted by [22], even algorithms with runtime quadratic in the size n of the input dataset are generally not sufficiently scalable to handle large datasets. In this paper, we present an algorithm with a running time that is nearly nearly linear. Specifically, we show

▶ **Theorem 4.** *There exists an algorithm running in time $O\left(n^{1+o(1)}d \cdot 2^{\eta \cdot k^2 \cdot \varepsilon^{-12}\log^2(k/(\varepsilon\delta))}\right)$, for some absolute constant η, that computes a $(1+\varepsilon)$-approximate solution to ℓ_2^2 k-MinSum Clustering with probability $1-\delta$.*

We again emphasize that the runtime of 4 is linear in the size n of the input dataset, though it has exponential dependencies in both the number k of clusters and the approximation parameter $\varepsilon > 0$. By contrast, the best previous PTAS uses runtime $n^{\mathcal{O}\left(k/\varepsilon^2\right)}$, which has substantially worse dependency on the size n of the input dataset.

Learning-augmented algorithms. Unfortunately, exponential dependencies on the number k of clusters can still be prohibitive for moderate values of k. To that end, we turn our attention to learning-augmented papers. We consider the standard *label oracle* model for clustering, where the algorithm has access to an oracle that provides a label for each input point. Formally, for each point x of the n input points, the oracle outputs a label $i \in [k]$ for x, so that the labels implicitly partition the input dataset into k clusters that induce an approximately optimal solution. However, the oracle also has some amount of adversarial error that respects the precision and recall of each cluster; we defer the formal definition to Definition 25. One of the reasons label oracles have been used for learning-augmented algorithms for clustering is their relative ease of acquisition via machine learning models

that are trained on a similar distribution of data. For example, a smaller separate dataset can be observed and used as a "training" data, an input to some heuristic to cluster the initial data, which we can then use to form a predictor for the actual input dataset. Indeed, implementations of label oracles have been shown to perform well in practice [28, 61].

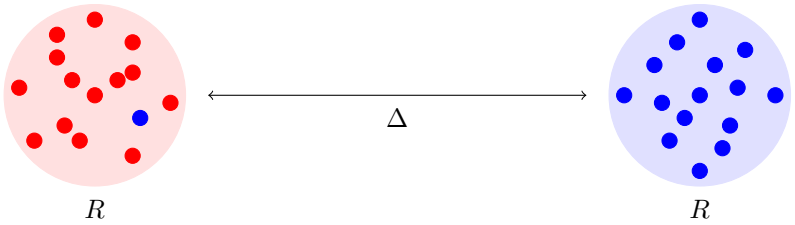


Figure 2 Note that with arbitrarily small error rate, i.e., $\frac{1}{n}$, a single mislabeled point among the n input points causes the resulting clustering to be arbitrarily bad for $\Delta \gg n^2 \cdot R$.

We also remark that perhaps counter-intuitively, a label oracle with arbitrarily high accuracy does not trivialize the problem. In particular, the naïve algorithm of outputting the clustering induced by the labels does not work. As a simple example, consider an input dataset where half of the n points are at $x = 0$ and the other half of the points are at $x = 1$. Then for $k = 2$, the clear optimal clustering is to cluster the points at the origin together, and cluster the points at $x = 1$ together, which induces the optimal cost of zero. However, if even one of the n points is incorrect, then the clustering output by the labels has cost at least 1. Therefore, even with error rate as small as $\frac{1}{n}$, the multiplicative approximation of the naïve algorithm can be arbitrarily bad. See Figure 2 for an illustration of this example. Of course, this example does not rule out more complex algorithms that combines the labels with structural properties of optimal clustering and indeed, our algorithm utilizes such properties.

We give a polynomial-time algorithm for the ℓ_2^2 min-sum k-clustering that can provide guarantees beyond the computational limits of Theorem 3, given a sufficiently accurate oracle.

▶ **Theorem 5.** *There exists a polynomial-time algorithm that uses a label predictor with error rate* $\alpha \in \left[0, \frac{1}{2}\right)$ *and outputs a* $\frac{1+\gamma\alpha}{(1-\alpha)^2}$*-approximation to the* ℓ_2^2 *min-sum* k*-clustering problem, where* $\gamma = 7.7$ *for* $\alpha \in \left[0, \frac{1}{7}\right)$ *or* $\gamma = \frac{5\alpha - 2\alpha^2}{(1-2\alpha)(1-\alpha)}$ *for* $\alpha \in \left[0, \frac{1}{2}\right)$.

We remark that Theorem 5 does not require the true error rate α as an input parameter. Because we are in an offline setting, where can run Theorem 5 multiple times with guesses for the true error rate α, in decreasing powers of $\frac{1}{\lambda}$ for any constant $\lambda > 1$. We can then compare the resulting clustering output by each guess for α and take the output the best clustering.

1.2 Technical Overview

Hardness of approximation. Recently, the authors of [19] put forth the Johnson Coverage Hypothesis (JCH) and introduced a framework to obtain (optimal) hardness of approximation results for k-median and k-means in ℓ_p-metrics. The proof of Theorem 3 builds on this framework. JCH roughly asserts that for large enough constant z, given as input an integer k and a collection of z-sets (i.e., sets each of size z) over some universe, it is NP-hard to distinguish the completeness case where there is a collection C of k many $(z-1)$-sets such

that every input set is covered[1] by some set in C, from the soundness case where every collection C of k many $(z-1)$-sets does not cover much more than $1-\frac{1}{e}$ fraction of the input sets (see Hypothesis 7 for a formal statement).

In this paper, we consider a natural generalization of JCH, called Balanced − JCH*, where we assume that the number of input sets is "dense", i.e., $\omega(k)$, and more importantly that in the completeness case, the collection C covers the input z-sets in a balanced manner, i.e., we can partition the input to k equal parts such that each part is completely covered by a single set in C (see Hypothesis 9 for a formal statement).

We now sketch the proof of Theorem 3 assuming Balanced − JCH*. Given a collection of m many z-sets over a universe $[n]$ as input, we create a point for each input set, which is simply the characteristic vector of the set as a subset of $[n]$, i.e., the points are all n-dimensional Boolean vectors of Hamming weight z. In the completeness case, from the guarantees of Balanced − JCH*, it is easy to see that the points created can be divided into k equal clusters of size m/k such that all the z-sets of a cluster are completely covered by a single $(z-1)$-set. This implies that the squared Euclidean distance between a pair of points within a cluster is exactly 2 and thus the ℓ_2^2 min-sum k-clustering cost is $k \cdot 2 \cdot (m/k)(m/k-1) \approx 2m^2/k$.

On the other hand, in the soundness case, we first use the density guarantees of Balanced − JCH* to argue that most clusters are not small. Then suppose that we had a low cost ℓ_2^2 min-sum k-clustering, we look at a typical cluster and observe that the squared distance of any two points in the cluster must be a positive even integer, and it is exactly 2 only when the two input sets corresponding to the points intersect on a $(z-1)$-set. Thus, if the cost of the clustering is close to $\alpha \cdot 2m^2/k$ (for some $\alpha \geq 1$), then we argue (using convexity) that for a typical cluster that there must be a $(z-1)$-set that covers $(1-\alpha')m/k$ many z-sets in that cluster, where α' depends on α. Thus, from this we decode k-many $(z-1)$-sets which cover a large fraction of the input z-sets. In order to obtain the unconditional NP-hardness result, much like in [19], we need to extend the above reduction to a more general problem. This is indeed established in Theorem 12, and after this we prove a special case of a generalization of Balanced − JCH* (when $z=3$) which is done in Theorem 11 and this involved proving additional properties of the reduction of [19] from the multilayered PCPs of [27, 46] to 3-Hypergraph Vertex Coverage.

Nearly Linear Time PTAS. An important feature of ℓ_2^2 Min-Sum Clustering is that we can use assignments of clusters to their mean to obtain the cost of the points in the cluster, an idea previously used in [37, 58, 63, 24]. We show how to reduce the number of candidate means to a constant (depending only on k and ε. The idea here is to use D^2 sampling methods akin to k-means++ [6]. Unfortunately, by itself, it is not sufficient as there may exist clusters that have significant Min-Sum clustering cost, but are not detectable by D^2 sampling. To this end, we augment D^2 sampling via a careful pruning strategy that removes high costing points, increasing the relative cost of clusters of high density. Thereafter, we show that given sufficiently many samples, we can find a small set of suitable candidate means that are induced by a nearly optimal clustering.

What remains to be shown is how to find an assignment of points to these centers with similar cost. For this, we could use a flow-based approach, but this results in a n^3 running time. Instead, we employ a discretization and bucketing strategy that allows us to sparsify the point set while preserving the Min-Sum clustering cost, akin to coresets.

[1] A $(z-1)$-set covers a z-set if the former is a subset of the latter.

Learning-augmented algorithm. Our starting point for our learning-augmented algorithm for min-sum k-clustering is the learning-augmented algorithms for k-means clustering by [28, 61]. We can rewrite the Min-Sum clustering cost in terms of weighted squared distances to the means or centroids. Our goal is therefore to quickly identify suitable centroids c_i. The algorithms note that the clustering objective can be decomposed across the points that are given each label $i \in [k]$. Thus we consider the subset P_i of points of the input dataset X that are given label i by the oracle.

The cluster P_i can have an α fraction of incorrect points. The main observation is that there can be two cases. Either P_i includes a number of "bad" points that are far from the true mean and thus easy to identify, or P_i includes a number of "bad" points that are difficult to identify but also are close to the true mean and thus do not largely affect the overall clustering cost. Thus the algorithm simply needs to prune away the points that are far away, which can be achieved by selecting the interval of $(1 - \mathcal{O}(\alpha))$ points that has the best clustering cost. Therefore, we have a set of centers for which there exists an assignment that obtains a good approximation of the cost of the optimal min-sum k-clustering; it remains to identify the actual clusters.

To find an assignment of points to candidate means, we use min-cost flow approach, similar to [24]. The constrained min-cost flow problem can be written as a linear program. Therefore to identify the overall clusters, we run any standard polynomial-time algorithm for solving linear programs [44, 66, 67, 52, 53, 16, 42]. It then follows by that well-known integrality theorems for min-cost flow, the resulting solution is integral and thus provides a valid clustering with approximately optimal ℓ_2^2 min-sum k-clustering objective.

2 Hardness of Approximation of ℓ_2^2 Min-Sum k-Clustering

In this section, we show the hardness of approximation of ℓ_2^2 min-sum k-clustering, i.e., Theorem 3. We first define the relevant formulations of Johnson Coverage Hypothesis in Section 2.1. Next, in Section 2.2 we provide the main reduction from the Johnson coverage problem to the ℓ_2^2 min-sum k-clustering problem. Finally, we prove a special case of a generalization of $\mathsf{Balanced} - \mathsf{JCH}^*$ which yields the unconditional NP-hardness factor claimed in Theorem 3.

2.1 Johnson Coverage Hypothesis

In this section, we recall the Johnson Coverage problem, followed by the Johnson Coverage hypothesis [19]. Let $n, z, y \in \mathbb{N}$ such that $n \geq z > y$. Let $E \subseteq \binom{[n]}{z}$ and $S \in \binom{[n]}{y}$. We define the coverage of S w.r.t. E, denoted by $\mathsf{cov}(S, E)$ as follows: $\mathsf{cov}(S, E) = \{T \in E \mid S \subset T\}$.

▶ **Definition 6** (Johnson Coverage Problem). *In the (α, z, y)-Johnson Coverage problem with $z > y \geq 1$, we are given a universe $U := [n]$, a collection of subsets of U, denoted by $E \subseteq \binom{[n]}{z}$, and a parameter k as input. We would like to distinguish between the following two cases:*

- **Completeness**: *There exists $\mathcal{C} := \{S_1, \ldots, S_k\} \subseteq \binom{[n]}{y}$ such that*

$$\mathsf{cov}(\mathcal{C}) := \bigcup_{i \in [k]} \mathsf{cov}(S_i, E) = E.$$

- **Soundness**: *For every $\mathcal{C} := \{S_1, \ldots, S_k\} \subseteq \binom{[n]}{y}$ we have $|\mathsf{cov}(\mathcal{C})| \leq \alpha \cdot |E|$.*

We call $(\alpha, z, z-1)$-Johnson Coverage as (α, z)-Johnson Coverage.

Notice that $(\alpha, 2)$-Johnson Coverage Problem is simply the well-studied vertex coverage problem (with gap α). Also, notice that if instead of picking the collection $\mathcal{C}$ from $\binom{[n]}{y}$, we replace it with picking the collection $\mathcal{C}$ from $\binom{[n]}{1}$ with a similar notion of coverage, then we simply obtain the Hypergraph Vertex Coverage problem (which is equivalent to the Max k-Coverage problem for unbounded z). In Figure 3 we provide a few examples of instances of the Johnson coverage problem.

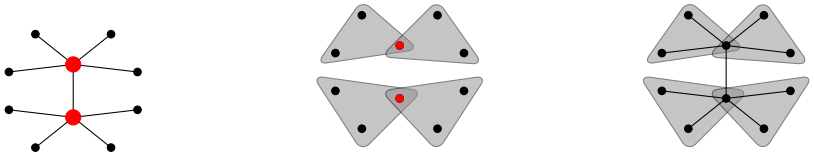


Figure 3 Examples of input instances of the Johnson Coverage Hypothesis for $k = 2$. Figure 3a shows an example of a completeness instance of $(0.7, 2, 1)$, since all subsets of size 2, i.e., all edges, can be covered by $k = 2$ choices of subset of size 1, i.e., two vertices. Figure 3b shows an example of a completeness instance of $(0.7, 3, 1)$, since all subsets of size 3 can be covered by $k = 2$ vertices. Figure 3c shows an example of a soundness instance of $(0.7, 3, 2)$, since at most $2 \leq 0.7 \cdot 4$ subsets of size 3 can be covered by any choice of $k = 2$ edges.

▶ **Hypothesis 7** (Johnson Coverage Hypothesis (JCH) [19])**.** *For every constant $\varepsilon > 0$, there exists a constant $z := z(\varepsilon) \in \mathbb{N}$ such that deciding the $\left(1 - \frac{1}{e} + \varepsilon, z\right)$-Johnson Coverage Problem is* NP*-Hard.*

Note that since Vertex Coverage problem is a special case of the Johnson Coverage problem, we have that the NP-Hardness of (α, z)-Johnson Coverage problem is already known for $\alpha = 0.944$ [7] (under unique games conjecture).

On the other hand, if we replace picking the collection $\mathcal{C}$ from $\binom{[n]}{z-1}$ by picking from $\binom{[n]}{1}$, then for the Hypergraph Vertex Coverage problem, we do know that for every $\varepsilon > 0$ there is some constant z such that the Hypergraph Vertex Coverage problem is NP-Hard to decide for a factor of $\left(1 - \frac{1}{e} + \varepsilon\right)$ [29]. For continuous clustering objectives, a dense version of JCH is sometimes needed to prove inapproximability results (see [19] for a discussion on this). Thus, we state:

▶ **Hypothesis 8** (Dense Johnson Coverage Hypothesis (JCH*) [19])**.** JCH *holds for instances (U, E, k) of Johnson Coverage problem where $|E| = \omega(k)$.*

More generally, let (α, z, y)-Johnson Coverage* problem be the special case of the (α, z, y)-Johnson Coverage problem where the instances satisfy $|E| = \omega(k \cdot |U|^{z-y-1})$. Then JCH* states that for any $\varepsilon > 0$, there exists $z = z(\varepsilon)$ such that $(1 - 1/e + \varepsilon, z, z - 1)$-Johnson Coverage* is NP-Hard. This additional property has always been obtained in literature by looking at the hard instances that were constructed. In [17], where the authors proved the previous best inapproximability results for continuous case k-means and k-median, it was observed that hard instances of $(0.94, 2, 1)$-Johnson Coverage constructed in [7] can be made to satisfy the above property. Now we are ready to define the variant of JCH needed for proving inapproximability of ℓ_2^2 min-sum k-clustering. For any two non-empty finite sets A, B, and a constant $\delta \in [0, 1]$, we say a function $f : A \to B$ is δ-balanced if for all $b \in B$ we have $|\{a \in A : f(a) = b\}| \leq (1 + \delta) \cdot \frac{|A|}{|B|}$. We then put forth the following hypothesis.

▶ **Hypothesis 9** (Dense and Balanced Johnson Coverage Hypothesis (Balanced − JCH*))**.** JCH *holds for instances* (U, E, k) *of Johnson Coverage problem where* $|E| = \omega(k)$ *and in the completeness case there exists* $\mathcal{C} := \{S_1, \ldots, S_k\} \subseteq \binom{[n]}{z-1}$ *and a* **0-balanced** *function* $\psi : E \to [k]$ *such that for all* $T \in E$ *we have* $S_{\psi(T)} \subset T$*.*

More generally, let (α, z, y, δ)-Balanced Johnson Coverage* problem be the special case of the (α, z, y)-Johnson Coverage* problem where the instances admit a δ-balanced function $\psi : E \to [k]$ in the completeness case which partitions E to k parts, say $E_1 \dot\cup \cdots \dot\cup E_k$ such that for all $i \in [k]$ we have $\mathsf{cov}(S_i, E_i) = E_i$ and $|E_i| \leq \frac{|E|}{k} \cdot (1 + \delta)$. Then Balanced − JCH* states that for any $\varepsilon > 0$, there exists $z = z(\varepsilon)$ such that $(1 - 1/e + \varepsilon, z, z - 1, 0)$-Balanced Johnson Coverage* is NP-Hard. As with the case of JCH*, the balanced addition to JCH* is also quite natural and candidate constructions typically give this property for free. To support this point, we will prove some special case of this. In [19] the authors had proved the following special case of JCH*.

▶ **Theorem 10** ([19])**.** *For any* $\varepsilon > 0$*, given a simple 3-hypergraph* $\mathcal{H} = (V, H)$ *with* $n = |V|$*, it is NP-hard to distinguish between the following two cases:*

- **Completeness:** *There exists* $S \subseteq V$ *with* $|S| = n/2$ *that intersects every hyperedge.*
- **Soundness:** *Any subset* $S \subseteq V$ *with* $|S| \leq n/2$ *intersects at most a* $(7/8 + \varepsilon)$ *fraction of hyperedges.*

Furthermore, under randomized reductions, the above hardness holds when $|H| = \omega(n^2)$*.*

▶ **Theorem 11.** *Theorem 10 holds even with the following additional completeness guarantee for all* $\delta > 0$*: there exists* $S := \{v_1, \ldots, v_k\} \subseteq V$ *and a* δ*-balanced function* $\psi : H \to [k]$ *such that for all* $e \in H$ *we have* $v_{\psi(e)} \in e$*.*

This result will be used to prove the unconditional NP-hardness of approximating ℓ_2^2 min-sum k-clustering problem.

2.2 Inapproximability of ℓ_2^2 min-sum k-clustering

▶ **Theorem 12.** *Assume* (α, z, y, δ)*-Balanced Johnson Coverage** *is* NP*-Hard. For every constant* $\varepsilon > 0$*, given a point-set* $P \subset \mathbb{R}^d$ *of size* n *(and* $d = \mathcal{O}(\log n)$*) and a parameter* k *as input, it is* NP*-Hard to distinguish between the following two cases:*

- **Completeness***: There exists partition* $P_1^* \dot\cup \cdots \dot\cup P_k^* := P$ *such that*

$$\sum_{i \in [k]} \sum_{p,q \in P_i^*} \|p - q\|_2^2 \leq (1 + 3\delta) \cdot (z - y) \cdot \rho n^2 / k,$$

- **Soundness***: For every partition* $P_1 \dot\cup \cdots \dot\cup P_k := P$ *we have*

$$\sum_{i \in [k]} \sum_{p,q \in P_i} \|p - q\|_2^2 \geq (1 - o(1)) \cdot \left(\alpha \cdot \sqrt{z - y} + (1 - \alpha) \cdot \sqrt{z - y + 1}\right)^2 \cdot \rho n^2 / k,$$

for some constant $\rho > 0$*.*

Putting together the above theorem with Theorem 11 (i.e., NP-hardness of $(7/8+\varepsilon, 3, 1, \delta)$-Balanced Johnson Coverage* problem for all $\varepsilon, \delta > 0$), we obtain the NP-hardness of approximating ℓ_2^2 min-sum k-clustering. The above theorem also immediately yields the hardness of approximating ℓ_2^2 min-sum k-clustering under Balanced − JCH* (i.e., conditional NP-hardness of $(1 - 1/e + \varepsilon, z, z - 1, 0)$-Balanced Johnson Coverage* problem for all $\varepsilon > 0$ and some $z = z(\varepsilon) \in \mathbb{N}$). This completes the proof of Theorem 3.

3 PTAS based on D^2 Sampling

For a set $A \subset \mathbb{R}^d$, let $\mu(A) := \frac{1}{|A|}\sum_{p\in A} p$ denote its mean. Let $\mathcal{C} = \{C_1, \dots C_k\}$ be an optimal k-MinSum clustering of a point set A. We use $\mu_i = \mu(C_i)$ to denote the mean of C_i and we use $\Delta_i = \frac{\sum_{p\in C_i}\|p-\mu_i\|^2}{|C_i|}$ to denote the average mean squared distance of C_i to μ_i. We further use C_i^β to denote the subset of C_i with $\|p-\mu_i\|^2 \le \beta\cdot\Delta_i$. Finally, let OPT denote the cost of an optimal solution. So, $\mathsf{OPT} = \sum_{i=1}^k |C_i|^2\cdot\Delta_i$.

▶ **Definition 13.** *We say that m is an ε-approximate mean of C_i if $\|m-\mu_i\|^2 \le \varepsilon\cdot\Delta_i$. We say that a set $S\subset A$ is an (ε,β)-mean seeding set for $C_i\in\mathcal{C}$, if there exists a subset $S'\cup\{s\}\subset S$ with $\|s-\mu_i\|^2\le\beta\cdot\Delta_i$ and a weight assignment $w: S'\to\mathbb{R}_{\ge 0}$ such that $\left\|\frac{1}{\sum_{p\in S'}w(p)}\sum_{p\in S'}w(p)\cdot p-\mu_i\right\|^2 \le \varepsilon\cdot\Delta_i$.*

We will use the following well-known identities for Euclidean means.

▶ **Lemma 14** ([36])**.** *Let $A\subset\mathbb{R}^d$ be a set of points. Then for any $c\in\mathbb{R}^d$:*

- $\sum_{p\in A}\|p-c\|^2 = \sum_{p\in A}\|p-\mu(A)\|^2 + |A|\cdot\|\mu(A)-c\|^2$.
- $\sum_{p,q\in A}\|p-q\|^2 = 2\cdot|A|\cdot\sum_{p\in A}\|p-\mu(A)\|^2$.

We also show that we only have to consider seeding sets with $\beta\in\Theta(\varepsilon^{-2})$.

▶ **Lemma 15.** *For any cluster C_i, $\varepsilon\in(0,1)$ and $\beta\ge 12\varepsilon^{-2}$, we have that $\mu_i(C_i^\beta) = \frac{1}{|C_i^\beta|}\sum_{p\in C_i^\beta} p$ is a ε-approximate mean of C_i.*

Finally, we also show how to efficiently extract a mean from a mean seeding set, while being oblivious to Δ_i.

▶ **Lemma 16.** *Let S be an $(\varepsilon/4,\beta)$-mean seeding set of a cluster C_j with mean μ_j. Then we can compute $\left(\frac{10\beta\cdot|S|}{\varepsilon}+1\right)^{|S|}$ choices of weights in time linear in the size of choices such that at least one of the computed choices satisfies $\left\|\frac{1}{\sum_{p\in S}w(p)}\sum_{p\in S}w(p)\cdot p-\mu_j\right\|^2\le\varepsilon\cdot\Delta_j$.*

Computing a Mean-Seeding Set via Uniform Sampling

▶ **Lemma 17.** *Let $\varepsilon\in(0,1)$ and $\beta > 48\varepsilon^2$. With probability at least $1-\delta$, a set of $32k\varepsilon^{-1}\log\delta^{-1}$ points S sampled uniformly at random with replacement from A contains is a (ε,β)-mean seeding set of any C_i with $|C_i|\ge\frac{n}{k}$.*

D^2 Subsampling

We now define an algorithm for sampling points that induce means from the target clusters. The high level idea is as follows. We construct a rooted tree in which every node is labeled by a set of candidate cluster means. For a parent and child pair of nodes, the parent's set is a subset of the child's set. The construction is iterative. Given an interior node, we construct its children by adding a candidate mean to the parent's set. The candidantes are generated using points sampled at random from a distribution that will be defined later. The goal is to have, eventually, an ε-approximate mean for every optimal cluster. This will be achieved with high probability at one of the leaves of the tree. The root of the tree is labeled with the empty set, and its children are constructed via uniform sampling. Subsequently, we refine the sampling distribution to account for various costs and densities of the clusters.

We now go into more detail for the various sampling stages of the algorithm.

Preprocessing: We ensure that all points are not too far from each other.

Initialization: We initialize the set of means via uniform sampling. Due to Lemma 17, we can enumerate over potential sets of ε-approximate means for all clusters of size $\frac{n}{k}$. Each candidate mean defines a child of the root.

Sampling Stage: Consider a node of the tree labeled with a non-empty set of candidate means M. We put $\Gamma_i = 2^{-i} \cdot \sum_{q \in A} \min_{m \in M} \|q-m\|^2$ for $i \in \{0, 1, \ldots, 13 \log(nk/\varepsilon)\}$, where η is an absolute constant to be defined later. Let $A_{i,M} = \{q \in A \colon \min_{m \in M} \|q-m\|^2 \leq \Gamma_i\}$. (Note that $A_{0,M}$ includes all the points.) Let $\mathbb{P}_i$ denote the probability distribution on $A_{i,M}$ induced by setting, for each $p \in A_{i,M}$, $\mathbb{P}_i[p] = \frac{\min_{m \in M} \|p-m\|^2}{\sum_{p \in A_{i,M}} \min_{m \in M} \|p-m\|^2}$ We'll use $\mathbb{P}$ to denote $\mathbb{P}_0$. For each i, we sample a sufficient (polynomial in k and ε, but independent of n) number of points independently from the distribution $\mathbb{P}_i$. Let S denote the set of sampled points.

Mean Extraction Stage: We enumerate over combinations of points in $M \cup S$, using some non-uniform weighing to fix a mean to add to M, see Lemma 16. Each choice of mean is added to M to create a child of the node labeled M.

Throughout this section we will use the following definition. Given a set of centers M, we say that a cluster C_i is ε-covered by M if $|C_i|^2 \cdot \min_{m \in M} \|\mu_i - m\|^2 \leq \frac{\varepsilon}{2} \cdot \left(\frac{1}{k} \cdot OPT + |C_i|^2 \Delta_i\right)$. Our goal will be to prove the following lemma.

▶ **Lemma 18.** *Let $\mathcal{C} = \{C_1, \ldots C_k\}$ be the clusters of an optimal Min-Sum k-clustering and let η be an absolute constant. For every $\delta, \epsilon > 0$, there is a randomized algorithm that outputs a collection of at most $n^{o(1)} \cdot 2^{\eta \cdot k^2 \cdot \varepsilon^{-12} \log^2(k/(\varepsilon\delta))}$ sets of at most k centers M, such that with probability $1-\delta$ at least one of them that ε-covers every $C_i \in \mathcal{C}$. The algorithm runs in time $n^{1+o(1)} \cdot d \cdot 2^{\eta \cdot k^2 \cdot \varepsilon^{-12} \log^2(k/(\varepsilon\delta))}$.*

Note that if all clusters of $\mathcal{C}$ are ε-covered, then there exists an assignment of points to centers, such that Min-Sum clustering cost of the resulting clustering is at most $(1+\varepsilon) \cdot \mathsf{OPT}$.

Preprocessing

The first lemma allows us to assume that all points are in some sense close to each other.

▶ **Lemma 19.** *Suppose $n > 20$. Given an set of n points $A \subset \mathbb{R}^d$, we can partition a point set into subsets $A_1, \ldots A_k$, such that $\|p-q\|^2 \leq n^{10} \cdot \mathsf{OPT}$ for any two points $p, q \in A_i$ and such that any cluster C_j is fully contained in one of the A_i. The partitioning takes time $\tilde{O}(nd + k^2)$.*

Computing a Mean-Seeding Set via D^2 Sampling

We now consider a slight modification of Lemma 17 to account for sampling points from a cluster non-uniformly. We introduce the notion of a distorted core as follows. Given a cluster C_j, a set of centers M, and parameters α, β, we say that a subset of $C_j^\beta \cup M$ is a (C_j, β, α, M)-distorted core (denoted $core(C_j, \beta, \alpha, M)$) iff it is the image of a mapping $\pi_{\alpha,M} : C_j^\beta \to C_j^\beta \cup M$ such that for any point $p \in C_j^\beta$, we have

$$\pi_{\alpha,M}(p) = \begin{cases} p & \text{if } \min_{m \in M} \|p-m\|^2 \geq \alpha \cdot \Delta_j \\ \operatorname{argmin}_{m \in M} \|p-m\|^2 & \text{if } \min_{m \in M} \|p-m\|^2 < \alpha \cdot \Delta_j \end{cases}.$$

We use $D(C_j, \beta, \alpha, M)$ to denote the set of points in C_j^β such that $\min_{m \in M} \|p-m\|^2 < \alpha \cdot \Delta_j$.

The following lemmas relate the goodness of a mean computed on an α-distorted core to the mean on the entire set of points when sampling points proportionate to squared distances. We start by proving an analogue of Lemma 15.

▶ **Lemma 20.** *Let $\alpha \leq \frac{\varepsilon}{4}$ and let $\beta \geq \frac{144}{\varepsilon^2}$. Given a set of centers M and a cluster C_j, let $\hat{\mu_j} = \frac{1}{|C_j^\beta|} \sum_{p \in C_j^\beta} \pi_{\alpha,M}(p)$. Then, $\|\hat{\mu_j} - \mu_j\|^2 \leq \varepsilon \cdot \Delta_j$.*

We now characterize when M either covers a cluster C_j, or when M is a suitable seeding set for C_j. The following lemma says that if M is not a seeding set of C_j, then there exist many points in the core C_j^β of C_j that are far from M.

▶ **Lemma 21.** *Given $\alpha \leq \frac{\varepsilon}{16}$, $\beta \geq \frac{2400}{\varepsilon^2}$, and $\gamma \leq \sqrt{\frac{\varepsilon}{16(\beta+\alpha)}}$, and a set of centers M, let C_j be a cluster for which $|D(C_j, \beta, \alpha, M)| \geq (1-\gamma) \cdot |C_j^\beta|$. Then M is an (ε, β)-mean seeding set of C_j.*

Next, we show that the marginal probability of picking a point from an uncovered cluster C_j cannot be significantly smaller than the marginal probability of picking a point from the union of covered clusters with larger cardinality than C_j.

▶ **Lemma 22.** *Let M be a set of centers, and let $\mathcal{C}$ denote a set of clusters that are ε-covered by M. Let $\mathcal{H}$ denote the set of points in all the clusters in $\mathcal{C}$. Let $\beta > \frac{2400}{\varepsilon^2}$. Consider a cluster $C_j \notin \mathcal{C}$. Let i be the largest index such that $C_i \in \mathcal{C}$. Suppose that M is not an (ε, β)-mean seeding set of C_j, and that $i < j$. Then $\mathbb{P}[p \in C_j^\beta \mid p \in \mathcal{H} \cup C_j] \geq \frac{\varepsilon^4 \cdot \beta^{-3/2}}{1088k}$.*

We now consider a cluster C_j that is small compared to the union of the clusters C_j' with $j' > j$. In this case, we show that one of the distance-proportional distributions that we use guarantees that the probability of sampling points from the core of C_j is large.

▶ **Lemma 23.** *Let M be a set of centers. Let $\beta > \frac{2400}{\varepsilon^2}$. Let j be the smallest index such that C_j is not ε-covered by M. If M is not an (ε, β)-mean seeding set for C_j, then there exists $i \in \{0, 1, \ldots, \eta \log(nk/\varepsilon)\}$ such that $C_j^\beta \in A_{i,M}$ and $\mathbb{P}_i[p \in C_j^\beta] \geq \frac{1}{4352 \cdot k} \cdot \left(\frac{\varepsilon}{\beta^{5/8}}\right)^4$*

Finally, we show how to account for the sampling bias when estimating the means.

▶ **Lemma 24.** *Let M be a set of centers. Let j be the smallest index such that C_j is not ε-covered by M. Suppose that M is not an $(\varepsilon/4, \beta)$-mean seeding set for C_j. Consider a set of points S' sampled iid from $\mathbb{P}_i$, and let $S = S' \cap C_j^\beta$. If $\beta \geq 2400\varepsilon^{-2}$ and $S > 17825792 \cdot k \left(\frac{\beta^{7/12}}{\varepsilon}\right)^6 \log(2/\delta)$, then with probability at least $1-\delta$, we have that $S' \cup M$ is an $(\varepsilon/4, \beta)$-mean seeding set of C_j.*

The proof of Lemma 18 now argues that, given a set of points M, that we can find a seeding set for the largest uncovered cluster via D^2 sampling with respect to a suitable distribution $\mathbb{P}_i$. The existence of such a distribution, as well as the number of samples is given via Lemma 24 and extracting a suitable mean can be done via Lemma 16. The overall number of candidate solutions is now exponential in the number of points sampled over the course of the procedure, which is bounded by $\text{poly}(k, \varepsilon^{-1}, \log \delta^{-1})$.

Obtaining the Parameterized PTAS

We complete this section by explaining how to funnel the mean-seeding procedure into a PTAS. This yields Theorem 4. For every candidate solution given by Lemma 18, we bucket points by squared distance to the respective centers. This number of buckets depends on a

guess of the optimum, which is inexpensive to obtain, as well as a discretization over possible different distances to centers, of which we show there cannot exist too many. Since the number of buckets are very small, we can efficiently find an assignment. Then we show that we can extract a clustering from the bucketed assignment in linear time.

4 Learning-Augmented ℓ_2^2 Min-Sum k-Clustering

In this section, we describe and analyze our learning-augmented algorithm, corresponding to Theorem 5. We first formally define the precision and recall guarantees of a label predictor.

▶ **Definition 25** (Label predictor). *Suppose that there is an oracle that produces a label $i \in [k]$ for each $x \in X$, so that the labeling partitions $X = P_1 \dot\cup \ldots \dot\cup P_k$ into k clusters $P_1, \ldots, P_k$, where all points in P_i have the same label $i \in [k]$. We say the oracle is a* label predictor *with error rate α if there exists some fixed optimal min-sum clustering $P_1^*, \ldots, P_k^*$ such that for all $i \in [k]$, $|P_i \cap P_i^*| \geq (1-\alpha)\max(|P_i|, |P_i^*|)$. We say that $P^* = \{P_1^*, \ldots, P_k^*\}$ is the clustering consistent with the label oracle.*

We also recall the following guarantees of previous work on learning-augmented k-means clustering for a label predictor with error rate $\alpha \in \left[0, \frac{1}{2}\right)$.

▶ **Theorem 26** ([61]). *Given a label predictor with error rate $\alpha < \frac{1}{2}$ consistent with some clustering $P^* = \{P_1^*, \ldots, P_k^*\}$ with centers $\{c_1^*, \ldots, c_k^*\}$, there exists a polynomial-time algorithm* LearnedCenters *that outputs a set of centers $\{c_1, \ldots, c_k\}$, so that for each $i \in [k]$, $\sum_{x \in P_i^*} \|x - c_i\|_2^2 \leq (1 + \gamma_\alpha \alpha) \sum_{x \in P_i^*} \|x - c_i^*\|_2^2$, where $\gamma_\alpha = 7.7$ for $\alpha \in \left[0, \frac{1}{7}\right)$ or $\gamma_\alpha = \frac{5\alpha - 2\alpha^2}{(1-2\alpha)(1-\alpha)}$ for $\alpha \in \left[0, \frac{1}{2}\right)$.*

Unfortunately, although the centers $\{c_1, \ldots, c_k\}$ returned by LearnedCenters are good centers for the clustering induced by a near-optimal ℓ_2^2 min-cost k-clustering, it is not clear what the resulting assignment should be. In fact, we emphasize that unlike k-means clustering, the optimal ℓ_2^2 min-cost k-clustering may not assign each point to its closest center.

Constrained min-cost flow. To that end, we now create a constrained min-cost flow problem as follows. We first create a source node s and a sink node t and require that $n = |X|$ flow must be pushed from s to t. We create a node u_x for each point $x \in X$ and create a directed edge from s to each node u_x with capacity 1 and cost 0. There are no more outgoing edges from s or incoming edges to each u_x. This ensures that to achieve n flow from s to t, a unit of flow must be pushed across each node u_x.

Algorithm 1 Learning-augmented min-sum k-clustering.

Input: Dataset X with partition $P_1, \ldots, P_k$ induced by label predictor with error rate α
Output: Labels for all points consistent with a $(1 + \mathcal{O}(\alpha))$-optimal min-sum k-clustering

1: Let $c_1, \ldots, c_k$ be the output centers of LearnedCenters on $P_1, \ldots, P_k$
2: Create a min-cost flow problem $\mathcal{F}$ with required flow n
3: Solve the flow problem $\mathcal{F}$
4: For each $x \in X$, let the flow from u_x be sent to the node v_{ℓ_x}, so that $\ell_x \in [k]$
5: Label x with ℓ_x

We then adjust an integrality theorem to handle capacitated edges, thereby showing that the resulting solution for the min-cost flow problem is integral, and show that since the constraint matrix is totally unimodular, i.e., all submatrices have determinant -1, 0, or 1, then a valid clustering can be recovered by using the output of a linear program solver. Thus, we have the following guarantees for our learning-augmented algorithm.

▶ **Theorem 27.** *There exists a polynomial-time algorithm that uses a label predictor with error rate α and outputs a $\frac{1+\gamma_\alpha \alpha}{(1-\alpha)^2}$-approximation to min-sum k-clustering, where γ_α is the fixed constant from Theorem 26.*

References

1 Anders Aamand, Justin Y. Chen, and Piotr Indyk. (optimal) online bipartite matching with degree information. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2022.

2 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009. doi:10.1007/S10994-009-5103-0.

3 Enver Aman, Karthik C. S., and Sharath Punna. On connections between k-coloring and Euclidean k-means. In *32nd Annual European Symposium on Algorithms, ESA 2024*, 2024. To appear.

4 Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. Online algorithms with multiple predictions. In *International Conference on Machine Learning, ICML*, pages 582–598, 2022. URL: https://proceedings.mlr.press/v162/anand22a.html.

5 Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM Trans. Algorithms*, 19(2):19:1–19:34, 2023. doi:10.1145/3582689.

6 David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

7 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory Comput.*, 7(1):27–43, 2011. doi:10.4086/TOC.2011.V007A003.

8 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 35–66, 2022. doi:10.1137/1.9781611977073.3.

9 Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

10 Sandip Banerjee, Rafail Ostrovsky, and Yuval Rabani. Min-sum clustering (with outliers). In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 16:1–16:16, 2021. doi:10.4230/LIPICS.APPROX/RANDOM.2021.16.

11 Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing*, pages 11–20, 2001. doi:10.1145/380752.380754.

12 Babak Behsaz, Zachary Friggstad, Mohammad R. Salavatipour, and Rohit Sivakumar. Approximation algorithms for min-sum k-clustering and balanced k-median. *Algorithmica*, 81(3):1006–1030, 2019. doi:10.1007/S00453-018-0454-1.

13 Justin Y. Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P. Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

14 Justin Y. Chen, Piotr Indyk, and Tal Wagner. Streaming algorithms for support-aware histograms. In *International Conference on Machine Learning, ICML*, pages 3184–3203, 2022. URL: https://proceedings.mlr.press/v162/chen22g.html.

15 Justin Y. Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning, ICML*, pages 3583–3602, 2022. URL: `https://proceedings.mlr.press/v162/chen22v.html`.

16 Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *J. ACM*, 68(1):3:1–3:39, 2021. `doi:10.1145/3424305`.

17 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 519–539, 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 *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2635–2648, 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 *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1493–1530, 2022. `doi:10.1137/1.9781611977073.63`.

20 Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, Chris Schwiegelshohn, and Omar Ali Sheikh-Omar. Improved coresets for euclidean k-means. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2022.

21 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 169–182, 2021. `doi:10.1145/3406325.3451022`.

22 Artur Czumaj and Christian Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Struct. Algorithms*, 30(1-2):226–256, 2007. `doi:10.1002/RSA.20157`.

23 Sami Davies, Benjamin Moseley, Sergei Vassilvitskii, and Yuyan Wang. Predictive flows for faster ford-fulkerson. In *International Conference on Machine Learning, ICML*, volume 202, pages 7231–7248, 2023. URL: `https://proceedings.mlr.press/v202/davies23b.html`.

24 Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 50–58, 2003. `doi:10.1145/780542.780550`.

25 Wenceslas Fernandez de la Vega and Claire Kenyon. A randomized approximation scheme for metric MAX-CUT. *J. Comput. Syst. Sci.*, 63(4):531–541, 2001. `doi:10.1006/JCSS.2001.1772`.

26 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 10393–10406, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html`.

27 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005. `doi:10.1137/S0097539704443057`.

28 Jon C. Ergun, Zhili Feng, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Learning-augmented k-means clustering. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

29 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. `doi:10.1145/285055.285059`.

30 Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 2319–2327, 2019. URL: `http://proceedings.mlr.press/v97/gollapudi19a.html`.

31 Elena Grigorescu, Young-San Lin, Sandeep Silwal, Maoyuan Song, and Samson Zhou. Learning-augmented algorithms for online linear and semidefinite programming. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2022.

32 Venkatesan Guruswami and Piotr Indyk. Embeddings and non-approximability of geometric problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 537–538, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644198.

33 Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for min-sum p-clustering. *Discret. Appl. Math.*, 89(1-3):125–142, 1998. doi:10.1016/S0166-218X(98)00100-0.

34 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR*, 2019.

35 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 2733–2743, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/161c5c5ad51fcc884157890511b3c8b0-Abstract.html.

36 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*, pages 332–339, 1994. doi:10.1145/177424.178042.

37 Piotr Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *40th Annual Symposium on Foundations of Computer Science, FOCS*, pages 154–159, 1999. doi:10.1109/SFFCS.1999.814587.

38 Piotr Indyk, Ali Vakilian, and Yang Yuan. Learning-based low-rank approximations. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, pages 7400–7410, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/1625abb8e458a79765c62009235e9d5b-Abstract.html.

39 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.

40 Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Chaining, group leverage score overestimates, and fast spectral hypergraph sparsification. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 196–206. ACM, 2023. doi:10.1145/3564246.3585136.

41 Shaofeng H.-C. Jiang, Erzhi Liu, You Lyu, Zhihao Gavin Tang, and Yubo Zhang. Online facility location with predictions. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

42 Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 823–832, 2021. doi:10.1145/3406325.3451058.

43 Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P. Woodruff. Learning-augmented data stream algorithms. In *8th International Conference on Learning Representations, ICLR*, 2020.

44 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984. doi:10.1007/BF02579150.

45 Misha Khodak, Maria-Florina Balcan, Ameet Talwalkar, and Sergei Vassilvitskii. Learning predictions for algorithms with predictions. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2022.

46 Subhash Khot. Hardness results for coloring 3 -colorable 3 -uniform hypergraphs. In *43rd Symposium on Foundations of Computer Science (FOCS), Proceedings*, pages 23–32, 2002. doi:10.1109/SFCS.2002.1181879.

47 Jon M. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS*, pages 446–453, 2002. URL: https://proceedings.neurips.cc/paper/2002/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html.

48 Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference*, pages 489–504, 2018. doi:10.1145/3183713.3196909.

49 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1859–1877, 2020. doi:10.1137/1.9781611975994.114.

50 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.

51 James R. Lee. Spectral hypergraph sparsification via chaining. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 207–218. ACM, 2023. doi:10.1145/3564246.3585165.

52 Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pages 230–249, 2015. doi:10.1109/FOCS.2015.23.

53 Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory, COLT*, pages 2140–2157, 2019. URL: http://proceedings.mlr.press/v99/lee19a.html.

54 Yi Li, Honghao Lin, Simin Liu, Ali Vakilian, and David P. Woodruff. Learning the positions in countsketch. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

55 Honghao Lin, Tian Luo, and David P. Woodruff. Learning augmented binary search trees. In *International Conference on Machine Learning, ICML*, pages 13431–13440, 2022. URL: https://proceedings.mlr.press/v162/lin22f.html.

56 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021. doi:10.1145/3447579.

57 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.

58 Jirí Matousek. On approximate geometric k-clustering. *Discret. Comput. Geom.*, 24(1):61–84, 2000. doi:10.1007/S004540010019.

59 Michael Mitzenmacher. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 462–471, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/0f49c89d1e7298bb9930789c8ed59d48-Abstract.html.

60 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. doi:10.1017/9781108637435.037.

61 Thy Dinh Nguyen, Anamay Chaturvedi, and Huy L. Nguyen. Improved learning-augmented algorithms for k-means and k-medians clustering. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

62 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS*, pages 9684–9693, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

63 Leonard J. Schulman. Clustering for edge-cost minimization (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 547–555, 2000. doi:10.1145/335305.335373.

64 Yongho Shin, Changyeol Lee, Gukryeol Lee, and Hyung-Chan An. Improved learning-augmented algorithms for the multi-option ski rental problem via best-possible competitive analysis. In *International Conference on Machine Learning, ICML*, pages 31539–31561, 2023. URL: https://proceedings.mlr.press/v202/shin23c.html.

65 Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2014.

66 Pravin M. Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science*, pages 332–337, 1989.

67 Pravin M. Vaidya. An algorithm for linear programming which requires o(((m+n)n^2 + (m+n)$^{1.5}$n)l) arithmetic operations. *Math. Program.*, 47:175–201, 1990. doi:10.1007/BF01580859.

68 Shufan Wang, Jian Li, and Shiqiang Wang. Online algorithms for multi-shop ski rental with machine learned advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

69 Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

70 Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005. doi:10.1109/TNN.2005.845141.

The Maximum Clique Problem in a Disk Graph Made Easy

J. Mark Keil
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada

Debajyoti Mondal[1]
Department of Computer Science, University of Saskatchewan, Saskatoon, Canada

Abstract

A disk graph is an intersection graph of disks in $\mathbb{R}^2$. Determining the computational complexity of finding a maximum clique in a disk graph is a long-standing open problem. In 1990, Clark, Colbourn, and Johnson gave a polynomial-time algorithm for computing a maximum clique in a unit disk graph. However, finding a maximum clique when disks are of arbitrary size is widely believed to be a challenging open problem. In this paper, we provide a new perspective to examine adjacencies in a disk graph that helps obtain the following results.

- We design an $\mathcal{O}^*(n^{2k})$-time algorithm, where $\mathcal{O}^*$ hides a polynomial factor, to find a maximum clique in a n-vertex disk graph with k different sizes of radii. This is polynomial for every fixed k, and thus settles the open question for the case when $k = 2$.
- Given a set of n unit disks, we show how to compute a maximum clique inside each possible axis-aligned rectangle determined by the disk centers in $O(n^5 \log n)$-time. This is at least a factor of $n^{4/3}$ faster than applying the fastest known algorithm for finding a maximum clique in a unit disk graph for each rectangle independently.
- We give an $\mathcal{O}^*(n^{2rk})$-time algorithm to find a maximum clique in a n-vertex ball graph with k different sizes of radii where the ball centers lie on r parallel planes. This is polynomial for every fixed k and r, and thus contrasts the previously known NP-hardness result for finding a maximum clique in an arbitrary ball graph.
- We design an $\mathcal{O}^*(n^{2k})$-time algorithm to find a maximum clique in the intersection graph of a set S of n L-visible convex polygons, where k is the number of distinct shapes in S. This contrasts the known hardness result on finding a maximum clique in the intersection graph of unit disks and axis-aligned rectangles.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Geometric Intersection Graphs, Disk Graphs, Ball Graphs, Maximum Clique

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.63

Related Version *Full Version*: https://arxiv.org/abs/2404.03751 [32]

Funding The work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

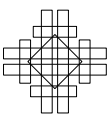
Acknowledgements We thank Soichiro Yamazaki for useful feedback.

1 Introduction

A *geometric intersection graph* consists of a set of geometric objects as vertices and a set of edges that are determined by the intersection of these objects, i.e., two vertices are considered adjacent if and only if the corresponding objects intersect. Geometric objects of different shapes and their intersection graphs are often used to model applied contexts,

[1] Corresponding author

© J. Mark Keil and Debajyoti Mondal;
licensed under Creative Commons License CC-BY 4.0



e.g., unit disks for problems in wireless networks [24, 28], line segments (time intervals) for task scheduling [34], trapezoids for channel routing in VLSI design [23], etc. Many NP-hard graph problems are known to admit polynomial-time solutions on various types of geometric intersection graphs (see e.g., [8, 16]). There also exist cases where a graph problem is known to be NP-hard, but its time complexity status is open for some intersection graph class (see e.g. the open problems posed in [2, 1, 5, 33]). In this paper, we restrict our attention to one such scenario, where the problem of interest is finding a *maximum clique*, i.e., a maximum subset of pairwise adjacent vertices, and the intersection graph we examine is a *disk graph*. Here, a *disk graph* G is an intersection graph of disks in $\mathbb{R}^2$, where each vertex of G corresponds to a disk and two vertices are adjacent in G if and only if their disks intersect.

The maximum clique problem is NP-hard for many well-known intersection graph classes such as intersection graph of rays [13], grounded strings [33], triangles [3], ellipses [3], and a combination of axis-aligned rectangles and unit disks [7, 21], whereas polynomial-time solvable for circle graphs [36], trapezoid graphs [23], circle trapezoid graphs [23], unit disk graphs [16], axis-aligned rectangle intersection graphs [29], and so on. However, determining the time complexity of computing a maximum clique in a disk graph is a long-standing open question [3, 4, 24]. In a seminal paper published in 1990, Clark, Colbourn, and Johnson [16] gave a polynomial-time algorithm for the case of unit disk graphs. However, the time-complexity question for general disk graphs has remained open since then. Although this was not posed as an open problem in [16], it has long been known to be a challenging problem [4], even before it was explicitly posed as an open problem (e.g., by Fishkin [24], Ambühl and Wagner [3] and Cabello [11, 12]). In fact, the problem has been emphasized in various ways in the literature, e.g., as "an intriguing open question", as "a notorious open question in computational geometry" [5], as being "elusive with no new positive or negative results" [4], and several times as a "long-standing open problem" [21, 25, 27].

Although no polynomial-time exact solution is known for finding a maximum clique in general disk graphs, a 2-approximate solution [3] can be obtained by guessing all stabbing sets of four points. There always exists a stabbing set of at most four points for every set of disks that forms a clique [17, 39, 14]. Therefore, one can take the largest solution obtained from all guesses. However, obtaining a better approximation ratio appears to be a challenge. Cabello [11, 12] even asked whether a 1.99 approximation can be achieved if we restrict the input to disks that have at most two different sizes of radii.

Several recent research has shown interesting progress on various fronts of the problem. The $O(n^{4.5})$-time algorithm for finding a maximum clique in a unit disk graph [16], which was improved first to $O(n^{3.5}\log n)$-time [9] and then to $O(n^3\log n)$-time [19], has been improved further in SoCG'23 to $O(n^{2.5}\log n)$ [21]. Chan [15] observed that the running time of [21] can be expressed as $O(n \cdot h(m)^{1+o(1)})$, where $h(m)$ is the time to compute a maximum matching in a bipartite graph that has a biclique cover[2] of size m, and m in this context is known to be $O(n^{4/3}\text{polylog}(n))$ [31]. One can leverage such a biclique cover to compute a maximum matching in $O(m^{1+o(1)})$ time [22]. Therefore, the overall running time becomes $O(n^{7/3+o(1)})$. In FOCS'18, the authors in [6] gave a QPTAS, a randomized EPTAS (and thus a fixed-parameter-tractable algorithm [7]), and a subexponential algorithm for computing a maximum clique in a disk graph. They also showed how to extend these results for *unit ball graphs*, which are intersection graphs of 3-dimensional balls of unit radius. In SODA'22, the authors in [35] designed further subexponential-time fixed-parameter-tractable algorithms

[2] A *biclique cover* of a bipartite graph G is a set of complete bipartite subgraphs of G that cover the edges of G. The size of the cover is the number of elements in the set.

for many other fundamental graph problems on disk graphs. While the maximum clique problem is open for disk graphs with arbitrary radii, it is NP-hard for ball graphs [6]. The hardness result holds in a very restricted setting with all radii falling within the interval $[1, 1+\epsilon]$, $\epsilon > 0$, where even a subexponential-time approximation scheme is unlikely unless the exponential-time-hypothesis [30] fails.

A rich body of research examined the maximum clique problem in the intersection graph of various types of convex shapes. A t-directional convex polygon is a polygon with sides parallel to one of the prespecified t directions. A t-directional convex graph is an intersection graph of t-directional polygons. The paper in [10] showed that a maximum clique in k-directional convex graphs can be solved in time $O(n^{k+3})$ time, and given a geometric representation, this can be improved to $O(n^{k+1})$. The maximum-clique problem can be solved in polynomial time for the intersection graph of unit disks and 2-pancakes [26], where a 2-pancake is defined using a segment on the x-axis. Specifically, a 2-pancake is the Minkowski sum of the unit disk centered at $(0, 0)$ and the line segment with endpoints $(x_1, 0)$ and $(x_2, 0)$, i.e., the union of all unit disks with center on the line segment x_1x_2. Bonnet, Grelier, and Miltzow [7] extends the polynomial-time algorithm for unit disks [16] to translates of any fixed convex set. This algorithm does not generalize any further, i.e., no polynomial-time algorithm is known for the translates of two fixed convex sets.

Our Contribution

We show that a maximum clique in a unit disk graph can be obtained by examining *slabs* (i.e., regions that are bounded by two parallel lines), whereas all prior algorithms depend on a more constrained lens-shaped region. We show how our new slab-based idea can be extended to compute a maximum clique in polynomial time when the number of different radii sizes is fixed, and even to address the case of ball graphs in some restricted settings. Specifically, we obtain the following results, where in all cases, we assume that an arrangement of disks, i.e., a disk representation of the disk graph, is given as an input.

Disk graph with k Different Sizes of Radii. We give an $\mathcal{O}^*(n^{2k})$-time algorithm, where $\mathcal{O}^*$ hides a polynomial factor, to find a maximum clique in a disk graph where k is the number of different types of radii. For every fixed k, the running time becomes polynomial. This settles the open question posed by Cabello [11, 12] on whether a polynomial-time algorithm exists when $k = 2$.

Range Query in a Unit disk graph. We show that our slab-based idea can help the range query version of the maximum clique problem, where the input unit disks should be preprocessed such that several queries that may appear at a later time can be answered efficiently. We examine the case of *axis-aligned rectangular query*, where a maximum clique needs to be reported over disks with centers in the query rectangle. A natural approach for handling rectangular range queries is to precompute the maximum cliques for all possible axis-aligned rectangles (determined by the given disk centers) such that given a query R, a solution can be obtained first by identifying the smallest enclosing rectangle R' for the disk centers in R, and then by a table look-up using R'. Applying an existing lens-based algorithm on all $O(n^4)$ possible axis-aligned rectangles determined by the input disk centers takes $O(n^{19/3+o(1)})$ time. In contrast, we can use our slab-based idea to precompute all such solutions in $O(n^5 \log n)$, achieving a speed up by at least a factor of $n^{4/3}$.

Ball graphs with k Different Sizes of Radii. Given a set of n balls in the Euclidean plane with k different sizes of radii where the ball centers lie on r parallel planes, we show how to compute a maximum clique in the corresponding ball graph in $\mathcal{O}^*(n^{2rk})$ time. For fixed k and r, the running time becomes polynomial in n. We then show that the restriction of the planes being parallel can be removed as long as the input planes are all perpendicular to a different plane. This result contrasts the known NP-hardness result [5] for finding a maximum clique in a ball graph where the radii of the balls are very close to each other, i.e., in the interval $[1, 1+\varepsilon]$, where $\varepsilon > 0$.

Intersection graphs of L-visible convex polygons. Given a set S of n convex polygons that are L-visible, we show how to find a maximum clique in their intersection graph in $\mathcal{O}^*(n^{2k})$-time, where k is the number of distinct shapes in S. This strengthens a result in [7]. Here two points p, q in a polygon are called *L-visible* if the axis parallel rectangle with diagonal pq is contained in the polygon. We refer to a polygon P as a *L-visible polygon* if it contains a point from which every point of the polygon is L-visible. For example, a disk and a regular hexagon with one side parallel to x-axis are L-visible, whereas a triangle is not necessarily an L-visible polygon. Our result implies that the maximum clique can be computed in polynomial time in the intersection graph of the translates of a unit disk and a set of fixed axis-aligned rectangles. In contrast, if the number of rectangle shapes is not bounded then the problem is known to be NP-hard, e.g., consider the case of unit disks and axis-aligned rectangles [7, 21].

2 Preliminaries

In this section we discuss some notation and preliminary results.

By $D_{a,r}$, we denote a disk with a center at point a and radius r. Given a pair of disks $D_{a,r}$ and $D_{b,r'}$, we refer to their common intersection region as a *lens* (Figure 1(a)). Let G be a disk graph. For simplicity, sometimes we say a pair of disks are adjacent in G, which means that the disks intersect and their corresponding vertices are adjacent in G. By $B_{a,r}$, we denote a ball with a center at point a and radius r.

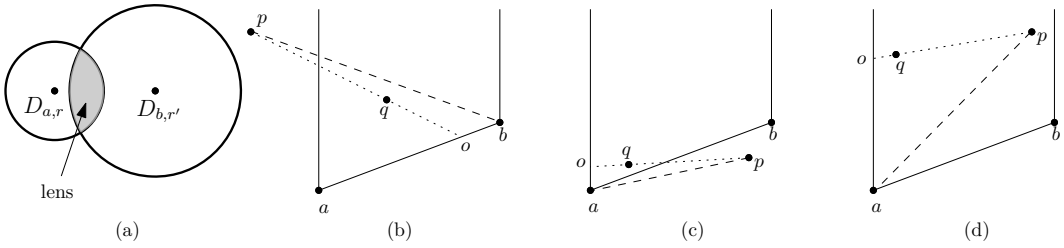


Figure 1 Illustration for (a) a lens, (b)–(d) Illustration for Lemma 1.

For a pair of points a and b, we denote the line passing through them as ℓ_{ab}. By ab and $|ab|$ we denote the line segment with endpoints a, b and the Euclidean distance between a and b, respectively. By ℓ_p^h and ℓ_p^v we denote the horizontal and vertical lines through a point p, respectively. By an *upper slab* U_{ab} of ab, we denote the region above ab, which is bounded by the lines ℓ_a^v and ℓ_b^v. Similarly, by a *lower slab* $\overline{U}_{ab}$ we denote the region below ab, which is bounded by ℓ_a^v and ℓ_b^v.

▶ **Lemma 1.** *Let ab be a line segment and let U_{ab} be the upper slab of ab. Let q be a point in U_{ab} and let p be a point (not necessarily in U_{ab}) with a y-coordinate equal to or larger than the y-coordinate of q. Then $|pq| \leq \max\{|pa|, |pb|\}$.*

Proof. We distinguish two scenarios depending on whether ab is vertical or not.

Scenario 1 (The x-coordinates of a and b are different): Without loss of generality assume that a has a strictly smaller x-coordinate than that of b. Since the y coordinate of p is equal to or larger than the y-coordinate of q, the ray that starts at p and passes through q must hit the boundary of U_{ab}. Let o be the point of intersection when the ray exits U_{ab} (e.g., Figure 1(b)–(d)). It suffices to show that $|po| \leq \max\{|pa|, |pb|\}$.

Consider first the case when o lies on ab (e.g., Figure 1(b)). Let m be the point on line ℓ_{ab} that minimizes the distance $|pm|$, i.e., ℓ_{pm} is perpendicular to ℓ_{ab}. We now move o away from m along line ℓ_{ab} until it hits either a or b. Since we are moving o away from m, the length $|po|$ increases monotonically, and the largest length it can attain is $\max\{|pa|, |pb|\}$.

Consider now the case when o lies on the vertical sides of U_{ab} (e.g., Figure 1(c)–(d)). We move o downward along the side of U_{ab} until it hits either a or b. Since p has a higher y-coordinate than q, moving o downward increases the length $|po|$ monotonically, and the largest length it can attain is $\max\{|pa|, |pb|\}$.

Scenario 2 (The x-coordinates of a and b are the same): In this case, ab is a vertical segment. If a has a smaller y-coordinate than that of b, then $|pq| \leq |pa|$. Otherwise, $|pq| \leq |pb|$. ◀

3 Maximum Clique in Disk Graphs with k Different Radii

In this section we give an $\mathcal{O}^*(n^{2k})$-time algorithm to find a maximum clique in a disk graph where k is the number of different types of radii.

We first introduce some notation. Let $\mathcal{D}_k$ be a disk graph where the number of different types of radii is at most k. For convenience, we denote these different types of radii as $r_1, r_2, \ldots, r_k$, where for every $1 \leq i < j \leq k$, we have $r_i < r_j$. By a *type-i disk* we denote a disk of radius r_i. Let $\mathcal{C}$ be a maximum clique of $\mathcal{D}_k$. By $\mathcal{C}_i$ we denote a maximal clique in $\mathcal{C}$ where all disks are of type i.

The idea of the algorithm is as follows. We guess the number of different types of radii that may appear in a maximum clique and we take the maximum over all the solutions computed from these 2^k guesses. We now describe how a solution is computed for a particular guess for a set of radii that may appear in the maximum clique $\mathcal{C}$. For each disk type i, we guess two disk centers a_i, b_i from $\mathcal{C}_i$, where a_i is the leftmost (i.e., with the smallest x-coordinate) and b_i is the rightmost (i.e., with the largest x-coordinate) over all the disk centers in $\mathcal{C}_i$. Note that a_i may coincide with b_i if $\mathcal{C}_i$ contains only one disk, or if the centers of all disks in $\mathcal{C}_i$ are on a vertical line. Let Ψ be the set of these $2k$ disks. For every upper slab $U_{a_ib_i}$, we construct a set X_i by taking every disk that has its center in $U_{a_ib_i}$ and intersects all the disks of Ψ. We show that the union of these disks, i.e., $X = (X_1 \cup X_2 \cup \ldots \cup X_k)$, is a clique in $\mathcal{D}_k$. Similarly, for each lower slab $\overline{U}_{a_ib_i}$, we construct a set Y_i by taking every disk that has its center in $\overline{U}_{a_ib_i}$ and intersects all the disks of Ψ. We show that their union $Y = (Y_1 \cup Y_2 \cup \ldots \cup Y_k)$ is a clique in $\mathcal{D}_k$. Since the disks of Ψ are assumed to be in $\mathcal{C}$, it is straightforward to observe that $\mathcal{C}$ is a subset of the disks in $(\Psi \cup X \cup Y)$. Since the complement of the disk graph determined by the disks $(X \cup Y)$ is a bipartite graph H, we can compute the maximum clique $\mathcal{C}$ from a maximum bipartite matching in H. It now suffices to show that the disks in X (similarly, the disks in Y) are mutually adjacent.

▶ **Lemma 2.** *For every i, j, where $1 \leq i, j \leq k$, the disks in $X_i \cup X_j$ are mutually adjacent.*

Proof. Let D_{p,r_i} and D_{q,r_j} be two disks, where p belongs to X_i and q belongs to X_j. We now show that D_{p,r_i} and D_{q,r_j} must mutually intersect.

Consider first the case when $i = j$. Without loss of generality assume that the y-coordinate of p is larger than or equal to that of q. Then by Lemma 1, $|pq| \leq \max\{|pa_i|, |pb_i|\}$. By the construction of X, the disk D_{p,r_i} intersects D_{a_i,r_i} and D_{b_i,r_i}. We thus have $|pq| \leq \max\{|pa_i|, |pb_i|\} \leq 2r_i$, and since p and q correspond to type-i disks, they must intersect.

Consider now the case when $i \neq j$.

If the y-coordinate of p is larger than or equal to that of q, then we apply Lemma 1 to obtain $|pq| \leq \max\{|pa_j|, |pb_j|\}$. By the construction of X, the disk D_{p,r_i} intersects D_{a_j,r_j} and D_{b_j,r_j}. We thus have $|pq| \leq \max\{|pa_j|, |pb_j|\} \leq r_i + r_j$. Consequently, D_{p,r_i} and D_{q,r_j} mutually intersect.

If the y-coordinate of p is smaller than that of q, then we apply Lemma 1 by swapping the role of p and q, i.e., by using the condition that $p \in U_{a_i,b_i}$ whereas q has a larger y-coordinate than that of p. We thus have $|pq| \leq \max\{|qa_i|, |qb_i|\}$. By the construction of X, we have $\max\{|qa_i|, |qb_i|\} \leq r_i + r_j$. Consequently, D_{p,r_i} and D_{q,r_j} mutually intersect. ◀

We now consider the running time. There are at most $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} = O(n^2)$ ways to guess the pair of disks for a particular disk type, irrespective of whether these two disks are distinct or not. Therefore, the number of ways we can guess k pairs is $O(n^{2k})$. For each guess of k pairs, it is straightforward to construct the sets X and Y in $O(kn)$ time. Since the corresponding bipartite graph and its maximum matching can be computed in polynomial time, the overall running time becomes $\mathcal{O}^*(n^{2k})$.

The following theorem summarizes the result of this section.

▶ **Theorem 3.** *Given a set of n disks in the Euclidean plane with k different types of radii, a maximum clique in the corresponding disk graph can be computed in $\mathcal{O}^*(n^{2k})$ time.*

4 Rectangular Range Query over Unit Disks

In this section, we consider the problem of reporting a maximum clique given a rectangular range query over an arrangement of unit disks. To this end, we show how to compute a maximum clique for every axis-aligned rectangle (determined by the input disk centers) in $O(n^5 \log n)$ time, which is at least a factor of $n^{4/3}$ faster than applying the algorithm of [21] separately for each axis-aligned rectangle.

To better explain such advantages of our technique, we first give an overview of the existing algorithms to compute a maximum clique in a unit disk graph. The idea of Clark, Colbourn, and Johnson's $O(n^{4.5})$-time algorithm [16] is to guess the farthest pair of vertices (v, w) in the clique, and then to find a maximum clique C that contains both v and w. Since v and w are the farthest pair, the set S of vertices that are adjacent to both v, w must lie in a lens-shaped region, i.e., the common intersection region of the two disks centered at v and w with radius $|vw|$. They showed that the lens can be partitioned by the line segment vw into two halves where the disks with centers in the same half are pairwise intersecting. Consequently, the complement of the disk graph corresponding to S is a bipartite graph H, and the maximum clique C can be computed by finding a maximum independent set in H. The running time for the unit disk case has subsequently been improved to $O(n^{3.5} \log n)$ by using non-trivial data structures [9], then to $O(n^3 \log n)$ by examining the lenses in a particular order so that the solution for each lens does not need to be computed from scratch [19, 20], and finally, to $O(n \cdot h(m)^{1+o(1)})$ by a combination of divide-and-conquer and plane sweep approach using a circular arc [21, 15]. Here $h(m)$ is

the time to compute a maximum matching in H, i.e., the complement of the disk graph inside a lens, and m is the size of the biclique cover of H. Since $m \in O(n^{4/3}\text{polylog}(n))$ [31] and a maximum matching can be computed using a maximum-flow algorithm in $O(m^{1+o(1)})$ time [22], the running time becomes $O(n^{7/3+o(1)})$. Consider now the scenario of computing a maximum clique for every possible rectangle determined by the input disk centers. Since lenses examined by these algorithms are not necessarily axis-aligned, finding an order of the lenses to compute solutions for the rectangles by dynamic point insertion and deletion appears to be challenging. A straightforward approach that solves each rectangular region independently takes $O(n^{6.33+o(1)})$ time.

The idea of our algorithm is to maintain solutions for all possible slabs where one can insert and delete points as necessary to find the solutions for all possible rectangles within the slab. For convenience, we assume that the centers of the disks are in general position, i.e., no two centers have the same y-coordinates. We now describe the details.

Consider first an alternative (slower) approach that uses slabs to compute a maximum clique that guesses the leftmost and rightmost points a, b of a maximum clique. For every such guess, one can find a maximum clique by considering the set S of disks that are adjacent to both a, b and have their centers in the slab bounded by ℓ_a^v and ℓ_b^v. By Lemma 1, the disks of S that have their centers in U_{ab} (similarly, in $\overline{U}_{a,b}$) are mutually adjacent. Consequently, the complement of the disk graph underlying S is a bipartite graph H, and a maximum clique can be computed by finding a maximum independent set in H.

We now consider maintaining the solutions for all possible slabs under point insertion and deletion. Initially, the solution for each slab is set to null. We now sweep the plane upward with two horizontal lines ℓ^b and ℓ^u in two phases.

In the first phase, ℓ^b is placed so that it passes through the disk center with the lowest y-coordinate, and ℓ^u is used to sweep the plane upward starting at ℓ_b. Each time ℓ^u moves to a new point, a point is inserted into the corresponding slabs and their solutions are updated. Every time the horizontal slab between ℓ^b and ℓ^u contains the guessed points of a slab (i.e., the two points determining a slab), we obtain a solution for a new rectangular region, and the corresponding solution is stored in a table. It is straightforward to observe that each insertion changes $O(n^2)$ slabs, and hence we have $O(n^3)$ insertion operations in total.

In the next phase, we sweep the plane upward using ℓ^b. Each time ℓ^b moves to a new point, a point is deleted from the corresponding rectangular regions we considered in the first phase. A deletion changes $O(n^3)$ previous solutions, i.e., the solutions corresponding to $O(n^2)$ slabs and $O(n)$ regions in each slab determined by the positions of ℓ_b. Therefore, we have $O(n^4)$ deletion operations in total.

Each insertion and deletion operation updates existing lenses, each of which is determined by a pair of guessed points. It is known that such updates can be done in $O(n \log n)$ time by performing an alternating path search to update the maximum independent set in the bipartite graph determined by the complement of the disk graph inside the lens [20, 19]. For $O(n^4)$ update operations, the running time becomes $O(n^5 \log n)$. By $S[p_\ell, p_r, p_t, p_b]$ we denote the size of a maximum clique inside a rectangle R with its left, right, top and bottom sides being determined by the lines through the disk centers p_ℓ, p_r, p_t, p_b, respectively, where $|p_\ell, p_r|$ is at most two units and p_ℓ, p_r appear on the left and right sides of R. If $|p_\ell, p_r|$ is larger than two units, then $S[p_\ell, p_r, p_t, p_b]$ is 0.

We now compute a maximum clique for each axis-aligned rectangle determined by the input disk centers. We first compute two sorted orders of the disk centers, one based on increasing x-coordinates and the other based on increasing y-coordinates. We now use a dynamic programming approach, where the base cases are the entries of S, and the $O(n^4)$

rectangles that do not contain any disk center in their proper interior. The latter can be computed and stored in a table in $O(n^4)$ time. Let $M[p_\ell, p_r, p_t, p_b]$ be the maximum clique of a rectangle R with its left, right, top and bottom sides being determined by the lines through p_ℓ, p_r, p_t, p_b, respectively. Note that the disk centers of the maximum clique may not appear on the sides of R. Let p'_ℓ and p'_r be the disk centers immediately after and before p_ℓ and p_r, respectively, which can be determined from the precomputed sorted order. Similarly, p'_t and p'_b are the disk centers immediately before and after p_t and p_b, respectively. We now can find a solution using $O(1)$ table look-ups as follows.

$$M[p_\ell, p_r, p_t, p_b] = \begin{cases} \max\{S[p_\ell, p_r, p_t, p_b], M[p'_\ell, p_r, p_t, p_b], \\ \qquad M[p_\ell, p'_r, p_t, p_b], M[p_\ell, p_r, p'_t, p_b], \\ \qquad M[p_\ell, p_r, p_t, p'_b]\}, & \text{if } p_\ell \text{ and } p_r \text{ appear on } R \\ \max\{M[p'_\ell, p_r, p_t, p_b], M[p_\ell, p'_r, p_t, p_b], \\ \qquad M[p_\ell, p_r, p'_t, p_b], M[p_\ell, p_r, p_t, p'_b]\}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

Since there are $O(n^4)$ cells in the table and each entry is computed by $O(1)$ table look-ups which are determined in $O(\log n)$ time, the overall running time remains $O(n^5 \log n)$.

▶ **Theorem 4.** *Given a set of n unit disks in the Euclidean plane, a maximum clique for every possible axis-aligned rectangle determined by the given disk centers can be computed in $O(n^5 \log n)$ time.*

5 Maximum Clique in Ball Graphs with k Different Radii

We now give an $\mathcal{O}^*(n^{2rk})$-time algorithm for computing a maximum clique in a ball graph, where the centers of the balls are contained in r planes which are all perpendicular to a different plane (Section 5.2). For simplicity, we first consider a scenario when the input planes are parallel to each other (Section 5.1).

5.1 Ball Centers are on r Parallel Planes

Without loss of generality, we assume that the r input planes are parallel to the xy-plane. We first show that a version of Lemma 1 holds in three dimensions, as follows.

▶ **Lemma 5.** *Let ab be a line segment on a plane L, where L is parallel to the xy-plane, and let U_{ab} be the upper slab of ab on L. Let q be a point in U_{ab}. Let p be a point (not necessarily on L) with a y-coordinate equal to or larger than the y-coordinate of q. Then $|pq| \leq \max\{|pa|, |pb|\}$.*

Proof. Let p' be the projection of p on L (Figure 2(a)). Then the y-coordinate of p' is equal to or larger than the y-coordinate of q. By Lemma 1, $|p'q| \leq \max\{|p'a|, |p'b|\}$ and thus

$$\begin{aligned} & \sqrt{|pq|^2 - |pp'|^2} \leq \max\{\sqrt{|pa|^2 - |pp'|^2}, \sqrt{|pb|^2 - |pp'|^2}\} \\ \implies & |pq|^2 - |pp'|^2 \leq \max\{|pa|^2 - |pp'|^2, |pb|^2 - |pp'|^2\} \\ \implies & |pq| \leq \max\{|pa|, |pb|\}. \end{aligned}$$

◀

Let B_k be a ball graph with k different types of radii. Similar to Section 3, we guess the number of different types of radii that may appear in a maximum clique and we take the maximum over all the solutions computed from these 2^k guesses. For each ball type i,

we guess at most $2r$ ball centers $\{a_i^1, b_i^1\}, \ldots, \{a_i^r, b_i^r\}$ from $\mathcal{C}_i$. Here a_i^j is the leftmost and b_i^j is the rightmost over all the ball centers in $\mathcal{C}_i$ on the jth plane, where $1 \leq j \leq r$. Note that a_i^j may sometimes coincide with b_i^j. Let Ψ be the set of these $2rk$ balls. For every upper slab $U_{a_i^j b_i^j}$ on the jth plane, we construct a set X_i^j by taking every ball that has its center in $U_{a_i^j b_i^j}$ and intersects all the balls of Ψ. We show that the union of these balls, i.e., $X = \bigcup_{1 \leq i \leq k} (X_i^1 \cup X_i^2 \cup \ldots \cup X_i^r)$, is a clique in $\mathcal{D}_k$. Similarly, for each lower slab $\overline{U}_{a_i b_i}$, we construct a set Y_i^j by taking every ball that has its center in $\overline{U}_{a_i^j b_i^j}$ and intersects all the balls of Ψ. We show that their union $Y = \bigcup_{1 \leq i \leq k} (Y_i^1 \cup Y_i^2 \cup \ldots \cup Y_i^r)$ is a clique in $\mathcal{D}_k$. It now suffices to show that the balls in X (similarly, the balls in Y) are mutually adjacent.

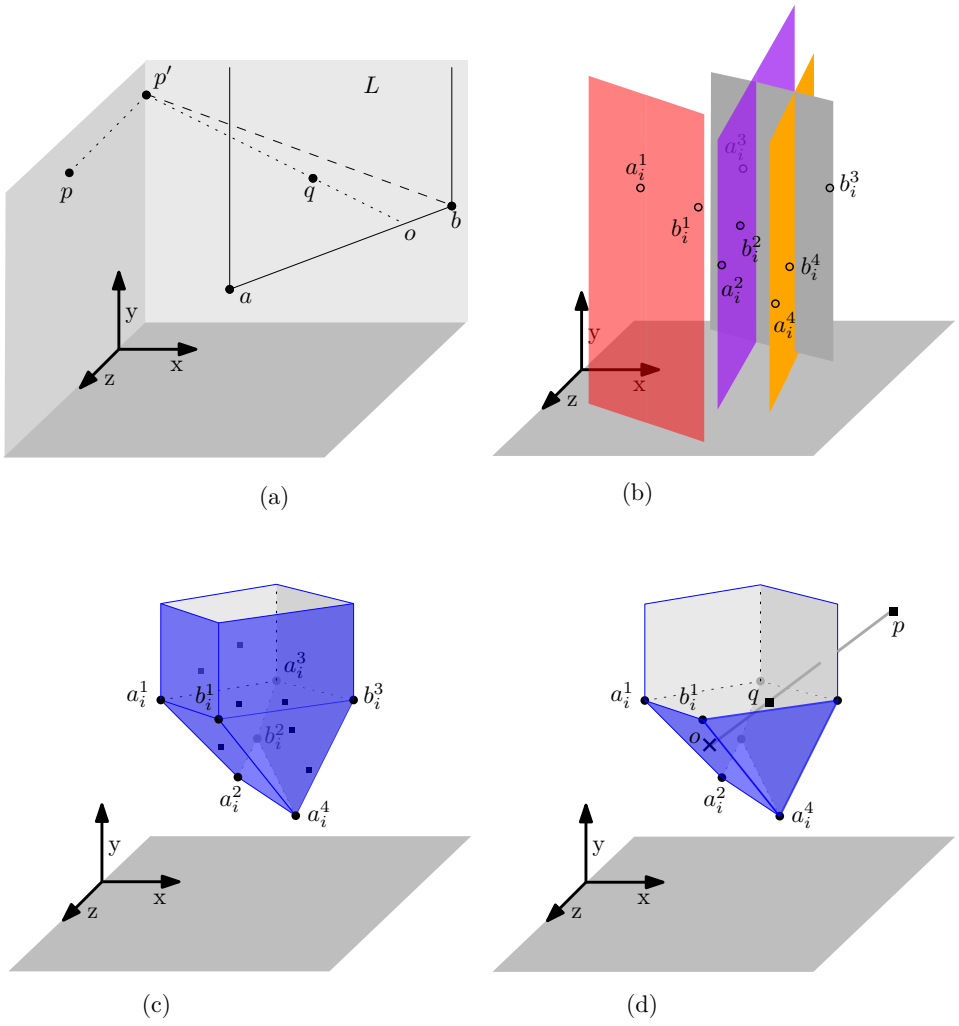


Figure 2 Illustration for (a) Lemma 5, and (b)–(d) Lemma 9.

▶ **Lemma 6.** *For every i, j, where $1 \leq i, j \leq k$, the balls in $X_i^g \cup X_j^h$, where $1 \leq g \leq h \leq r$, are mutually adjacent.*

Proof. Let B_{p,r_i} and B_{q,r_j} be two balls, where p belongs to X_i^g and q belongs to X_j^h. We now show that B_{p,r_i} and B_{q,r_j} must mutually intersect. If $g = h$, then the proof follows from Lemma 2. Therefore, we may assume that $g \neq h$. However, the proof in this case is the same as that of Lemma 2 except that we use Lemma 5 instead of Lemma 1 for the arguments, as follows.

Consider first the case when $i = j$. Without loss of generality assume that the y-coordinate of p is larger than or equal to that of q. Then by Lemma 5, $|pq| \leq \max\{|pa_j^h|, |pb_j^h|\}$. By the construction of X, the ball B_{p,r_i} intersects $B_{a_j^h,r_i}$ and $B_{b_j^h,r_i}$. We thus have $|pq| \leq \max\{|pa_j^h|, |pb_j^h|\} \leq 2r_j$, and since p and q correspond to type-j balls, they must intersect.

Consider now the case when $i \neq j$. If the y-coordinate of p is larger than or equal to that of q, then we apply Lemma 5 to obtain $|pq| \leq \max\{|pa_j^h|, |pb_j^h|\}$. By the construction of X, the ball B_{p,r_i} intersects $B_{a_j^h,r_j}$ and $B_{b_j^h,r_j}$. We thus have $|pq| \leq \max\{|pa_j^h|, |pb_j^h|\} \leq r_i + r_j$. Consequently, B_{p,r_i} and B_{q,r_j} mutually intersect. If the y-coordinate of p is smaller than that of q, then we apply Lemma 5 by swapping the role of p and q, i.e., by using the condition that $p \in U_{a_i^g,b_i^g}$ whereas q has a larger y-coordinate than that of p. We thus have $|pq| \leq \max\{|qa_i^g|, |qb_i^g|\}$. By the construction of X, we have $\max\{|qa_i^g|, |qb_i^g|\} \leq r_i + r_j$. Consequently, B_{p,r_i} and B_{q,r_j} mutually intersect. ◀

We now consider the running time. For a single plane, there are at most $O(n^2)$ ways to guess the pair of balls for a particular ball type, and thus $O(n^{2k})$ ways for k pairs considering all types. Since we have r planes, the total number of guesses is $O(n^{2rk})$. For each guess of rk pairs, it is straightforward to construct the sets X and Y in $O(rkn)$ time. Since the the ball graph determined by the balls $(X \cup Y)$ and the corresponding maximum matching can be computed in polynomial time. the running time becomes $\mathcal{O}^*(n^{2rk})$.

The following theorem summarizes the result of this section.

▶ **Theorem 7.** *Given a set of n balls in the Euclidean plane with k different types of radii where the ball centers are contained on r parallel planes, a maximum clique in the corresponding ball graph can be computed in $\mathcal{O}^*(n^{2rk})$ time.*

5.2 Ball Centers are on r Planes that are Perpendicular to Another Plane

We now show that the condition for the input planes being parallel to each other can be relaxed as long as the input planes are all perpendicular to a different plane. Without loss of generality assume that the r input planes are perpendicular to the xz-plane (e.g., Figure 2(b)). We first observe the following property of a maximum clique.

▶ **Lemma 8.** *Let B_k be a ball graph with k different radii with centers on r planes that are perpendicular to the xz-plane. Let $\mathcal{C}_i$ be all the balls of type i in a maximum clique of B_k. Let P be the projection of the ball centers in $\mathcal{C}_i$ on the xz-plane. Then the convex-hull boundary of P contains at most $2r$ vertices (i.e., they correspond to strictly convex corners).*

Proof. If three points of P come from the same original plane, then they are collinear in the xz-plane, thus only two of them can appear as strictly convex corners on the convex-hull boundary of P. Therefore, the r planes can contribute to at most $2r$ vertices in total. ◀

Let Q be a set of points in three dimensions. By a *convex hull* of Q we denote the smallest convex polyhedra that contains all points in Q. A *lower (upper) envelope* of Q consists of all points that lie at the boundary of the convex hull of Q such that the rays that start at these points and move along the negative y-axis (positive y-axis) do not contain any interior

point of the convex hull. An *extended lower envelope* of Q is a three-dimensional region that consists of the points on the lower envelope and all points that are hit by the rays that start at the lower envelope and move along the positive y-axis. Figure 2(c) illustrates an extended lower envelope. We also define an *extended upper envelope* of Q symmetrically.

Similar to Section 3, we guess the set of different types of radii that may appear in a maximum clique. For each ball type i, we guess at most $2r$ ball centers $\{a_i^1, b_i^1\}, \ldots, \{a_i^r, b_i^r\}$ from $\mathcal{C}_i$, where a_i^j is the leftmost and b_i^j is the rightmost over all the ball centers in $\mathcal{C}_i$ (e.g., Figure 2(b)) on the jth plane. It is straightforward to observe that the convex hull of the projection of the guessed ball centers on the xz-plane contains the projection of all ball centers of $\mathcal{C}_i$. Let Ψ be the set of at most $2rk$ balls that we guess over all radii types.

For every extended lower envelope U_i for the ball centers guessed for type i, we construct a set $\mathcal{X}_i$ by taking every ball that has its center in U_i and intersects all the balls of Ψ. We show that the union of these balls, i.e., $\mathcal{X} = (\mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_k)$, determines a clique. Similarly, for each extended upper envelope $\overline{U}_i$, we construct a set $\mathcal{Y}_i$ by taking every ball that has its center in $\overline{U}_i$ and intersects all the balls of Ψ. We show that their union $\mathcal{Y} = (\mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots \cup \mathcal{Y}_k)$, determines a clique. It now suffices to show that the balls in $\mathcal{X}$ (similarly, the balls in $\mathcal{Y}$) are mutually adjacent.

▶ **Lemma 9.** *For every i, j, where $1 \leq i, j \leq k$, the balls in $\mathcal{X}_i \cup \mathcal{X}_j$ are mutually adjacent.*

Proof. Let B_{p,r_i} and B_{q,r_j} be two balls, where p belongs to $\mathcal{X}_i$ and q belongs to $\mathcal{X}_j$. We now show that B_{p,r_i} and B_{q,r_j} must mutually intersect.

The argument in the rest of the proof works irrespective of whether $i = j$ or not. Without loss of generality assume that the y-coordinate of p is larger than or equal to that of q. Let R be the ray that starts at p and passes through q, and let o be the intersection point of R when R exits the extended lower envelope U_i. Figure 2(d) illustrates this when $i \neq j$.

Assume first that o hits a face F of the lower envelope, and let L be the plane determined by F. Let m be the point of L that minimizes the distance $|pm|$. Since F is convex, moving o away from m on F would increase the length $|po|$ monotonically and hit a vertex w of F. Since B_{p,r_i} intersects B_{w,r_j}, and since $|pq| \leq |po| \leq |pw| \leq r_i + r_j$, the balls B_{p,r_i} and B_{q,r_j} must intersect.

Assume now that o does not belong to the lower envelope but hits a side F of the extended lower envelope, and let L be the plane determined by F. Let m be the point of L that minimizes the distance $|pm|$. Since F is convex, moving o towards negative y-axis and away from m on F would increase the length $|po|$ monotonically and hit a vertex w of F. Since B_{p,r_i} intersects B_{w,r_j}, and since $|pq| \leq |po| \leq |pw| \leq r_i + r_j$, the balls B_{p,r_i} and B_{q,r_j} must intersect. ◀

The following theorem summarizes the result of this section.

▶ **Theorem 10.** *Given a set of n balls in the Euclidean plane with k different types of radii where the ball centers are contained on r planes that are perpendicular to a single different plane, a maximum clique in the corresponding ball graph can be computed in $\mathcal{O}^*(n^{2rk})$ time.*

6 Maximum Clique in Disk-Like Graphs

In this section we give an $\mathcal{O}^*(n^{2k})$-time algorithm to compute a maximum clique in the intersection graph of n L-visible convex polygons with k distinct shapes. Figure 3(a) illustrates an L-visible polygon, where every point of the polygon is L-visible from a center point c.

We first introduce some notation. Let $\mathcal{S}_k$ be an intersection graph of L-visible polygons, where the number of different types of shapes is at most k. For convenience, we denote these shapes as $S_1, S_2, \ldots, S_k$ and for each shape S_i, designate a center c_i, i.e., all points

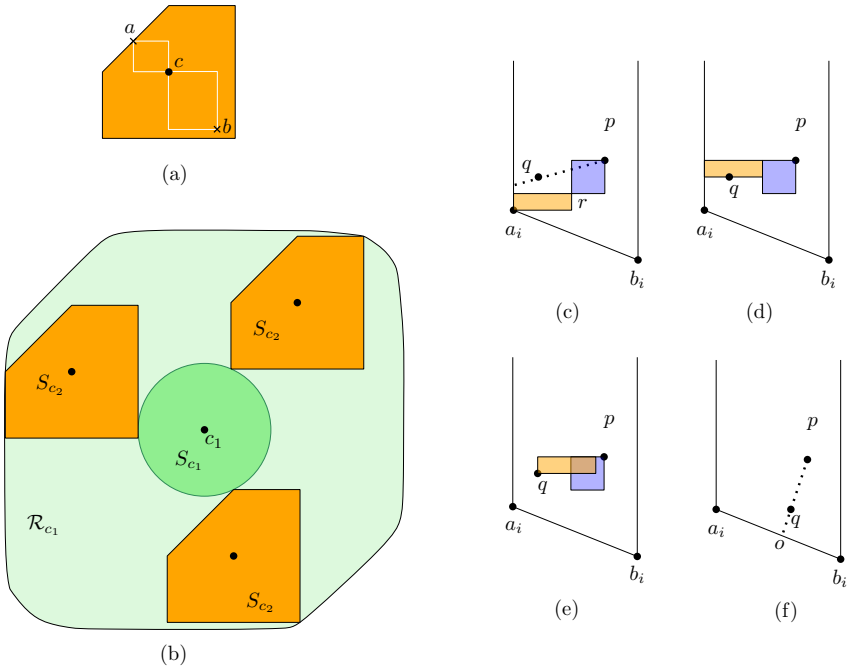


Figure 3 (a) Illustration for L-visibility. (b) $\mathcal{R}_{c_1}$. (c)–(f) Illustration for the proof of Lemma 12.

of S_i are L-visible from c_i. Let $\mathcal{C}$ be a maximum clique of $\mathcal{S}_k$. By $\mathcal{C}_i$ we denote a maximal clique in $\mathcal{C}$ where all shapes are of type i. We also use the following properties of L-visible shapes which are straightforward to verify using the concept of Minkowski sum of two convex polygons [18].

▶ Remark 11. Let S_1 and S_2 be two L-visible convex shapes with centers c_1 and c_2, respectively. Let $\mathcal{R}_{c_1}$ be the region that consists of points where S_2 can be placed to intersect S_1. Then $\mathcal{R}_{c_1}$ is convex (Figure 3(b)).

The idea of the algorithm is similar to that of Section 3. We guess the set of different types of shapes that may appear in a maximum clique and we take the maximum over all the solutions computed from these 2^k guesses. For each type i, we guess the leftmost and rightmost shape centers a_i and b_i, respectively, over the shapes in $\mathcal{C}_i$. Let Ψ be the set of these $2k$ shapes. For every upper slab $U_{a_ib_i}$, we construct a set X_i by taking every shape that has its center in $U_{a_ib_i}$ and intersects all the shapes of Ψ. We show that the union of these shapes, i.e., $X = (X_1 \cup X_2 \cup \ldots \cup X_k)$, is a clique in $\mathcal{S}_k$. Similarly, for each lower slab $\overline{U}_{a_ib_i}$, we construct a set Y_i by taking every shape that has its center in $\overline{U}_{a_ib_i}$ and intersects all the shapes of Ψ. We show that their union $Y = (Y_1 \cup Y_2 \cup \ldots \cup Y_k)$ is a clique in $\mathcal{S}_k$. A maximum clique can then be found from a maximum bipartite matching in the complement of the intersection graph of $(X \cup Y)$. The time to compute the graph and the matching is polynomial in n and m, where m is the *shape complexity*, i.e., upper bound on the number of smooth curves on the boundary of a shape.

▶ **Lemma 12.** *For every i, j, where $1 \leq i, j \leq k$, the shapes in $X_i \cup X_j$ are mutually adjacent.*

Proof. Let $S_q \in X_i$ and $S_p \in X_j$ be two shapes with centers q and p, respectively. We now show that S_p and S_q must mutually intersect.

Consider first the case when $i = j$, i.e., p and q both belong to the upper slab $U_{a_ib_i}$. Without loss of generality assume that the y-coordinate of p is larger than or equal to that of q. We now consider two scenarios.

If the ray starting at p passing through q does not intersect a_ib_i, then assume without loss of generality that it intersects the left side of $U_{a_ib_i}$ after passing through q (Figure 3(c)). Let r be a point that is common to both S_{a_i} and S_p. Let R be the axis-aligned rectangle with diagonal rp. We can then consider moving the center a_i along the left side of $U_{a_ib_i}$ until we reach the y-coordinate of q, and then move a_i horizontally until it coincides with q (Figure 3(d)–(e)). Throughout the process the axis-aligned rectangle with diagonal a_ir maintains an intersection region with R. Hence S_q must intersect S_p.

If the ray starting at p passing through q intersects a_ib_i at a point o (Figure 3(f)), then we prove the adjacency of S_p and S_q as follows. By Remark 11, $\mathcal{R}_p$ is convex. Since S_p intersects a_i and b_i, $\mathcal{R}_p$ also contains a_i and b_i. By the convexity of $\mathcal{R}_p$, the segment a_ib_i belongs to $\mathcal{R}_p$. We now move the center a_i to o along the line a_ib_i. Since $a_io \in \mathcal{R}_p$, throughout the process S_{a_i} intersects S_p. We now move a_i from o to q along the line op. Since L-visible shapes are convex, moving a pair of intersecting L-visible shapes closer along the the line determined by their centers keeps them intersected. Therefore, throughout the process S_{a_i} maintains an intersection with S_p. Hence S_q must intersect S_p.

Consider now the case when $i \neq j$. If the y-coordinate of p is larger than or equal to that of q, then the proof above still applies. The only difference is that p can now be outside of $U_{a_ib_i}$, but this leaves the above argument the same as we only used the property that the y-coordinate of p is larger than or equal to that of q. If the y-coordinate of p is smaller than that of q, then we can swap the role of p and q, i.e., by using the condition that $p \in U_{a_i,b_i}$ whereas q has a larger y-coordinate than that of p. ◀

The running time can now be computed in the same way as in Section 3.

▶ **Theorem 13.** *Given a set of n L-visible convex shapes in the Euclidean plane where the number of distinct shapes is k, a maximum clique in the corresponding intersection graph can be computed in $\mathcal{O}^*(n^{2k})$ time.*

7 Direction for Future Research

We gave an $\mathcal{O}^*(n^{2k})$-time algorithm to find a maximum clique in a n-vertex disk graph with k different radii. Designing faster algorithms is a natural direction to explore. Existing techniques [38] that computes maximum clique on unit disk graphs even when a geometric representation is not given do not generalize to our context. Therefore, finding efficient algorithms when a geometric representation of the disk graph is not given would be interesting.

We showed that given a set of n unit disks, one can compute a maximum clique for each possible axis-aligned rectangle determined by the input disk centers in $O(n^5 \log n)$ time. Consequently, given a rectangular range query R, the maximum clique C inside R can be reported in $O(\log \log n + |C|)$ time, where the $O(\log \log n)$ term is to locate the rectangle R' (using range searching data structures [37]) for which a solution is precomputed. Improving the $O(n^5 \log n)$ running time or establishing tight time-space trade-offs can be interesting.

For ball graphs, we gave an $\mathcal{O}^*(n^{2rk})$-time algorithm to find a maximum clique, where k is the number of different radii and the ball centers are contained in r planes that are perpendicular to a single different plane. One may attempt to relax our perpendicularity constraint. The NP-hardness reduction for computing a maximum clique in a ball graph [6] allows for arbitrary radii whereas our result provides polynomial-time algorithms when $r, k \in O(1)$. The case when $k \in o(n)$ can be explored. Finally, can we find a polynomial-time algorithm for computing maximum clique in the intersection graph of convex shapes that are not necessarily L-visible, where the number of distinct shapes is fixed?

References

1 Ranendu Adhikary, Kaustav Bose, Satwik Mukherjee, and Bodhayan Roy. Complexity of maximum cut on interval graphs. In *Proceedings of the 37th International Symposium on Computational Geometry, (SoCG)*, volume 189 of *LIPIcs*, pages 7:1–7:11, 2021. doi:10.4230/LIPICS.SOCG.2021.7.

2 Ranendu Adhikary, Kaustav Bose, Satwik Mukherjee, and Bodhayan Roy. Complexity of maximum cut on interval graphs. *Discret. Comput. Geom.*, 70(2):307–322, 2023. doi:10.1007/S00454-022-00472-Y.

3 Christoph Ambühl and Uli Wagner. The clique problem in intersection graphs of ellipses and triangles. *Theory of Computing Systems*, 38(3):279–292, 2005. doi:10.1007/s00224-005-1141-6.

4 J. Bang-Jensen, B. Reed, M. Schacht, R. Šámal, B. Toft, and U. Wagner. *Topics in Discrete Mathematics, Dedicated to Jarik Nešetřil on the Occasion of his 60th birthday*, volume 26 of *Algorithms and Combinatorics*, pages 613–627. Springer, 2006.

5 Marthe Bonamy, Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Panos Giannopoulos, Eun Jung Kim, Pawel Rzazewski, Florian Sikora, and Stéphan Thomassé. EPTAS and subexponential algorithm for maximum clique on disk and unit ball graphs. *J. ACM*, 68(2):9:1–9:38, 2021. doi:10.1145/3433160.

6 Marthe Bonamy, Edouard Bonnet, Nicolas Bousquet, Pierre Charbit, and Stéphan Thomassé. EPTAS for max clique on disks and unit balls. In Mikkel Thorup, editor, *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 568–579. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00060.

7 Édouard Bonnet, Nicolas Grelier, and Tillmann Miltzow. Maximum clique in disk-like intersection graphs. In *Proceedings of the 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020)*, volume 182 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.FSTTCS.2020.17.

8 Prosenjit Bose, Paz Carmi, J. Mark Keil, Anil Maheshwari, Saeed Mehrabi, Debajyoti Mondal, and Michiel Smid. Computing maximum independent set on outerstring graphs and their relatives. *Comput. Geom.*, 103:101852, 2022. doi:10.1016/j.comgeo.2021.101852.

9 Heinz Breu. *Algorithmic aspects of constrained unit disk graphs*. PhD thesis, University of British Columbia, 1996.

10 Valentin E Brimkov, Konstanty Junosza-Szaniawski, Sean Kafer, Jan Kratochvíl, Martin Pergel, Paweł Rzążewski, Matthew Szczepankiewicz, and Joshua Terhaar. Homothetic polygons and beyond: Maximal cliques in intersection graphs. *Discrete Applied Mathematics*, 247:263–277, 2018. doi:10.1016/J.DAM.2018.03.046.

11 Sergio Cabello. Maximum clique for disks of two sizes. *Open problems from Geometric Intersection Graphs: Problems and Directions CG Week Workshop, Eindhoven, June 25*, 2015.

12 Sergio Cabello. Open problems presented at the algorithmic graph theory. *Adriatic Coast workshop, Koper, Slovenia, June 16–19*, 2015.

13 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discret. Comput. Geom.*, 50(3):771–783, 2013. doi:10.1007/s00454-013-9538-5.

14 Paz Carmi, Matthew J. Katz, and Pat Morin. Stabbing pairwise intersecting disks by four points. *Discret. Comput. Geom.*, 70(4):1751–1784, 2023. doi:10.1007/S00454-023-00567-0.

15 Timothy Chan. Personal communication. June 6, 2023.

16 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discret. Math.*, 86(1-3):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

17 Ludwig Danzer. Zur lösung des gallaischen problems über kreisscheiben in der euklidischen ebene. *Studia Sci. Math. Hungar*, 21(1-2):111–134, 1986.

18 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: Algorithms and applications*. Springer Science & Business Media, 2000.

19 David Eppstein. Graph-theoretic solutions to computational geometry problems. In Christophe Paul and Michel Habib, editors, *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 1–16, 2009. doi:10.1007/978-3-642-11409-0_1.

20 David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discret. Comput. Geom.*, 11:321–350, 1994. doi:10.1007/BF02574012.

21 Jared Espenant, J. Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In Erin W. Chambers and Joachim Gudmundsson, editors, *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.30.

22 Tomás Feder and Rajeev Motwani. Clique partitions, graph compression and speeding-up algorithms. *J. Comput. Syst. Sci.*, 51(2):261–272, 1995. doi:10.1006/JCSS.1995.1065.

23 Stefan Felsner, Rudolf Müller, and Lorenz Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discret. Appl. Math.*, 74(1):13–32, 1997. doi:10.1016/S0166-218X(96)00013-3.

24 Aleksei V. Fishkin. Disk graphs: A short survey. In Klaus Jansen and Roberto Solis-Oba, editors, *Proceedings of Approximation and Online Algorithms, First International Workshop (WAOA)*, volume 2909 of *Lecture Notes in Computer Science*, pages 260–264. Springer, 2003. doi:10.1007/978-3-540-24592-6_23.

25 Nicolas Grelier. Computing a maximum clique in geometric superclasses of disk graphs. *J. Comb. Optim.*, 44(4):3106–3135, 2022. doi:10.1007/S10878-022-00853-2.

26 Nicolas Grelier. Computing a maximum clique in geometric superclasses of disk graphs. *Journal of Combinatorial Optimization*, 44(4):3106–3135, 2022. doi:10.1007/S10878-022-00853-2.

27 Nicolas Grelier. *Maximum Clique in Generalisations of Disk Graphs and Plane Geometric Graphs on Degenerate Point Sets*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2022. doi:10.3929/ETHZ-B-000574739.

28 Mark L. Huson and Arunabha Sen. Broadcast scheduling algorithms for radio networks. In *Proceedings of MILCOM'95*, volume 2, pages 647–651. IEEE, 1995.

29 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *J. Algorithms*, 4(4):310–323, 1983. doi:10.1016/0196-6774(83)90012-3.

30 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 653–663. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743516.

31 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM J. Comput.*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.

32 J. Mark Keil and Debajyoti Mondal. The maximum clique problem in a disk graph made easy, 2024. doi:10.48550/arXiv.2404.03751.

33 J. Mark Keil, Debajyoti Mondal, Ehsan Moradi, and Yakov Nekrich. Finding a maximum clique in a grounded 1-bend string graph. *Journal of Graph Algorithms and Applications*, 26(4), 2022. doi:10.7155/JGAA.00608.

34 Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison Wesley, 2006.

35 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031. SIAM, 2022. doi:10.1137/1.9781611977073.80.

36 Nicholas Nash and David Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Inf. Process. Lett.*, 110(16):630–634, 2010. doi:10.1016/j.ipl.2010.05.016.

37 Yakov Nekrich. New data structures for orthogonal range reporting and range minima queries. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1191–1205. SIAM, 2021. doi:10.1137/1.9781611976465.73.

38 Vijay Raghavan and Jeremy P. Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003. doi:10.1016/S0196-6774(03)00048-8.

39 L Stachó. A gallai-féle körletüzési probléma megoldása, mat. *Lapok*, 32:19–47, 1981.

Super-Polynomial Growth of the Generalized Persistence Diagram

Donghan Kim
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Woojin Kim
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Wonjun Lee
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Abstract

The Generalized Persistence Diagram (GPD) for multi-parameter persistence naturally extends the classical notion of persistence diagram for one-parameter persistence. However, unlike its classical counterpart, computing the GPD remains a significant challenge. The main hurdle is that, while the GPD is defined as the Möbius inversion of the Generalized Rank Invariant (GRI), computing the GRI is intractable due to the formidable size of its domain, i.e., the set of all connected and convex subsets in a finite grid in $\mathbb{R}^d$ with $d \geq 2$. This computational intractability suggests seeking alternative approaches to computing the GPD.

In order to study the complexity associated to computing the GPD, it is useful to consider its classical one-parameter counterpart, where for a filtration of a simplicial complex with n simplices, its persistence diagram contains at most n points. This observation leads to the question: *Given a d-parameter simplicial filtration, could the cardinality of its GPD (specifically, the support of the GPD) also be bounded by a polynomial in the number of simplices in the filtration?* This is the case for $d = 1$, where we compute the persistence diagram directly at the simplicial filtration level. If this were also the case for $d \geq 2$, it might be possible to compute the GPD directly and much more efficiently without relying on the GRI.

We show that the answer to the question above is negative, demonstrating the inherent difficulty of computing the GPD. More specifically, we construct a sequence of d-parameter simplicial filtrations where the cardinalities of their GPDs are not bounded by any polynomial in the number of simplices. Furthermore, we show that several commonly used methods for constructing multi-parameter filtrations can give rise to such "wild" filtrations.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation

Keywords and phrases Persistent homology, Möbius inversion, Multiparameter persistence, Generalized persistence diagram, Generalized rank invariant

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.64

Related Version *Full Version*: `https://arxiv.org:2412.04889` [38]

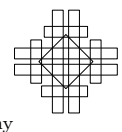
Funding This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-00515946).

Acknowledgements WK thanks Jisu Kim and Facundo Mémoli for their feedback on parts of an earlier draft of this paper.

1 Introduction

(Multi-parameter) Persistent Homology. Persistent homology is a central concept in Topological Data Analysis (TDA), which is useful for studying the multi-scale topological features of datasets [16, 17, 27]. In the classical one-parameter setting, topological features in datasets are summarized in the form of persistence diagrams or barcodes [23, 65], which serve as complete, discrete invariants of the associated persistence modules.

© Donghan Kim, Woojin Kim, and Wonjun Lee;
licensed under Creative Commons License CC-BY 4.0



Multi-parameter persistent homology generalizes persistent homology. In the d-parameter setting, topological features of datasets are indexed by the poset $\mathbb{R}^d$ or a subposet of $\mathbb{R}^d$ [11, 18]. This generalization allows for a finer extraction of topological features, enabling a more detailed analysis of complex datasets [19, 24, 46, 47, 59, 63, 64]. However, the algebraic structure of multi-parameter persistence modules, i.e. functors from the poset $\mathbb{R}^d$ ($d \geq 2$) to the category $\mathbf{vec}_{\mathbb{F}}$ of finite dimensional vector spaces over a field $\mathbb{F}$, is significantly more complex, and in fact no complete discrete invariant exists for multi-parameter persistence modules [18].

Generalized Persistence Diagram (GPD). Many incomplete, but potentially useful invariants for multi-parameter persistence modules have been studied; e.g. [3, 4, 8, 12, 21, 34, 35, 44, 45, 51, 54, 58]. The Generalized Persistence Diagram (GPD) is one such invariant and is a natural extension of the persistence diagram for one-parameter persistence [39, 55]. The GPD is defined as the Möbius inversion of the Generalized Rank Invariant (GRI)[1], which captures "persistence" in multi-parameter persistence modules or, more generally, any linear representations of posets [40]. Many aspects of the GPD and GRI–such as stability, discriminating power, computation, connections to other invariants, and generalizations–have been studied; e.g. [5, 12, 22, 25, 26, 31, 41, 42]. Also, a vectorization method for the (restricted) GRI has recently been proposed and utilized in a machine learning context [53, 64]. There are also other works on invariants of multi-parameter persistence modules that are closely related to the GPD and GRI; e.g. [3, 4, 5, 8, 15, 34, 36].

Challenges in Computing the GPD. While many properties of the GPD have been clarified in the aforementioned works, computing the GPD, unlike its classical counterpart, remains a significant challenge. The main hurdle is that, while the GPD is defined as the Möbius inversion of the GRI, computing the GRI is intractable mainly due to the formidable size of its domain. For instance, the domain of the GRI of a persistence module over a finite grid $G \subset \mathbb{R}^d$ is the set of all *intervals* in G (cf. Definitions 2 and 5). The cardinality of the domain is huge relative to the cardinality $|G|$ of G even when $d = 2$; cf. [2, Theorem 31]. This computational intractability suggests seeking alternative approaches to computing the GPD.

In order to study the complexity associated to computing the GPD, it is useful to consider its classical one-parameter counterpart, where for a filtration of a simplicial complex with N simplices, its persistence diagram contains at most N points. This observation leads to the question: *Given a d-parameter simplicial filtration, could the cardinality of its GPD (more precisely, the support of the GPD) also be bounded by a polynomial in the number of simplices in the filtration?* In what follows, we make this question more precise.

Size of the GPD. Let K be an abstract simplicial complex, and let ΔK be the set of subcomplexes of K ordered by inclusion. A monotone map $\mathcal{F} : \mathbb{R}^d \to \Delta K$ such that there exists a $p \in \mathbb{R}^d$ with $\mathcal{F}_p := \mathcal{F}(p) = K$ is called a **(d-parameter simplicial) filtration (of K)**. The number of simplices in the filtration $\mathcal{F}$ is defined as the number of simplices in K.

[1] The term "generalized persistence diagram" sometimes refers to different concepts; e.g. [14, 52, 56]. Also, we note that the notion of the GRI stems from the concept of the rank invariant [18].

▶ **Definition 1** ([45]). *We call the filtration $\mathcal{F}$ **finite**, if K is finite, and in $\mathcal{F}$, every simplex $\sigma \in K$* is born at finitely many points of $\mathbb{R}^d$, *i.e. there exist finitely many points $q_1, \ldots, q_{t(\sigma)} \in \mathbb{R}^d$ s.t. $\sigma \in \mathcal{F}_p$ for $p \in \mathbb{R}^d$ if and only if $q_i \leq p$ in $\mathbb{R}^d$ for some $i = 1, \ldots, t(\sigma)$. We call those points $q_1, \ldots, q_{t(\sigma)}$ the **birth indices** of σ. We also say that σ is **born** at $q_1, \ldots, q_{t(\sigma)} \in \mathbb{R}^d$.*

If $\mathcal{F}$ is finite, then for each homology degree $m \in \mathbb{Z}_{\geq 0}$, the persistence module $\mathrm{H}_m(\mathcal{F}; \mathbb{F}) : \mathbb{R}^d \to \mathbf{vec}_{\mathbb{F}}$ is finitely presentable (cf. Definition 3 and Remark 4). This guarantees that the ***m*-th GPD** of any finite d-parameter filtration is well-defined as an integer-valued function on a *finite* subset of intervals of $\mathbb{R}^d$ (cf. Definition 5 and Remark 7), denoted by $\mathrm{dgm}_m(\mathcal{F})$. By the **size** of $\mathrm{dgm}_m(\mathcal{F})$ we mean the cardinality of the *support* of $\mathrm{dgm}_m(\mathcal{F})$, i.e. $|\{I \in \mathrm{Int}(\mathbb{R}^d) : \mathrm{dgm}_m(\mathcal{F})(I) \neq 0\}|$ (cf. Definition 2). Since the notion of the GPD reduces to that of the persistence diagram when $d = 1$ [22, Section 3], in this case, the size of the GPD becomes the number of points in the persistence diagram. Our question is as follows.

▶ **Question.** Does there exist $k \in \mathbb{Z}_{\geq 0}$ such that for *any* homology degree $m \geq 0$, the size of the m-th GPD of *any* finite simplicial filtration $\mathcal{F}$ over $\mathbb{R}^d$ is $O(N^k)$ where N stands for the number of simplicies in $\mathcal{F}$?

Note that when $d = 1$, the answer to the question is *yes*, and k can be taken to be 1, in which case we compute the persistence diagram directly at the simplicial filtration level [28]. If the answer is also *yes* when $d \geq 2$, it might be possible to compute the GPD of multi-parameter filtrations directly and much more efficiently without relying on the GRI of the persistence module $\mathrm{H}_m(\mathcal{F}; \mathbb{F}) : \mathbb{R}^d \to \mathbf{vec}_{\mathbb{F}}$.

Our contributions

1. We show that the answer to **Question** above is negative, demonstrating the inherent difficulty of computing the GPD (Theorem 13). More specifically, we construct a sequence $(\mathcal{F}_n)$ of finite d-parameter simplicial filtrations where the sizes of their GPDs are not bounded by any polynomial in the number of simplices in $\mathcal{F}_n$.
2. We also show that several well-known methods for constructing multi-parameter filtrations – Sublevel-Rips, Sublevel-Čech, Degree-Rips, and Degree-Čech – can give rise to such "wild" filtrations (Theorems 24 and 25).
3. We find that computing the value of the GPD for a 2-parameter filtration containing $O(n^s)$ simplices, for some $s \in \mathbb{N}$, via the Möbius inversion of the GRI can result in the problem of summing $\Theta(2^n)$ integers, which is an EXPTIME problem (Corollary 26). This implies that, even with the fully computed GRI, computing the GPD through Möbius inversion of the GRI remains computationally challenging.

It is noteworthy that in the proofs of some of our results, we employ techniques inspired by some recent works on the GRI [12, 22] as well as Rota's Galois connections [33]; see the proof of Theorem 13 for the case of $d > 2$ and that of Lemma 21.

Other related works. Botnan, Oppermann, Oudot investigated the computational complexity of the minimal rank decomposition of the standard rank invariant as part of their results [12, Remark 5.3]. Also, Botnan, Oppermann, Oudot, Scoccola found an upper bound on the size of the rank *exact* decomposition of a persistence module, which is a polynomial in the size of the (usual) multigraded Betti numbers of the module [13].

Clause, Kim, Mémoli identified a *tame* persistence module M over $\mathbb{Z}^2$ (a concept introduced by Miller [51]) whose GRI does not admit a Möbius inversion [22, Theorem B]. Our construction of persistence modules in Section 3 is reminiscent of that of M, making it natural to interpret M as a limit object of the types of persistence modules we construct.

Morozov and Patel elucidated a connection between 1- and 2-parameter persistence settings in order to find an output-sensitive algorithm for computing the Möbius inversion of the birth-death function [52]. We also note that there are many recent examples of utilizing Möbius inversion to extract information from (multi-parameter) persistent (co)homology: see e.g. [7, 33, 34, 49, 50, 52, 54, 56, 62].

Alonso, Kerber, Skraba showed that many multi-parameter filtration constructions, such as density- or degree-Rips bifiltrations, and across a general category of point samples in Euclidean space, the probability of the homology-induced persistence module decomposing into interval modules approaches zero as the sample size tends to infinity [1].

Organization. In Section 2, we review basic terminology and well-known theorems related to persistence modules, as well as the GPD and GRI. In Sections 3 and 4, we establish our main results, as mentioned above. Finally, in Section 5, we discuss future research directions.

2 Preliminaries

Persistence modules and barcodes. Throughout this paper, $P = (P, \leq)$ is a poset, regarded as the category whose objects are the elements of P, and for any pair $p, q \in P$, there exists a unique morphism $p \to q$ if and only if $p \leq q$. All vector spaces in this paper are over a fixed field $\mathbb{F}$. Let $\mathbf{vec}_{\mathbb{F}}$ denote the category of finite-dimensional vector spaces and linear maps over $\mathbb{F}$. A functor $P \to \mathbf{vec}_{\mathbb{F}}$ will be referred to as a **persistence module over P** or simply a **P-module**. A morphism between P-modules is a natural transformation between the P-modules. For any P-modules M and N, their direct sum $M \oplus N$ is defined pointwisely. A P-module M is **trivial** if $M(p) = 0$ for all $p \in P$, and we write $M = 0$. A nontrivial P-module M is **indecomposable** if the assumption $M \cong M' \oplus M''$ for some P-modules M' and M'' implies that either $M' = 0$ or $M'' = 0$. By Krull-Remak-Schmidt-Azumaya's theorem [6, 9], any P-module is isomorphic to a direct sum of indecomposable P-modules, and this decomposition is unique up to isomorphism and permutation of summands.

▶ **Definition 2.** *An **interval** I of P is a subset $I \subseteq P$ such that:*

- **(i)** *I is nonempty.*
- **(ii)** *If $p, q \in I$ and $p \leq r \leq q$, then $r \in I$.*
- **(iii)** *I is **connected**, i.e. for any $p, q \in I$, there is a sequence $p = p_0, p_1, \cdots, p_\ell = q$ of elements of I with p_i and p_{i+1} comparable for $0 \leq i \leq \ell - 1$.*

By $\mathrm{Int}(P)$ we denote the set of all intervals of P.

For any $p \leq q$ in P, the set $[p, q] := \{r \in P : p \leq r \leq q\}$ is called a **segment**. The collection of all segments (resp. intervals) in P will be denoted by $\mathrm{Seg}(P)$ (rep. $\mathrm{Int}(P)$). It is not difficult to see that $\mathrm{Seg}(P) \subset \mathrm{Int}(P)$.

Given any $I \in \mathrm{Int}(P)$, the **interval module** $\mathbb{F}_I$ is the P-module, with

$$\mathbb{F}_I(p) := \begin{cases} \mathbb{F} & \text{if } p \in I \\ 0 & \text{otherwise.} \end{cases}, \qquad \mathbb{F}_I(p \leq q) := \begin{cases} \mathrm{id}_{\mathbb{F}} & \text{if } p \leq q \in I \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Every interval module is indecomposable [10, Proposition 2.2]. A P-module M is called **interval-decomposable** if it is isomorphic to a direct sum of interval modules. The **barcode** of an interval decomposable P-module $M \cong \bigoplus_{j\in J} \mathbb{F}_{I_j}$ is defined as the multiset $\mathrm{barc}(M) := \{I_j : j \in J\}$.

For $p \in P$, let $p^{\uparrow}$ (resp. $p^{\downarrow}$) denote the set of points $q \in P$ such that $p \le q$ (resp. $q \le p$). Clearly, both $p^{\uparrow}$ and $p^{\downarrow}$ belong to $\mathrm{Int}(P)$.

▶ **Definition 3.** *A P-module is called **finitely presentable** if it is isomorphic to the cokernel of a morphism $\bigoplus_{b\in B} \mathbb{F}_{p_b^{\uparrow}} \to \bigoplus_{a\in A} \mathbb{F}_{p_a^{\uparrow}}$, where $\{p_a : a \in A\}$ and $\{p_b : b \in B\}$ are finite multisets of elements of P.*

▶ Remark 4. It is well known that, for any finite d-parameter simplicial filtration $\mathcal{F}$ (Definition 1), the induced $\mathbb{R}^d$-module $\mathrm{H}_m(\mathcal{F};\mathbb{F})$, for each $m \in \mathbb{Z}_{\ge 0}$, is finitely presentable. Namely, $\mathrm{H}_m(\mathcal{F};\mathbb{F})$ is isomorphic to the cokernel of a morphism $\bigoplus_{j\in J} \mathbb{F}_{p_j^{\uparrow}} \to \bigoplus_{i\in I} \mathbb{F}_{p_i^{\uparrow}}$, where $\{p_j : J \in J\} \cup \{p_i : i \in I\}$ is a subset of the smallest finite d-d grid in $\mathbb{R}^d$ that contains all the birth indices of m- and $(m+1)$-simplices in $\mathcal{F}$. See, e.g. [45].

Generalized Rank invariant and Generalized Persistence Diagram. Any P-module M admits both a **limit** and a **colimit** [48, Chapter V]: A limit of M, denoted by $\varprojlim M$, consists of a vector space L together with a collection of linear maps $\{\pi_p : L \to M(p)\}_{p\in P}$ such that

$$M(p \le q) \circ \pi_p = \pi_q \text{ for every } p \le q \text{ in } P. \tag{2}$$

A colimit of M, denoted by $\varinjlim M$, consists of a vector space C together with a collection of linear maps $\{i_p : M(p) \to C\}_{p\in P}$ such that

$$i_q \circ M(p \le q) = i_p \text{ for every } p \le q \text{ in } P. \tag{3}$$

Both $\varprojlim M$ and $\varinjlim M$ satisfy certain universal properties, making them unique up to isomorphism.

Let us assume that P is connected. The connectedness of P alongside the equalities given in Equations (2) and (3) imply that $i_p \circ \pi_p = i_q \circ \pi_q : L \to C$ for any $p, q \in P$. This fact ensures that the **canonical limit-to-colimit map** $\psi_M : \varprojlim M \longrightarrow \varinjlim M$ given by $i_p \circ \pi_p$ for any $p \in P$ is well-defined. The **(generalized) rank** of M is defined to be[2] $\mathrm{rank}(M) := \mathrm{rank}(\psi_M)$, which is finite as $\mathrm{rank}(M) = \mathrm{rank}(i_p \circ \pi_p) \le \dim(M(p)) < \infty$ for *any* $p \in P$. The rank of M is a count of the "persistent features" in M that span the entire indexing poset P [40].

Now, we refine the rank of a P-module, which is a single integer, into an integer-valued function. Options for the domain of the function include $\mathrm{Seg}(P)$ or the larger set $\mathrm{Int}(P)$.

▶ **Definition 5** ([22, 39])**.** *Let M be a P-module.*

(i) *The **generalized rank invariant (GRI)** is the map $\mathrm{rk}_M : \mathrm{Int}(P) \to \mathbb{Z}_{\ge 0}$ given by $I \mapsto \mathrm{rank}(M|_I)$ where $M|_I$ is the restriction of M to I.*

(ii) *The **generalized persistence diagram (GPD) of M** is defined as the function $\mathrm{dgm}_M : \mathrm{Int}(P) \to \mathbb{Z}$ that satisfies the equality*[3]

$$\forall I \in \mathrm{Int}(P), \quad \mathrm{rk}_M(I) = \sum_{\substack{J\in\mathrm{Int}(P)\\ J\supseteq I}} \mathrm{dgm}_M(J), \tag{4}$$

where the right-hand side of the equation includes only finitely many nonzero summands.

[2] This construction was considered in the study of quiver representations [43].
[3] This equality generalizes *the fundamental lemma of persistent homology* [29].

We also remark that, in Definition 5, by replacing all instances of $\mathrm{Int}(P)$ by $\mathrm{Seg}(P)$, we obtain the notions of the *rank invariant* and its *signed barcode* [12, 13, 18].

▶ **Theorem 6** (Existence and Uniqueness of the GPD). *Let M be a P-module.*

(i) *If P is finite, then the GPD of M exists.*

(ii) *If $P = \mathbb{R}^d$ $(d = 1, 2, \ldots)$, and M is finitely presentable, then the GPD of M exists. Furthermore, dgm_M is finitely supported, i.e. there exist only finite many $I \in \mathrm{Int}(\mathbb{R}^d)$ such that $\mathrm{dgm}_M(I) \neq 0$.*

(iii) *In each of the previous two cases, the GPD is unique.*

Proof. (i): If P is finite, then $\mathrm{Int}(P)$ is finite. Thus, the claim directly follows from the Möbius inversion formula [57]. (ii): This statement is precisely that of [22, Theorem C(iii)]. (iii): In the setting of (i), the uniqueness is also a direct consequence of the Möbius inversion formula. In the setting of (ii), the uniqueness is proved in [22, Proposition 3.2]. ◀

▶ Remark 7 ([22, Theorem C (iii)]). For any finitely presentable $\mathbb{R}^d$-module M, assume that M is the cokernel of a morphism $\bigoplus_{b\in B} \mathbb{F}_{p_b^{\uparrow}} \to \bigoplus_{a\in A} \mathbb{F}_{p_a^{\uparrow}}$, where A and B are finite index sets. Let $\mathcal{C} := \{p_a : a \in A\} \cup \{p_b : b \in B\} \subset \mathbb{R}^d$. Consider the finite full subposet $P := \prod_{i=1}^d \pi_i(\mathcal{C}) \subset \mathbb{R}^d$, where $\pi_i : \mathbb{R}^d \to \mathbb{R}$ is the canonical projection to the i-th coordinate. Then, the GPD of M is directly obtained from the GPD of the restriction $M|_P$, as follows.

We extend $\mathbb{R}^d$ to $\mathbb{R}^d \cup \{-\infty\}$ by declaring that $-\infty \leq x$ for all $x \in \mathbb{R}^d$. Let $\lfloor - \rfloor_P : \mathbb{R}^d \to P \cup \{-\infty\}$ be the map sending each $x \in \mathbb{R}^d$ to the maximal point $p =: \lfloor x \rfloor_P$ in $P \cup \{-\infty\}$ such that $p \leq x$ in $\mathbb{R}^d \cup \{-\infty\}$. For any $I \subseteq \mathbb{R}^d$, let $\lfloor I \rfloor_P$ denote the set of the points $\lfloor x \rfloor_P$ for $x \in I$. Then, the GPD of M, i.e. $\mathrm{dgm}_M : \mathrm{Int}(\mathbb{R}^d) \to \mathbb{Z}$, is equal to

$$I \mapsto \begin{cases} \mathrm{dgm}_{M|_P}(\lfloor I \rfloor_P), & \text{if } I = \lfloor - \rfloor_P^{-1}(J) \text{ for some } J \in \mathrm{Int}(P) \\ 0, & \text{otherwise.} \end{cases}$$

This implies that **there exists a bijection between the supports of $\mathbf{dgm}_M$ and $\mathbf{dgm}_{M|_P}$.**

We also remark that if P is a finite poset, the condition given in Equation (4) holds if and only if

$$\forall I \in \mathrm{Int}(P), \quad \mathrm{dgm}_M(I) = \sum_{\substack{J \in \mathrm{Int}(P) \\ J \supseteq I}} \mu(J, I) \cdot \mathrm{rk}_M(J), \tag{5}$$

where μ is the *Möbius function* of the poset $(\mathrm{Int}(P), \supseteq)$; see [61, Section 3.7] for a general reference on Möbius inversion, and [40] for a discussion of Möbius inversion in this specific context.

We define the **size** of rk_M, denoted by $\|\mathrm{rk}_M\|$, as the cardinality of its support. Likewise, the size of dgm_M is defined as the cardinality of its support and is denoted by $\|\mathrm{dgm}_M\|$.

▶ Remark 8. For any P-module M, the following holds.

(i) (Monotonicity) $\mathrm{rk}_M(I) \leq \mathrm{rk}_M(J)$ for any pair $I \supseteq J$ in $\mathrm{Int}(P)$ [39, Proposition 3.8].

(ii) For any $I \in \mathrm{Int}(P)$, if $\mathrm{rk}_M(I) = 0$, then $\mathrm{dgm}_M(I) = 0$. Hence, we have $\|\mathrm{dgm}_M\| \leq \|\mathrm{rk}_M\|$ [40, Remark 8].

(iii) If M is interval decomposable and $I \in \mathrm{Int}(P)$, then $\mathrm{dgm}_M(I)$ equals the multiplicity of I in $\mathrm{barc}(M)$ [25, Theorem 2.10].

In establishing results in later sections, we will utilize the following well-known concrete formulation for the limit of a P-module M (see, for instance, [39, Appendix E]):

▶ **Convention 9.** The limit of a P-module M is the pair $(L, (\pi_p)_{p\in P})$ described as:

$$L := \left\{ (\ell_p)_{p\in P} \in \prod_{p\in P} M(p) : \ \forall p \le q \in P, M(p \le q)(\ell_p) = \ell_q \right\}$$

where for each $p \in P$, the map $\pi_p : L \to M(p)$ is the canonical projection. Elements of L are called **sections** of M.

On 2-d grid lattices. A **join** (a.k.a. least upper bound) of $S \subseteq P$ is an element $q_0 \in P$ such that (i) $s \le q_0$, for all $s \in S$, and (ii) for any $q \in P$, if $s \le q$ for all $s \in S$, then $q_0 \le q$. A **meet** (a.k.a. greatest lower bound) of $S \subseteq P$ is an element $r_0 \in P$ such that (i) $r_0 \le s$ for all $s \in S$, and (ii) for any $r \in P$, if $r \le s$ for all $s \in S$, then $r \le r_0$. If a join and a meet of S exist, then they are unique. Hence, whenever they exist, we refer to them as *the* join (denoted by $\bigvee S$) and *the* meet (denoted by $\bigwedge S$), respectively. When S consists of exactly two points s_1 and s_2, we also use $s_1 \vee s_2$ and $s_1 \wedge s_2$ instead of $\bigvee S$ and $\bigwedge S$, respectively. A **lattice** is a poset such that for any pair of elements, their meet and join exist. For any $n \in \mathbb{Z}_{\ge 0}$, let $[n] := \{0 < 1 < \cdots < n\}$. By a **finite *d*-d grid lattice**, we mean a poset isomorphic to the lattice $[n_1] \times [n_2] \times \cdots \times [n_d]$ for some $n_1, n_2, \cdots, n_d \in \mathbb{Z}_{\ge 0}$.

Let p and q be any two points in a poset P. We say that q **covers** p if $p < q$ and there is no $r \in P$ such that $p < r < q$. By $\mathrm{Cov}(p)$, we denote the set of all points $q \in P$ that cover p.

▶ **Lemma 10** ([4, Theorem 5.3]). *Let μ be the Möbius function of the poset $(\mathrm{Int}([n]^2), \supseteq)$. Then, for all $J, I \in \mathrm{Int}([n]^2)$ with $J \supseteq I$,*

$$\mu(J, I) = \begin{cases} 1, & \text{if } I = J, \\ \displaystyle\sum_{\substack{J = \bigwedge S \\ \emptyset \neq S \subseteq \mathrm{Cov}(I)}} (-1)^{|S|}, & \text{otherwise,} \end{cases}$$

where $|S|$ denotes the cardinality of S. Note that when $I \neq J$ and there is no nonempty $S \subseteq \mathrm{Cov}(I)$ such that $J = \bigwedge S$, the sum above is empty, and thus $\mu(J, I) = 0$. We also remark that $\bigwedge S$ represents the smallest interval in $[n]^2$ that contains all the intervals in S.

A subposet $L \subset P$ (resp. $U \subset P$) is called a **lower (resp. upper) fence** of P if L is connected, and for any $q \in P$, the intersection $L \cap q^{\downarrow}$ (resp. $U \cap q^{\uparrow}$) is nonempty and connected [25, Definition 3.1].

▶ **Lemma 11** ([25, Proposition 3.2]). *Let L and U be a lower and an upper fence of a connected poset P, respectively. Given any P-module M, we have*

$$\varprojlim M \cong \varprojlim M|_L \quad \text{and} \quad \varinjlim M \cong \varinjlim M|_U.$$

Let I be any subset of P. By $\min(I)$ and $\max(I)$, we denote the collections of minimal and maximal elements of I, respectively. A **zigzag poset** of n points is $\bullet_1 \leftrightarrow \bullet_2 \leftrightarrow \ldots \bullet_{n-1} \leftrightarrow \bullet_n$ where $\leftrightarrow$ stands for either $\le$ or $\ge$.

▶ **Definition 12** ([25, Definition 3.5]). *For $I \in \mathrm{Int}([n]^2)$ with $\min(I) = \{p_0, p_1, \ldots, p_k\}$, and $\max(I) = \{q_0, q_1, \ldots, q_l\}$, we define the following two zigzag posets*

$$\begin{aligned} \min_{\mathrm{ZZ}}(I) &:= \{p_0 < (p_0 \vee p_1) > p_1 < (p_1 \vee p_2) > \cdots < (p_{k-1} \vee p_k) > p_k\} \\ &= \min(I) \cup \{p_i \vee p_{i+1} : i = 0, \ldots, k-1\}, \\ \max_{\mathrm{ZZ}}(I) &:= \{q_0 > (q_0 \wedge q_1) < q_1 > (q_1 \wedge q_2) < \cdots > (q_{l-1} \wedge q_l) > q_l\} \\ &= \max(I) \cup \{q_i \wedge q_{i+1} : i = 0, \ldots, l-1\}, \end{aligned}$$

which are lower and upper fences of I, respectively [25, Section 3.2].

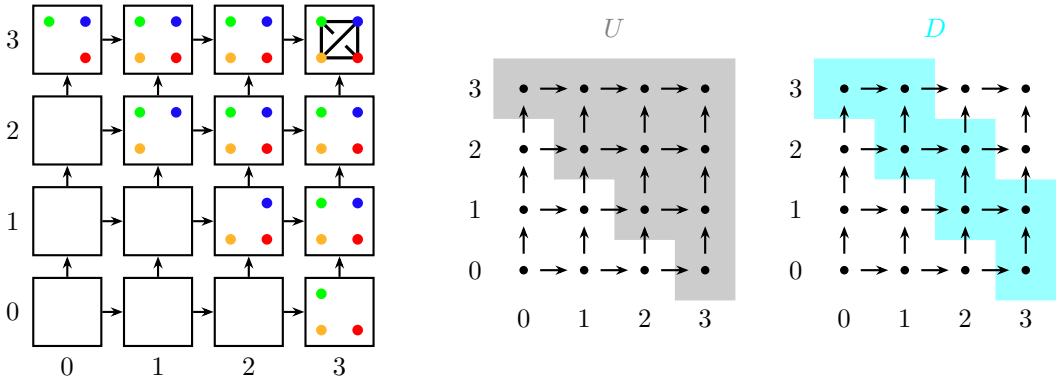


Figure 1 For $n = 3$, the filtration over $[n]^2$ defined in Equation (6), along with the intervals U and D of $[n]^2$, as described in Equation (9).

3 Super-Polynomial Growth of the GPD

The goal of this section is to establish the following theorem.

▶ **Theorem 13** (Super-polynomial Growth of the GPD). *Let $d, m \in \mathbb{Z}_{\geq 0}$ with $d \geq 2$. There does* not *exist $k \in \mathbb{Z}_{\geq 0}$ such that, for* all *finite filtrations $\mathcal{F}$ over $\mathbb{R}^d$ containing N simplices, $\|\mathrm{dgm}_m(\mathcal{F})\|$ belongs to $O(N^k)$.*

Let K be an abstract simplicial complex and let $m \in \mathbb{Z}_{\geq 0}$. Let $C_m(K;\mathbb{F})$ and $Z_m(K;\mathbb{F})$ be the group of m-chains and that of m-cycles of K, respectively. Let K^m be the m-skeleton of K.

To prove the statement, we construct a sequence $(\mathcal{F}_n)$ of filtrations where $\mathcal{F}_n$ contains $O(n^t)$ simplices for some $t \in \mathbb{Z}_{\geq 0}$, but $\|\mathrm{dgm}_m(\mathcal{F}_n)\| \notin O(n^k)$ for *any* $k \in \mathbb{Z}_{\geq 0}$. By Remarks 4 and 7, it suffices to construct filtrations $\mathcal{F}_n$ over the discrete posets $[n]^d$.

▶ Remark 14. In [45], the *size* of a finite d-parameter filtration of a simplicial complex K is defined as the total number of birth indices across all $\sigma \in K$. Even if we adopt this quantity as N in the above theorem, the statement remains true.

We also remark that some techniques used in the following proof are similar to those employed in the proof of [22, Theorem B]. We first prove the statement for the case of $d = 2$.

Proof of Theorem 13 when $d = 2$. We consider the cases of $m = 0$ and $m \geq 1$ separately.

Case 1 ($m = 0$). Fix $n \in \mathbb{Z}_{\geq 0}$. Let $K := K_n$ be the simplicial complex on the vertex set $K^0 = \{z_0, z_1, \ldots, z_n\}$ that consists of the 1-simplices $\{z_i, z_j\}$ $(0 \leq i < j \leq n)$ and their faces. We define a filtration $\mathcal{F} := \mathcal{F}_n$ of K over $[n]^2$ as follows. For $(i,j) \in [n]^2$, let (see Figure 1):

$$\mathcal{F}_{(i,j)} := \begin{cases} \emptyset, & \text{if } i+j < n \\ K^0 - \{\{z_i\}\}, & \text{if } i+j = n \\ K^0, & \text{if } i+j > n \text{ and } (i,j) \neq (n,n) \\ K, & \text{if } (i,j) = (n,n). \end{cases} \tag{6}$$

Let $M := \mathrm{H}_0(\mathcal{F};\mathbb{F}) : [n]^2 \to \mathbf{vec}_{\mathbb{F}}$. Then, we have:

$$M(i,j) \cong \begin{cases} 0, & \text{if } i+j < n \\ \mathbb{F}^n, & \text{if } i+j = n \\ \mathbb{F}^{n+1}, & \text{if } i+j > n \text{ and } (i,j) \neq (n,n) \\ \mathbb{F}, & \text{if } (i,j) = (n,n). \end{cases} \tag{7}$$

Let $i \in [n]$. Then, for every $(x,y) \in [n]^2$ that covers $(i, n-i)$, we have that

$$M((i,n-i) \leq (x,y)) \quad = \quad \iota_i^n : \quad \mathbb{F}^n \to \mathbb{F}^{n+1}$$
$$(v_1, \dots, v_n) \mapsto (v_1, \dots, v_i, 0, v_{i+1}, \dots, v_n). \tag{8}$$

In what follows, we use ι_i in place of ι_i^n. Now, we consider the following two intervals of $[n]^2$ (see Figure 1):

$$U := \{(i,j) \in [n]^2 : \ n \leq i+j\} \ \text{ and } \ D := \{(i,j) \in [n]^2 : n \leq i+j \leq n+1\}. \tag{9}$$

By proving the following claims, we complete the proof for the case of $m = 0$.

▷ Claim 15. If $J \in \mathrm{Int}([n]^2)$ is not a proper subset of U, then $\mathrm{rk}_M(J) = 0$.

Proof. First, assume that there exists a point $p \in J - U$. Then, by the construction of M, $\dim(M(p)) = 0$ and thus $\mathrm{rk}_M(J) = 0$ by monotonicity of rk_M (Remark 8 (i)). Next, assume that $J = U$. Since $\mathrm{rk}_M(U) = \mathrm{rank}(\varprojlim M|_U \to \varinjlim M|_U)$, it suffices to show $\varprojlim M|_U = 0$. Again, since $D = \min_{\mathrm{ZZ}}(U)$, by Lemma 11, it suffices to show that $\varprojlim M|_D = 0$. By Equation (8), the diagram $M|_D$ is given by

$$\mathbb{F}^n \xrightarrow{\iota_0} \mathbb{F}^{n+1} \xleftarrow{\iota_1} \mathbb{F}^n \xrightarrow{\iota_1} \mathbb{F}^{n+1} \xleftarrow{\iota_2} \mathbb{F}^n \xrightarrow{\iota_2} \mathbb{F}^{n+1} \xleftarrow{\iota_3} \mathbb{F}^n \xrightarrow{\iota_3} \cdots \xleftarrow{\iota_n} \mathbb{F}^n.$$

From this, it is not difficult to see that $\varprojlim M|_D = 0$: For $p \in D$, let $\pi_p : \varprojlim M|_D \to M(p)$ be the canonical projection. For any $i = 0, \dots, n$ and for any $v \in \varprojlim M|_D$, let $\pi_{(i,n-i)}(v) =: (v_1^i, v_2^i, \dots, v_n^i) \in \mathbb{F}^n = M(i, n-i)$. Then, Equations (2) and (8) imply the following pair of equalities for each $i = 1, \dots, n$:

$$\begin{aligned} \pi_{(i,n-i+1)}(v) &= \iota_{i-1} \circ \pi_{(i-1,n-i+1)}(v) &&= (v_1^{i-1}, \dots, v_{i-1}^{i-1}, 0, v_i^{i-1}, v_{i+1}^{i-1}, \dots, v_n^{i-1}), \\ \pi_{(i,n-i+1)}(v) &= \iota_i \circ \pi_{(i,n-i)}(v) &&= (v_1^i, \dots, v_{i-1}^i, v_i^i, 0, v_{i+1}^{i-1}, \dots, v_n^i), \end{aligned}$$

and thus $v_i^i = v_i^{i-1} = 0$ for $i = 1, \dots n$. From this, we deduce that $v_j^i = 0$ for all $i = 0, 1, \dots, n$ and $j = 1, 2, \dots, n$. Therefore, $\pi_{(i,n-i)}(v) = 0$ for all $i = 0, 1, \dots, n$ and in turn $v = 0$. Since $v \in \varprojlim M|_D$ was arbitrary, we have $\varprojlim M|_D = 0$. ◁

Consider the following intervals of $[n]^2$ (see Figure 2):

$$U_i := U - \{(i, n-i)\} \ \text{ for each } i \in [n]. \tag{10}$$

▷ Claim 16. Let $J \in \mathrm{Int}([n]^2)$ such that $\bigcap_{i=0}^n U_i \subseteq J \subsetneq U$. Then, $\mathrm{rk}_M(J) = 1$.

Proof. Since $J \subsetneq U$, there exists $i \in [n]$ such that $J \subseteq U_i$. Fix such i. Since $\max(J) = \{(n,n)\}$, by Lemma 11, we have the isomorphism $\varinjlim M|_J \overset{(*)}{\cong} M(n,n)$. Observe that for every $p \in J \subseteq U_i$, $\{z_i\} \in \mathcal{F}_p$. Hence, by Convention 9 and the definition of M, we have that

$$([z_i])_{p \in J} \in \varprojlim M|_J \subset \bigoplus_{p \in J} \mathrm{H}_0(\mathcal{F}_p; \mathbb{F}).$$

Clearly, this tuple $([z_i])_{p \in J}$ maps to $[z_i] \in \mathrm{H}_0(\mathcal{F}_{(n,n)}; \mathbb{F}) = M(n,n)$ via the limit-to-colimit map of M over J, followed by the isomorphism $(*)$. Since $[z_i] \in M(n,n)$ is nonzero and $\dim M(n,n) = 1$ (cf. Equation (7)), we have that $\mathrm{rk}_M(J) = 1$, as desired. ◁

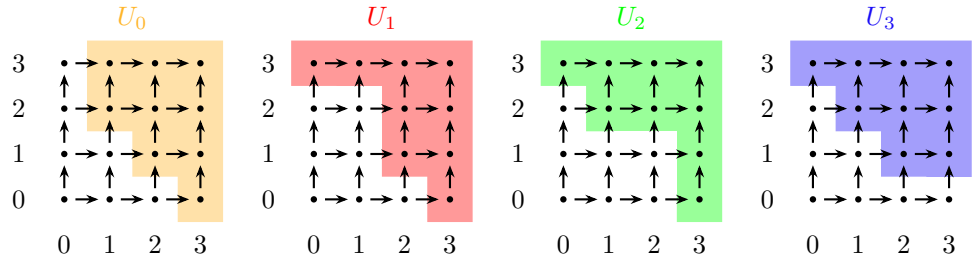


Figure 2 For $n = 3$, the intervals U_0, U_1, U_2, U_3 of $[n]^2$ that are defined in Equation (10).

▷ Claim 17. $\mathrm{dgm}_M(U_i) = 1$ for every $i \in [n]$.

Proof. We have:

$$\begin{aligned} 1 &= \mathrm{rk}_M(U_i) && \text{by Claim 16} \\ &= \sum_{\substack{I \in \mathrm{Int}([n]^2) \\ I \supseteq U_i}} \mathrm{dgm}_M(I) && \text{by Equation (4)} \\ &= \mathrm{dgm}_M(U_i) && \text{by Claim 15 and Remark 8 (ii).} \end{aligned}$$ ◁

For each nonempty $S \subset [n]$, we define $U_S := \bigcap_{i \in S} U_i$, which is an interval of $[n]^2$.

▷ Claim 18. For each nonempty $S \subset [n]$, we have that $\mathrm{dgm}_M(U_S) = (-1)^{|S|+1}$.

Proof. Let $I := U_S$. When $|S| = 1$, there exists $i \in [n]$ such that $I = U_i$. We already proved that $\mathrm{dgm}_M(U_i) = (-1)^{1+1} = 1$. If $|S| > 1$, then

$$\begin{aligned} \mathrm{dgm}_M(I) &= \sum_{\substack{J \in \mathrm{Int}(P) \\ J \supseteq I}} \mu(J, I) \cdot \mathrm{rk}_M(J) && \text{by Equation (5)} \\ &= \sum_{\substack{J \in \mathrm{Int}(P) \\ U \supsetneq J \supseteq I}} \mu(J, I) \cdot \mathrm{rk}_M(J) && \text{by Claim 15} \\ &= \sum_{\substack{J \in \mathrm{Int}(P) \\ U \supsetneq J \supseteq I}} (-1)^{|J-I|} \cdot \mathrm{rk}_M(J) && \text{by Lemma 10} \\ &= \sum_{S' \subsetneq S} (-1)^{|S'|} && \text{by Claim 16, letting } S' = J - I \\ &= (1-1)^{|S|} - (-1)^{|S|} = (-1)^{|S|+1} && \text{by the binomial theorem} \end{aligned}$$

as desired. The number of nonempty subsets $S \subset [n]$ is $2^{n+1} - 1$ and each S corresponds to the interval U_S such that $\mathrm{dgm}_M(U_S) \neq 0$. Hence, $\|\mathrm{dgm}_0(\mathcal{F})\| = \|\mathrm{dgm}_M\| \geq 2^{n+1} - 1$. In particular, there is no $k \in \mathbb{Z}_{\geq 0}$ such that $2^{n+1} - 1 \in O(N^k)$ for $N := (n^2 + n)/2$, the number of simplices in the filtration $\mathcal{F}$, completing the proof. ◁

Case 2 ($m \geq 1$). Let $K' := K_{n,m}$ be the smallest simplicial complex on the vertex set $\{x_0, \cdots, x_n, y_0, \cdots, y_m\}$, containing all m-simplices τ, τ_{ij}, and $(m+1)$-simplices σ_{ij} that are the underlying sets of the following oriented simplices (see Figure 3a):

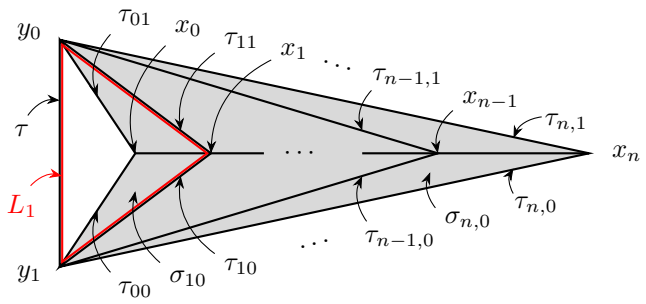
$$\begin{aligned} \tau^+ &:= [y_0, \cdots, y_m] \\ \tau_{ij}^+ &:= [x_i, y_0, \cdots, y_{j-1}, y_{j+1}, \cdots, y_m], && \text{for } 0 \leq i \leq n,\ 0 \leq j \leq m, \\ \sigma_{ij}^+ &:= [x_{i-1}, x_i, y_0, \cdots, y_{j-1}, y_{j+1}, \cdots, y_m], && \text{for } 1 \leq i \leq n,\ 0 \leq j \leq m. \end{aligned}$$

For $0 \le k \le n$, let L_k be the m-dimensional subcomplex of K', whose m-simplices are exactly τ and $\{\tau_{kj} : 0 \le j \le m\}$ (see Figure 3a).

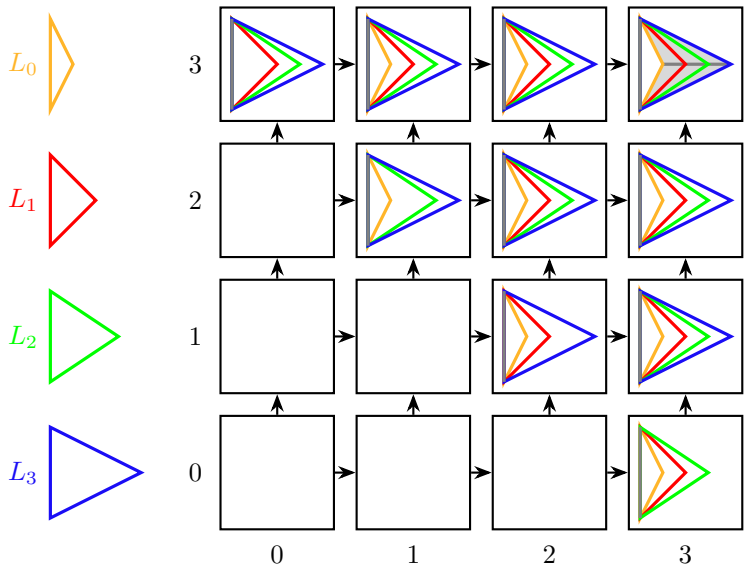
We define the filtration $\mathcal{F}' := \mathcal{F}'_{n,m} : [n]^2 \to \Delta K'$ $(= \Delta K_{n,m})$ as (see Figure 3b):

$$\mathcal{F}'_{(i,j)} := \begin{cases} \emptyset, & \text{if } i+j < n \\ \bigcup_{k \in [n]-\{i\}} L_k, & \text{if } i+j = n \\ \bigcup_{k \in [n]} L_k, & \text{if } i+j > n \text{ and } (i,j) \neq (n,n) \\ K', & \text{if } (i,j) = (n,n). \end{cases} \tag{11}$$

Note that, for each $0 \le i \le n$, the m-chain $c_i = \tau^+ - \sum_{j=0}^{m} (-1)^j \tau_{ij}^+$ in $C_m(K'; \mathbb{F})$ satisfies $\partial c_i = 0$, i.e. c_i is an m-cycle. Moreover, $\mathrm{H}_m(L_i; \mathbb{F})$, which is isomorphic to $Z_m(L_i; \mathbb{F})$, is the 1-dimensional vector space generated by the homology class $[c_i]$.



(a) For $m = 1$, the simplicial complex $K' = K_{n,m}$ given in **Case 2**.



(b) For $m = 1$ and $n = 3$, the filtration $\mathcal{F}' = \mathcal{F}'_{n,m}$ given in **Case 2**.

Figure 3 Illustration of the constructions described in **Case 2**.

Let $N := \mathrm{H}_m(\mathcal{F}'; \mathbb{F}) : [n]^2 \to \mathbf{vec}_{\mathbb{F}}$. We claim that N is isomorphic to M, given in Equation (7). Indeed,

$$N(i,j) \cong \begin{cases} 0, & \text{if } i+j < n \\ \langle [c_0], \cdots, [c_{i-1}], [c_{i+1}], \cdots, [c_n] \rangle \cong \mathbb{F}^n, & \text{if } i+j = n \\ \langle [c_0], \cdots, [c_n] \rangle \cong \mathbb{F}^{n+1}, & \text{if } i+j > n \text{ and } (i,j) \neq (n,n) \\ \mathbb{F}, & \text{if } (i,j) = (n,n) \end{cases}$$

and for each pair $p \leq q$ in $[n]^2$, $N(p \leq q) \cong M(p \leq q)$. Therefore, there is no $k \in \mathbb{Z}_{\geq 0}$ such that $\|\mathrm{dgm}_m(\mathcal{F}')\| = \|\mathrm{dgm}_N\| = \|\mathrm{dgm}_M\| = 2^{n+1} - 1 \notin O(N^k)$ for N, the number of simplices in the filtration $\mathcal{F}'$, which belongs to $O(n^{m+2})$. ◀

In order to prove Theorem 13 for the case where $d > 2$, we utilize a result from [12, Section 3.2] and techniques present in [22, Section 3]. See the full version [38] for details.

▶ Remark 19. Notably, the supports of the GPDs we have considered contain many overlapping intervals with opposite-signed values. This makes it difficult to interpret the information the GPDs convey, unlike persistence diagrams for one-parameter persistence modules.

4 Super-Polynomial Growth of the GPDs Induced by Finite Metric Spaces

In this section, we show that the sizes of the 1-st GPDs of the sublevel-Rips, sublevel-Čech, degree-Rips, and degree-Čech bifiltrations are *not* bounded by *any* polynomial in the number of points in the input metric space (Theorems 24 and 25). We prove these results by constructing specific point sets in $\mathbb{R}^3$ that induce persistence modules isomorphic, or structurally similar to, those appearing in the proof of Theorem 13. Also, we make use of Rota's Galois connection.

A **Galois connection** between two posets P and Q is a pair of monotone maps $g_1 : P \rightleftarrows Q : g_2$ satisfying $g_1(p) \leq q$ iff $p \leq g_2(q)$ for any $p \in P$ and $q \in Q$. Let $f : P \to Q$ be any function. For any $h : P \to \mathbb{Z}$, the **pushforward** of h along f is the function $f_\sharp h : Q \to \mathbb{Z}$ given by $f_\sharp h(q) = \sum_{p \in f^{-1}(q)} h(p)$. For any $\ell : Q \to \mathbb{Z}$, the **pullback** of ℓ along f is the function $f^\sharp \ell : P \to \mathbb{Z}$ defined by $f^\sharp \ell(p) = (\ell \circ f)(p)$. For any poset P, let ∂_P denote the Möbius inversion operator over P. To say that the GPD of a P-module M exists is equivalent to stating that $\partial_{\mathrm{Int}(P)} \mathrm{rk}_M$ is well-defined and equals dgm_M.

▶ **Lemma 20** ([33, Theorem 3.1]). *Let P be a finite poset, Q be any poset, and $g_1 : P \rightleftarrows Q : g_2$ be a Galois connection. Then, $\partial_Q \circ g_2^\sharp = {g_1}_\sharp \circ \partial_P$.*

Lemma 20 is crucial for establishing the following lemma, whose second item is essential for the proofs of both Theorem 24 and Theorem 25. See the full version [38] for the proof.

▶ **Lemma 21.** *Let P be a finite poset, M be a P-module, and $I \subset P$ be an interval of P. Then,*

(i) *for every $J \in \mathrm{Int}(I)$, we have $\mathrm{dgm}_{M|_I}(J) = \sum_{\substack{J' \in \mathrm{Int}(P) \\ J \in \Pi(J' \cap I)}} \mathrm{dgm}_M(J')$, where $\Pi(J' \cap I)$ denotes the coarsest partition of $J \cap I$ into disjoint intervals of I.*

(ii) *Furthermore, if P and I are finite 2-d grid, then $\|\mathrm{dgm}_M\| \geq \|\mathrm{dgm}_{M|_I}\|$.*

▶ Remark 22. Lemma 21 (ii) does not generally hold in higher dimensions, i.e. for $d \geq 3$, there exist d-d grids P and I with $I \subseteq P$ and a P-module M such that $\|\mathrm{dgm}_M\| < \|\mathrm{dgm}_{M|_I}\|$.

4.1 Sublevel-Rips and Sublevel-Čech Bifiltrations

For $r \in \mathbb{R}_{\geq 0}$, the **Vietoris-Rips complex** of a metric space (X, d_X) at scale r is defined as

$$\mathrm{Rips}((X,\mathsf{d}_X))_r = \left\{ \text{finite } \sigma \subset X : \max_{x,y\in\sigma} \mathsf{d}_X(x,y) \leq r \right\}.$$

Now assume that $X \subset \mathbb{R}^d$, where $\mathbb{R}^d$ is equipped with a metric d, and $\mathsf{d}_X = \mathsf{d}|_{X\times X}$. The **Čech complex of (X, d_X) in $(\mathbb{R}^d, \mathsf{d})$ at scale r** is

$$\check{\mathrm{C}}\mathrm{ech}((X,\mathsf{d}_X))_r = \left\{ \text{finite } \sigma \subset X : \bigcap_{x\in\sigma} B_r(x) \neq \emptyset \right\},$$

where $B_r(x) = \{y \in \mathbb{R}^d : \mathsf{d}(x,y) \leq r\}$.

▶ **Lemma 23** ([32, Lemma VII]). *Let $(\mathbb{R}^d, \rho)$ be the Euclidean space with the supremum metric ρ. For any finite $X \subset \mathbb{R}^d$ and any $r \in \mathbb{R}_{\geq 0}$, we have* $\check{\mathrm{C}}\mathrm{ech}((X,\rho|_{X\times X}))_r = \mathrm{Rips}((X,\rho|_{X\times X}))_{2r}$.

Let $\gamma : X \to \mathbb{R}$ be a function. For any $a \in \mathbb{R}$ and any $r \in \mathbb{R}_{\geq 0}$ (see e.g. [18]),

(i) the **sublevel-Rips complex at scale r and threshold a** is $\mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma))_{a,r} := \mathrm{Rips}(\gamma^{-1}((-\infty,a]))_r$. The **sublevel-Rips bifiltration** of (X,d,γ) is the simplicial filtration $\mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma)) := \{\mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma))_{a,r}\}_{a\in\mathbb{R},r\in\mathbb{R}_{\geq 0}}$ over $\mathbb{R}\times\mathbb{R}_{\geq 0}$.

(ii) the **sublevel-Čech complex at scale r and threshold a** is $\check{\mathrm{C}}\mathrm{ech}^{\downarrow}((X,\mathsf{d},\gamma))_{a,r} := \check{\mathrm{C}}\mathrm{ech}(\gamma^{-1}((-\infty,a]))_r$. The **sublevel-Čech bifiltration** of (X,d,γ) is the simplicial filtration $\check{\mathrm{C}}\mathrm{ech}^{\downarrow}((X,\mathsf{d},\gamma)) := \{\check{\mathrm{C}}\mathrm{ech}^{\downarrow}((X,\mathsf{d},\gamma))_{a,r}\}_{a\in\mathbb{R},r\in\mathbb{R}_{\geq 0}}$ over $\mathbb{R}\times\mathbb{R}_{\geq 0}$.

Let

$$\Gamma_X := \mathrm{im}(\gamma) \subset \mathbb{R} \quad \text{and} \quad T_X := \mathrm{im}(\mathsf{d}) \subset \mathbb{R}_{\geq 0}. \tag{12}$$

By Remarks 4 and 7, for each $m \in \mathbb{Z}_{\geq 0}$, the size of the GPD of $M := \mathrm{H}_m(\mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma)),\mathbb{F}) : \mathbb{R}\times\mathbb{R}_{\geq 0} \to \mathbf{vec}_{\mathbb{F}}$ equals $\|\mathrm{dgm}_{M|_{\Gamma_X\times T_X}}\|$.

▶ **Theorem 24** (Super-polynomial GPD of Sublevel-Rips and Sublevel-Čech filtrations). *There does* not *exist $k \in \mathbb{Z}_{\geq 0}$ such that for* every *metric space (X,d) and* every *function $\gamma : X \to \mathbb{R}$, $\|\mathrm{dgm}_1(\mathcal{F}_{(X,\mathsf{d},\gamma)})\|$ belongs to $O(|X|^k)$, where $\mathcal{F}_{(X,\mathsf{d},\gamma)} \in \{\mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma)), \check{\mathrm{C}}\mathrm{ech}^{\downarrow}((X,\mathsf{d},\gamma))\}$.*

Given any set $A \subset \mathbb{R}^d$ and $r \in \mathbb{R}$, let $rA := \{ra \in \mathbb{R}^d : a \in A\}$. When $d \geq 2$, let $\pi : \mathbb{R}^d \to \mathbb{R}^2$ be the projection $(x_1,\ldots,x_d) \mapsto (x_1,x_2)$.

Proof Sketch. By Lemma 23, it suffices to show the statement when $\mathcal{F}_{(X,\mathsf{d},\gamma)} = \mathrm{Rips}^{\downarrow}((X,\mathsf{d},\gamma))$. Let $n \geq 1$. We construct an $X_n \subset \mathbb{R}^3$, which consists of $4n(n+2)$ points and inherits the metric d_n from the supremum metric on $\mathbb{R}^3$. Our goal is to show that $\|\mathrm{dgm}_1(\mathrm{Rips}^{\downarrow}((X_n,\mathsf{d}_n,\gamma_n)))\|$ does not belong to $O(n^k)$ for any k, which then does not belong to $O(|X_n|^k)$ for any k. Let $\epsilon_n := 1/n^2$. Let $X_n = \left(\bigcup_{i=0}^{n} Y_i\right) \cup \left(\bigcup_{i=0}^{n-1} Z_i\right)$ (see Figure 4), where, for $i \in [n]$, Y_i is the subset of the plane $z = 2in$ in $\mathbb{R}^3$ whose projected image πY_i onto $\mathbb{R}^2$ is described below, and Z_i is the subset of the plane $z = (2i+1)n$ in $\mathbb{R}^3$ whose projected image πZ_i onto $\mathbb{R}^2$ is described below. Let $A := \{(1,0),(0,1),(-1,0),(0,-1)\}$, $b_j := n - j\epsilon_n$ for $j \in [n]$, $\pi Y_i := \bigcup_{\substack{0\leq j\leq n \\ j\neq i}} b_j A$, and $\pi Z_i := b_0 A$. Note that $|X_n| = \left|\bigcup_{i=0}^{n} Y_i\right| + \left|\bigcup_{i=0}^{n-1} Z_i\right| =$

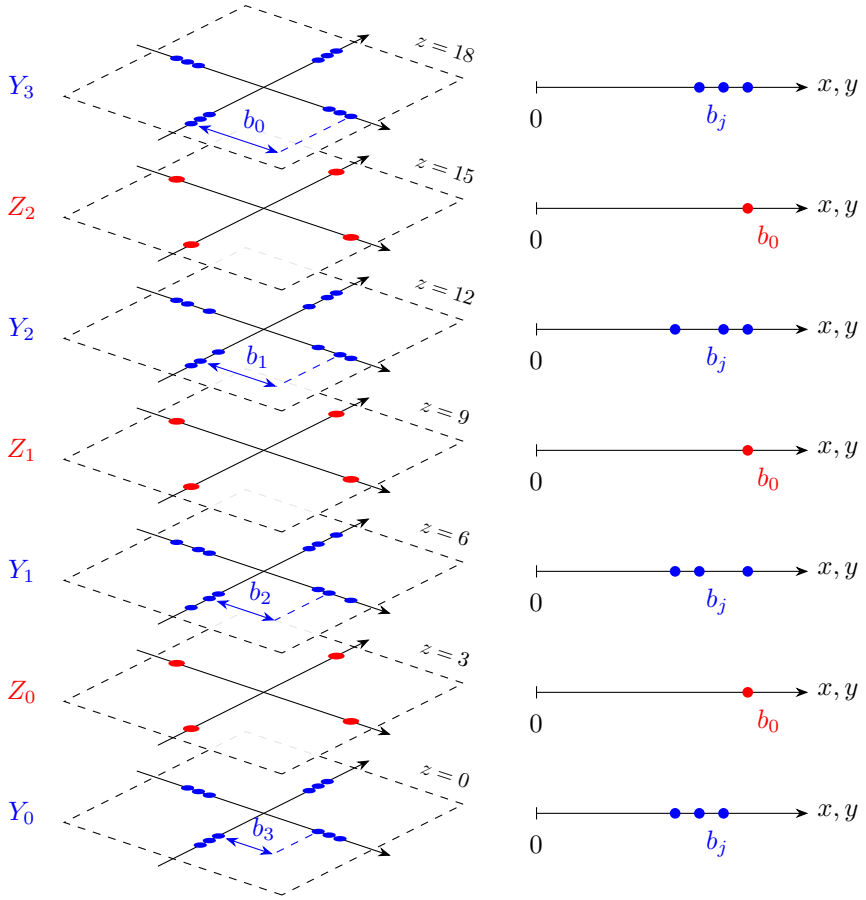


Figure 4 For $n = 3$, the set X_n that is defined in the proof of Theorem 24.

$4n(n+1) + 4n = 4n(n+2)$. Let $\gamma_n : X_n \to \mathbb{R}$ be

$$\gamma_n(w) = \begin{cases} j, & \text{if } w \in \bigcup_{i=0}^{n} Y_i \text{ and } \pi(w) \in b_j A \text{ for some } 0 \le j \le n, \\ n, & \text{if } w \in \bigcup_{i=0}^{n-1} Z_i \end{cases}$$

so that $\Gamma_{X_n} = \mathrm{im}(\gamma_n) = \{0 < 1 < \ldots < n\} \subset \mathbb{R}$. In the full version [38], we show that for $\mathcal{F}_n := \mathrm{Rips}^{\downarrow}((X_n, \mathsf{d}_n, \gamma_n))$ on $\Gamma_{X_n} \times T_{X_n}$ (Figure 5), $\|\mathrm{dgm}_1(\mathcal{F}_n)\|$ does not belong to $O(|X_n|^k)$ for any k. In particular, we identify an interval of $\Gamma_{X_n} \times T_{X_n}$ on which the restriction of $M_n := \mathrm{H}_1(\mathcal{F}_n; \mathbb{F})$ is isomorphic to M from Equation (7), and then apply Lemma 21 (ii). ◀

4.2 Degree-Rips and Degree-Čech Bifiltrations

Let (X, d) be a metric space. For $d \in \mathbb{Z}$ and $r \in \mathbb{R}_{\ge 0}$,

(i) the **Degree-Rips complex at scale r and degree d** [45], denoted $\mathrm{DRips}((X, \mathsf{d}))_{d,r}$, is the maximal subcomplex of $\mathrm{Rips}((X, \mathsf{d}))_r$ whose vertices have degree at least $d-1$ in the 1-skeleton of $\mathrm{Rips}((X, \mathsf{d}))_r$. The **Degree-Rips bifiltration** of (X, d) is the simplicial filtration $\mathrm{DRips}((X, \mathsf{d})) := \{\mathrm{DRips}((X, \mathsf{d}))_{d,r}\}_{d \in \mathbb{Z}^{\mathrm{op}}, r \in \mathbb{R}_{\ge 0}}$ over $\mathbb{Z}^{\mathrm{op}} \times \mathbb{R}_{\ge 0}$.

(ii) The **Degree-Čech complex at scale r and degree d** [30], denoted $\mathrm{DČech}((X, \mathsf{d}))_{d,r}$, is the maximal subcomplex of $\check{\mathrm{C}}\mathrm{ech}((X, \mathsf{d}))_r$ whose vertices have degree at least $d-1$ in the 1-skeleton of $\check{\mathrm{C}}\mathrm{ech}((X, \mathsf{d}))_r$. The **Degree-Čech bifiltration** of (X, d) is the simplicial filtration $\mathrm{DČech}((X, \mathsf{d})) := \{\mathrm{DČech}((X, \mathsf{d}))_{d,r}\}_{d \in \mathbb{Z}^{\mathrm{op}}, r \in \mathbb{R}_{\ge 0}}$ over $\mathbb{Z}^{\mathrm{op}} \times \mathbb{R}_{\ge 0}$.

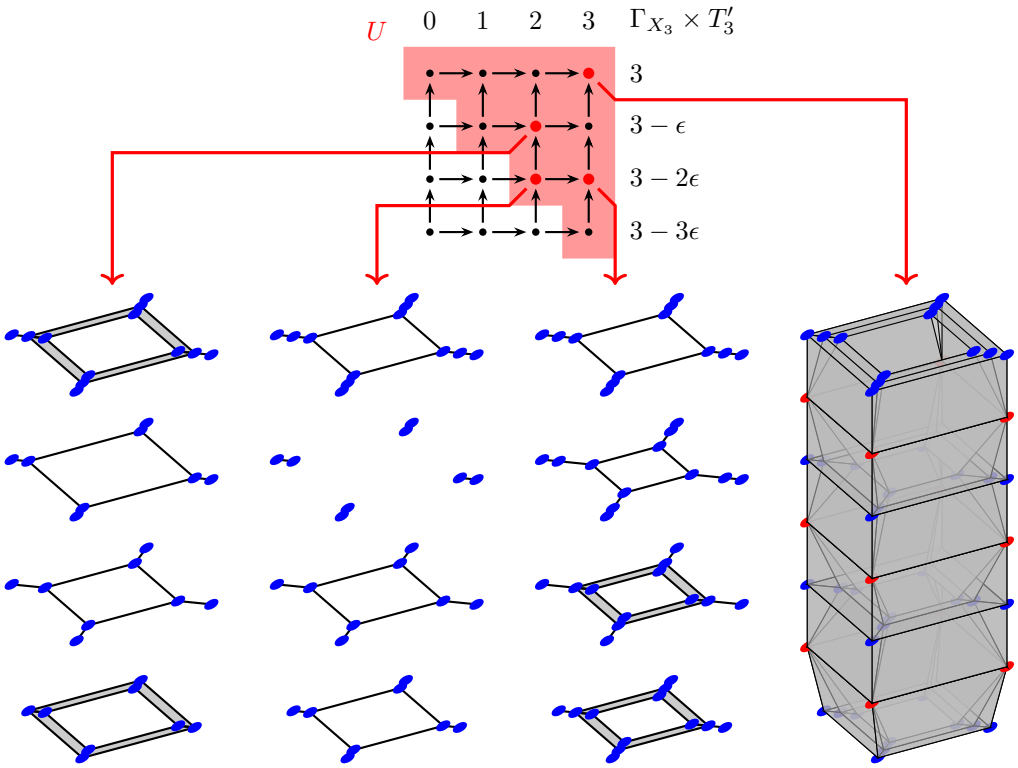


Figure 5 Partial illustration of the filtration $\mathcal{F}_3$ on $\Gamma_{X_3} \times T'_3$. The body of $\mathcal{F}_3(3,3)$ is homotopy equivalent to the cylinder $S^1 \times [0,1]$. The 1-cycles in $\mathcal{F}_3(2, 3-\epsilon_3)$, $\mathcal{F}_3(2, 3-2\epsilon_3)$, and $\mathcal{F}_3(3, 3-2\epsilon_3)$ become homologous in $\mathcal{F}_3(3,3)$.

We consider the finite subposets

$$J_X := \{|X| - 1 < |X| - 2 < \ldots < 1\} \subset \mathbb{Z}^{\mathrm{op}} \quad \text{and} \quad T_X := \mathrm{im}(\mathsf{d}) \subset \mathbb{R}_{\geq 0}.$$

By Remarks 4 and 7, for each $m \in \mathbb{Z}_{\geq 0}$, the size of the GPD of $M := \mathrm{H}_m(\mathrm{DRips}((X,\mathsf{d})), \mathbb{F}) : \mathbb{Z}^{\mathrm{op}} \times \mathbb{R}_{\geq 0} \to \mathbf{vec}_{\mathbb{F}}$ equals $\|\mathrm{dgm}_{M|_{J_X \times T_X}}\|$.

▶ **Theorem 25** (Super-polynomial GPD of Degree-Rips and Degree-Čech filtrations). *There does* not *exist $k \in \mathbb{Z}_{\geq 0}$ such that for* every *finite metric space (X,d), $\|\mathrm{dgm}_1(\mathcal{F}_{(X,\mathsf{d})})\|$ belongs to $O(|X|^k)$, where $\mathcal{F}_{(X,\mathsf{d})} \in \{\mathrm{DRips}((X,\mathsf{d})), \mathrm{D\check{C}ech}((X,\mathsf{d}))\}$.*

Proof Sketch. By Lemma 23, it suffices to show the statement when $\mathcal{F}_{(X,\mathsf{d})} = \mathrm{DRips}((X,\mathsf{d}))$.

Let $n \geq 1$. We construct an $X_n \subset \mathbb{R}^3$, which consists of $4(n+1)^2$ points and inherits the metric d_n from the supremum metric on $\mathbb{R}^3$. Our goal is to show that $\|\mathrm{dgm}_1(\mathrm{DRips}((X_n,\mathsf{d}_n)))\|$ does not belong to $O(n^k)$ for any k, which then does not belong to $O(|X_n|^k)$ for any k. Let $\epsilon_n := 1/n^2$. Let $X_n := \bigcup_{i=0}^{n} Y_i$ (see Figure 6), where, for $i \in [n]$, Y_i is the subset of the plane $z = (n+\epsilon_n)i$ in $\mathbb{R}^3$ whose projected image πY_i onto $\mathbb{R}^2$ is described below. Let $A := \{(1,0),(0,1),(-1,0),(0,-1)\}$, $B := \{(1,\epsilon_n),(-\epsilon_n,1),(-1,-\epsilon_n),(\epsilon_n,-1)\}$, and $a := n - n\epsilon_n$. For $j \in [n+1]$, let $b_j := a + j\epsilon_n = n + (j-n)\epsilon_n$. Let

$$\pi Y_i := aA \cup \Big(\bigcup_{\substack{0 \leq j \leq n \\ j \neq i}} (a+b_i)A \Big) \cup (a+b_{i+1})B.$$

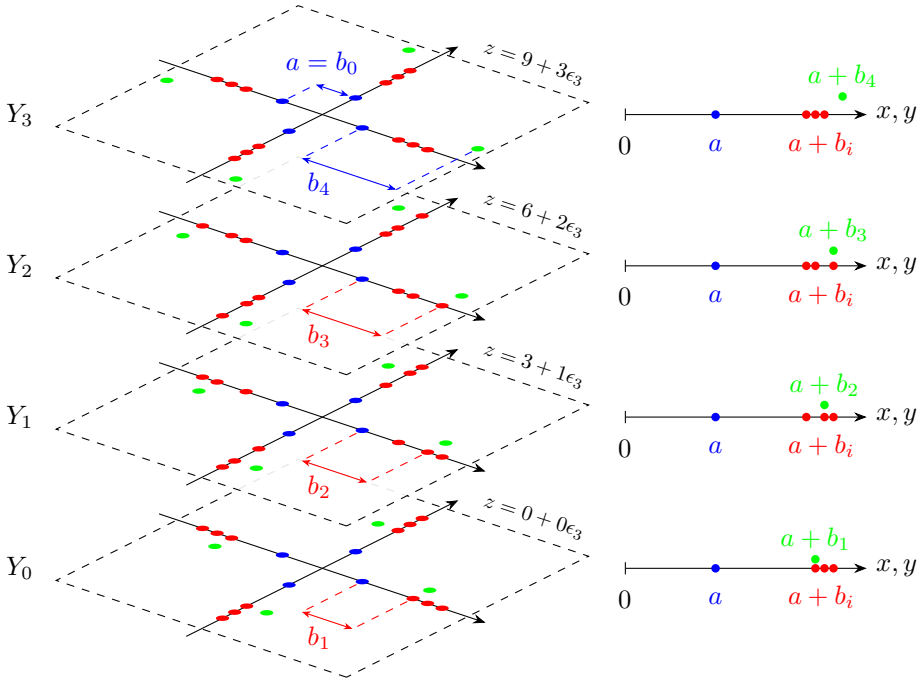


Figure 6 For $n = 3$, the set X_n that is defined in the proof of Theorem 25.

In the full version [38], we show that for the filtration $\mathcal{F}_n := \mathrm{DRips}((X_n, \mathsf{d}_n))$, $\|\mathrm{dgm}_1(\mathcal{F}_n)\|$ does not belong to $O(|X_n|^k)$ for any k. For this, we identify an interval of $\Gamma_{X_n} \times T_{X_n}$ on which the restriction of $M_n := \mathrm{H}_1(\mathcal{F}_n; \mathbb{F})$ is similar to M from Equation (7). Then, we prove claims analogous to Claim 15 through 18, and apply Lemma 21 (ii). ◀

It is known that, for $m \in \mathbb{Z}_{\geq 0}$, adding m integers has time complexity $O(m)$ [37]. Hence, as a direct corollary of Theorems 13, 24, and 25, we obtain:

▶ **Corollary 26.** *Let $n, d \in \mathbb{Z}_{\geq 0}$, $d \geq 2$, and $I \in \mathrm{Int}([n]^d)$. For an arbitrary $[n]^d$-module M, computing $\mathrm{dgm}_M(I)$ via the Möbius inversion of the GRI of M requires at least $\Theta(2^n)$ in time. In particular, M can arise as the homology of a simplicial filtration over $[n]^d$, with the number of simplices being polynomial in n.*

Proof. Let M be the persistence module defined in the proof of Theorem 13 for $d = 2$ and $m = 0$. Let $I := U_{[n]}$, as described immediately above Claim 18 in that proof. By Lemma 10, we have $\mathrm{dgm}_M(I) = \sum_{S' \subsetneq [n]} (-1)^{|S'|} = (-1)^{n+2}$. This sum contains $2^{n+1} - 1$ nonzero summands, requiring $\Theta(2^n)$ time for computation. ◀

5 Conclusion

We showed that the size of the GPD can be super-polynomial in the number of simplices in a given multi-parameter filtration, and such sizes can arise even from well-known constructions. Furthermore, we noted that the GRI can be "locally densely supported" (see Claim 18), which suggests that using the Möbius inversion of the GRI to compute the GPD can be intractable. These findings highlight the need to compute the GPD (or any invariant approximating it) directly, without relying on the GRI.

From a different perspective, developing an output-sensitive algorithm for computing the GPD might be a more practical approach, potentially extending or adapting ideas from [52]. Moreover, combining optimization techniques to address challenges in computing the GPD appears to be a promising research direction. In line with this, [20] proposes a method for "sparsifying" the GPD via gradient descent.

It also remains an interesting question whether special types of multi-parameter filtrations not explored in this work can also give rise to GPDs of super-polynomial size. Examples include multicover and barycentric (subdivision) bifiltrations [60].

References

1 Ángel Javier Alonso, Michael Kerber, and Primoz Skraba. Probabilistic analysis of multi-parameter persistence decompositions into intervals. In *40th International Symposium on Computational Geometry (SoCG 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

2 Hideto Asashiba, Mickaël Buchet, Emerson G Escolar, Ken Nakashima, and Michio Yoshiwaki. On interval decomposability of 2D persistence modules. *Computational Geometry*, 105:101879, 2022. doi:10.1016/J.COMGEO.2022.101879.

3 Hideto Asashiba, Emerson G Escolar, Ken Nakashima, and Michio Yoshiwaki. Approximation by interval-decomposables and interval resolutions of persistence modules. *Journal of Pure and Applied Algebra*, 227(10):107397, 2023.

4 Hideto Asashiba, Emerson G Escolar, Ken Nakashima, and Michio Yoshiwaki. On approximation of 2D persistence modules by interval-decomposables. *Journal of Computational Algebra*, 6:100007, 2023.

5 Hideto Asashiba, Etienne Gauthier, and Enhao Liu. Interval replacements of persistence modules. *arXiv preprint*, 2024. arXiv:2403.08308.

6 Gorô Azumaya. Corrections and supplementaries to my paper concerning Krull-Remak-Schmidt's theorem. *Nagoya Mathematical Journal*, 1:117–124, 1950.

7 Leo Betthauser, Peter Bubenik, and Parker B Edwards. Graded persistence diagrams and persistence landscapes. *Discrete & Computational Geometry*, 67(1):203–230, 2022. doi:10.1007/S00454-021-00316-1.

8 Benjamin Blanchette, Thomas Brüstle, and Eric J Hanson. Homological approximations in persistence theory. *Canadian Journal of Mathematics*, 76(1):66–103, 2024.

9 Magnus Botnan and William Crawley-Boevey. Decomposition of persistence modules. *Proceedings of the American Mathematical Society*, 148(11):4581–4596, 2020.

10 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & Geometric topology*, 18(6):3133–3204, 2018.

11 Magnus Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of Algebras and Related Structures*, pages 77–150. EMS Press, 2023.

12 Magnus Bakke Botnan, Steffen Oppermann, and Steve Oudot. Signed barcodes for multi-parameter persistence via rank decompositions and rank-exact resolutions. *Foundations of Computational Mathematics*, 2024. doi:10.1007/s10208-024-09672-9.

13 Magnus Bakke Botnan, Steffen Oppermann, Steve Oudot, and Luis Scoccola. On the bottleneck stability of rank decompositions of multi-parameter persistence modules. *Advances in Mathematics*, 451:109780, 2024.

14 Peter Bubenik and Iryna Hartsock. Topological and metric properties of spaces of generalized persistence diagrams. *Journal of Applied and Computational Topology*, pages 1–53, 2024.

15 Chen Cai, Woojin Kim, Facundo Mémoli, and Yusu Wang. Elder-rule-staircodes for augmented metric spaces. *SIAM Journal on Applied Algebra and Geometry*, 5(3):417–454, 2021. doi:10.1137/20M1353605.

16 Gunnar Carlsson. Topology and Data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

17 Gunnar Carlsson and Mikael Vejdemo-Johansson. *Topological data analysis with applications*. Cambridge University Press, 2021.

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 Mathieu Carrière and Andrew Blumberg. Multiparameter persistence image for topological machine learning. *Advances in Neural Information Processing Systems*, 33:22432–22444, 2020.

20 Mathieu Carriere, Seunghyun Kim, and Woojin Kim. Sparsification of the generalized persistence diagrams for scalability through gradient descent. *arXiv preprint*, 2024. doi:10.48550/arXiv.2412.05900.

21 Wojciech Chacholski, Andrea Guidolin, Isaac Ren, Martina Scolamiero, and Francesca Tombari. Effective computation of relative homological invariants for functors over posets. *arXiv preprint*, 2022. arXiv:2209.05923.

22 Nate Clause, Woojin Kim, and Facundo Memoli. The Generalized Rank Invariant: Möbius invertibility, Discriminating power, and Connection to Other Invariants. *arXiv preprint*, 2024. arXiv:2207.11591v5.

23 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.

24 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.

25 Tamal K Dey, Woojin Kim, and Facundo Mémoli. Computing generalized rank invariant for 2-parameter persistence modules via zigzag persistence and its applications. *Discrete & Computational Geometry*, 71(1):67–94, 2024. doi:10.1007/S00454-023-00584-Z.

26 Tamal K Dey, Aman Timalsina, and Cheng Xin. Computing generalized ranks of persistence modules via unfolding to zigzag modules. *arXiv preprint*, 2024. doi:10.48550/arXiv.2403.08110.

27 Tamal Krishna Dey and Yusu Wang. *Computational topology for data analysis*. Cambridge University Press, 2022.

28 Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002. doi:10.1007/S00454-002-2885-2.

29 Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2008.

30 Herbert Edelsbrunner and Georg Osang. The multi-cover persistence of euclidean balls. *Discrete & Computational Geometry*, 65:1296–1313, 2021. doi:10.1007/S00454-021-00281-9.

31 Emerson G Escolar and Woojin Kim. Barcoding invariants and their equivalent discriminating power. *arXiv preprint*, 2024. arXiv:2412.04995.

32 Robert Ghrist and Abubakr Muhammad. Coverage and hole-detection in sensor networks via homology. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 254–260. IEEE, 2005. doi:10.1109/IPSN.2005.1440933.

33 Aziz Burak Gülen and Alexander McCleary. Galois connections in persistent homology. *arXiv preprint*, 2022. arXiv:2201.06650.

34 Aziz Burak Gülen, Facundo Mémoli, and Zhengchao Wan. Orthogonal Möbius inversion and grassmannian persistence diagrams. *arXiv preprint*, 2023. arXiv:2311.06870.

35 Heather A Harrington, Nina Otter, Hal Schenck, and Ulrike Tillmann. Stratifying multiparameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 3(3):439–471, 2019. doi:10.1137/18M1224350.

36 Yasuaki Hiraoka, Ken Nakashima, Ippei Obayashi, and Chenguang Xu. Refinement of interval approximations for fully commutative quivers. *arXiv preprint*, 2023. arXiv:2310.03649.

37 Emil Jeřábek. A note on the complexity of addition. *arXiv preprint*, 2023. arXiv:2306.08513.

38 Donghan Kim, Woojin Kim, and Wonjun Lee. Super-polynomial growth of the generalized persistence diagram. *arXiv preprint*, 2025. arXiv:2412.04889.

39 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *Journal of Applied and Computational Topology*, 5(4):533–581, 2021. doi:10.1007/S41468-021-00075-1.

40 Woojin Kim and Facundo Mémoli. Persistence over posets. *Notices of the American Mathematical Society*, 70(08), 2023.

41 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.

42 Woojin Kim and Samantha Moore. Bigraded Betti numbers and generalized persistence diagrams. *Journal of Applied and Computatioal Topology*, 2024. doi:10.1007/s41468-024-00180-x.

43 Ryan Kinser. The rank of a quiver representation. *Journal of Algebra*, 320(6):2363–2387, 2008.

44 Claudia Landi. The rank invariant stability via interleavings. In *Research in computational topology*, pages 1–10. Springer, 2018.

45 Michael Lesnick and Matthew Wright. Interactive visualization of 2d persistence modules. *arXiv preprint arXiv:1512.00180*, 2015. arXiv:1512.00180.

46 David Loiseaux, Mathieu Carrière, and Andrew Blumberg. A framework for fast and stable representations of multiparameter persistent homology decompositions. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates, Inc., 2023.

47 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, 2024.

48 Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

49 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.

50 Facundo Mémoli, Anastasios Stefanou, and Ling Zhou. Persistent cup product structures and related invariants. *To appear in Journal of Applied and Computational Topology, arXiv preprint arXiv:2211.16642*, 2022. arXiv:2211.16642.

51 Ezra Miller. Homological algebra of modules over posets. *arXiv preprint*, 2020. arXiv:2008.00063.

52 Dmitriy Morozov and Amit Patel. Output-sensitive computation of generalized persistence diagrams for 2-filtrations. *arXiv preprint arXiv:2112.03980*, 2021. arXiv:2112.03980.

53 Soham Mukherjee, Shreyas N Samaga, Cheng Xin, Steve Oudot, and Tamal K Dey. D-gril: End-to-end topological learning with 2-parameter persistence. *arXiv preprint*, 2024. doi:10.48550/arXiv.2406.07100.

54 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.

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 Amit Patel and Tatum Rask. Poincaré duality for generalized persistence diagrams of (co) filtrations. *Journal of Applied and Computational Topology*, pages 1–16, 2024.

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, 1964.

58 Florian Russold and Michael Kerber. Graphcode: Learning from multiparameter persistent homology using graph neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

59 Luis Scoccola, Siddharth Setlur, David Loiseaux, Mathieu Carrière, and Steve Oudot. Differentiability and optimization of multiparameter persistent homology. In *41st International Conference on Machine Learning (ICML 2024)*, volume 235, pages 43986–44011. PMLR, 2024.

60 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.

61 Richard P Stanley. Enumerative combinatorics volume 1 second edition. *Cambridge studies in advanced mathematics*, 2011.

62 Ashleigh Linnea Thomas. *Invariants and metrics for multiparameter persistent homology*. PhD thesis, Duke University, 2019.

63 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. URL: https://jmlr.org/papers/v21/19-054.html.

64 Cheng Xin, Soham Mukherjee, Shreyas N Samaga, and Tamal K Dey. Gril: A 2-parameter persistence based vectorization for machine learning. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 313–333. PMLR, 2023.

65 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005. doi:10.1007/S00454-004-1146-Y.

Recognizing 2-Layer and Outer k-Planar Graphs

Yasuaki Kobayashi
Hokkaido University, Sapporo, Japan

Yuto Okada
Nagoya University, Japan

Alexander Wolff
Universität Würzburg, Germany

Abstract

The *crossing number* of a graph is the least number of crossings over all drawings of the graph in the plane. Computing the crossing number of a given graph is NP-hard, but fixed-parameter tractable (FPT) with respect to the natural parameter. Two well-known variants of the problem are *2-layer crossing minimization* and *circular crossing minimization*, where every vertex must lie on one of two *layers*, namely two parallel lines, or a circle, respectively. In both cases, edges are drawn as straight-line segments. Both variants are NP-hard, but admit FPT-algorithms with respect to the natural parameter.

In recent years, in the context of beyond-planar graphs, a local version of the crossing number has also received considerable attention. A graph is *k-planar* if it admits a drawing with at most k crossings per edge. In contrast to the crossing number, recognizing k-planar graphs is NP-hard even if $k = 1$ and hence not likely to be FPT with respect to the natural parameter k.

In this paper, we consider the two above variants in the local setting. The k-planar graphs that admit a straight-line drawing with vertices on two layers or on a circle are called *2-layer k-planar* and *outer k-planar* graphs, respectively. We study the parameterized complexity of the two recognition problems with respect to the natural parameter k. For $k = 0$, the two classes of graphs are exactly the caterpillars and outerplanar graphs, respectively, which can be recognized in linear time. Two groups of researchers independently showed that outer 1-planar graphs can also be recognized in linear time [Hong et al., Algorithmica 2015; Auer et al., Algorithmica 2016]. One group asked explicitly whether outer 2-planar graphs can be recognized in polynomial time.

Our main contribution consists of XP-algorithms for recognizing 2-layer k-planar graphs and outer k-planar graphs, which implies that both recognition problems can be solved in polynomial time for every fixed k. We complement these results by showing that recognizing 2-layer k-planar graphs is XNLP-complete and that recognizing outer k-planar graphs is XNLP-hard. This implies that both problems are W[t]-hard for every t and that it is unlikely that they admit FPT-algorithms. On the other hand, we present an FPT-algorithm for recognizing 2-layer k-planar graphs where the order of the vertices on one layer is specified.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases 2-layer k-planar graphs, outer k-planar graphs, recognition algorithms, local crossing number, bandwidth, FPT, XNLP, XP, W[t]

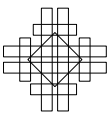
Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.65

Related Version *Full Version*: https://arxiv.org/abs/2412.04042

Funding *Yasuaki Kobayashi*: Supported by JSPS KAKENHI Grant Numbers JP23K28034, JP24H00686, and JP24H00697.
Yuto Okada: Supported by JST SPRING, Grant Number JPMJSP2125 and JSPS KAKENHI, Grant Number JP22H00513 (Hirotaka Ono).

Acknowledgements We thank Hirotaka Ono and the AFSA project (Creation and Organization of Innovative Algorithmic Foundations for Social Advancement) for supporting our work. We thank Boris Klemz and Marie Diana Sieper for useful discussions.

© Yasuaki Kobayashi, Yuto Okada, and Alexander Wolff;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

When evaluating the quality of a graph drawing, one of the established metrics is the number of crossings, whose importance is supported by user experiments [42]. Unfortunately, computing the crossing number of a given graph, that is, the minimum number of crossings over all drawings of the graph, is NP-hard [27], even for graphs that become planar after removal of a single edge [14]. On the other hand, the problem is fixed-parameter tractable (FPT) with respect to the natural parameter, that is, the number of crossings [29, 33]. Many variants of the crossing number have been studied; see Schaefer's survey [44]. Two variants with geometric restrictions have attracted considerable attention: *2-layer crossing minimization* and *circular* (or *convex*, or *1-page*) *crossing minimization*, where the placement of the vertices is restricted to two parallel lines (called *layers*) and to a circle, respectively. In both cases, edges are drawn as straight-line segments. Circular crossing minimization is NP-hard, but admits FPT-algorithms with respect to the natural parameter [6, 34]. In practice, often the so-called *sifting heuristic* is used [7]. Circular crossing minimization can be seen as a special case of a *book embedding problem*, where vertices must lie on a straight line, the *spine* of the book, and each edge must be drawn on one of a given number of halfplanes called *pages* whose intersection is the spine. In this setting, crossing minimization is interesting even if the order of the vertices along the spine is given [8, 38].

The 2-layer variant comes in two settings: *one-sided crossing minimization* (*OSCM*) and *two-sided crossing minimization* (*TSCM*). In OSCM, the input consists of a (bipartite) graph and a linear order for the vertices on one side of the bipartition; the task is to find a linear order for the vertices on the other side that minimizes the total number of crossings. In TSCM, the linear orders on both layers can be chosen freely. OSCM is an important step in the so-called Sugiyama framework for drawing hierarchical graphs [46], that is, graphs where each vertex is assigned to a specific layer. OSCM was the topic of the Parameterized Algorithms and Computational Experiments Challenge (PACE[1]) 2024. Both OSCM and TSCM are NP-hard; OSCM even for the disjoint union of 4-stars [39] and for trees [19]. On the positive side, OSCM admits a subexponential FPT-algorithm; it runs in $O(k2^{\sqrt{2k}} + n)$ time [35]. TSCM also admits an FPT-algorithm; it runs in $2^{O(k)} + n^{O(1)}$ time [36].

In the context of beyond-planar graphs, a local version of the crossing number has also received considerable attention [18, 22]. A graph is *k-planar* if it admits a drawing with at most k crossings per edge. The *local crossing number* of a graph is the smallest k such that the graph is k-planar. The recognition of 1-planar graphs has long been known to be NP-hard [28]. Later, it turned out that the recognition of k-planar graphs is NP-hard for every k [47]. Hence, it is unlikely that FPT- or XP-algorithms exist with respect to the natural parameter k. On the other hand, recognizing 1-planar graphs is fixed-parameter tractable with respect to tree-depth and cyclomatic number [5]. The problem remains NP-hard, however, for graphs of bounded bandwidth (and hence, pathwidth and treewidth). The local crossing number has also been studied in the context of book embeddings [1, 37].

In this paper, we study the above-mentioned geometric restrictions, but with respect to the *local* crossing number. The resulting graph classes are called *2-layer k-planar* graphs and *outer k-planar* graphs; see Figure 1. The former were studied by Angelini, Da Lozzo, Förster, and Schneck [3]. Among others, they gave bounds on the edge density of these graphs and characterized 2-layer k-planar graphs with the maximum edge density for $k \in \{2, 4\}$. They concluded that "the general recognition and characterization of 2-layer k-planar graphs

[1] https://pacechallenge.org/2024/

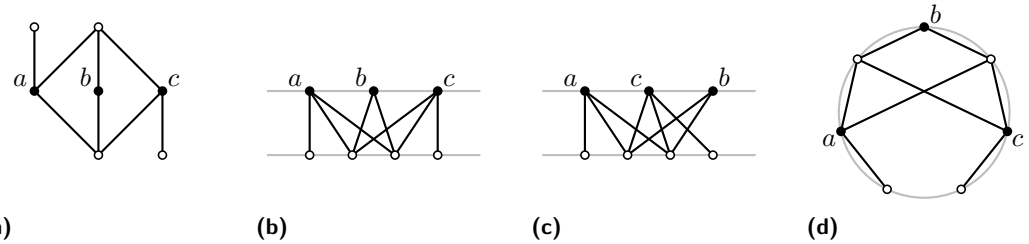


Figure 1 Drawings of the same bipartite graph with optimal local crossing number in different settings: (a) planar drawing, (b) 2-layer 2-planar drawing without restriction, (c) 2-layer 3-planar drawing where the vertex order on the upper layer is fixed, (d) outer 1-planar drawing.

remain important open problems". According to Schaefer's survey [44] on crossing numbers, Kainen [32] introduced the "local outerplanar crossing number", which minimizes, over all circular drawings, the largest number of crossings along any edge. Outer k-planar graphs have been studied by Pach and Tóth [40], who showed that any outer k-planar graph with n vertices has at most $4.1\sqrt{k}n$ edges. For $k \leq 3$, they established a better bound $(k+3)(n-2)$, which is tight for $k \in \{1,2\}$. For $k \geq 5$, the constant factor was later improved to $\sqrt{243/40} \approx 2.46$ [2].

We study the parameterized complexity of the two recognition problems with respect to the natural parameter k. For $k = 0$, the two classes of graphs are exactly the caterpillars and outerplanar graphs, respectively, which can be recognized in linear time. There are also linear-time algorithms for recognizing outer 1-planar graphs [4,30]. The authors of [30] posed the existence of polynomial-time algorithms for recognizing outer 2-planar graphs as an open problem. A partial answer has been given by Hong and Nagamochi [31], showing that *full* outer 2-planar graphs can be recognized in linear time. Outer k-planar drawings are *full* if no crossing appears on the boundary of the outer face. The authors of [16] generalized this result and showed that, for every integer k, full outer k-planarity is testable in $O(f(k) \cdot n)$ time, for a computable function f. They also showed that outer k-planar graphs can be recognized in quasi-polynomial time, which implies that, for every integer k, testing outer k-planarity is not NP-hard unless the Exponential-Time Hypothesis fails.

Parameterized complexity. We assume that the reader is familiar with basic concepts in parameterized complexity theory (see [17,20,25] for definitions of these concepts). Zehavi [48] gives a survey specifically on the parameterized analysis of crossing minimization problems. The class XNLP consists of all parameterized problems that can be solved non-deterministically in time $f(k)n^{O(1)}$ and space $f(k)\log n$, where f is some computable function, n is the input size, and k is the parameter. A parameterized problem $L_2 \subseteq \Sigma^* \times \mathbb{N}$ is said to be XNLP-*hard* if for any $L_1 \in$ XNLP, there is a *parameterized logspace reduction* from L_1 to L_2, that is, there is an algorithm $\mathcal{A}$ and computable functions f and g that satisfy the following: Given $(x_1, k_1) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ computes $(x_2, k_2) \in \Sigma^* \times \mathbb{N}$ such that $(x_1, k_1) \in L_1$ if and only if $(x_2, k_2) \in L_2$, $k_2 \leq g(k_1)$, and $\mathcal{A}$ runs in space $O(f(k_1) + \log |x_1|)$. A parameterized problem is said to be XNLP-*complete* if it is XNLP-hard and belongs to XNLP. The class XNLP contains the class W[t] for every $t \geq 1$ [13]. Moreover, Pilipczuk and Wrochna [41] conjectured that an XNLP-hard problem does not admit an algorithm that runs in $n^{f(k)}$ time and $f(k) \cdot n^{O(1)}$ space for a computable function f, where n is the size of the instance and k is the parameter. We refer to [13,23] for more information.

Recently, the authors of [10] showed the first graph-drawing problem to be XNLP-complete, namely *ordered level planarity*, parameterized by the number of levels. Ordered level planarity is a restricted version of level planarity, where for each level, the vertices on that level are given in order (and the problem is to route the edges in a y-monotone and crossing-free way).

Our contribution. We present XP-algorithms for recognizing 2-layer k-planar graphs and outer k-planar graphs, which implies that both recognition problems can be solved in polynomial time for every fixed k; see Sections 4.1 and 5.1, respectively. This solves the open problem regarding the recognition of outer 2-planar graphs posed by the authors of [30]. We complement these results by showing that recognizing 2-layer k-planar graphs is XNLP-complete even for trees (Section 4.2) and that recognizing outer k-planar graphs is XNLP-hard (Section 5.3). This implies that both problems are W[t]-hard for every t [13] and that it is unlikely that they admit FPT-algorithms. On the other hand, we present an FPT-algorithm for recognizing 2-layer k-planar graphs where the order of the vertices on one layer is specified; see Figure 1c and Section 3.1. We prove that two *edge-weighted* versions of this problem are NP-hard; see Section 3.2. Finally, we show that the local circular crossing number cannot be approximated even for graphs that are almost trees (that is, graphs with feedback vertex number 1); see Section 5.2. We conclude with open problems; see Section 6.

The proofs of statements marked with a (clickable) $\star$ are in the full version.

2 Preliminaries

Let G be a graph. We let $V(G)$ and $E(G)$ denote the sets of vertices and edges of G, respectively. For a vertex v of G, let $N_G(v)$ be the set of neighbors of v in G, and let $\delta_G(v)$ be the set of edges incident to v. For $U \subseteq V(G)$, we define $N_G(U) = (\bigcup_{v\in U} N_G(v)) \setminus U$, $N_G[U] = N_G(U) \cup U$, and $\delta_G(U) = \bigcup_{v\in U} \delta_G(v)$. We may omit the subscript G when it is clear from the context. The subgraph of G induced by $U \subseteq V(G)$ is denoted by $G[U]$. A vertex is called a *leaf* if it has exactly one neighbor.

We use $[\ell]$ as shorthand for $\{1, 2, \dots, \ell\}$. Let $n = |V(G)|$, and let $\sigma\colon V(G) \to [n]$ be a linear order of the vertices of G (i.e., a bijection between $V(G)$ and $[n]$). For $\{u, v\} \in E$, the *stretch* of edge $\{u, v\}$ in σ is defined as $|\sigma(u) - \sigma(v)|$. The *bandwidth* of σ (with respect to G) is the maximum stretch of an edge of G in σ. The *bandwidth* of G, denoted by $\mathrm{bw}(G)$, is the minimum integer k such that G has a linear order σ of V with bandwidth at most k.

We define a *circular drawing* of a graph G to be a cyclic order $D = (v_1, \dots, v_n)$ of $V(G)$. We say that an edge $\{v_i, v_j\}$ with $i < j$ *pierces* a pair of (not necessarily adjacent) vertices $\{v_{i'}, v_{j'}\}$ with $i' < j'$ if either $1 \le i < i' < j < j' \le n$ or $1 \le i' < i < j' < j \le n$ holds. In particular, if $\{v_{i'}, v_{j'}\}$ is an edge of G, we say that $\{v_i, v_j\}$ *crosses* $\{v_{i'}, v_{j'}\}$. For an edge e, let $\mathrm{cr}_D(e)$ denote the number of edges that cross e in D. A circular drawing D is *k-planar* (or an *outer k-planar drawing*) if every edge crosses at most k edges in D. Since whether two edges cross is determined only by the cyclic vertex order, in this paper, we allow the edges to be drawn arbitrarily.

Let G be a bipartite graph with $V(G) = X \cup Y$, $X \cap Y = \emptyset$, and $E(G) \subseteq X \times Y$. A *2-layer drawing* of G is a pair $D = (<_X, <_Y)$ of (strict) linear orders $<_X$ and $<_Y$ defined on X and on Y, respectively. A *crossing* in D is defined by a pair of edges $\{x, y\}$ and $\{x', y'\}$ with distinct endpoints $x, x' \in X$ and distinct endpoints $y, y' \in Y$ such that $x <_X x'$ and $y' <_Y y$. The notation $\mathrm{cr}_D(e)$ is defined as above. Moreover, for two distinct edges e and e', let $\mathrm{cr}_D(e, e')= 1$ if e crosses e'; $\mathrm{cr}_D(e, e') = 0$ otherwise. For distinct $x, x' \in X$, we say that x is *to the left of* x' in D if $x <_X x'$. Equivalently, we say that x' is *to the right of* x. The *leftmost* (resp. *rightmost*) of X in D is the smallest (resp. largest) vertex in X under the linear order $<_X$. We also use these notions for vertices in Y. For an integer $k \ge 0$, a 2-layer drawing D of G is said to be *k-planar* if, for each edge e of G, $\mathrm{cr}_D(e) \le k$. For (not necessarily disjoint) vertex sets $A, B \subseteq X \cup Y$ and 2-layer drawings D_A of $G[A]$ and D_B of $G[B]$, we say that D_A is *compatible* with D_B (or equivalently D_B is compatible with D_A) if, for every pair $\{z, z'\} \subseteq A \cap B$ of vertices that both are in the same set of the bipartition $Z \in \{X, Y\}$, we have that $z <_Z z'$ in D_A if and only if $z <_Z z'$ in D_B.

3 Recognizing 2-Layer k-Planar Graphs – The One-Sided Case

In this section, we design an FPT-algorithm for recognizing 2-layer k-planar graphs when the order of the vertices on one layer is given as input. The problem is defined as follows.

Problem: One-Sided k-Planarity
Input: A bipartite graph $(X \cup Y, E)$, an integer $k \geq 0$, and a linear order $<_X$ of X.
Question: Does Y admit a linear order $<_Y$ such that $(<_X, <_Y)$ is a 2-layer k-planar drawing of $(X \cup Y, E)$?

Degree reduction. Let $G = (X \cup Y, E)$ be a bipartite graph, and let k be a non-negative integer. We describe two simple reduction rules that yield an equivalent instance of One-Sided k-Planarity where every vertex in X has degree at most $2k + 2$.

▶ **Observation 1** (⋆). *Let $G = (X \cup Y, E)$ be a bipartite graph that contains a vertex with more than $2k + 2$ non-leaf neighbors. Then G is not 2-layer k-planar.*

▶ **Lemma 2** (⋆). *Let $(G, <_X, k)$ be an instance of One-Sided k-Planarity. If G contains a vertex $v \in X$ with $\deg(v) > 2k + 2$ and with a leaf neighbor $y \in Y$, then $(G, <_X, k)$ is a* YES-instance if and only if $(G - y, <_X, k)$ is a YES-instance.

Hence, in the following, we assume that every vertex in X has degree at most $2k + 2$.

3.1 An FPT-Algorithm

Let n be the number of vertices in G. In this section, we prove the following result.

▶ **Theorem 3.** *One-Sided k-Planarity can be solved in time $2^{O(k \log k)} n^{O(1)}$, that is, One-Sided k-Planarity is fixed-parameter tractable when parameterized by k.*

We assume that G has no isolated vertices; otherwise, we simply remove them. For a 2-layer drawing D of a subgraph G' of G, we say that D *respects* $<_X$ if, for every $x, x' \in V(G')$, it holds that x is to the left of x' in D if and only if $x <_X x'$.

We first give a simpler algorithm with running time $2^{O(k^2 \log k)} n^{O(1)}$. Let $x_1, \ldots, x_{|X|}$ be the vertices of X appearing in this order in $<_X$. If $|X| < 2k + 1$, then $|Y| \leq 2k(2k + 2)$, and we can simply enumerate all the possible 2-layer drawings in time $2^{O(k^2 \log k)} n^{O(1)}$. Thus, we assume $|X| \geq 2k + 1$. Let $\ell = 2k$. For $i \in [|X| - \ell]$, let $X_{\leq i} = \{x_1, \ldots, x_{i+\ell}\}$ and $X_i = \{x_i, \ldots, x_{i+\ell}\}$. Correspondingly, let $G_{\leq i} = G[N[X_{\leq i}]]$ and $G_i = G[N[X_i]]$. Our algorithm recursively decides whether $G_{\leq i}$ admits a 2-layer k-planar drawing that extends a prescribed partial drawing D of G_i. To be more precise, let D be a 2-layer k-planar drawing of G_i that respects $<_X$. Given $i \in [|X| - \ell]$, a partial drawing D of G_i respecting $<_X$, and a function $\chi \colon \delta(X_i) \to \{0, \ldots, k\}$, we define a Boolean value $\texttt{draw}(i, D, \chi)$ to be `true` if and only if $G_{\leq i}$ admits a 2-layer k-planar drawing $D_{\leq i}$ such that

- $D_{\leq i}$ is compatible with D and
- for every edge $e \in \delta(X_i)$, it holds that $\chi(e) = \mathrm{cr}_{D_{\leq i}}(e)$.

Our goal is to compute $\texttt{draw}(|X| - \ell, D, \chi)$ for some partial drawing D of $G_{|X|-\ell}$ and χ, which is done by the following dynamic programming algorithm.

For the base case $i = 1$, we compute the table entries by brute force: For each possible 2-layer k-planar drawing D of G_1 respecting $<_X$ and for every function $\chi \colon \delta(X_1) \to \{0, \ldots, k\}$, we set $\texttt{draw}(1, D, \chi) = \texttt{true}$ if $\chi(e) = \mathrm{cr}_D(e)$ for every $e \in \delta(X_1)$; otherwise, we set $\texttt{draw}(1, D, \chi) = \texttt{false}$. Now we show that, in any 2-layer k-planar drawing, if two edges have endpoints in X that are far apart, then the edges do not cross.

▶ **Lemma 4 (⋆).** *If $e = (x_p, y)$ and $f = (x_q, y')$ are distinct edges such that $p + \ell < q$, then, in every 2-layer k-planar drawing $D = (<_X, <_Y)$ of G, it holds that $y <_Y y'$ or that $y = y'$.*

This suggests the following recurrence for the dynamic program.

▶ **Lemma 5.** *Let $2 \leq i \leq |X| - \ell$, let D be a 2-layer k-planar drawing of G_i respecting $<_X$, and let $\chi\colon \delta(X_i) \to \{0, \dots, k\}$ such that $\chi(e) = \mathrm{cr}_D(e)$ for every $e \in \delta(x_{i+\ell})$. Then,*

$$\texttt{draw}(i, D, \chi) = \bigvee_{D_{i-1}, \chi_{i-1}} \texttt{draw}(i-1, D_{i-1}, \chi_{i-1}),$$

where the D_{i-1} are taken over all 2-layer k-planar drawings of G_{i-1} that are compatible with D, and the χ_{i-1} are taken over all functions $\delta(X_{i-1}) \to \{0, \dots, k\}$ that satisfy

$$\chi_{i-1}(e) = \chi(e) - \sum_{f \in \delta(x_{i+\ell})} \mathrm{cr}_D(e, f)$$

for every edge $e \in \delta(X_i) \cap \delta(X_{i-1})$.

Proof. Suppose that $\texttt{draw}(i, D, \chi) = \texttt{true}$. Then, there is a 2-layer k-planar drawing $D_{\leq i}$ of $G_{\leq i}$ that is compatible with D and, for every edge $e \in \delta(X_i)$, it holds that $\chi(e) = \mathrm{cr}_{D_{\leq i}}(e)$. We need to show that there exists a triplet $t = (i-1, D_{i-1}, \chi_{i-1})$ such that $\texttt{draw}(t) = \texttt{true}$. Let $D_{\leq i-1}$ and D_{i-1} be subdrawings of $D_{\leq i}$ induced by $G_{\leq i-1}$ and G_{i-1}, respectively. Then $D_{\leq i-1}$ is compatible with D_{i-1}. By Lemma 4, edges incident to $x_{i+\ell}$ do not cross edges incident to x_{i-1}. Thus, for every edge $e \in \delta(x_{i-1})$, we have $\mathrm{cr}_{D_{\leq i}}(e) = \mathrm{cr}_{D_{\leq i-1}}(e)$. Moreover, every edge $e \in \delta(X_i) \cap \delta(X_{i-1})$ has exactly $\chi(e) - \sum_{f \in \delta(x_{i+\ell})} \mathrm{cr}_D(e, f)$ crossings in $D_{\leq i-1}$. Define χ_{i-1} by setting $\chi_{i-1}(e) = \mathrm{cr}_{D_{\leq i-1}}(e)$ for every edge $e \in \delta(X_{i-1})$. Then, by definition, $\texttt{draw}(i-1, D_{i-1}, \chi_{i-1}) = \texttt{true}$ since $D_{\leq i-1}$ is a 2-layer k-planar drawing of $G_{\leq i-1}$ compatible with D_{i-1} and, for every $e \in \delta(X_{i-1})$, it trivially holds that $\chi_{i-1}(e) = \mathrm{cr}_{D_{\leq i-1}}(e)$.

We omit the converse direction, which readily follows by reversing the above argument. ◀

Our algorithm evaluates the recurrence of Lemma 5 in a dynamic programming manner. To see the runtime bound, observe that, for each $i \in [|X| - \ell]$, the number of possible 2-layer k-planar drawings of G_i is upper bounded by $|N(X_i)|! \leq ((\ell+1) \cdot (2k+2))! = 2^{O(k^2 \log k)}$ and the number of possible functions from $\delta(X_i)$ to $\{0, \dots, k\}$ is upper bounded by $(k+1)^{|\delta(X_i)|} = 2^{O(k^2 \log k)}$. Hence, we can evaluate the recurrence in time $2^{O(k^2 \log k)} n^{O(1)}$.

We can improve the exponential dependency of our running time as follows. Instead of fixing the "window size" to $2k+1$, for every i, we dynamically take the smallest ℓ_i such that $\delta(\{x_i, \dots, x_{i+\ell_i}\})$ consists of at least $2k+1$ edges. It is easy to verify that Lemma 4 (and hence Lemma 5) still holds for this dynamic window size. Since the degree of every vertex in X is at most $2k+2$, we have that $|\delta(\{x_i, \dots, x_{i+\ell_i}\})| \leq 4k+2$. This improves the running time to $2^{O(k \log k)} n^{O(1)}$, completing the proof of Theorem 3.

3.2 NP-Hardness of the Weighted Version

We can generalize One-Sided k-Planarity to weighted settings. Let $G = (X \cup Y, E)$ be a bipartite graph, and let $w\colon E \to \mathbb{N}_{>0}$ be an edge-weight function. A 2-layer drawing D of (G, w) is said to be k-planar if, for each edge e of G, it holds that

$$\sum_{f \text{ crosses } e \text{ in } D} w(f) \leq k. \tag{1}$$

It is straightforward to extend our algorithm to this weighted setting. Although we believe that One-Sided k-Planarity is NP-hard, we can only show the following weaker result.

▶ **Theorem 6** (⋆). *The weighted* One-Sided k-Planarity *is (weakly)* NP*-hard under (1).*

Proof (sketch). The claim is shown by performing a reduction from Partition, which is known to be (weakly) NP-hard [26]. The problem asks, given a set of n integers $A = \{a_1, \ldots, a_n\}$, whether the set can be partitioned into two subsets of equal sum. We construct a bipartite graph G consisting of a path of length 2 with two edges e_0 and e_{n+1} of weight 1, and n isolated edges with weight proportional to the integers in A. By appropriately defining the order $<_X$ on X, we can ensure that each isolated edge crosses either e_0 or e_{n+1}; see Figure 2. Setting k properly induces two balanced subsets of A. ◀

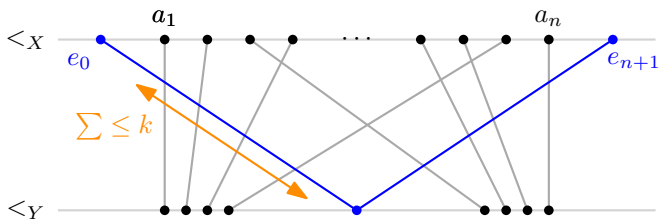


Figure 2 The graph G and $<_X$ we construct. $\sum$ is the sum of weights of edges that cross e_0.

We remark that there is another reasonable definition of crossings in a weighted graph: A 2-layer drawing is defined to be k-planar if, for each edge $e \in E$, it holds that

$$\sum_{f \text{ crosses } e \text{ in } D} w(e) \cdot w(f) \leq k. \tag{2}$$

By making $w(e_0)$ and $w(e_{n+1})$ sufficiently large, a similar reduction will work.

▶ Remark 7. The weighted One-Sided k-Planarity is (weakly) NP-hard under (2).

4 Recognizing 2-Layer k-Planar Graphs – The Two-Sided Case

The algorithm in Theorem 3 exploits the prescribed order $<_X$ on X, which is not specified in the two-sided case. This difference is reflected by the parameterized complexity of the two problems. The two-sided case turns out to be XNLP-hard, meaning that it is unlikely to be fixed-parameter tractable. On the other hand, we design a polynomial-time algorithm for the two-sided case, provided that k is fixed. We use this algorithm also to show that the problem is contained in XNLP. Formally, the problem is defined as follows.

Problem: Two-Sided k-Planarity
Input: A bipartite graph $G = (X \cup Y, E)$ and an integer $k \geq 0$.
Question: Does G admit a 2-layer k-planar drawing?

4.1 An XP-Algorithm

To solve Two-Sided k-Planarity, we extend the algorithm for One-Sided k-Planarity presented in Section 3. Let $G = (X \cup Y, E)$ be a bipartite graph, let n be the number of vertices of G, and let $k \in \mathbb{N}$. We assume that G is connected; otherwise, the problem can be solved independently for each connected component. Moreover, by applying Observation 1 and applying Lemma 2 first to X and then to Y, we assume that every vertex has degree at most $2k + 2$. Our algorithm employs a dynamic programming approach analogous to that

presented in Section 3. Instead of a "window", we specify a subset $X_i \subseteq X$ of $\ell + 1 = 2k + 1$ vertices, which plays the same role as the window $\{x_i, \dots, x_{i+\ell}\}$. However, this subset does not specify the graph $G_{\le i}$ on the left of the window, preventing us from defining the same type of subproblems as there. We overcome this obstacle by applying an idea similar to that of Saxe [43] for recognizing bandwidth-k graphs. To properly define the subproblems, we observe that $N[X_i]$ separates the subdrawings of the components of $G[V(G) \setminus N[X_i]]$ into left and right parts.

▶ **Lemma 8.** *Let $D = (<_X, <_Y)$ be a 2-layer k-planar drawing of G, and let $S \subseteq X$ be a set of $\ell + 1$ vertices that appears consecutively in D. Let x and x' be the leftmost vertex and the rightmost vertex of S in D, respectively. Then, for each component C of $G[V(G) \setminus N[S]]$, the vertices in $C \cap X$ are either entirely to the left of x or entirely to the right of x'.*

Proof. Suppose that C has two vertices $u, v \in X \setminus S$ such that u is to the left of x and v is to the right of x' in D. Let P be a path between u and v in $G[C]$. We can assume that P has exactly two edges, e and f. Observe that each edge incident to a vertex in S crosses either e or f. Since each vertex in S has at least one incident edge, at least one of e and f involves more than k crossings. ◀

Suppose that G has a 2-layer k-planar drawing $D = (<_X, <_Y)$. For a family $\mathcal{D} \subseteq 2^{V(G)}$, we use $\mathcal{D}^X$ as shorthand for $\bigcup_{C \in \mathcal{D}} C \cap X$. Let $x_1, \dots, x_{|X|}$ be the vertices of X appearing in this order in $<_X$. For $1 \le i \le |X| - \ell$, let $\mathcal{C}_i$ be the set of connected components in $G[V(G) \setminus N[\{x_i, \dots, x_{i+\ell}\}]]$. By Lemma 8, we have $\mathcal{C}^X = \{x_1, \dots, x_{i-1}\}$ for some $\mathcal{C} \subseteq \mathcal{C}_i$.

Now, we can formally define our subproblems. Let $S \subseteq X$ with $|S| = \ell + 1$, let D be a 2-layer k-planar drawing of $G[N[S]]$, let $\chi\colon \delta(S) \to \{0, \dots, k\}$, and let $\mathcal{C} \subseteq \mathcal{C}_S$, where $\mathcal{C}_S$ is the set of components in $G[V(G) \setminus N[S]]$. We define a Boolean value $\texttt{draw}(S, D, \chi, \mathcal{C})$ to be `true` if and only if there is a 2-layer k-planar drawing D^* of $G[N[S \cup \mathcal{C}^X]]$ such that

- D^* is compatible with D and
- for every edge $e \in \delta(S)$, it holds that $\chi(e) = \mathrm{cr}_{D^*}(e)$.

Hence, G has a 2-layer k-planar drawing if and only if $\texttt{draw}(S, D, \chi, \mathcal{C}_S) = \texttt{true}$ for some $S \subseteq X$, D, χ, and $\mathcal{C}_S$.

To compute the values $\texttt{draw}(S, D, \chi, \mathcal{C})$ for $S \subseteq X$ with $|S| = \ell + 1$, D, χ, and $\mathcal{C} \subseteq \mathcal{C}_S$, we first compute the base cases where $\mathcal{C} = \emptyset$ and then the other cases in ascending order of $|S \cup \mathcal{C}^X|$. This can be done by using a recurrence similar to the one in Lemma 5.

To see the running time bound of the above algorithm, observe that the number of possible choices for S, D, χ, and $\mathcal{C}$ is at most

$$\sum_{S \subseteq X} |S|! \cdot |N(S)|! \cdot (k+1)^{|\delta(S)|} \cdot 2^{|\mathcal{C}_S|} = n^{\ell+1} \cdot 2^{O(k^2 \log k)} \cdot 2^{|\mathcal{C}_S|}.$$

The third factor can be bounded by $2^{O(k^3)}$ as follows: Since G is connected, each connected component of $G[V(G) \setminus N[S]]$ contains at least one vertex in $N(N(S)) \setminus S$. This implies that the number of components in $G[V(G) \setminus N[S]]$ is at most $|N(N(S)) \setminus S| \le (2k+2)(2k+1)^2$.

▶ **Theorem 9.** *Two-Sided k-Planarity can be solved in time $2^{O(k^3)} n^{2k+O(1)}$, that is, Two-Sided k-Planarity is polynomial-time solvable when k is fixed.*

The above algorithm easily turns into a non-deterministic algorithm that runs in polynomial time and space $k^{O(1)} \log n$, which implies the following.

▶ **Corollary 10** ($\star$). *Two-Sided k-Planarity is in* XNLP.

4.2 XNLP-Completeness

To complement the positive result in the previous subsection, we show that Two-Sided k-Planarity is XNLP-hard even on trees. In contrast to our result, TSCM can be solved in polynomial time on trees [45].

▶ **Theorem 11** (⋆). *Two-Sided k-Planarity is* XNLP*-complete w.r.t. k even on trees.*

Proof (sketch). Membership in XNLP follows from Corollary 10. We prove the claim by showing a parameterized logspace reduction from Bandwidth, which is known to be XNLP-hard even on trees [11,13]. Let T be a tree. We subdivide each edge e of T once by introducing a vertex w_e, and we add ℓ leaves adjacent to each original vertex of T for some $\ell = \Theta(b^2)$. Let G be the graph obtained in this way. Next, we show that $\mathrm{bw}(T) \leq b$ if and only if G has a 2-layer k-planar drawing for some $k = \Theta(b^3)$.

Let $X = V(T)$, let $Y = V(G) \setminus X$, and let σ be a vertex order of T with bandwidth b. Define a vertex order $<_X$ on X by setting $<_X = \sigma$. Since the stretch of each edge in σ is at most b, there are at most $b-1$ vertices between its endpoints. This implies that we can place the vertices in Y so that there will be only $O(b^3)$ crossings per edge; see Figure 3. Conversely, in any 2-layer k-planar drawing of G, the endpoints of every edge $e = \{u, v\}$ of T are close to each other, as each vertex between u and v causes at least ℓ crossings on the path (u, w_e, v). Hence, the order on X turns into a vertex order of T with bandwidth at most b. ◀

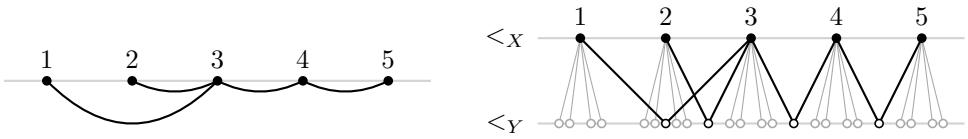


Figure 3 A minimum-bandwidth order of T and the 2-layer drawing of G that we construct.

5 Recognizing Outer k-Planar Graphs

In this section, we discuss the parameterized complexity of recognizing outer k-planar graphs.

Problem: Outer k-Planarity
Input: A graph G with n vertices and an integer k.
Question: Does G admit an outer k-planar drawing?

5.1 An XP-Algorithm

In this subsection, we show our main result, an XP-algorithm for Outer k-Planarity with respect to k. Note that a graph is outer k-planar if and only if its biconnected components are outer k-planar; this can be shown in a similar manner as [31, Theorem 4] for $k = 2$. Hence, we assume the input graph to be biconnected.

▶ **Theorem 12.** *Outer k-Planarity can be solved in time $2^{O(k \log k)} n^{3k+O(1)}$, that is, Outer k-Planarity is polynomial-time solvable when k is fixed.*

Let G be the input graph, let $\vec{e} = (u, v)$ be an ordered pair of two distinct vertices of G, and let R be a subset of $V(G) \setminus \{u, v\}$ such that there are at most k edges between R and L, where $L = V(G) \setminus (\{u, v\} \cup R)$. Let C be the set of these edges. Let τ be a linear order of

C, let $c_1, \dots, c_\ell$ denote the vertices of C in the order given by τ, and let $\chi\colon C \to \{0, \dots, k\}$. Let $G_{\tau,\vec{e},R}$ be the graph obtained by adding ℓ vertices $t_1^\tau, t_2^\tau, \dots, t_\ell^\tau$ to the induced subgraph $G[\{u,v\} \cup R]$ and by connecting t_i^τ and the endpoint of c_i in R for every $i \in [\ell]$. Then we define a Boolean value $\texttt{draw}(\vec{e}, R, \tau, \chi)$ to be `true` if and only if $G_{\tau,\vec{e},R}$ admits an outer k-planar drawing D with the following properties:

(P1) the cyclic order of D contains $(u, t_1^\tau, t_2^\tau, \dots, t_\ell^\tau, v)$ as a consecutive subsequence, and

(P2) for every edge $c_i \in C$, it holds that $\chi(c_i) = \mathrm{cr}_D(c_i)$.

Clearly, the graph G admits an outer k-planar drawing if and only if there exists a vertex pair $\vec{e}$ such that $\texttt{draw}(\vec{e}, V(G) \setminus \{u,v\}, f_\emptyset, f_\emptyset) = \texttt{true}$, where $f_\emptyset$ is the empty function.

We evaluate the recurrence as follows. For every base case, namely where $R = \emptyset$, $\texttt{draw}(\vec{e}, R, \tau, \chi)$ is `true` since C is also empty.

When $R \neq \emptyset$, we compute $\texttt{draw}(\vec{e}, R, \tau, \chi)$ for smaller sets of type R. In this case, with the same technique as that used in [24, Lemma 6], we can show the following.

▶ **Lemma 13** (⋆). *If $G_{\tau,\vec{e},R}$ admits an outer k-planar drawing D with properties P1 and P2, there is $w \in R$ such that vertex pairs $\{u,w\}$ and $\{v,w\}$ are pierced by at most k edges in D.*

Hence, we can compute $\texttt{draw}(\vec{e}, R, \tau, \chi)$ by checking all the ways to split the instance at the vertex w. Let $\{R_1, R_2\}$ be a partition of $R \setminus \{w\}$, let $L_1 = V(G_{\tau,\vec{e},R}) \setminus (\{u,w\} \cup R_1)$ and let $L_2 = V(G_{\tau,\vec{e},R}) \setminus (\{v,w\} \cup R_2)$. For $i \in [2]$, let C_i be the set of edges between R_i and L_i, let $\ell_i = |C_i|$, let τ_i be a linear order of C_i, and let $\chi_i\colon C_i \to \{0, \dots, k\}$ be a function. We say that $w, R_1, \tau_1, \chi_1, R_2, \tau_2, \chi_2$ are *consistent* if the following holds ($\mathrm{cr}_{\tau,\tau_1,\tau_2}$ is defined below):

- there are at most k edges between L_1 and R_1 and at most k edges between L_2 and R_2,
- for every edge c between L and R_i, $\chi_i(c) = \chi(c) - \mathrm{cr}_{\tau,\tau_1,\tau_2}(c)$ holds for $i \in [2]$,
- for every edge c between L and $\{w\}$, $\chi(c) = \mathrm{cr}_{\tau,\tau_1,\tau_2}(c)$,
- for every edge c between $\{v\}$ and R_1, $\chi_1(c) + \mathrm{cr}_{\tau,\tau_1,\tau_2}(c) \leq k$,
- for every edge c between $\{u\}$ and R_2, $\chi_2(c) + \mathrm{cr}_{\tau,\tau_1,\tau_2}(c) \leq k$, and
- for every edge c between R_1 and R_2, $\chi_1(c) + \chi_2(c) + \mathrm{cr}_{\tau,\tau_1,\tau_2}(c) \leq k$.

Informally, the value $\mathrm{cr}_{\tau,\tau_1,\tau_2}(c)$ is the number of crossings on c inside the "triangle" consisting of $\{u,v,w\}$. To define it formally, let us consider a circular drawing $D_H = (u, t_1^\tau, \dots, t_\ell^\tau, v, t_{\ell_2}^{\tau_2}, \dots t_1^{\tau_2}, w, t_{\ell_1}^{\tau_1}, \dots, t_1^{\tau_1})$ of a graph H. For each edge $c \in (E \cap \{u,v\}) \cup C \cup C_1 \cup C_2$, the graph H contains an edge $f(c)$ defined as follows. If $c = \{u,v\}$, $f(c)$ simply connects u and v. Suppose that c is incident to exactly one vertex $x \in \{u,v,w\}$. This implies that c is contained in exactly one of C, C_1, and C_2, which also means that c is contained in the domain of exactly one $\tau' \in \{\tau, \tau_1, \tau_2\}$. Then $f(c)$ connects x and $t_{\tau'(c)}^{\tau'}$. Otherwise, c is contained in exactly two of C, C_1, and C_2, since c is not contained in $C \cap C_1 \cap C_2$. Similarly to the previous case, c is contained in the domains of distinct $\tau', \tau'' \in \{\tau, \tau_1, \tau_2\}$. Then $f(c)$ connects $t_{\tau'(c)}^{\tau'}$ and $t_{\tau''(c)}^{\tau''}$. Now we define $\mathrm{cr}_{\tau,\tau_1,\tau_2}(c) = \mathrm{cr}_{D_H}(f(c))$.

We are ready to state Lemma 14, which formalizes the above idea of splitting an instance $((u,v), R, \tau, \chi)$ at a vertex w in R into two subinstances $((u,w), R_1, \tau_1, \chi_1)$ and $((w,v), R_2, \tau_2, \chi_2)$; see Figure 4, where some edges are curved for better visualization.

▶ **Lemma 14.** *For $\vec{e} = (u,v)$, it holds that*

$$\texttt{draw}(\vec{e}, R, \tau, \chi) = \bigvee_{\substack{w, R_1, \tau_1, \chi_1, R_2, \tau_2, \chi_2 \\ \textit{consistent}}} \texttt{draw}((u,w), R_1, \tau_1, \chi_1) \wedge \texttt{draw}((w,v), R_2, \tau_2, \chi_2).$$

Proof. Suppose that $\texttt{draw}(\vec{e}, R, \tau, \chi) = \texttt{true}$, that is, there is an outer k-planar drawing $D = (u, t_1^\tau, \dots, t_\ell^\tau, v, v_1, \dots, v_r)$ of $G_{\tau,e,R}$ that satisfies properties P1 and P2. We assume that edges are drawn as straight-line segments in D. By Lemma 13 there is a vertex $w \in R$ for

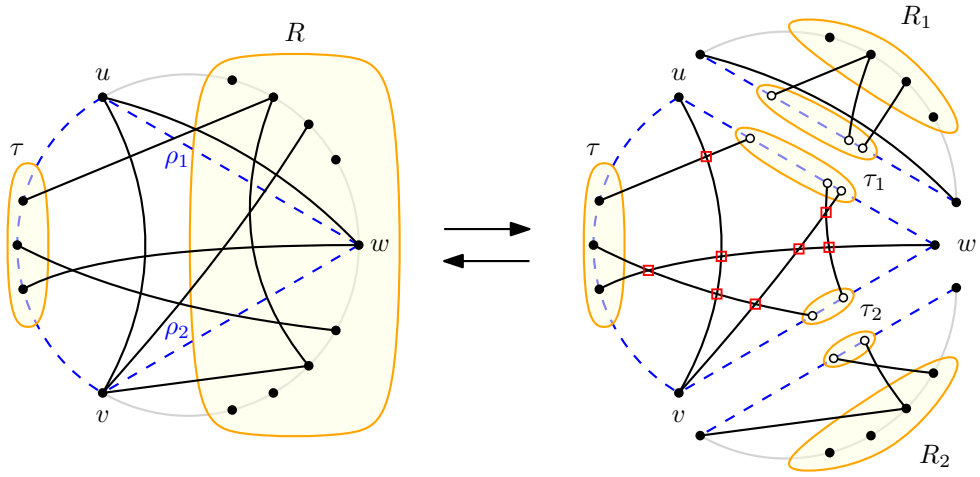


Figure 4 An image of Lemma 14. The red boxes are the crossings considered in $\mathrm{cr}_{\tau,\tau_1,\tau_2}$.

some $w = v_i$ such that both $\{u, w\}$ and $\{v, w\}$ have at most k piercing edges. The vertices u, v, w divide the circumference into the three arcs $\rho_{\bar{u}}, \rho_{\bar{v}}, \rho_{\bar{w}}$, where $\rho_{\bar{x}}$ is the arc between vertices other than x that does not pass through x for each $x \in \{u, v, w\}$. Then, following the line through u and w, we can take a curve ρ_1 between u and w, inside the circle, such that it crosses exactly the piercing edges, does not pass through any crossing between piercing edges, and separates v and the segment representing the edge $\{u, w\}$ (if it exists). We can take a curve ρ_2 between v and w similarly. Let $R_1 = \{v_{i+1}, \ldots, v_r\}$ and $R_2 = \{v_1, \ldots, v_{i-1}\}$. We then define a linear order τ_1 on the edges between R_1 and $L_1 \coloneqq V(G_{\tau,\vec{e},R}) \setminus (\{u, w\} \cup R_1)$ in such a way that τ_1 orders those edges in ascending order of the distance between u and the crossing with the curve ρ_1. Similarly, τ_2 is defined in such a way that τ_2 orders the edges R_2 and $L_2 \coloneqq V(G_{\tau,\vec{e},R}) \setminus (\{v, w\} \cup R_2)$ in ascending order of the distance between w to the crossing with the curve ρ_2. If we cut the drawing D along the curves as Figure 4, the drawing D can be decomposed into three subdrawings D_H, D_1, and D_2: D_H is the drawing inside the region surrounded by arcs $\rho_{\bar{w}}$, ρ_2, and ρ_1; D_1 is the drawing inside the region surrounded by arcs $\rho_{\bar{v}}$ and ρ_1; D_2 is the drawing inside the region surrounded by arcs $\rho_{\bar{u}}$ and ρ_2. Since each crossing on the edges between R_1 and L_1 is contained in exactly one of D_H, D_1, D_2, we have $\chi_1(c) = \chi(c) - \mathrm{cr}_{\tau,\tau_1,\tau_2}(c)$ for each edge c between L and R_1, and we have $\chi_1(c) + \mathrm{cr}_{\tau,\tau_1,\tau_2}(c) \leq k$ for each edge c between $\{v\}$ and R_1. As D_1 is a circular drawing of $G_{\tau_1,(u,w),R_1}$ that satisfies P1 and P2, we have $\texttt{draw}((u, w), R_1, \tau_1, \chi_1) = \texttt{true}$, and similarly, we have $\texttt{draw}((w, v), R_2, \tau_2, \chi_2) = \texttt{true}$. Since each edge c between R_1 and R_2 satisfies $\chi_1(c) + \chi_2(c) + \mathrm{cr}_{\tau,\tau_1,\tau_2}(c) \leq k$, we conclude that $w, R_1, \tau_1, \chi_1, R_2, \tau_2, \chi_2$ are consistent.

Suppose that there are consistent w, R_1, τ_1, χ_1, R_2, τ_2, χ_2 such that $\texttt{draw}((u, w), R_1, \tau_1, \chi_1) = \texttt{draw}((w, v), R_2, \tau_2, \chi_2) = \texttt{true}$. Let D_1 and D_2 be circular drawings of $G_{\tau_1,(u,w),R_1}$ and $G_{\tau_2,(w,v),R_2}$, respectively, that satisfy P1 and P2. Let $\sigma_1 = (w, \ldots, u)$ and $\sigma_2 = (v, \ldots, w)$ be the linear orders of $\{w, u\} \cup R_1$ and $\{v, w\} \cup R_2$ obtained from D_1 and D_2 by removing the vertices $t_i^{\tau_1}$ and $t_j^{\tau_2}$ for $i \in [\ell_1]$ and $j \in [\ell_2]$, respectively. Then we obtain a cyclic order σ of $G_{\tau,e,R}$ by concatenating $\sigma_2, \sigma_1, (u, t_1^{\tau}, \ldots, t_\ell^{\tau}, v)$ in this order, identifying the two occurrences of each of w, u, v with each other. It is not difficult to see that, by combining D_H, D_1, and D_2 as in Figure 4, we obtain a drawing D with linear order σ that satisfies P1 and P2. In other words, $\texttt{draw}(\vec{e}, R, \tau, \chi) = \texttt{true}$. ◀

Naïvely, the number of R's to consider is $\Theta(2^n)$, which does not give an XP-algorithm. However, the following lemma assures that it is not so large.

▶ **Lemma 15** (⋆). *Let G be a biconnected graph that admits an outer k-planar drawing D. Let $\{u, v\}$ be a pair of distinct vertices of G that has at most k piercing edges in D. Then the number of R's such that $\texttt{draw}((u, v), R, \tau, \chi) = \texttt{true}$ for some τ and χ is at most $2^{O(k)}m^{k+O(1)}$, where $m = |E(G)|$. Moreover, such R's can be enumerated in $2^{O(k)}m^{k+O(1)}$ time.*

Proof (sketch). We show that, given the set of edges piercing $\{u, v\}$, there are $2^{O(k)}$ possibilities for R that are separated by these piercing edges. Since R is a union of components in the graph obtained from $G[V(G) \setminus \{u, v\}]$ by deleting the piercing edges, it suffices to show that there are only $O(k)$ components in this graph. The proof shares the same underlying idea with Lemma 8, but it is more involved as the maximum degree is no longer bounded. The upper bound can be obtained by considering that there are at most k piercing edges of $\{u, v\}$ and the number of components in $G[V(G) \setminus \{u, v\}]$ is at most $2k + 3$. ◀

With Lemma 15 and the fact that $m = O(\sqrt{k}n)$ [40], the number of combinations of arguments $\{e, R, \sigma, \chi\}$ to consider is at most

$$n^2 \cdot 2^{O(k)}m^{k+O(1)} \cdot k! \cdot (k + 1)^k = 2^{O(k \log k)}n^{k+O(1)}.$$

To compute the value $\texttt{draw}(\vec{e}, R, \sigma, \chi)$ as in Lemma 14, we guess at most

$$n \cdot (2^{O(k \log k)}n^{k+O(1)})^2 = 2^{O(k \log k)}n^{2k+O(1)}$$

possible combinations of $w, R_1, \tau_1, \chi_1, R_2, \tau_2, \chi_2$. For each guess, checking the consistency takes $n^{O(1)}$ time. Hence, the total running time to fill the table is $2^{O(k \log k)}n^{3k+O(1)}$. This completes the proof of Theorem 12.

5.2 NP-Hardness of Approximation

In this subsection, we show an inapproximability result for Outer k-Planarity even for graphs that are almost trees, whereas trees can be drawn without any crossings.

▶ **Theorem 16.** *For any fixed $c \geq 1$, there is no polynomial-time c-approximation algorithm for Outer k-Planarity unless $\mathsf{P} = \mathsf{NP}$, even for graphs with feedback vertex number 1.*

Our proof is by reduction from Bandwidth on trees, which is NP-hard to approximate within any constant factor [21]. In other words, given a tree T, there is no polynomial-time algorithm to distinguish between the cases $\mathrm{bw}(T) \leq b$ and $\mathrm{bw}(T) > cb$ for any constant $c \geq 1$, unless $\mathsf{P} = \mathsf{NP}$.

Let T be a tree, and let n denote $|V(T)|$. We construct a graph G from T by adding a vertex w and making it adjacent to all vertices of T. Clearly, G has feedback vertex number 1 since $G[V(G) \setminus \{w\}]$ is a tree.

▶ **Lemma 17** (⋆). *If G is outer k-planar, then $\mathrm{bw}(T) \leq k - 1$.*

▶ **Lemma 18** (⋆). *Let $b \geq 1$. If $\mathrm{bw}(T) \leq b$, then G is outer $(5b - 5)$-planar.*

Proof (sketch). From a vertex order $(v_1, \ldots, v_n)$ of T with bandwidth at most b, we construct a circular drawing D of G as $D = (w, v_1, \ldots, v_n)$. Each edge $\{w, v_i\}$ incident to w has at most $2b - 2$ crossings in D since these crossing edges lie between vertices that are "close" to v_i. For other edge $\{v_i, v_j\} \in E(T)$ with $i < j$, it only crosses (1) edges incident to w and (2)

edges in T. There are at most $b-1$ edges of (1) since the stretch of $\{v_i, v_j\}$ is at most b, and at most $4b-4$ edges of (2) since these edges lie between vertices that are "close" to v_i or v_j. Hence, each edge has at most $5b-5$ crossings in total. ◀

Suppose that there is a polynomial-time c-approximation algorithm $\mathcal{A}$ for Outer k-Planarity. Let T be a tree, and let $b = \mathrm{bw}(T)$. By Lemma 18, $\mathcal{A}$ would output an outer $5bc$-planar drawing D of G. By Lemma 17, D can be transformed into a linear order of $V(T)$ with bandwidth at most $5bc$. Thus, we can find a $5c$-approximate solution for Bandwidth in polynomial time, which is impossible under $\mathsf{P} \neq \mathsf{NP}$. This completes the proof of Theorem 16.

5.3 XNLP-Hardness

In the proof of Theorem 16, we reduced the gap-version of Bandwidth to Outer k-Planarity. We exploited the gap to accommodate the crossings between edges in the original instance, which may increase the crossing number of each edge by $O(b)$. However, if we allow parallel edges, we can reduce from (the exact version of) Bandwidth by making the edges incident to w so thick that we can ignore the $O(b)$ increase in the crossing numbers. The following theorem is shown by emulating those parallel edges with rigid structures.

▶ **Theorem 19** ($\star$). *Outer k-Planarity is* XNLP*-hard when parameterized by k.*

Proof (sketch). As in Theorem 11, we give a parameterized logspace reduction from Bandwidth on trees. The idea of the reduction is similar to that used in Theorem 16. Instead of connecting w with each v_i, we replace each vertex v_i with a clique path gadget that appears consecutively in any outer k-planar drawing for some $k = \Theta(b^4)$ and connect w with sufficiently many vertices in the gadget. Since there are many edges between w and each gadget, two adjacent gadgets are placed closely in any outer k-planar drawing. ◀

6 Open Problems

We conclude with a number of problems that we have left open in this paper.

- Is One-Sided k-Planarity NP-hard?
- We conjecture that Outer k-Planarity is XALP-complete (see [12] for the definition).
- Can we extend the algorithm for Two-Sided k-Planarity to obtain an XP-algorithm for ℓ-layer k-planarity parameterized by $\ell + k$?
- Another way to extend k-planarity is to consider *min-k-planarity*, which is also called *weak k-planarity* [9, 15]. In a min-k-planar drawing, in every crossing, at least one of the two edges must have at most k crossings. Can 2-layer min-k-planar graphs and outer min-k-planar graphs be recognized by XP-algorithms with respect to k?
- Two-Sided k-Planarity can be seen as a restricted version of Outer k-Planarity for bipartite graphs where the vertices of the two sets of the bipartition must not interleave in the cyclic vertex order. This can be generalized as follows: For $k \geq 3$, can we efficiently recognize k-partite graphs that admit a k-planar straight-line drawing on the regular k-gon? A related question has been investigated for fixed-order book embedding [1].

References

1 Akanksha Agrawal, Sergio Cabello, Michael Kaufmann, Saket Saurabh, Roohani Sharma, Yushi Uno, and Alexander Wolff. Eliminating crossings in ordered graphs. In Hans Bodlaender, editor, *19th Scand. Symp. Algorithm Theory (SWAT)*, volume 294 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SWAT.2024.1.

2 Oswin Aichholzer, Johannes Obenaus, Joachim Orthaber, Rosna Paul, Patrick Schnider, Raphael Steiner, Tim Taubner, and Birgit Vogtenhuber. Edge partitions of complete geometric graphs. In Xavier Goaoc and Michael Kerber, editors, *38th Int. Symp. Comput. Geom. (SoCG)*, volume 224 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.6.

3 Patrizio Angelini, Giordano Da Lozzo, Henry Förster, and Thomas Schneck. 2-Layer k-planar graphs: Density, crossing lemma, relationships and pathwidth. *The Computer Journal*, 67(3):1005–1016, 2023. doi:10.1093/comjnl/bxad038.

4 Christopher Auer, Christian Bachmaier, Franz J. Brandenburg, Andreas Gleißner, Kathrin Hanauer, Daniel Neuwirth, and Josef Reislhuber. Outer 1-planar graphs. *Algorithmica*, 74(4):1293–1320, 2016. doi:10.1007/S00453-015-0002-1.

5 Michael Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *Journal of Graph Algorithms and Applications*, 22(1):23–49, 2018. doi:10.7155/jgaa.00457.

6 Michael Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. *Journal of Graph Algorithms and Applications*, 22(4):577–606, 2018. doi:10.7155/jgaa.00479.

7 Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In Juraj Hromkovic, Manfred Nagl, and Bernhard Westfechtel, editors, *30th Int. Workshop Graph-Theoretic Concepts Comput. Sci. (WG)*, volume 3353 of *LNCS*, pages 332–343. Springer, 2004. doi:10.1007/978-3-540-30559-0_28.

8 Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for book embedding problems. *Journal of Graph Algorithms and Applications*, 24(4):603–620, 2020. doi:10.7155/jgaa.00526.

9 Carla Binucci, Aaron Büngener, Giuseppe Di Battista, Walter Didimo, Vida Dujmovic, Seok-Hee Hong, Michael Kaufmann, Giuseppe Liotta, Pat Morin, and Alessandra Tappini. Min-k-planar drawings of graphs. *Journal of Graph Algorithms and Applications*, 28(2):1–35, 2024. doi:10.7155/JGAA.V28I2.2925.

10 Vacláv Blažej, Boris Klemz, Felix Klesen, Marie Diana Sieper, Alexander Wolff, and Johannes Zink. Constrained and ordered level planarity parameterized by the number of levels. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th Annu. Sympos. Comput. Geom. (SoCG'24)*, volume 293 of *LIPIcs*, pages 21:1–16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SoCG.2024.20.

11 Hans L. Bodlaender. Parameterized complexity of bandwidth of caterpillars and weighted path emulation. In Łukasz Kowalik, Michał Pilipczuk, and Paweł Rzążewski, editors, *Graph-Theoretic Concepts Comput. Sci. (WG)*, volume 12911 of *LNCS*, pages 15–27. Springer, 2021. doi:10.1007/978-3-030-86838-3_2.

12 Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Marcin Pilipczuk, and Michał Pilipczuk. On the complexity of problems on tree-structured graphs. In Holger Dell and Jesper Nederlof, editors, *17th Int. Symp. Paramet. & Exact Comput. (IPEC)*, volume 249 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.6.

13 Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. In *62nd IEEE Ann. Symp. Foundat. Comput. Sci. (FOCS)*, pages 193–204, 2021. doi:10.1109/FOCS52979.2021.00027.

14 Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM Journal on Computing*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

15 Rutger Campbell, Katie Clinch, Marc Distel, J. Pascal Gollin, Kevin Hendrey, Robert Hickingbotham, Tony Huynh, Freddie Illingworth, Youri Tamitegama, Jane Tan, and David R. Wood. Product structure of graph classes with bounded treewidth. *Combinatorics, Probability and Computing*, 33(3):351–376, 2024. doi:10.1017/S0963548323000457.

16 Steven Chaplick, Myroslav Kryven, Giuseppe Liotta, Andre Löffler, and Alexander Wolff. Beyond outerplanarity. In Fabrizio Frati and Kwan-Liu Ma, editors, *25th Int. Symp. Graph Drawing & Network Vis. (GD)*, volume 10692 of *LNCS*, pages 546–559. Springer, 2018. doi:10.1007/978-3-319-73915-1_42.

17 Marek Cygan, Fedor V. Fomin, Łukasz 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 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

19 Alexander Dobler. A note on the complexity of one-sided crossing minimization of trees, 2023. arXiv. doi:10.48550/arXiv.2306.15339.

20 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

21 Chandan Dubey, Uriel Feige, and Walter Unger. Hardness results for approximating the bandwidth. *Journal of Computer and System Sciences*, 77(1):62–90, 2011. Celebrating Karp's Kyoto Prize. doi:10.1016/j.jcss.2010.06.006.

22 Vida Dujmović, Seok-Hee Hong, Michael Kaufmann, János Pach, and Henry Förster. Beyond-planar graphs: Models, structures and geometric representations (Dagstuhl seminar 24062). *Dagstuhl Reports*, 14(2):71–94, 2024. doi:10.4230/DagRep.14.2.71.

23 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.

24 Oksana Firman, Grzegorz Gutowski, Myroslav Kryven, Yuto Okada, and Alexander Wolff. Bounding the treewidth of outer k-planar graphs via triangulations. In Stefan Felsner and Karsten Klein, editors, *32nd Int. Symp. Graph Drawing & Network Vis. (GD)*, volume 320 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.14.

25 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

26 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

27 Michael R. Garey and David S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983. doi:10.1137/0604033.

28 Alexander Grigoriev and Hans L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/S00453-007-0010-X.

29 Martin Grohe. Computing crossing numbers in quadratic time. *Journal of Computer and System Sciences*, 68(2):285–302, 2004. doi:10.1016/j.jcss.2003.07.008.

30 Seok-Hee Hong, Peter Eades, Naoki Katoh, Giuseppe Liotta, Pascal Schweitzer, and Yusuke Suzuki. A linear-time algorithm for testing outer-1-planarity. *Algorithmica*, 72(4):1033–1054, 2015. doi:10.1007/S00453-014-9890-8.

31 Seok-Hee Hong and Hiroshi Nagamochi. A linear-time algorithm for testing full outer-2-planarity. *Discrete Applied Mathematics*, 255:234–257, 2019. doi:10.1016/j.dam.2018.08.018.

32 Paul C. Kainen. The book thickness of a graph. II. In *20th Southeastern Conf. Combin., Graph Theory, & Comput. (Boca Raton, FL, 1989)*, volume 71, pages 127–132, 1990.

33 Ken-ichi Kawarabayashi and Bruce Reed. Computing crossing number in linear time. In *39th Ann. ACM Symp. Theory Comput. (STOC)*, pages 382–390, 2007. doi:10.1145/1250790.1250848.

34 Yasuaki Kobayashi, Hiromu Ohtsuka, and Hisao Tamaki. An improved fixed-parameter algorithm for one-page crossing minimization. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th Int. Symp. Paramet. & Exact Comput. (IPEC)*, volume 89 of *LIPIcs*, pages 25:1–25:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.IPEC.2017.25.

35 Yasuaki Kobayashi and Hisao Tamaki. A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization. *Algorithmica*, 72:778–790, 2015. doi:10.1007/s00453-014-9872-x.

36 Yasuaki Kobayashi and Hisao Tamaki. A faster fixed parameter algorithm for two-layer crossing minimization. *Information Processing Letters*, 116(9):547–549, 2016. doi:10.1016/j.ipl.2016.04.012.

37 Yunlong Liu, Jie Chen, and Jingui Huang. Parameterized algorithms for fixed-order book drawing with bounded number of crossings per edge. In Weili Wu and Zhongnan Zhang, editors, *Proc. 14th Int. Conf. Combin. Optim. Appl. (COCOA)*, volume 12577 of *LNCS*, pages 562–576. Springer, 2020. doi:10.1007/978-3-030-64843-5_38.

38 Yunlong Liu, Jie Chen, Jingui Huang, and Jianxin Wang. On parameterized algorithms for fixed-order book thickness with respect to the pathwidth of the vertex ordering. *Theor. Comput. Sci.*, 873:16–24, 2021. doi:10.1016/j.tcs.2021.04.021.

39 Xavier Muñoz, Walter Unger, and Imrich Vrt'o. One sided crossing minimization is NP-hard for sparse graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *9th Int. Symp. Graph Drawing (GD)*, volume 2265 of *LNCS*, pages 115–123. Springer, 2001. doi:10.1007/3-540-45848-4_10.

40 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

41 Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comput. Theory*, 9(4), 2018. doi:10.1145/3154856.

42 Helen C. Purchase, Christopher Pilcher, and Beryl Plimmer. Graph drawing aesthetics – created by users, not algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):81–92, 2012. doi:10.1109/TVCG.2010.269.

43 James B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Algebraic Discret. Methods*, 1(4):363–369, 1980. doi:10.1137/0601042.

44 Marcus Schaefer. The graph crossing number and its variants: A survey. *Electronic Journal of Combinatorics*, DS21, 2024. doi:10.37236/2713.

45 Farhad Shahrokhi, Ondrej Sýkora, László A. Székely, and Imrich Vrto. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 30(6):1773–1789, 2001. doi:10.1137/S0097539797331671.

46 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

47 John C. Urschel and Jake Wellens. Testing gap k-planarity is NP-complete. *Information Processing Letters*, 169:106083, 2021. doi:10.1016/j.ipl.2020.106083.

48 Meirav Zehavi. Parameterized analysis and crossing minimization problems. *Computer Science Review*, 45:100490, 2022. doi:10.1016/j.cosrev.2022.100490.

Lipschitz Decompositions of Finite ℓ_p Metrics

Robert Krauthgamer
Weizmann Institute of Science, Rehovot, Israel

Nir Petruschka
Weizmann Institute of Science, Rehovot, Israel

Abstract

Lipschitz decomposition is a useful tool in the design of efficient algorithms involving metric spaces. While many bounds are known for different families of finite metrics, the optimal parameters for n-point subsets of ℓ_p, for $p > 2$, remained open, see e.g. [Naor, SODA 2017]. We make significant progress on this question and establish the bound $\beta = O(\log^{1-1/p} n)$. Building on prior work, we demonstrate applications of this result to two problems, high-dimensional geometric spanners and distance labeling schemes. In addition, we sharpen a related decomposition bound for $1 < p < 2$, due to Filtser and Neiman [Algorithmica 2022].

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Sparsification and spanners

Keywords and phrases Lipschitz decompositions, metric embeddings, geometric spanners

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.66

Related Version *arXiv Version*: https://arxiv.org/abs/2502.01120

Funding *Robert Krauthgamer*: Work partially supported by the Israel Science Foundation grant #1336/23, by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center, and by a research grant from the Estate of Harry Schutzman.

Acknowledgements We thank Arnold Filtser and Ofer Neiman for helpful discussions.

1 Introduction

The pursuit of approximating metric spaces by simpler structures has inspired the development of fundamental concepts, such as graph spanners [47, 46] and low-distortion embeddings into various spaces [37, 7], both of which have a wide range of algorithmic applications. Many of these results, including for instance [7, 48, 17, 30, 6, 19], rely on various notions of decomposition of a metric space into low-diameter clusters, and these decompositions are most often randomized. One extensively studied notion, see e.g. [13, 24, 17, 25], is *Lipschitz decomposition* (also called *separating decomposition*), which informally is a random partition of a metric space into low-diameter clusters, with a guarantee that nearby points are likely to belong to the same cluster.

▶ **Definition 1.1** (Lipschitz decomposition [7]). *Let (X, ρ) be a metric space. A distribution $\mathcal{D}$ over partitions of X is called* (β, Δ)-Lipschitz *if*

1. *for every partition $P \in \mathrm{supp}(\mathcal{D})$, all clusters $C \in P$ satisfy* $\mathrm{diam}(C) \le \Delta$*; and*
2. *for all $x, y \in X$,*

$$\Pr_{P \in \mathcal{D}}[P(x) \neq P(y)] \le \beta \cdot \frac{\rho(x,y)}{\Delta},$$

where $P(z)$ denotes the cluster of P containing $z \in X$ and $\mathrm{diam}(C) := \sup_{x,y \in C} \rho(x, y)$.

© Robert Krauthgamer and Nir Petruschka;
licensed under Creative Commons License CC-BY 4.0

Typical applications require such decompositions where Δ is not known in advance, or even multiple values of Δ (say for every power of 2). We naturally seek small β and thus define the (optimal) *decomposition parameter of* (X, ρ) as

$$\beta^*(X) := \inf_{\beta \geq 1} \Big\{ \beta : \forall \Delta > 0, \text{ every finite } X' \subseteq X \text{ admits a } (\beta, \Delta)\text{-Lipschitz decomposition} \Big\},$$

and we extend this to a family of metric spaces $\mathcal{X}$, by defining $\beta^*(\mathcal{X}) := \sup_{X \in \mathcal{X}} \beta^*(X)$.

Obtaining bounds on the decomposition parameter of various metrics (and families of metrics) is of significant algorithmic importance, and we list in Table 1 several known bounds. One fundamental example where we know of (nearly) tight bounds is the metric space ℓ_p^d, for $p \geq 1$, which stands for $\mathbb{R}^d$ equipped with the ℓ_p norm. For $p \in [1, 2]$, we have $\beta^*(\ell_p^d) = \Theta(d^{1/p})$ due to [13], and for $p \in [2, \infty]$ we have $\beta^*(\ell_p^d) = \tilde{\Theta}(d^{1/2})$ due to [40] (see discussion therein about an incorrect claim made in [13]).[1] Observe that an upper bound for $X = \ell_p^d$ immediately extends to all subsets of it, implying in particular a bound for the family $\mathcal{X}$ of all finite subsets of ℓ_p^d. These bounds depend on d, and are thus most suitable for low-dimensional settings.

We focus on finite metrics X, aiming to bound $\beta^*(X)$ in terms of $n = |X|$, which is often useful in high-dimensional settings. For instance, it is well-known that $\beta^* = \Theta(\log n)$ the family of all n-point metric spaces [7]. To write this assertion more formally, define $\beta_n^*(X) := \beta^*(\{X' \subseteq X : \ |X| = n\})$ and then the above asserts that $\beta_n^*(\ell_\infty) = \Theta(\log n)$, where we used that every finite metric embeds isometrically in ℓ_∞. For the family of n-point ℓ_2 metrics, combining $\beta^*(\ell_2^d) = \tilde{\Theta}(\sqrt{d})$ with the famous JL Lemma [27] immediately yields $\beta_n^*(\ell_2) = O(\sqrt{\log n})$, which is tight by [13]. For n-point ℓ_p metrics, $1 < p < 2$, we have $\beta_n^*(\ell_p) = \frac{O(\log^{1/p} n)}{p-1}$ due to [35, 40], nearly matching the lower bound of $\beta_n^*(\ell_p) = \Omega(\log^{1/p} n)$ from [13]. However, for n-point ℓ_p metrics, $p > 2$, to the best of our knowledge, the only known upper bound is $\beta_n^*(\ell_p) = O(\log n)$, obtained by trivially applying the results for general n-point metric spaces. The following question was raised by Naor [40, Question 1], see also [41, Question 83].

▶ **Question 1.2** ([40]). *Is it true that for every $p \in (2, \infty)$, $\beta_n^*(\ell_p) = o(\log n)$? More ambitiously, is it true that $\beta_n^*(\ell_p) = O_p(\sqrt{\log n})$?*

Our main result, in Theorem 1.3, answers the first part of this question in the affirmative. Additionally, we show in Section 2 an analogous result for another notion of decomposability that was introduced in [22] (and we call capped decomposition) and is particularly suited for high-dimensional geometric spanners.

Geometric Spanners. A *spanner with stretch* $t \geq 1$ (in short a *t-spanner*) for a finite metric $M = (X, \rho)$ is a graph $G = (X, E)$, that satisfies $\rho(x, y) \leq \rho_G(x, y) \leq t \cdot \rho(x, y)$ for all $x, y \in X$, meaning that the shortest-path distance ρ_G in the graph G approximates the original distance $\rho(x, y)$ within factor t, where by definition every edge $\{u, v\} \in E$ has weight $\rho(u, v)$. Of particular interest are spanners that are *sparse*, meaning they contain a small number of edges, ideally linear in $n = |X|$. Another important parameter is the *lightness* of a spanner, defined as the total weight of its edges divided by the weight of a minimum spanning tree of X. Clearly, the lightness is at least 1. These spanners are called geometric because the input is a metric space (rather than a graph). They are natural and useful

[1] Throughout, the notation $\tilde{O}(f)$ hides $\text{poly}(\log f)$ factors, and $O_\alpha(\cdot)$ hides a factor that depends only on α.

Table 1 Known bounds on the decomposition parameter of some important families of metrics.

Family of Metrics	β^* or β_n^*	Reference	Comments
ℓ_p^d spaces $1 \le p \le 2$	$\Theta(d^{1/p})$	[13]	
ℓ_p^d spaces $p \ge 2$	$\tilde{\Theta}(\sqrt{d})$	[40]	
finite metrics	$\Theta(\log n)$	[7]	
ℓ_2 space (Euclidean)	$\Theta(\sqrt{\log n})$	[13]	
ℓ_p spaces $1 \le p \le 2$	$\Theta_p(\log^{1/p} n)$	[35, 40]	
ℓ_p spaces $p \ge 2$	$O(\log^{1-1/p} n)$	Theorem 1.3	conjectured $\beta_n^* = \Theta(\sqrt{\log n})$
doubling constant λ	$\Theta(\log \lambda)$	[24]	
K_r-minor-free graphs	$O(r)$	[1, 18]	conjectured $\beta^* = \Theta(\log r)$
graphs with genus g	$\Theta(\log g)$	[36, 1]	
graphs with treewidth w	$\Theta(\log w)$	[20]	

representations of a metric, and as such, have been studied extensively, see the surveys [16, 49, 2]. Spanners for n-point metrics in low-dimensional spaces (e.g., in fixed-dimensional Euclidean space or doubling metrics) are well-studied and well-understood. For instance, metrics with doubling dimension ddim admit $(1+\varepsilon)$-spanners with near-optimal sparsity $n(1/\varepsilon)^{O(\text{ddim})}$ and lightness $(1/\varepsilon)^{O(\text{ddim})}$ [12, 34].

However, in high-dimensional spaces, our understanding of spanners is rather limited. Har-Peled, Indyk, and Sidiropoulos [25] showed that every n-point Euclidean metric admits, an $O(t)$-spanner with $\tilde{O}(n^{1+1/t^2})$ edges for every $t \ge 1$. Filtser and Neiman [22] extended this result to all metric spaces that admit a certain decomposition that we call capped decomposition (Definition 2.4), showing that in those spaces, it is possible to construct spanners that are both sparse and light. In particular, they showed that every n-point subset of ℓ_p, $1 < p \le 2$, has an $O(t)$-spanner with $n^{1+\tilde{O}(1/t^p)}$ edges and lightness $n^{\tilde{O}(1/t^p)}$ for every $t \ge 1$. It remained open whether the spaces ℓ_p for $p \in (2, \infty)$ admit the aforementioned capped decomposition. To the best of our knowledge, all known spanners for these spaces have a tradeoff of stretch $O(t)$ with sparsity $O(n^{1+1/t})$.

1.1 Our Results

Our main contribution is the construction of a Lipschitz decomposition for finite ℓ_p metrics, $p \ge 2$, as follows.

▶ **Theorem 1.3.** *Let $p \in [2, \infty]$. Then $\beta_n^*(\ell_p) = O(\log^{1-1/p} n)$. That is, for every n-point metric $X \subset \ell_p$ and $\Delta > 0$, there exists an $(O(\log^{1-1/p} n), \Delta)$-Lipschitz decomposition of X.*

Previously, this bound was known only for the extreme values $p = 2, \infty$, and in these cases it is actually tight. More precisely, for $p = 2$ our bound coincides with the well-known result $\beta_n^*(\ell_2) = \Theta(\sqrt{\log n})$ [13], and for $p = \Omega(\log n)$ it is known that $\beta_n^*(\ell_p) = \Theta(\log n)$, because all n-point metrics embed into ℓ_p with $O(1)$-distortion [38]. For intermediate values, say fixed $p \in (2, \infty)$, our bound is the first one to improve over $O(\log n)$, which applies to all n-point metrics, and leaves a gap from the $\Omega(\sqrt{\log n})$ lower bound that follows from Dvoretzky's Theorem [15].

We compare our bound with those for other metric spaces in Table 1.

The proof of Theorem 1.3 appears in Section 2.1, and has interesting technical features. It relies on *two* known decompositions of finite metrics, one for general metrics and one for Euclidean metrics, that are composed via a metric-embedding tool called the Mazur map. Our decomposition method is data-dependent, i.e., not oblivious to the data, and we discuss this intriguing aspect in Sections 2.1 and 5.

Note added in proof. Shortly after this work was posted online, two groups working in parallel to each other [31, 42] improved our result in Theorem 1.3 to $\beta_n^*(\ell_p) = O_p(\sqrt{\log n})$ for every $2 < p < \infty$, thereby resolving in the affirmative also the second part of Question 1.2. These two papers design a recursive process that relies on the technique developed here for proving Theorem 1.3, see each paper for its dependence on p.

Geometric Spanners for $p \geq 2$. We then use similar ideas to obtain a new bound for another notion of decomposability, that was introduced in [22] and we call capped decomposition; and this immediately yields geometric spanners in ℓ_p, for $p \geq 2$. While for $p = 2$ these spanners coincide with the known bounds from [25, 22], for fixed $2 < p < \infty$, our spanners are the first improvement over the trivial bounds that hold for all metric spaces.

▶ **Theorem 1.4.** *Let $p \in [2, \infty)$ and $t \geq 1$. Then every n-point metric $X \subset \ell_p$ admits an $O(t)$-spanner of size $\tilde{O}\left(n^{1+1/t^q}\right)$ and lightness $\tilde{O}\left(n^{1/t^q}\right)$, where $q \in (1, 2)$ is such that $\frac{1}{p} + \frac{1}{q} = 1$.*

The proof of this theorem appears in Section 2.2, and includes both the spanner construction, which follows [22], and our new bound for capped decomposition, which is the main technical result.

Geometric Spanners for $p \leq 2$. We also sharpen the known spanner results for ℓ_p spaces with $1 < p < 2$, which say that every n-point subset admits an $O(t)$-spanner with $n^{1+O(\log^2 t/t^p)}$ edges and lightness $n^{O(\log^2 t/t^p)}$ for every $t \geq 1$ [22]. We improve upon this result by eliminating the $\log^2 t$ factor in the exponent.

▶ **Theorem 1.5.** *Let $p \in (1, 2]$ and $t \geq 1$. Then every n-point metric $X \subset \ell_p$ admits an $O(t)$-spanner of size $\tilde{O}(n^{1+1/t^p})$ and lightness $\tilde{O}(n^{1/t^p})$.*

The proof of this theorem, presented in Section 3, follows the construction of [22], but replacing a key step, in which they rely on results from [43], with results from [4]. Interestingly, our improved spanner bound "matches" the bounds of Theorem 1.4, up to duality between p and q.

Distance Labeling Schemes. Distance labeling for a metric space (X, ρ) assigns to each point $x \in X$ a label $l(x)$, so that one can later recover (perhaps approximately) the distance between any two points in X based only on their labels (without knowledge of the metric space). It was formulated in [45], motivated by applications in distributed computing, and has been studied intensively, see e.g. [23, 21]. An immediate corollary of our main result in Theorem 1.3 is a distance labeling scheme for finite metrics in ℓ_p for $p > 2$, as follows.

▶ **Theorem 1.6.** *Let $p \in (2, \infty)$. Then the family of n-point metrics in ℓ_p with pairwise distances in the range $[1, \Delta_{max}]$ admits a distance labeling scheme with approximation $O(\log^{1/q} n)$ and label size $O(\log n \log \Delta_{max})$ bits, where $q \in (1, 2)$ is such that $\frac{1}{p} + \frac{1}{q} = 1$.*

A formal definition of the distance labeling model and a proof of Theorem 1.6 appear in Section 4.

1.2 Related Work

We focus on Lipschitz decomposition and on capped decomposition, that was introduced in [22], but the literature studies several different decompositions of metric spaces into low-diameter clusters, see e.g. [39, 19]. In particular, the notion of padded decomposition [48, 29]

is closely related and was used extensively, see for example [48, 8, 35, 39, 30]. While a Lipschitz decomposition guarantees that nearby points are likely to be clustered together, a padded decomposition guarantees that each point is, with good probability, together with all its nearby points in the same cluster. Remarkably, if a metric space admits a padded decomposition then it admits also a Lipschitz decomposition with almost the same parameters [35], however the other direction is not true, as demonstrated by ℓ_2^d.

The problem of computing efficiently the optimal decomposition parameters for an input metric space (X, ρ) was studied in [32]. Specifically for Lipschitz decomposition, they show that $\beta^*(X)$ can be $O(1)$-approximated in polynomial time (in n).

2 Decompositions and Spanners in ℓ_p for $p > 2$

In this section we consider finite subsets of ℓ_p for $p \in (2, \infty)$. We first present (in Section 2.1) a new Lipschitz decomposition, which proves Theorem 1.3. Next, we show (in Section 2.2) a new construction of capped decomposition, which is a related notion of decomposability that was introduced in [22] without a concrete name. Finally we obtain (in Section 2.3) new spanners, which prove Theorem 1.4. This is actually an immediate corollary of our capped decomposition, by following the spanner construction of [22].

2.1 Lipschitz Decomposition in ℓ_p for $p \in (2, \infty)$

Before presenting the proof of Theorem 1.3, we first provide the intuition behind the proof. A common approach in many algorithms for metric spaces is to embed the given metric into a simpler one (e.g., a tree metric), solve the problem in the target metric, and then pull back this solution to the original metric. For our purpose, of constructing a Lipschitz decomposition of $X \subset \ell_p$, $p > 2$, a natural idea is to seek a low-distortion embedding of X into ℓ_2, because we already have decompositions for that space, namely, $\beta_n^*(\ell_2) = O(\sqrt{\log n})$. Ideally, the embedding into ℓ_2 would be *oblivious*, meaning that it embeds the entire ℓ_p (not only X) into ℓ_2, but unfortunately such an embedding does not exist (it would imply oblivious dimension reduction in ℓ_p for $p > 2$, which is provably impossible [14]). We get around this limitation by employing a *data-dependent* approach, where the decomposition depends on the input set X. More precisely, we use Mazur maps, which provide a low-distortion embedding from ℓ_p to ℓ_2, but only for sets of bounded diameter (see Corollary 2.3). We thus first decompose X into bounded-diameter subsets by applying a standard Lipschitz decomposition (that is applicable for every n-point metric). The final decomposition is obtained by pulling back the solution (clusters) we found in ℓ_2.

We proceed to introduce some technical results needed for our proof of Theorem 1.3. The first one is a well-known bound for Lipschitz decomposition of a finite metric.

▶ **Theorem 2.1** ([7]). *Every n-point metric (X, ρ) admits an $(O(\log n), \Delta)$-Lipschitz decomposition for every $\Delta > 0$.*

Next, we define the *Mazur map*, which is an explicit embedding $M_{p,q} : \ell_p^m \to l_q^m$ for $1 < q < p < \infty$. The image of an input vector v is computed in each coordinate separately, by raising the absolute value to power p/q while keeping the original sign. The next theorem appears in [10], where it is stated as an adaptation of [11], and we will actually need the immediate corollary that follows it.

▶ **Theorem 2.2** ([11, 10]). *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 $x, y \in \ell_p$ such that $||x||_p, ||y||_p \le C_0$,*

$$\tfrac{q}{p}(2C_0)^{1-p/q}||x-y||_p^{p/q} \le ||M(x)-M(y)||_q \le ||x-y||_p.$$

▶ **Corollary 2.3.** *Let $2 < p < \infty$. Every n-point set $X \subset \ell_p$ with diameter at most $C_0 > 0$ admits an embedding $f : X \to \ell_2$ such that*

$$\forall x, y \in X, \qquad \tfrac{2}{p}(2C_0)^{1-p/2}\|x-y\|_p^{p/2} \le \|f(x)-f(y)\|_2 \le \|x-y\|_p.$$

Proof of Theorem 1.3. Let $\Delta > 0$, and let $X \subset \ell_p$ be an n-point metric space for $p \in (2, \infty)$. Construct a partition of X in the following steps:

1. Construct for X an $(O(\log n), \log^{1/p} n \cdot \Delta/4)$-Lipschitz decomposition $P_{\text{init}} = \{K_1, \ldots, K_t\}$ using Theorem 2.1.
2. Embed each cluster $K_i \subset \ell_p$ into ℓ_2 using the embedding f^{K_i} provided by Corollary 2.3 for $C_0 := \log^{1/p} n \cdot \Delta/4$.
3. For each embedded cluster $f^{K_i}(K_i)$, construct an $(O(\sqrt{\log n}), \frac{1}{2}\Delta/\log^{1/2-1/p} n)$-Lipschitz decomposition $P_i = \{K_i^1, \ldots, K_i^{k_i}\}$ using [13] and the JL Lemma [27].
4. The final decomposition P_{out} is obtained by taking the preimage of every cluster of every P_i.

It is easy to see that P_{out} is indeed a partition of X, consisting of $\sum_{i=1}^{t} k_i$ clusters. Next, consider $x, y \in X$ and let us bound $\Pr[P_{\text{out}}(x) \ne P_{\text{out}}(y)]$. Observe that a pair of points can be separated only in steps 1 or 3. Therefore,

$$\begin{aligned}
&\Pr\Big[P_{\text{out}}(x) \ne P_{\text{out}}(y)\Big] \\
&\quad \le \Pr\Big[P_{\text{init}}(x) \ne P_{\text{init}}(y)\Big] + \Pr\Big[P_i(f^{K_i}(x)) \ne P_i(f^{K_i}(y)) \mid P_{\text{init}}(x) = P_{\text{init}}(y) = K_i\Big] \\
&\quad \le O(\log n)\frac{\|x-y\|_p}{\log^{1/p} n \cdot \Delta/4} + O(\sqrt{\log n})\frac{\|f^{K_i}(x) - f^{K_i}(y)\|_2}{\frac{1}{2}\Delta/\log^{1/2-1/p} n} \\
&\quad \le O(\log^{1-1/p} n)\frac{\|x-y\|_p}{\Delta},
\end{aligned}$$

where the last inequality follows because each f^{K_i} is non-expanding on its cluster $K_i \subset \ell_p$.

It remains to show that the final clusters all have diameter at most Δ. Let $x, y \in X$ be in the same cluster, i.e., $P_{\text{out}}(x) = P_{\text{out}}(y)$. Then $P_{\text{init}}(x) = P_{\text{init}}(y) = K_i$ and $P_i(f^{K_i}(x)) = P_i(f^{K_i}(y))$. Combining the maximum possible diameter of $P_{\text{init}}(x)$ and $P_i(f^{K_i}(x))$ with the contraction guarantees of $f = f^{K_i}$, we get

$$\frac{2}{p}\Big(2(\log^{1/p} n)\frac{\Delta}{4}\Big)^{1-p/2}\|x-y\|_p^{p/2} \le \|f(x)-f(y)\|_2 \le \frac{\Delta}{2}\log^{1/p-1/2} n.$$

Rearranging this, we obtain $\|x-y\|_p \le \frac{\sqrt[p/2]{p/2}}{2}\Delta \le \Delta$, which completes the proof. ◀

2.2 Capped Decomposition in ℓ_p for $p \in (2, \infty)$

We now present our construction of capped decomposition, which is a notion of decomposability that was introduced in [22] without a concrete name. We start with its definition, and then present our construction.

▶ **Definition 2.4.** *Let (X, ρ) be a metric space. A distribution $\mathcal{D}$ over partitions of X is called* (t, Δ, η)-capped *if*

1. *for every partition $P \in \mathrm{supp}(\mathcal{D})$, all clusters $C \in P$ have $\mathrm{diam}(C) \leq \Delta$; and*
2. *for every $x, y \in X$ such that $\rho(x, y) \leq \frac{\Delta}{t}$,*

$$\Pr_{P \in \mathcal{D}}[P(x) = P(y)] \geq \eta.$$

Observe that here, unlike in Lipschitz decomposition, we have a guarantee on the probability that points $x, y \in X$ are clustered together only if they are within distance $\frac{\Delta}{t}$ of each other, hence the name "capped decomposition". Moreover, the probability bound does not depend on the exact value of $\rho(x, y)$. We say that (X, ρ) admits a (t, η)-capped decomposition, where $\eta = \eta(|X|, t)$, if it admits a (t, Δ, η)-capped decomposition for every $\Delta > 0$. A family of metrics admits a (t, η)-capped decomposition if every metric in the family admits a (t, η)-capped decomposition.

▶ **Theorem 2.5.** *Let $p \in (2, \infty)$. Then every n-point metric in ℓ_p admits a $(t, n^{-O(1/t^q)})$-capped decomposition for all $t \geq 1$, where $q \in (1, 2)$ is such that $\frac{1}{p} + \frac{1}{q} = 1$.*

Previously, such a decomposition was known only for the extreme case $p = 2$ by [22], see Proposition 2.6, and our bound above in fact converges to their bound when $p \to 2$. Our proof of Theorem 2.5 is similar to Theorem 1.3, and relies on two known capped decompositions, that we introduce next, together with the Mazur map Corollary 2.3.

▶ **Proposition 2.6** ([22])**.** *Every n-point subset of ℓ_2 admits a $(t, n^{-O(1/t^2)})$-capped decomposition for all $t \geq 1$.*

▶ **Proposition 2.7** (Implicit in [39])**.** *Every n-point metric space admits a $(t, n^{-O(1/t)})$-capped decomposition for all $t \geq 1$.*

Proof of Theorem 2.5. Let $\Delta > 0$ and $t \geq 1$. Let $X \subset \ell_p$ be an n-point subset of $p \in (2, \infty)$, where q is such that $\frac{1}{p} + \frac{1}{q} = 1$. Construct a partition of X in the following steps:

1. Construct for X a $(t_1 := t^q/4, \Delta_1 := \Delta/4t^{1-q}, n^{-O(1/t^q)})$-capped decomposition $P_{\mathrm{init}} = \{K_1, \ldots, K_t\}$ using Proposition 2.7.
2. Embed each cluster $K_i \subset \ell_p$ into ℓ_2 using the embedding f^{K_i} provided by Corollary 2.3 for $C_0 := \Delta_1$.
3. For each embedded cluster $f^{K_i}(K_i)$ construct a $(t_2 := t^{q/2}/2, \Delta_2 := \Delta/2t^{1-q/2}, n^{-O(1/t^q)})$-capped decomposition $P_i = \{K_i^1, \ldots, K_i^{k_i}\}$ using Proposition 2.6.
4. The final decomposition P_{out} is obtained by taking the preimage of every cluster of every P_i.

It is easy to see that that P_{out} is indeed a partition of X, consisting of $\sum_{i=1}^{t} k_i$ clusters. Next, consider $x, y \in X$ with $\|x - y\|_p \leq \Delta/t$ and let us bound $\Pr[P_{\mathrm{out}}(x) = P_{\mathrm{out}}(y)]$. Observe that $\Delta_1/t_1 = \Delta_2/t_2 = \Delta/t$, and therefore

$$\begin{aligned}
&\Pr\Big[P_{\mathrm{out}}(x) = P_{\mathrm{out}}(y)\Big] \\
&\quad = \Pr\Big[P_{\mathrm{init}}(x) = P_{\mathrm{init}}(y)\Big] \cdot \Pr\Big[P_i(f^{K_i}(x)) = P_i(f^{K_i}(y)) \mid P_{\mathrm{init}}(x) = P_{\mathrm{init}}(y) = K_i\Big] \\
&\quad \geq n^{-O(1/t^q)} \cdot n^{-O(1/t^q)} = n^{-O(1/t^q)},
\end{aligned}$$

where the inequality follows because each f^{K_i} is non-expanding on its cluster $K_i \subset \ell_p$.

It remains to show that each cluster has diameter at most Δ. Let $x, y \in X$ be in the same cluster, i.e., $P_{\text{out}}(x) = P_{\text{out}}(y)$. Then $P_{\text{init}}(x) = P_{\text{init}}(y) = K_i$ and $P_i(f^{K_i}(x)) = P_i(f^{K_i}(y))$. Combining the maximum possible diameter of $P_{\text{init}}(x)$ and $P_i(f^{K_i}(x))$ with the contraction guarantees of $f = f^{K_i}$, we get

$$\frac{2}{p}\Big(2\frac{\Delta}{4t^{1-q}}\Big)^{1-p/2}\|x-y\|_p^{p/2} \le \|f(x)-f(y)\|_2 \le \frac{\Delta}{2t^{1-q/2}}.$$

Rearranging this, we obtain $\|x-y\|_p \le \frac{\sqrt[p/2]{p/2}}{2}\Delta \le \Delta$, which completes the proof. ◀

2.3 Spanners in ℓ_p for $p \in (2, \infty)$

We can now prove Theorem 1.4, by applying the following spanner construction of [22].

▶ **Theorem 2.8** ([22]). *Let (X, ρ) be an n-point metric space admitting a (t, η)-capped decomposition for some $t \ge 1$. Then, for every $\epsilon \in (0, 1/8)$, there exists a $(2+\epsilon)t$-spanner for X with $O_\epsilon(\frac{n}{\eta} \cdot \log n \cdot \log t)$ edges and lightness $O_\epsilon(\frac{t}{\eta} \cdot \log^2 n)$.*

Proof of Theorem 1.4. The proof follows directly by combining Theorem 2.5 and Theorem 2.8, as we can assume $t = O(\log n)$ without loss of generality. ◀

3 Spanners in ℓ_p for $p \in (1, 2)$

This section presents an improved construction of geometric spanners in ℓ_p for $p \in (1, 2)$. Previously, $O(t)$-spanners of size $O(n^{1+\log^2 t/t^p})$ for all $t \ge 1$ were constructed in [22]; in particular, setting $t = (\log n \log\log n)^{1/p}$ yields an $O(t)$-spanner of near-linear size $\tilde{O}(n)$. We first present in Section 3.1 two different constructions of near-linear-size spanners with a slightly better stretch. Then in Section 3.2 we use yet another technique, namely Locality Sensitive Hashing (LSH), to slightly improve the construction of [22] of spanners with general stretch $O(t)$.

3.1 Spanners of Near-Linear Size

We slightly improve the near-linear size spanner construction of [22] by shaving the $(\log\log n)^{1/p}$ factor from the stretch, as follows.

▶ **Theorem 3.1.** *For every fixed $p \in (1, 2)$, every n-point metric $X \subset \ell_p$ admits an $O(\log^{1/p} n)$-spanner of size $\tilde{O}(n)$.*

We present two related but different proofs for this theorem. Both are based on modifying the spanner algorithm for ℓ_2 from [25], and therefore we start with an overview of that algorithm. Given an input set $X \subseteq \ell_2$, the algorithm begins by constructing a hierarchical set of 2^i-nets $X = N_0 \supseteq N_1 \supseteq \cdots \supseteq N_{\log \Delta_X}$, where we assume that the minimum and maximum distances in X are 1 and Δ_X, respectively. Then, for each level i, it constructs an $(O(\sqrt{\log n}), O(\sqrt{\log n}) \cdot 2^{i+1})$-Lipschitz decomposition of N_i by combining the JL Lemma [27] with the Lipschitz decomposition of [13]. For each cluster in it, the algorithm add to the spanner edges in a star-like fashion, meaning that all cluster points are connected to one arbitrary point within the cluster. The last two steps are repeated $O(\log n)$ times to ensure that with high probability, for each level i, every $x, y \in N_i$ with $\|x-y\|_2 \le 2^{i+1}$ are clustered together in at least one of the $O(\log n)$ repetitions. It is shown in [25] that this algorithm constructs an $(O(\sqrt{\log n}))$-spanner of size $\tilde{O}(n)$.

Proof of Theorem 3.1 via Lipschitz Decomposition. Observe that the above algorithm of [25] uses the fact that the points lie in ℓ_2 only for the construction of Lipschitz Decompositions, and relies on an optimal decomposition for finite ℓ_2 metrics to conclude that the spanner's stretch is $O(\beta_n^*(\ell_2))$. For finite ℓ_p metrics, $p \in (1,2)$, we can use instead a Lipschitz decomposition from [35, 40], which has $\beta = \frac{O(\log^{1/p} n)}{p-1}$, to conclude the claimed stretch. ◀

We next present a proof that modifies the algorithm of [25] differently, and relies on a decomposition that is similar to a Lipschitz decomposition but has slightly weaker guarantees. Interestingly, this technique yields a slightly stronger result than Theorem 3.1, where p need not to be fixed and can depend on n (e.g., $p \to 1$). We proceed to introduce some technical results from [9] regarding a weak form of dimensionality reduction in ℓ_p, for $p \in [1,2]$, which are needed for our proof.

▶ **Definition 3.2** ([44]). *Let (X,ρ), (Y,τ) be metric spaces and $[a,b]$ be a real interval. An embedding $f : X \to Y$ is called* $[a,b]$-range preserving with distortion $D \geq 1$ *if there exists $c > 0$ such that for all $x, x' \in X$:*

1. *If $a \leq \rho(x,x') \leq b$, then $\rho(x,x') \leq c \cdot \tau(f(x), f(x')) \leq D \cdot \rho(x,x')$.*
2. *If $\rho(x,x') > b$, then $c \cdot \tau(f(x), f(x')) \geq b$.*
3. *If $\rho(x,x') < a$, then $c \cdot \tau(f(x), f(x')) \leq D \cdot a$.*

We say that (X,ρ) admits *an R-range preserving embedding* into (Y,τ) with distortion D, if for all $u > 0$, there exists a $[u, uR]$-range preserving embedding into Y with distortion D.

▶ **Theorem 3.3** ([9]). *Let $1 \leq p \leq 2$. For every n-point set $S \subset \ell_p$, and for every range parameter $R > 1$, there exists an R-range preserving embedding $f : S \to \ell_p^k$ with distortion $1+\epsilon$, such that $k = O\left(\frac{R^{O(1/\epsilon)} \cdot \log n}{\epsilon}\right)$.*

Proof of Theorem 3.1 via Weak Dimension Reduction. Observe that the above algorithm of [25] only requires the decomposition of each net N_i to ensure that points $x, y \in N_i$ with $\|x-y\|_2 \leq 2^{i+1}$ are clustered together with constant probability, and that the diameter of all clusters is at most $O(\sqrt{\log n}) \cdot 2^i$; of course, for $X \subset \ell_p$, $p \in (1,2)$, we replace the $O(\sqrt{\log n})$ factor with $O(\log^{1/p} n)$. A careful examination shows that these properties are preserved by first reducing the dimension using the range-preserving embedding provided by Theorem 3.3 with $\varepsilon = \frac{1}{2}$ and $R = 2$, and then constructing a Lipschitz decomposition for the image points in $\ell_p^{O(\log n)}$ using [13]. ◀

3.2 Spanners with Stretch-Size Tradeoff

We now present, in Theorem 1.5, a construction of $O(t)$-spanners in ℓ_p, where $p \in (1,2)$, of size $\tilde{O}(n^{1+1/t^p})$ for all $t \geq 1$, which slightly improves over the $O(t)$-spanners of size $\tilde{O}(n^{1+\log^2 t/t^p})$ from [22]. It is worth noting that Theorem 1.5 generalizes the results of Theorem 3.1, and thus provides an alternative proof for it.

▶ **Definition 3.4** (LSH [26]). *Let $\mathcal{H}$ be a family of hash functions mapping a metric (X,ρ) to some universe U. We say that $\mathcal{H}$ is* (r, tr, p_1, p_2)-sensitive *if for every $x, y \in X$, the following is satisfied:*

1. *If $\rho(x,y) \leq r$, then $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \geq p_1$.*
2. *If $\rho(x,y) > tr$, then $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \leq p_2$.*

Such $\mathcal{H}$ is called an LSH family *with parameter $\gamma := \frac{\log(1/p_1)}{\log(1/p_2)}$.*

▶ **Lemma 3.5** ([22]). *Let (X,ρ) be a metric space such that for every $r > 0$, there exists a (r, tr, p_1, p_2)-sensitive LSH family with parameter γ. Then (X,ρ) admits a $(t, n^{-\mathcal{O}(\gamma)})$-capped decomposition.*

For $p = 2$, the LSH family constructed in [3] can be used in Lemma 3.5 to conclude that ℓ_2 admits a $(t, n^{-O(1/t^2)})$-capped decomposition for every $t \geq 1$ [22], thereby proving Theorem 1.5 for this case of $p = 2$. In a similar fashion, an LSH family constructed in [43] for $p \in (1, 2)$ was used in [22] to show that these spaces admit a $(t, n^{-O(\log^2 t/t^p)})$-capped decomposition. We observe that this result can be improved by replacing the LSH family from [43], with an alternative one that is briefly mentioned in [4], and consequently prove Theorem 1.5. For completeness, we reproduce this LSH family for ℓ_p, where $p \in (1, 2)$.

▶ **Lemma 3.6** ([4]). *Let $p \in (1, 2)$, $r > 0$, and large enough $t > 1$. Then there exists a (r, tr, p_1, p_2)-sensitive LSH family for ℓ_p with parameter $\gamma = \frac{1}{t^p} + o(1)$.*

Proof. Let $p \in (1, 2)$, $r > 0$, and sufficiently large $t > 1$. Let $f : \ell_p \to \ell_2$ be the isometric embedding of the $(p/2)$-snowflake of ℓ_p into ℓ_2 from [28, Theorem 4.1]. Take $r' = r^{p/2}$ and $t' = t^{p/2}$, and let $\mathcal{H}$ be the $(r', t'r', p_1, p_2)$-sensitive LSH family for ℓ_2 with parameter $\gamma = \frac{1}{t'^2} + o(1)$ from [3]. Observe that, for every $x, y \in \ell_p$, if $\|x - y\|_p \leq r$, then $\|f(x) - f(y)\|_2 = \|x - y\|_p^{p/2} \leq r^{p/2} = r'$, and thus

$$\Pr_{h \in \mathcal{H}}[h(f(x)) = h(f(y))] \geq p_1.$$

Similarly, if $\|x - y\|_p > tr$, then $\|f(x) - f(y)\|_2 = \|x - y\|_p^{p/2} > (tr)^{p/2} = t'r'$, and hence

$$\Pr_{h \in \mathcal{H}}[h(f(x)) = h(f(y))] \leq p_2.$$

We therefore conclude that $\mathcal{H} \circ f$ is an (r, tr, p_1, p_2)-sensitive LSH family for ℓ_p with parameter $\gamma = \frac{1}{t^p} + o(1)$. ◀

Proof of Theorem 1.5. The proof follows immediately by constructing a capped decomposition based on Lemma 3.5 and Lemma 3.6, and using it in the spanner construction from Theorem 2.8. ◀

▶ Remark 3.7. While [28, Theorem 4.1] does not provide an efficiently computable embedding, one can compute such an embedding for a finite set of points in polynomial time by [37].

4 Distance Labeling

In the distance labeling model, a scheme is designed for an entire a family $\mathcal{X}$ of n-point metrics (and in some scenarios, all these metrics have the same point set X, e.g., different graphs on the same vertex set). A *scheme* is an algorithm that preprocesses each metric X in $\mathcal{X}$ and assigns to each point $x \in X$ a label $l(x)$.

▶ **Definition 4.1.** *A scheme is* a distance labeling *with approximation $D \geq 1$ and label size of k if*

1. *every label (for every point in every metric in $\mathcal{X}$) consists of at most k bits; and*
2. *there is an algorithm $\mathcal{A}$ that, given the labels $l(x), l(y)$ of two points x, y in a metric $(X, \rho) \in \mathcal{X}$ (but not given (X, ρ) or the points x, y), outputs an estimate $\mathcal{A}(l(x), l(y))$ that satisfies*

$$\rho(x, y) \leq \mathcal{A}(l(x), l(y)) \leq D \cdot \rho(x, y).$$

The following theorem was presented in [24] with limited details, and we include a proof of it below for completeness.

▶ **Theorem 4.2** ([24]). *Let $\mathcal{X}$ be a family of n-point metrics, and assume that all the pairwise distances in all metrics (X, ρ) in $\mathcal{X}$ are in the range $[1, \Delta_{max}]$. Then $\mathcal{X}$ admits a distance-labeling scheme with approximation $O(\beta^*(\mathcal{X}))$ and label size $O(\log n \log \Delta_{\max})$ bits.*

It is straightforward to see that Theorem 1.6 follows by combining Theorem 4.2 and Theorem 1.3.

Proof of Theorem 4.2. We first describe the preprocessing algorithm, denoting $\beta := \beta^*(\mathcal{X})$. Perform the following steps for all levels $i = 0, \ldots, \log \Delta_{\max}$. Begin by constructing a $(\beta, \Delta_i := 4\beta 2^i)$-Lipschitz decomposition, and observe that every two points $x, y \in X$ with $\rho(x, y) \leq 2^i$ are separated with probability at most $\frac{1}{4}$. Then, assign a random bit to each cluster, and observe that if two points are at distance greater than Δ_i, they always fall in different clusters, hence, the probability that they are assigned the same bit is exactly $\frac{1}{2}$, and if they are at distance at most $2^i = \Delta_i/(4\beta)$ they are assigned the same bit with probability at least $\frac{3}{4}$. Repeat the last two steps $k = O(\log n)$ times, and then with high probability, every two points x, y are assigned the same bit at least $\frac{5}{8}k$ times if $\rho(x, y) \leq \Delta_i/(4\beta)$ and fewer than $\frac{5}{8}k$ times if $\rho(x, y) > \Delta_i$. Finally, label each point by concatenating the bit assigned to its cluster in all the repetitions at all levels.

The label-size analysis is straightforward. It remains to show that, given two labels $l(x), l(y)$, it is possible to approximate the distance $\rho(x, y)$ within factor $O(\beta)$. This can be achieved by identifying the smallest level i such that x and y are assigned the same bit at least $\frac{5}{8}k$ times, and then the above analysis (used in contrapositive form) implies that $\Delta_{i-1}/(4\beta) < \rho(x, y) \leq \Delta_i$, where by convention $\Delta_{-1} := 1$. ◀

5 Future Directions

Lipschitz Decompositions. We stress that our decomposition in Theorem 1.3 employs a data-dependent approach, and is not oblivious to the input set X (as, say, the decomposition for ℓ_2 in [13], even when applied together with the JL Lemma). In retrospect, this feature is perhaps not very surprising, because data-dependent approaches have been already shown to be effective for central problems, such as nearest neighbor search [5, 33]. We thus mention that a major open problem in the field is whether dimension reduction is possible in ℓ_p for $p \neq 1, 2, \infty$; we know that for $p > 2$ this is not possible via an oblivious mapping [14], raising the question whether data-dependent mappings can overcome this limitation.

Geometric Spanners. The geometric spanners in [25, 22] for ℓ_p, $1 < p \leq 2$, are not known to be optimal, i.e., we do not know of matching lower bounds, except for the more restricted case of 2-hop spanners [25]. We conjecture that tight instances exist in these spaces, i.e., the spanner bounds obtained in [25, 22] are optimal for every stretch t. We similarly do not know of matching lower bounds for the geometric spanners in ℓ_p, for fixed $2 \leq p < \infty$, that we obtain in Theorem 1.4, and it is quite plausible that our upper bounds are not tight. We do know however, based on known results, that for every n, there exist tight instances in ℓ_p for $p = \Omega(\log n)$.

References

1 Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 79–88. Association for Computing Machinery, 2014. doi:10.1145/2591796.2591849.

2 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020. doi:10.1016/j.cosrev.2020.100253.

3 A. Andoni and P. 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.

4 A. Andoni and P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 43, pages 1135–1150. CRC Press, 3rd edition, 2017. doi:10.1201/9781315119601.

5 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.

6 S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. *J. Amer. Math. Soc.*, 21(1):1–21, 2008.

7 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

8 Y. Bartal. Graph decomposition lemmas and their role in metric embedding methods. In *12th Annual European Symposium on Algorithms*, volume 3221 of *LNCS*, pages 89–97. Springer, 2004. doi:10.1007/978-3-540-30140-0_10.

9 Yair Bartal and Lee-Ad Gottlieb. Dimension reduction techniques for ℓ_p $(1 < p < 2)$ with applications. In *32nd International Symposium on Computational Geometry, SoCG 2016*, volume 51 of *LIPIcs*, pages 16:1–16:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.SOCG.2016.16.

10 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.

11 Yoav Benyamini and Joram Lindenstrauss. *Geometric nonlinear functional analysis*, volume 48. American Mathematical Soc., 1998.

12 Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2371–2379, 2019. doi:10.1137/1.9781611975482.145.

13 M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *39th Annual Symposium on Foundations of Computer Science*, pages 379–388, 1998.

14 Moses Charikar and Amit Sahai. Dimension reduction in the ℓ_1 norm. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 551–560. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181979.

15 A. Dvoretzky. Some results on convex bodies and Banach spaces. In *Proc. Internat. Sympos. Linear Spaces (Jerusalem, 1960)*, pages 123–160. Jerusalem Academic Press, Jerusalem, 1961.

16 David Eppstein. Spanning trees and spanners. In *Handbook of Computational Geometry*, pages 425–461. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50010-3.

17 J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

18 Arnold Filtser. On strong diameter padded decompositions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*, volume 145 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.APPROX-RANDOM.2019.6.

19 Arnold Filtser. Scattering and sparse partitions, and their applications. *ACM Trans. Algorithms*, 20(4):30:1–30:42, 2024. doi:10.1145/3672562.

20 Arnold Filtser, Tobias Friedrich, Davis Issac, Nikhil Kumar, Hung Le, Nadym Mallek, and Ziena Zeif. Optimal padded decomposition for bounded treewidth graphs. *CoRR*, abs/2407.12230, 2024. doi:10.48550/arXiv.2407.12230.

21 Arnold Filtser, Lee-Ad Gottlieb, and Robert Krauthgamer. Labelings vs. embeddings: On distributed and prioritized representations of distances. *Discrete & Computational Geometry*, 71:849–871, 2024. doi:10.1007/s00454-023-00565-2.

22 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.

23 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004. doi:10.1016/J.JALGOR.2004.05.002.

24 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2003. doi:10.1109/SFCS.2003.1238226.

25 Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 804–809. SIAM, 2013. doi:10.1137/1.9781611973105.57.

26 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.

27 W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, pages 189–206. Amer. Math. Soc., Providence, RI, 1984.

28 Nigel J. Kalton. The nonlinear geometry of Banach spaces. *Revista Matematica Complutense*, 21(1):7–60, 2008. URL: http://eudml.org/doc/42299.

29 R. Krauthgamer and J. R. Lee. The intrinsic dimensionality of graphs. *Combinatorica*, 27(5):551–585, 2007. doi:10.1007/s00493-007-2183-y.

30 R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. *Geometric And Functional Analysis*, 15(4):839–858, 2005. doi:10.1007/s00039-005-0527-6.

31 Robert Krauthgamer, Nir Petruschka, and Shay Sapir. The power of recursive embeddings for ℓ_p metrics, 2025. doi:10.48550/arXiv.2503.18508.

32 Robert Krauthgamer and Tim Roughgarden. Metric clustering via consistent labeling. *Theory of Computing*, 7(5):49–74, 2011. doi:10.4086/toc.2011.v007a005.

33 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.

34 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*, pages 295–308. ACM, 2023. doi:10.1145/3564246.3585185.

35 J. R. Lee and A. Naor. Metric decomposition, smooth measures, and clustering. Unpublished manuscript, 2003. URL: https://web.math.princeton.edu/~naor/homepagefiles/cluster.pdf.

36 James R. Lee and Anastasios Sidiropoulos. Genus and the geometry of the cut graph. In *21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 193–201. SIAM, 2010. doi:10.1137/1.9781611973075.18.

37 N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. doi:10.1007/BF01200757.

38 J. Matoušek. On embedding expanders into l_p spaces. *Israel J. Math.*, 102:189–197, 1997.

39 M. Mendel and A. Naor. Ramsey partitions and proximity data structures. *J. Eur. Math. Soc.*, 9(2):253–275, 2007.

40 Assaf Naor. Probabilistic clustering of high dimensional norms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 690–709. SIAM, 2017. doi:10.1137/1.9781611974782.44.

41 Assaf Naor. *Extension, separation and isomorphic reverse isoperimetry*, volume 11 of *Mem. Eur. Math. Soc.* European Mathematical Society (EMS), 2024. doi:10.4171/MEMS/11.

42 Assaf Naor and Kevin Ren. ℓ_p has nontrivial Euclidean distortion growth when $2 < p < 4$, 2025. arXiv:2502.10543.

43 Huy L. Nguyen. Approximate nearest neighbor search in ℓ_p. *CoRR*, abs/1306.3601, 2013. arXiv:1306.3601.

44 R. Ostrovsky and Y. Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002. doi:10.1145/506147.506149.

45 D. Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000. doi:10.1002/(SICI)1097-0118(200003)33:3\%3C167::AID-JGT7\%3E3.0.CO;2-5.

46 D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

47 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989. doi:10.1137/0218050.

48 S. Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Proceedings of the 15th Annual Symposium on Computational Geometry*, pages 300–306. ACM, 1999. doi:10.1145/304893.304983.

49 Uri Zwick. Exact and approximate distances in graphs – A survey. In *Algorithms – ESA 2001, 9th Annual European Symposium*, volume 2161 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2001. doi:10.1007/3-540-44676-1_3.

Hard Diagrams of Split Links

Corentin Lunel
INRIA Université Côte d'Azur, Montpellier, France

Arnaud de Mesmay
Laboratoire d'Informatique Gaspard Monge, 5 Boulevard Descartes, Champs-sur-Marne, 77454 Marne-la-Vallée Cedex 2, France

Jonathan Spreer
School of Mathematics and Statistics, The University of Sydney, Australia

Abstract

Deformations of knots and links in ambient space can be studied combinatorially on their diagrams via local modifications called Reidemeister moves. While it is well-known that, in order to move between equivalent diagrams with Reidemeister moves, one sometimes needs to insert excess crossings, there are significant gaps between the best known lower and upper bounds on the required number of these added crossings. In this article, we study the problem of turning a diagram of a split link into a split diagram, and we show that there exist split links with diagrams requiring an arbitrarily large number of such additional crossings. More precisely, we provide a family of diagrams of split links, so that any sequence of Reidemeister moves transforming a diagram with c crossings into a split diagram requires going through a diagram with $\Omega(\sqrt{c})$ extra crossings. Our proof relies on the framework of bubble tangles, as introduced by the first two authors, and a technique of Chambers and Liokumovitch to turn homotopies into isotopies in the context of Riemannian geometry.

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

Keywords and phrases Knot theory, hard knot and link diagrams, Reidemeister moves, extra crossings, split links, bubble tangles, compression representativity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.67

Related Version *Full Version*: `https://arxiv.org/abs/2412.03372`

Funding *Corentin Lunel*: Partially supported by the ANR project ANR-20-CE48-0007 (AlgoKnot). *Jonathan Spreer*: Supported by the Australian Research Council under the Discovery Project scheme (grant number DP220102588).

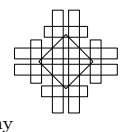
Acknowledgements We would like to thank Clément Maria for helpful discussions and Stephan Tillmann for his comments and suggestions on an earlier version of the paper. Moreover, the authors want to thank the anonymous referees for their helpful comments.

1 Introduction

The Reidemeister theorem [32] is a fundamental and powerful result in low-dimensional topology. It states that any two link diagrams represent equivalent links[1] if and only if they can be related by a sequence of planar isotopies and local moves, called **Reidemeister moves**, pictured in Figure 1. This theorem is at the heart of many theoretical results as well as computational applications. Indeed, many knot invariants, from the most basic ones such as tri-colourability [1, Section 1.5] to more advanced ones such as the Jones polynomial [20]

[1] A link is an embedding of a collection of circles into $\mathbb{R}^3$ and two links are considered equivalent if they are ambient isotopic. Link diagrams are projections of links into a plane as shown, for instance, in Figure 2. We refer to Section 2 for standard definitions in knot theory.

© Corentin Lunel, Arnaud de Mesmay, and Jonathan Spreer;
licensed under Creative Commons License CC-BY 4.0



or Khovanov homology [3], can be shown to be invariants by demonstrating that they are not modified by Reidemeister moves. From a computational point of view, the Reidemeister theorem allows for a discretisation of the space of possible transformations to consider when testing for knot equivalence. This fact is at the root of a straightforward algorithm to study algorithmic problems in knot theory: at the level of diagrams, apply Reidemeister moves in a random or brute-force manner until a desired property is verified.

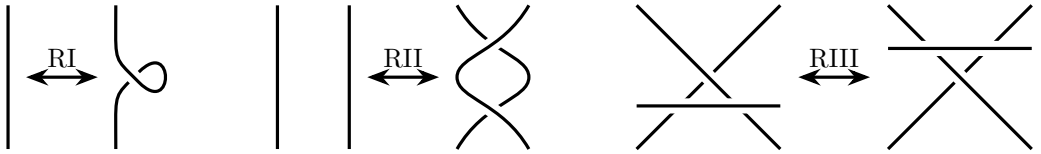


Figure 1 The three Reidemeister moves RI, RII, and RIII.

A primary example of such a knot theory problem is recognising the **unknot**, that is, the unique knot admitting a diagram with no crossings. This is a first instance of the fundamental problem of knot theory of deciding whether two knots are equivalent. It turns out that some unknot diagrams, called **hard unknot diagrams** or **culprits** [6, 21], exhibit an unwanted behaviour for the above algorithm. Namely, given an initial diagram $\mathcal{D}$ of the unknot, the largest number of crossings of a diagram in any sequence of Reidemeister moves from $\mathcal{D}$ to the 0-crossing diagram must be larger than in $\mathcal{D}$. This means one first needs to *add* crossings before being able to reach the 0-crossing diagram. The existence of such diagrams implies that it is not possible to untangle an unknot diagram by only applying Reidemeister moves that do not increase the number of crossings. An example of such a hard unknot diagram, called the **Goeritz culprit**, is shown in Figure 2. In [6], it is shown that, when diagrams in $\mathbb{S}^2$ are considered, at least one extra crossing is required to untangle this unknot. We do not know of systematic techniques or methods to prove easily that a given diagram of the unknot is hard. Instead, all proofs known to the authors resort to an exhaustive search in the graph of Reidemeister moves, which quickly becomes infeasible. Recently, new techniques based on reinforcement learning have been applied to find large numbers of hard unknot diagrams [2], but the proofs of hardness still involved exhaustive enumerations.

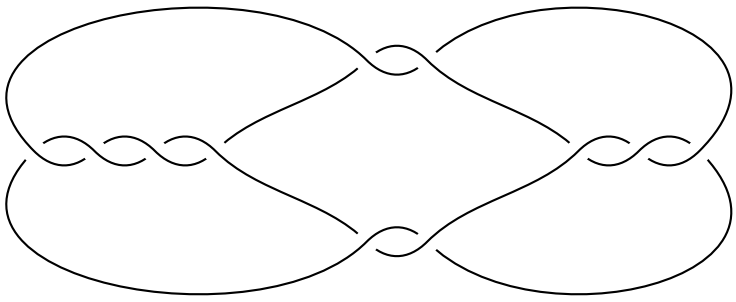


Figure 2 The Goeritz culprit: using Reidemeister moves in $\mathbb{S}^2$, one must add at least one extra crossing to untangle this unknot diagram.

Following the notations of [6, 21] we denote the number of crossings in a diagram $\mathcal{D}$ by $\mathrm{cr}(\mathcal{D})$. For diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ of equivalent links, and a sequence R of r Reidemeister moves transforming $\mathcal{D}_1$ into $\mathcal{D}_2$, we define $\mathbf{Top}(\boldsymbol{\mathcal{D}_1}, \boldsymbol{R}) = \max_{i\in\{0,1,\dots,r\}}\{\mathrm{cr}(\mathcal{D}^i) - \mathrm{cr}(\mathcal{D}_1)\}$ where $\mathcal{D}^i$, $0 \leq i \leq r$, is the diagram after performing the first i moves of the sequence. The minimal number of extra crossings to pass from $\mathcal{D}_1$ to $\mathcal{D}_2$ is denoted by $\mathbf{Add}(\boldsymbol{\mathcal{D}_1}, \boldsymbol{\mathcal{D}_2})$,

which, formally, is the minimum of $\mathrm{Top}(\mathcal{D}_1, R)$ taken over all sequences of Reidemeister moves R transforming $\mathcal{D}_1$ into $\mathcal{D}_2$. Hence, $\mathrm{Add}(\mathcal{D}_1, \mathcal{D}_2)$ is a lower bound on the number of crossings we must add in any sequence of Reidemeister moves performed on $\mathcal{D}_1$ to reach $\mathcal{D}_2$.

Let $\mathcal{D}$ be a diagram of the unknot and $\hat{\mathcal{D}}$ be its 0-crossing diagram. Naturally, $\mathcal{D}$ is a hard unknot diagram if and only if $\mathrm{Add}(\mathcal{D}, \hat{\mathcal{D}})$ is positive. This measure of complexity is called m in [6] (see also the *recalcitrance* in [21]). Studying these complexity measures and hard unknot diagrams is trickier than one might think. In fact, one of the purposes of [6] is to confirm or invalidate claims about previously known hard unknot diagrams using an exhaustive computer search over all possible sequences of Reidemeister moves. Still, the "hardest" known diagrams of the unknot have only been verified to require at least three extra crossings before they can be untangled. In contrast, the following conjecture is folklore.

▶ **Conjecture 1.** *Let m be an integer and let $\hat{\mathcal{D}}$ be the 0-crossing diagram of the unknot. Then there exists a diagram of the unknot $\mathcal{D}$ with n crossings such that any sequence of Reidemeister moves from $\mathcal{D}$ to $\hat{\mathcal{D}}$ passes through a diagram with at least $n + m$ crossings.*

Note that a proof of Conjecture 1, formulated in terms of recalcitrance, is claimed in [21], but concerns about the correctness of this proof are raised in [6].

In this article, as a possible step towards a proof of Conjecture 1, we shift our focus to **split links**. A link L is said to be split if there exists a sphere disjoint from L separating at least two link components of L. If such a sphere exists, there exists a link diagram in which two sublinks of L are disjoint: they are separated by a circle in the plane. Such a diagram is called a **split diagram**. By capping off the aforementioned circle with one disc above and one disc below the plane of projection, we can verify that conversely a split diagram witnesses a split link. Determining if a link is split is known as the **splitting problem**.

Given a split link L with a diagram $\mathcal{D}_1$, we study $\mathrm{Add}(\mathcal{D}_1, \mathcal{D}_2)$ where $\mathcal{D}_2$ is a split diagram of L. If the minimum of $\mathrm{Add}(\mathcal{D}_1, \mathcal{D}_2)$ over all split diagrams $\mathcal{D}_2$ of L, called the **crossing-complexity** and denoted by $\mathbf{CC}(D_1)$, is positive, we call $\mathcal{D}_1$ a **hard split link**.

Our results. We exhibit a family of link diagrams $\mathcal{D}(p, q)$ of split links $L(p, q)$ with two unlinked sublinks. The first sublink $M = M(p, q)$ is made of two linked torus knots $T_{p,q}$, and the second is an unknot U surrounding one of the torus knots (see Figure 3 for an illustration). For any split diagram $\mathcal{D}'(p, q)$ of $L(p, q)$, we prove Theorem 2, implying that $\mathrm{Add}(\mathcal{D}(p, q), \mathcal{D}'(p, q)) = \Omega(\min(p, q))$. More precisely, we have the following main theorem.

▶ **Theorem 2.** *For all $n \geq 2$, any sequence of Reidemeister moves transforming diagram $\mathcal{D}(n, n + 1)$ of the link $L(n, n + 1)$ with $2n^2 + 2$ crossings into a split diagram passes through a diagram with at least $2n^2 + \frac{2}{3}n$ crossings. In particular, there exist hard split links of arbitrarily large crossing-complexity.*

This is to our knowledge the first construction of split links with super-constant crossing-complexity, and we cannot rely on exhaustive search methods to prove Theorem 2. Instead, we develop new techniques to lower-bound the crossing-complexity. It was proved by Dynnikov [12, Theorem 2] that $\mathrm{CC}(D) = O(|D|^2)$, where $|D|$ denotes the number of crossings in a diagram D, so there is still a significant gap between this upper bound and the $\Omega(\sqrt{|D|})$ lower bound provided by our theorem.

▶ Remark 3. Theorem 2 is one of the many knot-theoretical statements that seems intuitively clear but is surprisingly delicate to prove: inspecting Figure 3, "clearly" the only way to split U from the two components of M is through an overlay of U on top of one of these

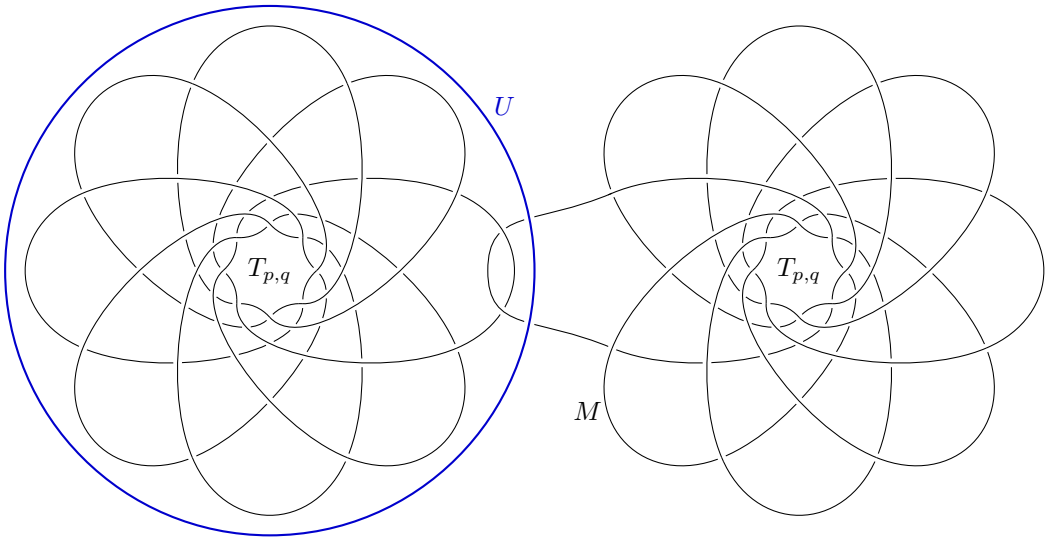


Figure 3 The link diagram $\mathcal{D}(p,q)$, $(p,q) = (7,8)$: two linked torus knots $T_{7,8}$ and an unknot U.

components, yielding the claimed increase in the number of crossings. In fact, the following idea to prove Theorem 2 may seem straightforward: if there is a sequence of Reidemeister moves splitting U from M without adding too many crossings, then we can find a continuous family of planes $(P_t)_{t\in\{0,1\}}$ in $\mathbb{R}^3$ sweeping one of the torus knots in M such that none of the planes P_t intersects M in too many points. Known results on *bridge position* [30, 31] and *thin position* [14, 29] of torus knots then imply that such a family cannot exist[2].

However, this proof idea has two issues. First, during a sequence of Reidemeister moves, the unknot U might come to intersect itself, making it difficult to define a continuous family of planes that mimics its intersections with M. Second, the Reidemeister moves might lead U to move back and forth, leading to a non-monotone behaviour of the planes P_t, which is not allowed in the aforementioned positions commonly used in knot theory. This leads us to rely on more advanced tools to prove Theorem 2.

Overview of the proof. The proof of Theorem 2 works by contradiction. That is, we start by assuming that there exists a sequence of Reidemeister moves R transforming $\mathcal{D}(n,n+1)$ into a split diagram and such that $\mathrm{Top}(\mathcal{D}(n,n+1),R)$ remains small. This implies that, throughout performing R, the number of crossings involving U and M always remains small.

The main tool that we rely on is the framework of bubble tangles introduced by the first and second authors in [26]. More precisely, we use the evolution of U throughout R to define a collection of 2-dimensional spheres that continuously sweep $\mathbb{S}^3$ (a *sweep-out*) and that – according to our assumptions – all have a small number of intersections with M. This sweep-out resembles the sphere decompositions of [26], but presents two notable differences. On the one hand it is simpler: it is linear and features no double bubbles. On the other hand, it is not monotone: a sphere involved in this sweep-out may go back-and-forth, while this behaviour is not allowed in the sphere decompositions from [26]. Despite this last difference, the bubble tangles, which are obstructions to thin sphere decompositions developed in [26], are versatile enough to show that the existence of this sweep-out leads to a contradiction.

[2] This argument is the basis of the proof in [8] showing that torus knot diagrams have high tree-width.

As mentioned in Remark 3, building the sweep-out is not straightforward. Intuitively, one would like to lift each unknot U to a sphere by capping it off above and below the diagram. However, the projection of the unknot U may intersect itself during the sequence of Reidemeister moves, which complicates the process. We alleviate this problem using results and methods from Chambers and Liokumovich [7] to transform homotopies of curves on a Riemannian surface into isotopies of similar length. The connection is the following: we think of the projection of the link M as a discrete metric for curves in the projection plane. In this discrete model, the **length** of a curve is given by its number of intersections with M, similarly to the cross-metric model commonly used in computational topology of surfaces (see for example [11]). Our assumptions imply that there exists a homotopy of the projection of U where intermediate curves all have small length. The techniques of [7] show that this implies that there also exists an isotopy of the projection of U with the same bounds on lengths. Since such an isotopy must consist of simple curves, we can then lift it into a sweep-out of $\mathbb{S}^3$ with 2-spheres, which all have a controlled number of intersections with M. However, a subtlety is that we cannot use the results of [7] out of the box. Indeed, the discrete metric defined by the link M is not fixed but evolves with the Reidemeister moves in R. In Section 3 we explain why the proof techniques in [7] can be adapted to deal with this issue.

The tools of the first and second authors [26] are then used to find a contradiction between the existence of the sweep-out and the topological properties of our link M. Namely, $L(n, n+1)$ consists of two torus knots $T_{n,n+1}$, which are embedded on tori in a specific way (they both have high *compression-representativity*). Hence, they can be used to define two **bubble tangles** using [26, Theorem 1.2] (In a nutshell, a bubble tangle is a way of choosing a "small side" for each sphere of the sweep-out, where the intuition is that the small side should be easy to sweep, see Section 2 for a formal definition). We then prove that, since in the initial diagram U lies in-between the two tori, different small sides are chosen for the corresponding sphere by each of the two bubble tangles. But the existence of the sweep-out with a small number of intersections with M forces the small sides of both bubble tangles to agree, leading to a contradiction.

Related work. The splitting problem has been studied several times as a useful and easier problem for understanding the unknot recognition problem [12, 23]. In 1961, Haken used normal surface theory to show that it is decidable [15]. Later, it was determined to be in **NP** [16] and also in co-**NP** [24, Theorem 1.6].

Several decision problems related to the splitting problem have been studied as well. For instance, the problem of deciding whether changing at most k crossings can transform a link diagram into the diagram of a split link is known to be **NP**-hard [22], and Lackenby provided an algorithm to detect links which can be split using exactly one crossing change [25].

Another natural question is to ask for the minimal number of Reidemeister moves needed to split a diagram. An exponential bound for this number was first provided by [19], and this bound was later greatly improved by Lackenby in [23], where he provided a polynomial bound using a combination of normal surface theory [16] and Dynnikov's work on grid diagrams [12]. There is a quadratic lower bound on the number of moves needed to untangle a specific unknot diagram in [17], and it was shown in [9, 10] that finding the shortest sequence of Reidemeister moves to untangle an unknot is **NP**-hard.

Organisation of this paper. After going through our setup in Section 2, we explain how to use the results of [7] in Section 3. This step is crucial for our definition of sweep-outs. Then, we exploit obstructions from [26] to prove Theorem 2 in Section 4.

2 Setup and definitions

Knots and links. Many concepts from this paper come from knot theory: while we strive to be as self-contained as possible, we refer to standard textbooks [5, 28] for an introduction to this topic and to Hatcher [18] for background on algebraic topology. Throughout this article, we work in the piecewise-linear (PL) category, which means that all the objects and functions that we consider are piecewise-linear with respect to a fixed polyhedral decomposition of the ambient space (generally $\mathbb{S}^3$). Two embeddings i_1 and i_2 in a topological space S are **(ambient) isotopic** if there exists a continuous family of homeomorphisms $h : S \times [0,1] \to S$ such that $h(i_1, 0) = i_2$ and $h(\cdot, 1)$ is the identity. A **knot**, resp. a **link**, is an embedding of the circle $\mathbb{S}^1$, resp. of a disjoint union of circles, into $\mathbb{S}^3$. In the following, we introduce the main definitions for knots for simplicity, but they apply identically to links. Two knots K_1 and K_2 are considered to be equivalent if they are isotopic. Since every knot misses at least one point of $\mathbb{S}^3$, via stereographic projection we can equivalently consider knots to be embedded in $\mathbb{R}^3$, and we freely switch between these two perspectives. The **unknot** is, up to equivalence, the unique embedding of S^1 in $\mathbb{R}^3$ with image a triangle. A **torus knot** $T_{p,q}$ is a knot embedded on a surface of an unknotted torus $\mathbb{T}$ in $\mathbb{S}^3$, for example a standard torus of revolution. It winds p times around the revolution axis, and q times around the core of the torus. We refer to Figure 3 for an illustration of two torus knots and an unknot.

A **knot diagram** $\mathcal{D}$ is a planar four-regular graph, where each crossing is decorated to indicate which strands are above and below. From such a diagram, one can easily obtain the data of a knot $K \hookrightarrow \mathbb{R}^3$ and a linear **projection** map $p : \mathbb{R}^3 \to P \simeq \mathbb{R}^2$ so that $p(K) = \mathcal{D}$ (respecting the decorations), where P is a plane of $\mathbb{R}^3$ called the **projection plane**. The **crossing number** of a knot is the minimal number of crossings among all of its diagrams. The Reidemeister theorem [32] shows that two diagrams represent equivalent knots if and only if they can be connected by a sequence of planar isotopies and local moves called **Reidemeister moves** and pictured in Figure 1.

Finally, recall that a **homotopy** between two simple closed curves of a surface Σ, $\gamma_0 : \mathbb{S}^1 \hookrightarrow \Sigma$ and $\gamma_1 : \mathbb{S}^1 \hookrightarrow \Sigma$ is a continuous map $\gamma : \mathbb{S}^1 \times [0,1] \to \Sigma$ such that $\gamma(\mathbb{S}^1, 0) = \gamma_0$ and $\gamma(\mathbb{S}^1, 1) = \gamma_1$. In particular, closed curves are allowed to self-intersect in a homotopy, while this is disallowed in an isotopy. Throughout this paper, to distinguish between links and their projections, we use calligraphic letters to designate diagrams, and capital letters to designate links so that a diagram of the link M is denoted by $\mathcal{M}$.

▶ Remark 4. We work with the standard setting of knot diagrams in $\mathbb{R}^2$, but it is also possible to work with diagrams in $\mathbb{S}^2$, which therefore allow for more Reidemeister moves (this is the perspective taken in [6]). Our results also hold in that setting, since a knot diagram in $\mathbb{S}^2$ lifts to a knot $K \hookrightarrow \mathbb{S}^2 \times [-\varepsilon, \varepsilon] \subseteq \mathbb{R}^3$, with the natural projection map $p : \mathbb{S}^2 \times [-\varepsilon, \varepsilon], (s,t) \mapsto (s, 0)$. The definitions of the spheres obtained from the diagrams in Section 4 can be directly adapted to this setting, and the rest of the proof is identical.

Our link diagrams. Throughout this article, we write $L(p,q)$ for the split link consisting of two linked torus knots $T_{p,q}$, denoted by $M(p,q)$ and shown in Figure 3, and an unlinked unknot component U. We consider two diagrams of $L(p,q)$. The first, denoted by $\mathcal{D}(p,q)$, is shown in Figure 3, and the second diagram $\mathcal{D}'(p,q)$ is any split link diagram of $L(p,q)$.

Let R be a sequence of Reidemeister moves turning $\mathcal{D}(p,q)$ into $\mathcal{D}'(p,q)$, such that $\mathrm{Top}(\mathcal{D}(p,q), R) \leq k$, $k \geq 0$. Our goal is to prove that k cannot be smaller than a function depending only on p and q. Since the values of p and q are mostly fixed and have little influence on our arguments, we mostly omit the parameters (p,q) from L, M, $\mathcal{D}$, and $\mathcal{D}'$.

The following lemma directly follows from known results on torus knots.

▶ **Lemma 5.** *Let $n \geq 2$, and let $\mathcal{D}'$ be a link diagram equivalent to $\mathcal{D}(n, n+1)$ by Reidemeister moves. Then $\mathrm{cr}(\mathcal{D}') \geq 2n^2$, that is, $\mathcal{D}'$ has at least $2n^2$ crossings.*

Proof. Let $\mathcal{D}'$ be a diagram of $L = L(n, n+1)$. First note that, since $n \geq 2$, it follows from a theorem of Murasugi [27, Proposition 7.5] that each of the torus knot components $T_{n,n+1}$ of L has at least $(n+1) \times (n-1) = n^2 - 1$ crossings. Since the two torus knots are linked, they share at least 2 crossings in each diagram of L, and it follows that $\mathrm{cr}(\mathcal{D}) \geq 2n^2$. ◀

From a sequence of Reidemeister moves to continuous operations. As detailed above, we work with a sequence of Reidemeister moves R turning the link diagram $\mathcal{D}$ of L into the split diagram $\mathcal{D}'$. From this sequence of Reidemeister moves, we can obtain an ambient isotopy $\Phi_R : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ and a projection $p : \mathbb{R}^3 \to \mathbb{R}^2$ so that the diagrams $p(\Phi_R(L,t))$ follow the evolution of $\mathcal{D}$ under the moves R, and in particular the combinatorial types of the diagrams $p(\Phi_R(L,t))$ only change for a finite number of values of t, one for each Reidemeister move. The projection p is regular except at the **critical times** of R, which are times where the projection $p \circ \Phi_R(L,t)$ displays a tangency or a triple point, see Figure 4 for an illustration. For any non-critical time t, we write $L_t = \Phi_R(L,t)$ and we denote the diagram defined by $p(L_t) = p(\Phi_R(L,t))$ by $\mathcal{D}_t$. Naturally, we have $\mathcal{D}_0 = \mathcal{D}$ and $\mathcal{D}_1 = \mathcal{D}'$.

Figure 4 Critical times corresponding to Reidemeister moves RII (left) and RIII (right).

Note that the definition of $\mathrm{Top}(\mathcal{D}, R)$ naturally coincides with $\sup_{t\in[0,1]}\{\mathrm{cr}(\mathcal{D}_t) - \mathrm{cr}(\mathcal{D}_0)\}$. Indeed, the diagram $\mathcal{D}_t$ at critical times has fewer intersections than one of $\mathcal{D}_{t+\epsilon}$ or $\mathcal{D}_{t-\epsilon}$ for ϵ small enough. Furthermore, $\mathrm{cr}(\mathcal{D}_t)$ is constant for all t between two critical times.

In Section 3 we consider the movements of U and of M under the Reidemeister moves separately. We use $\mathcal{M}_t = p(\Phi_R(M,t))$ as a shorthand for the diagram of the sublink $M \subset L$ at time t. For $\mathcal{U}$ we consider the homotopy $\phi_U : \mathbb{S}^1 \times [0,1] \to \mathbb{R}^2$ in the plane induced by the projection $p(\Phi_R(U,t))$. We denote the corresponding curves by $\mathcal{U}_t = \phi_U(\mathbb{S}^1, t)$ and emphasise that we consider these as immersions of closed curves in the plane. That is, we forget which strand is over which at each self-crossing of $\mathcal{U}_t$. See Figure 5 for a summary of this setup.

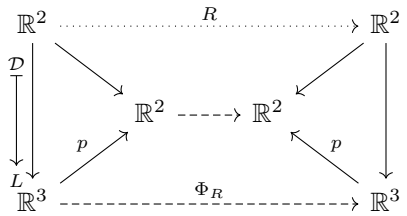


Figure 5 Definition of our homotopies from the sequence of Reidemeister moves R.

3 From homotopies to isotopies

We work with the definitions of Section 2. We start with a sequence of Reidemeister moves R turning $\mathcal{D}$ into $\mathcal{D}'$, and consider the induced homotopy ϕ_U of the link component $\mathcal{U}$ in the plane of projection. The goal of this section is to use results from [7] in order to locally alter the images $\mathcal{U}_t = \phi_U(\mathbb{S}^1, t)$ into simple closed curves, and hence to obtain an isotopy taking $\mathcal{U}_0$ to $\mathcal{U}_1$. After these modifications, the altered simple closed curves representing U still sweep over the remainder $\mathcal{M}_t$ of the diagrams $\mathcal{D}_t$, but this sweep no longer necessarily corresponds to a sequence of Reidemeister moves on the original link L.

Nevertheless, in the next section we use the isotopy from $\mathcal{U}_0$ to $\mathcal{U}_1$ to define a family of 2-spheres in $\mathbb{S}^3$ sweeping through M. This setup then allows us to use bubble tangles to obstruct small values of $\mathrm{Top}(\mathcal{D}, R)$ in the initial sequence of Reidemeister moves R.

The key points of this process are given by the following statement.

▶ **Proposition 6.** *Let R be a sequence of Reidemeister moves turning $\mathcal{D}$ into the split diagram $\mathcal{D}'$ such that for all $t \in [0,1]$, $\mathrm{cr}(\mathcal{D}_t) \leq m$ for some integer $m \geq 0$. Then there exists an ambient isotopy $\Phi' : \mathbb{S}^3 \times [0,1] \to \mathbb{S}^3$ and an isotopy $h : \mathbb{S}^1 \times [0,1] \to \mathbb{R}^2$ such that*

1. *$h(\mathbb{S}^1, 0) = \mathcal{U}_0$ and $h(\mathbb{S}^1, 1) = \mathcal{U}_1$; and*
2. *for all $t \in [0,1]$, the total number of crossings in the overlay of $p(\Phi'(M,t))$ and $h(\mathbb{S}^1, t)$ in $\mathbb{R}^2$ is at most m.*

We emphasise that in the second item of Proposition 6, we only consider the sublink M in $\Phi'(M,t)$, and not the entire link. That is, the proposition provides an ambient isotopy for M in $\mathbb{R}^3$ and an isotopy for $p(U)$ in $\mathbb{R}^2$, while preserving a bound on the total number of intersections of $p(\Phi'(M,t))$ and $h(\mathbb{S}^1, t)$ in the plane of projection. This proposition follows from the techniques of Chambers and Liokumovich developed in Chapter 2 of [7], the proof is detailed in the full version of the article for completeness. We first state one of their main results.

▶ **Definition 7** (Chambers, Liokumovich [7, Definition 1.3])**.** *Two curves α and β on a Riemannian 2-manifold M are **ϵ-image equivalent**, $\alpha \sim_\epsilon \beta$, if (i) there exists a finite collection of disjoint intervals $\bigsqcup_{i=1}^n I_i \subset \mathbb{S}^1$ such that $\|\alpha(\mathbb{S}^1 \setminus \bigsqcup I_i)\| + \|\beta(\mathbb{S}^1 \setminus \bigsqcup I_i)\| < \epsilon$, and (ii) there exists a permutation σ of $\{1, \ldots, n\}$ and a map $f : \{1, \ldots, n\} \to \{0, 1\}$, such that $\alpha|_{I_i} = (-1)^{f(i)} \beta|_{I_{\sigma(i)}}$ for all i. Here, $\|\alpha\|$ denotes the length of the curve α.*

The main result we use for the proof of Proposition 6 is given by the following statement.

▶ **Theorem 8** (Chambers, Liokumovich [7, Theorem 1.1'])**.** *Suppose that γ is a smooth homotopy of closed curves on a Riemannian 2-manifold M and that γ_0 is a simple closed curve. Then, for every $\epsilon > 0$, there exists an isotopy $\overline{\gamma}$ such that $\overline{\gamma_0} = \gamma_0$ and $\overline{\gamma_1}$ is ϵ-image equivalent to a small perturbation of γ_1. Additionally, for every t, there exists a t' such that $\overline{\gamma_t}$ is ϵ-image equivalent to a small perturbation of $\gamma_{t'}$. If γ_1 is simple, or is a point, then this homotopy also ends at γ_1, up to a change in orientation.*

Informally, we think of Definition 7 and Theorem 8 as follows. When a curve α self-intersects in the plane, each crossing point can be **resolved** in one of two ways by reconnecting the endpoints in a small ball around the crossing point, see Figure 6 on the right. When all the crossings have been resolved in some way and we obtain a simple closed curve α', we say that α' is a **resolution** of α. The theorem states that if we have a homotopy γ on a surface

between two simple curves γ_0 and γ_1, one can obtain an *isotopy* $\overline{\gamma}$ between γ_0 and γ_1[3] where each intermediate curve $\overline{\gamma_t}$ is a resolution of some intermediate curve $\gamma_{t'}$. Note that, here, the times t and t' may not coincide.

If M is endowed with a metric (for example, a Riemannian one), the lengths of γ_t and of $\overline{\gamma_{t'}}$ differ by an arbitrarily small quantity. Hence, Theorem 8 immediately implies that, for all $\varepsilon > 0$, if there exists a homotopy between two simple curves γ_0 and γ_1 where each intermediate curve has length at most ℓ, then there also exists an isotopy between γ_0 and γ_1 where each intermediate curve has length at most $\ell + \varepsilon$.

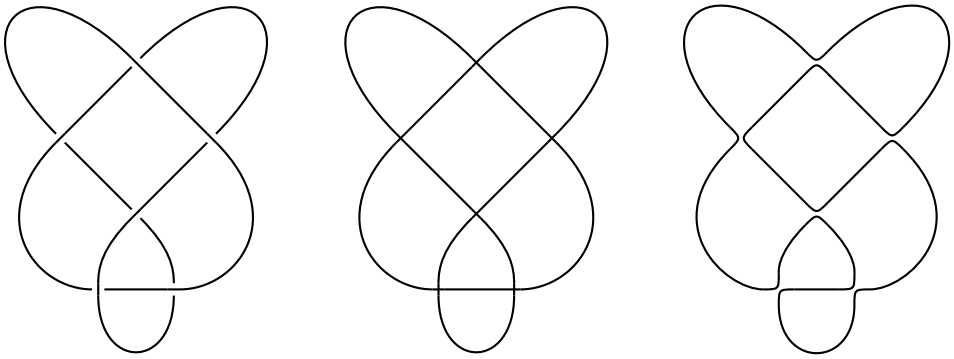


Figure 6 Left: A diagram of the unknot U. Middle: A projection $\mathcal{U}_t$ of this unknot, where the crossing information has been forgotten. Right: A resolution of $\mathcal{U}_t$.

Theorem 8 can be applied as a black-box to prove Proposition 6 in the particular case where the sublink M stays invariant throughout the sequence of Reidemeister moves R, i.e., $\Phi_R(M,t) = \Phi_R(M,0)$ for all $t \in [0,1]$. Indeed, in this case, we can think of $\mathcal{M}_t$ as a discrete metric measuring the length of a curve $\mathcal{U}_t$ by its number of intersections with $\mathcal{M}_t$. More formally, we can take γ to be the homotopy ϕ_U between $\mathcal{U}_0$ and $\mathcal{U}_1$, which are both simple closed curves by the definition of the link diagram $\mathcal{D}$. Applying Theorem 8 provides us with an isotopy h between $\mathcal{U}_0$ and $\mathcal{U}_1$ where all intermediate curves $\mathcal{U}'_t$ are obtained from resolving intersections of some $\mathcal{U}_{t'}$. In particular, for all $t \in [0,1]$, the number of intersections between $\mathcal{U}'_t$ and $\mathcal{M}_t$ is at most the number of intersections between $\mathcal{U}_{t'}$ and $\mathcal{M}_t$, which is at most k since $\mathcal{M}_t = \mathcal{M}_0$ does not depend on t.

A careful reading of [7] shows that the general case of Proposition 6, where the diagram $\mathcal{M}_t$ of the sublink M evolves during the sequence of Reidemeister moves, can also be obtained using the same proof techniques: The basic idea of the proof of Theorem 8 is to first decompose the homotopy of γ into a sequence of local moves, and then replace each curve γ_t by one of its resolutions.

For the first step, they track discrete times where the self-intersection pattern (i.e., the homeomorphism type) of γ_t changes. By an argument similar to the proof of the Reidemeister theorem, one can assume that this only happens at **critical events**, when a curve γ_t undergoes a homotopy move, which is a transformation analogue to a Reidemeister move but without any crossing information. This step is formalised by their Proposition 2.1 and Lemma 2.2. Note that this step is unnecessary for us, since by construction the local transformations undergone by ϕ_U are projections of Reidemeister moves.

[3] Or γ_1 with its orientation reversed. Since it is safe to disregard orientations for our purpose, we always consider curves up to orientation reversal.

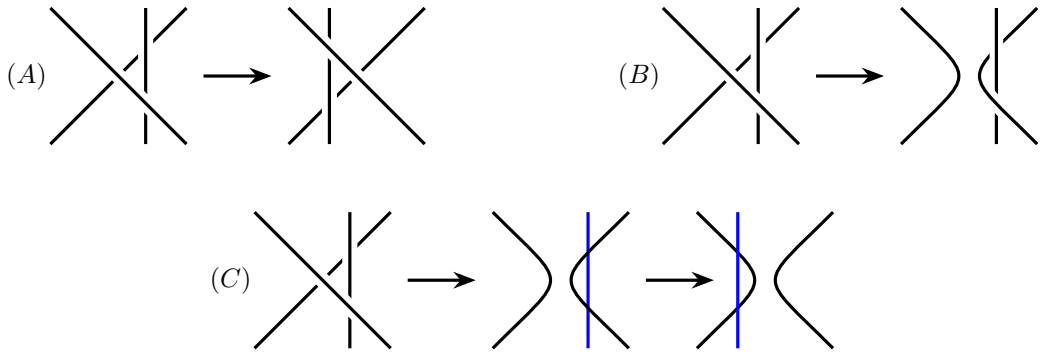


Figure 7 (A) Reidemeister III move. (B) a problematic resolution of $\mathcal{U}_t$. (C) The unknot $\mathcal{U}_t$ (black) seen as a curve sweeping through $\mathcal{M}$ (blue).

Between these critical events, any homotopy of a curve γ_t can be applied similarly to any of its resolutions, yielding an isotopy. Hence, the idea is to replace each of these Reidemeister moves by a resolution of the move, as explained by their Figure 2. However, doing so in a naive way runs into discontinuity issues, a basic example of which is detailed in [7, Example 2], which is associated to their Figures 3 and 4. Therefore, the authors provide a more intricate workaround: the key to the proof of Theorem 8 is to show how to choose the correct resolutions and connect their isotopies together. This is achieved by defining an auxiliary graph of resolutions (see their Figure 7), synthesising how they are connected by local isotopies. The precise definition for this graph follows their Figure 8. The proof is then finalised by finding a path through this graph by using the handshaking lemma [13].

In our case, the critical events are exactly the times t when $\mathcal{U}_t$ undergoes a Reidemeister move. In-between these critical events, there are other Reidemeister moves involving either (i) both $\mathcal{U}_t$ and $\mathcal{M}_t$, or (ii) only $\mathcal{M}_t$. Note that in case (i), the Reidemeister move only changes the relative position of $\mathcal{U}_t$ and $\mathcal{M}_t$, and hence such a move can be treated as leaving the isotopy type of $\mathcal{M}_t$ unchanged. Therefore the homotopy of $\mathcal{U}_t$ can be used as an isotopy of any of its resolutions in this case. In case (ii), $\mathcal{M}_t$ changes, but $\mathcal{U}_t$, considered up to isotopy, does not. Therefore, by applying the same Reidemeister moves to $\mathcal{M}_t$, any motion of $\mathcal{U}_t$ between critical events can be applied to any of its resolutions while preserving the number of intersections with $\mathcal{M}_t$. These motions can then be connected using the same handshaking argument as in the proof of Theorem 8. Summarising, the proof technique directly adapts to the case of an evolving metric, as long as these evolutions are applied appropriately throughout the new sequence of isotopies.

▶ Remark 9. We emphasise that in this section, we apply the framework of Theorem 8 to the *projection* of U without crossing information, which is thus considered as a closed, non-embedded, curve in the projection plane $\mathbb{R}^2$. Instead, it might be tempting to apply these techniques *directly at the level of Reidemeister moves*. This way one may hope to prove that in any sequence of Reidemeister moves splitting $\mathcal{U}$ from $\mathcal{M}$, one can assume that $\mathcal{U}$ remains simple while preserving a bound on the number of added crossings. However, this proof strategy fails because some resolutions applied to $\mathcal{U}$ can block the application of RIII moves involving $\mathcal{U}$ and $\mathcal{M}$. Such a case is pictured in Figure 7(B).

4 Sweep-outs, bubble tangles and proof of the lower bound

For the remainder of this article, we use Proposition 6 to assume that the homotopy $\phi_U : \mathbb{S}^1 \times [0,1] \to \mathbb{R}^2$ sweeping U across M is an isotopy. In other words, let h be the isotopy of $\mathcal{U}$ in $\mathbb{R}^2$ and Φ' be the ambient isotopy of M in $\mathbb{S}^3$ from Proposition 6, and for all $t \in [0,1]$, we replace $\phi_U(\mathbb{S}^1, t)$ by $h(\mathbb{S}^1, t) = \mathcal{U}_t$ which is a simple closed curve.

4.1 A sweep-out of 2-spheres

For each t, we associate a 2-sphere S_t to the simple curve $\mathcal{U}_t$, by extending $\mathcal{U}_t$ infinitely towards the direction of the projection p. Seen in $\mathbb{S}^3$ this produces a torus pinched at ∞, and cutting the surface at ∞ yields S_t (see Figure 8). By construction, S_t intersects $M_t = \Phi'(M, t)$ only in the pre-images of $\mathcal{U}_t \cap \mathcal{M}_t$ by the projection p. Altogether, this produces a continuous family of spheres S_t sweeping a continuous family of links M_t. By applying the ambient isotopy Φ'^{-1} to S_t and M_t, we can assume that M_t is fixed. We slightly abuse notation and make this assumption, while still denoting the family of spheres by S_t.

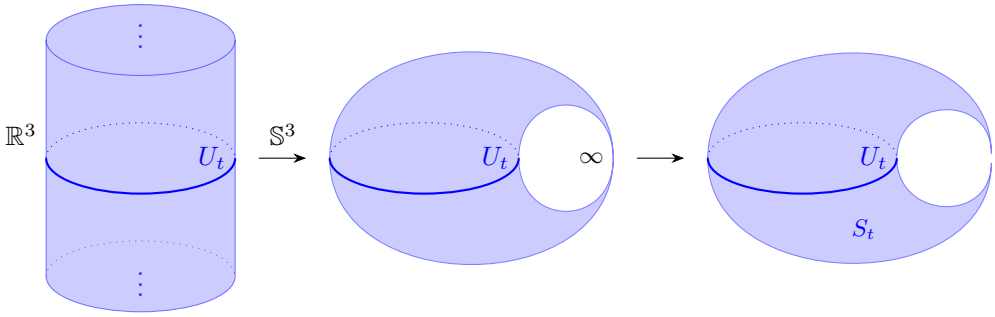


Figure 8 Gluing two infinite annuli on U and cutting the resulting pinched torus at ∞ in $\mathbb{S}^3$.

We now relate $\mathrm{Top}(\mathcal{D}(n, n+1), R)$ to our continuous family of spheres $\{S_t\}_{t\in[0,1]}$.

▶ **Lemma 10.** *We have* $\sup_{t\in[0,1]} |S_t \cap M_t| - 2 \leq \mathrm{Top}(\mathcal{D}(n, n+1), R)$.

Proof. For $\mathcal{M}$ and $\mathcal{M}'$ two components of a link diagram, we denote by $\mathrm{cr}(\mathcal{M}, \mathcal{M}')$ the number of crossings involving strands of $\mathcal{M}$ and $\mathcal{M}'$. If $\mathcal{M} = \mathcal{M}'$, then $\mathrm{cr}(\mathcal{M}, \mathcal{M}')$ is the number of self-crossings of $\mathcal{M}$.

Since $\mathcal{U}_t$ is simple for all times t, we have $\mathrm{cr}(\mathcal{D}_t) = \mathrm{cr}(\mathcal{M}_t, \mathcal{U}_t) + \mathrm{cr}(\mathcal{M}_t, \mathcal{M}_t)$. By definition, $\mathcal{D}_t$ is the diagram obtained from $\mathcal{D}(n, n+1)$ at time t of the sequence of Reidemeister moves R, and is hence equivalent to $\mathcal{D}(n, n+1)$. It follows from Lemma 5, that $\mathrm{cr}(\mathcal{M}_t, \mathcal{M}_t) \geq 2n^2$ at all times t. Furthermore, $\mathrm{cr}(\mathcal{M}_t, \mathcal{U}_t) = |S_t \cap M_t|$ by construction of S_t. By definition of $\mathcal{D}(n, n+1) = \mathcal{D}_0$, we have $\mathrm{cr}(\mathcal{D}_0) = 2n^2 + 2$. Hence, $\mathrm{cr}(\mathcal{D}_t) - \mathrm{cr}(\mathcal{D}_0) \geq 2n^2 + |M_t \cap S_t| - 2n^2 - 2$.

Thus, $\sup_{t\in[0,1]} |S_t \cap M_t| - 2 \leq \sup_{t\in[0,1]} \{\mathrm{cr}(\mathcal{D}_t) - \mathrm{cr}(\mathcal{D}_0)\} = \mathrm{Top}(\mathcal{D}(n, n+1), R)$. ◀

4.2 Bubble tangles

The aim for the remainder of this section is to use results from [26] to exhibit an obstruction to $|S_t \cap M_t|$ remaining small throughout the sweep-out defined in Section 4.1.

The key objects that we rely on are bubble tangles. Before introducing those, we need the following definitions. A **double bubble** consists of two spheres intersecting on a disc (see the left of Figure 9 for an illustration). The complement of a double bubble consists of

three balls, and we say that they are **induced** by the double bubble. A 3-ball in $\mathbb{S}^3$ is said to be ***L*-trivial**, if it intersects $L \subset \mathbb{S}^3$ in a single unknotted segment or in the empty set, see Figure 9 on the middle for an example and on the right for a non-example.

▶ **Definition 11** ([26], Definition 2.3). *Let $L \in \mathbb{S}^3$ be a link and let $k \in \mathbb{N}$. A **bubble tangle $\mathcal{T}$ of order $k \geq 2$**, is a collection of closed balls in $\mathbb{S}^3$ such that:*

1. *For all balls $B \in \mathcal{T}$, we have $|\partial B \cap L| < k$.*
2. *For all 2-spheres $S \in \mathbb{S}^3$ transverse to L, if $|S \cap L| < k$, then exactly one of the two[4] balls B_i, $i \in \{1, 2\}$ with boundary S is in $\mathcal{T}$.*
3. *For all triples of balls $\{B_1, B_2, B_3\}$, if $\{B_1, B_2, B_3\}$ induces a double bubble transverse to L, then $\{B_1, B_2, B_3\} \not\subset \mathcal{T}$.*
4. *For every closed ball B in $\mathbb{S}^3$, if B is L-trivial and $|\partial B \cap L| < k$, then $B \in \mathcal{T}$.*

Informally, we think of a bubble tangle as a way to choose, for each sphere S having less than k intersections with L, one of the two balls that it bounds. We refer to this ball as the **small side** of S, with the intuition that this small side intersects L in a simpler pattern than the other side. The essential property of a bubble tangle is the requirement that we cannot cover all of $\mathbb{S}^3$ using three small sides of a triple of spheres arranged in a double bubble. One of the main results of [26] states that if a bubble tangle of L of order k exists, any sweep-outs of $\mathbb{S}^3$ by 2-spheres contains at least one sphere with at least k intersections with L. As discussed in the introduction, our sweep-outs differ from those in [26]. However, we show that despite their differences, a bubble tangle can also be used to preclude the existence of a sweep-out S_t having few intersections with M.

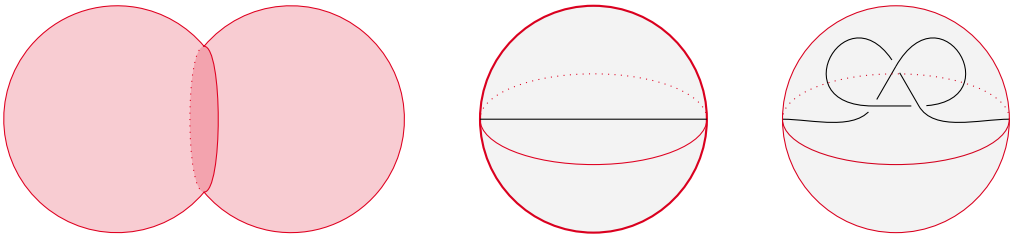


Figure 9 Left: a double bubble. Middle: an L-trivial ball. Right: a ball that is not L-trivial.

We denote by M^1 and M^2 the two torus links composing M. By definition, each of them can be embedded on a standard torus $\mathbb{T}$ as pictured in Figure 11. A simple curve on $\Sigma \subseteq \mathbb{S}^3$ is **compressible** if it is non-contractible on Σ and bounds a disk in $\mathbb{S}^3 \setminus \Sigma$. The **compression-representativity** of a knot K embedded on a surface Σ is the minimum number of intersections between K and a compressible curve in Σ. The compression-representativity of M_1 and M_2 is n, see [26, Section 5]. Note that, since M^1 and M^2 are linked, their underlying tori intersect, but this is of no consequence for us. By [26, Theorem 1.2], there exist two bubble tangles $\mathcal{T}^1, \mathcal{T}^2$ of order $\frac{2}{3}n$, where each $\mathcal{T}^i$ is the bubble tangle defined by M^i while completely forgetting about the other torus knot. Thus, $\mathcal{T}^i$ is a bubble tangle *induced* by the torus on which M^i is embedded. Such bubble tangles induced by a surface are called **compression bubble tangles**: given a link embedded on a surface Σ, the small sides of the bubble tangle are the balls B which are either disjoint from Σ or intersect Σ trivially, i.e., the inclusion $i : B \cap \Sigma \to \Sigma$ is π_1-trivial, which means that it

[4] By the PL Schoenflies theorem [4, Theorem XIV.1], a PL 2-sphere in $\mathbb{S}^3$ bounds exactly two balls.

induces a trivial map on fundamental groups. For example, the annulus and disc obtained by intersecting the torus and green ball of Figure 10 have a π_1-trivial inclusion in the torus, but the annulus stemming from the intersection between the red ball and the torus does not.

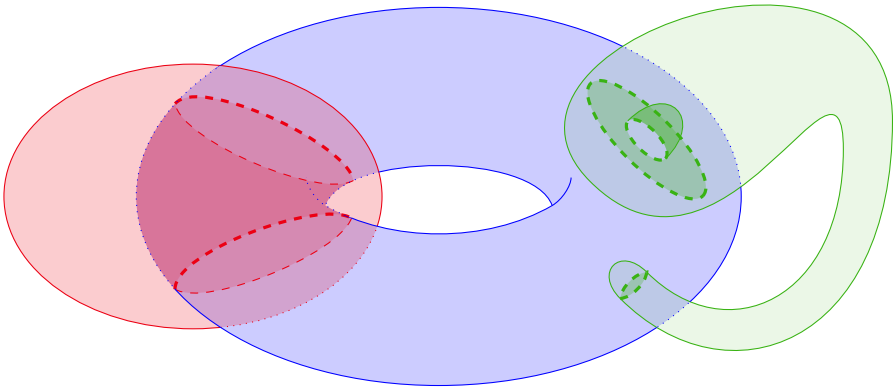


Figure 10 The green ball intersects the torus trivially while the red ball does not.

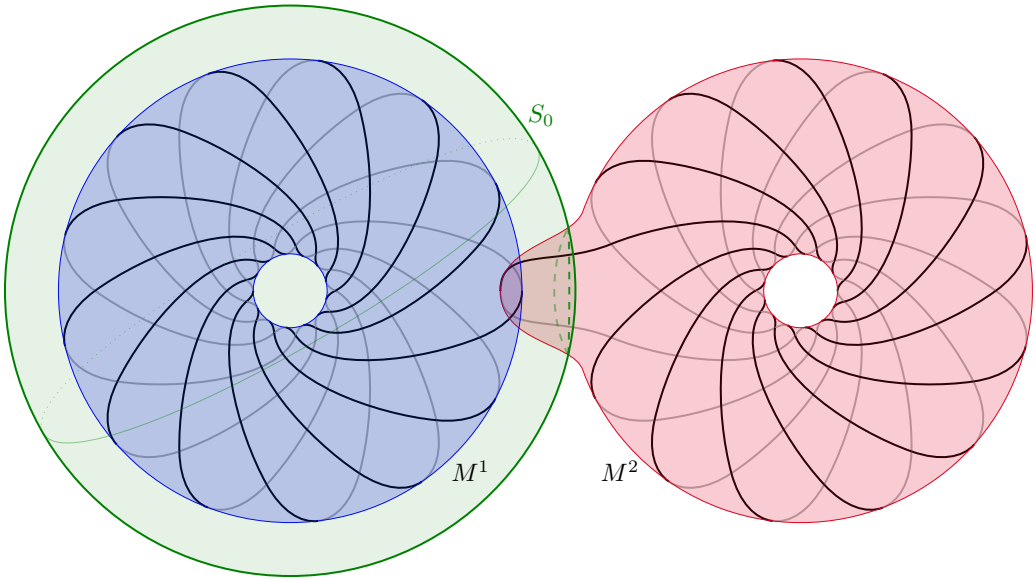


Figure 11 Link components M^1 and M^2 embedded on tori, and sphere S_0.

4.3 Proof of Theorem 2

In order to prove Theorem 2, we assume that $\mathrm{Top}(\mathcal{D}(n, n+1), R) < k = \frac{2}{3}n - 2$. By virtue of Lemma 10, it follows that $|M_t^i \cap S_t| \leq |M_t \cap S_t| < \frac{2}{3}n$ for $i \in \{1, 2\}$. Now, changes to $|M_t \cap S_t|$ can only happen at critical times where the number of crossings involving $\mathcal{U}_t$ in $\mathcal{D}_t$ increases or decreases. In particular, we can ignore RIII moves between $\mathcal{U}_t$ and $\mathcal{M}_t$. Since we only consider moves involving both $\mathcal{U}_t$ and $\mathcal{M}_t$, only RII moves are relevant for our study. We denote the critical times of RII moves involving both $\mathcal{U}_t$ and $\mathcal{M}_t$ by $(t_j)_{1 \leq j \leq s} \subset [0, 1]$. These times are the only times where S_t is not transverse to M_t, but **finitely tangent**, meaning that M_t is tangent to S_t, but their number of intersections is still finite.

Recall that, up to applying ϕ'^{-1}, we assume that for all $t \in [0,1]$ M_t is fixed; we denote it by M. By 2, for all times $t \in [0,1] \setminus \{t_j\}_{1\leq j\leq s}$, S_t has a small side corresponding to a 3-ball $B_t^i \in \mathcal{T}^i$. We study how these small sides evolve throughout the sweep-out. Special attention must be paid to tangencies at times $\{t_j\}_{1\leq j\leq s}$, where small sides are not defined. For convenience, we denote by $\Theta : \mathbb{S}^3 \times [0,1] \to \mathbb{S}^3$ an isotopy describing the evolution of S_t, i.e., such that $\Theta(S_0, t) = S_t$.

We start by defining the **agreement** between bubble tangles $\mathcal{T}^1$ and $\mathcal{T}^2$ as a map $a : [0,1] \setminus \{t_j\}_{1\leq j\leq s} \to \{0,1\}$, such that $a(t) = 1$ if $B_t^1 = B_t^2$ and 0 otherwise. With this definition, and under the assumption that $\mathrm{Top}(\mathcal{D}(n, n+1), R) < k$, we can infer that $a(0) = 0$ and $a(1) = 1$: First, note that $a(1) = 1$. Indeed, $\mathcal{U}_1$ is disjoint from $\mathcal{M}$ so that S_1 is disjoint from M. It follows that one side of S_1 is an empty ball with respect to both M^1 and M^2 and thus $a(1) = 1$ by 4. Second, the small side of S_0 in $\mathcal{T}^1$ is, by 4, the ball not containing M^1. For $\mathcal{T}^2$, as it is illustrated by Figure 11, S_0 contains an M^2-trivial ball and M^1 on its small side, and the remaining of M^2 on the other side. It follows that the small sides disagree at time 0, and hence $a(0) = 0$.

We want to show that a stays constant on its domain of definition. For this, we introduce the following definition: Two disjoint spheres $S, S' \subset \mathbb{S}^3$ are **braid-equivalent** with respect to some link $L \subset \mathbb{S}^3$, if L forms a braid in the product region between S and S', or, equivalently, if the product region between S and S' is homeomorphic to $S_\ell \times [0,1]$ where S_ℓ is the 2-sphere with ℓ holes, $\ell \geq 0$, see Figure 12 for an illustration.

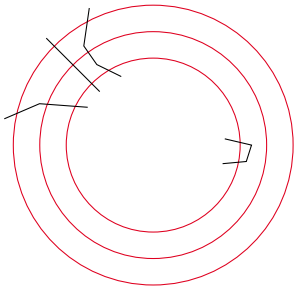


Figure 12 A cross-section view of three spheres and a link. The two outermost spheres are braid-equivalent, but the innermost one is not braid-equivalent to the others.

We have the following statement for two disjoint braid-equivalent spheres.

▶ **Lemma 12** ([26], Lemma 3.2)**.** *Let $\mathcal{T}$ be a bubble tangle and $S, S' \subset \mathbb{S}^3$ be two braid-equivalent spheres. Define $\mathbb{S}^3 \setminus S = \{B_1, B_2\}$ and $\mathbb{S}^3 \setminus S' = \{B_1', B_2'\}$ such that $B_1 \subset B_1'$. If $B_1 \in \mathcal{T}$ then $B_1' \in \mathcal{T}$.*

The compression bubble tangles furthermore have the following property.

▶ **Lemma 13.** *Let $\mathcal{T}$ be a compression bubble tangle of L, induced by an embedding of L into some surface $\Sigma \subset \mathbb{S}^3$, and let k be the order of $\mathcal{T}$. Then, for two closed 3-balls $A, B \subset \mathbb{S}^3$, if $A \in \mathcal{T}$, $B \subset A$, and $|\partial B \cap L| < k$, then $B \in \mathcal{T}$. That is, $\mathcal{T}$ is* stable by inclusion *up to 1.*

Proof. Let A and B be two closed balls of $\mathbb{S}^3$ such that $|\partial A \cap L|, |\partial B \cap L| < k$, and $A \in \mathcal{T}$. By definition, $A \cap \Sigma$ is π_1-trivial. Since $B \subset A$, we have $B \cap \Sigma \subset A \cap \Sigma$ so that the inclusion morphisms i_* satisfy $\pi_1(B \cap \Sigma) \to \pi_1(A \cap \Sigma) \to 0$. Hence, $B \cap \Sigma$ is π_1-trivial and $B \in \mathcal{T}$. ◀

To handle the back-and-forths of our sweep-out, we introduce a new equivalence relation on spheres. Two intersecting spheres $S, S' \subset \mathbb{S}^3$ are **intersection-equivalent** if there exists an isotopy between them which stays constant on their intersection $S \cap S'$, see Figure 13. Note that, by this definition, a sphere S is intersection-equivalent with itself.

The structure of the remainder of the proof is to show that the agreement a is constant on $[0,1]$ by showing that it is locally constant via Lemmas 15 and 16.

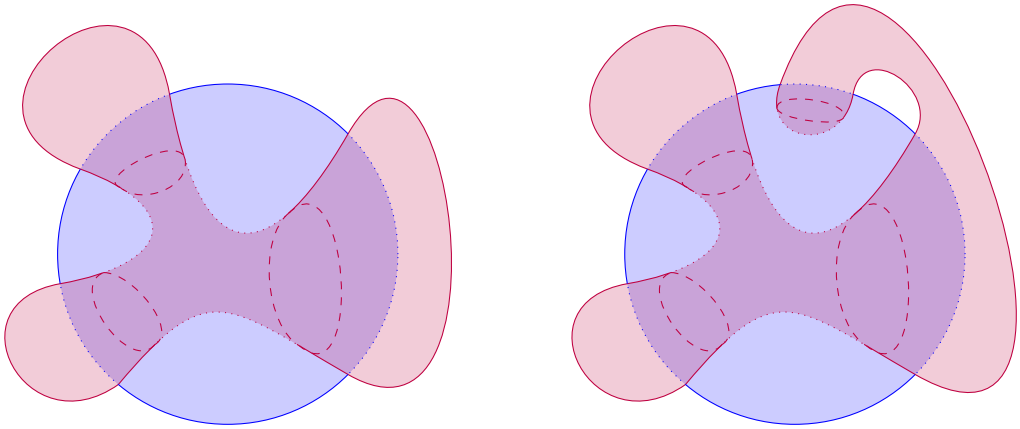


Figure 13 Left: two intersection-equivalent spheres. Right: two spheres that are not intersection-equivalent: the red sphere has an annulus component that cannot be mapped to a component of the blue one. Indeed, the components of the blue sphere are discs and a sphere with 4 punctures.

Since we work in the piecewise-linear setting, we have the following observation:

▶ **Lemma 14.** *Let $t \in [0,1]$. There exists a neighbourhood $V \subset [0,1]$ of t such that for all spheres S_v, $v \in V$, S_v and S_t are either disjoint or intersection-equivalent.*

The proofs of the two following lemmas, given in the full version of the article, are similar.

▶ **Lemma 15.** *Let $t \in [0,1] \setminus \{t_j\}_{1 \leq j \leq s}$ be a non-critical time. There exists a neighbourhood $V \subset [0,1] \setminus \{t_j\}_{1 \leq j \leq s}$ of t such that a is constant on V.*

Sketch of proof. The idea is to keep track of the small sides of two close spheres for each $\mathcal{T}^i$. Either the two spheres are disjoint and the result follows from Lemma 12 and Lemma 13, or they are intersection-equivalent and the proof proceeds similarly by applying these two lemmas in succession on the parts inside and outside of a small side.

▶ **Lemma 16.** *Let $t \in [0,1]$. Then there exists a neighbourhood $V \subset [0,1]$ of t such that a is constant on $V \setminus \{t\}$.*

▶ **Proposition 17.** *The agreement a is constant on $[0,1] \setminus \{t_j\}_{1 \leq j \leq s}$ and hence $a(0) = a(1)$.*

Proof. We can cover $[0,1]$ by open discs from Lemma 16 on which a is constant. Since $[0,1]$ is compact, only finitely many of them are enough to cover it. On each connected component of $[0,1] \setminus \{t_j\}_{1 \leq j \leq s}$ the agreement function is constant by the continuity of a (a is locally constant). Moreover, we know from Lemma 16 that for $u < t_j < v$ close enough we have $a(u) = a(v)$. Hence, a is constant on $[0,1] \setminus \{t_j\}_{1 \leq j \leq s}$, and $a(0) = a(1)$. ◀

Proof of Theorem 2. The initial discussion of this subsection states that $a(0) = 0$ and $a(1) = 1$. This contradicts Proposition 17. Thus, our assumption that $\mathrm{Top}(\mathcal{D}(n, n+1), R) < \frac{2}{3}n - 2$ does not hold. Therefore, during the sequence, at least one diagram has at least $2n^2 + \frac{2}{3}n$ crossings. ◀

In the full version of the article, we discuss variants of our construction for which our arguments also provide lower bounds on the crossing-complexity.

References

1 Colin C. Adams. *The knot book*. American Mathematical Society, 1994.

2 Taylor Applebaum, Sam Blackwell, Alex Davies, Thomas Edlich, András Juhász, Marc Lackenby, Nenad Tomašev, and Daniel Zheng. The unknotting number, hard unknot diagrams, and reinforcement learning. *arXiv preprint*, 2024. doi:10.48550/arXiv.2409.09032.

3 Dror Bar-Natan. On Khovanov's categorification of the Jones polynomial. *Algebraic & Geometric Topology*, 2(1):337–370, 2002.

4 R.H. Bing. *The geometric topology of 3-manifolds*, volume 40. American Mathematical Society, 1983.

5 Gerhard Burde and Heiner Zieschang. *Knots*. Walter de gruyter, 2002.

6 Benjamin A. Burton, Hsien-Chih Chang, Maarten Löffler, Clément Maria, Arnaud de Mesmay, Saul Schleimer, Eric Sedgwick, and Jonathan Spreer. Hard diagrams of the unknot. *Experimental Mathematics*, 0(0):1–19, 2023.

7 Gregory Chambers and Yevgeny Liokumovich. Converting homotopies to isotopies and dividing homotopies in half in an effective way. *Geometric and Functional Analysis*, 24, November 2013.

8 Arnaud de Mesmay, Jessica Purcell, Saul Schleimer, and Eric Sedgwick. On the tree-width of knot diagrams. *Journal of Computational Geometry*, 10(1):164–180, 2019. doi:10.20382/JOCG.V10I1A6.

9 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. The unbearable hardness of unknotting. In *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.SOCG.2019.49.

10 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. The unbearable hardness of unknotting. *Advances in Mathematics*, 381:107648, 2021.

11 Éric Colin De Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010. doi:10.1137/090761653.

12 I. A. Dynnikov. Arc-presentations of links: Monotonic simplification. *Fundamenta Mathematicae*, 190(1):29–76, 2006.

13 Leonhard Euler. Solutio problematis ad geometriam situs pertinentis (in latin). *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

14 David Gabai. Foliations and the topology of 3-manifolds. iii. *Journal of Differential Geometry*, 26(3):479–536, 1987.

15 Wolfgang Haken. Theorie der Normalflächen. *Acta Mathematica*, 105(3):245–375, 1961.

16 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM (JACM)*, 46(2):185–211, 1999. doi:10.1145/301970.301971.

17 Joel Hass and Tahl Nowik. Unknot diagrams requiring a quadratic number of Reidemeister moves to untangle. *Discrete & Computational Geometry*, 44:91–95, 2010. doi:10.1007/S00454-009-9156-4.

18 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge ; New York, 2002.

19 Chuichiro Hayashi. The number of Reidemeister moves for splitting a link. *Mathematische Annalen*, 332:239–252, 2005.

20 Louis H. Kauffman. The Jones polynomial, knots, diagrams, and categories. *Bulletin of the American Mathematical Society*, 2023.

21 Louis H. Kauffman and Sofia Lambropoulou. Hard unknots and collapsing tangles. *Introductory Lectures on Knot Theory*, pages 187–247, 2011.

22 Dale Koenig and Anastasiia Tsvietkova. Unlinking, splitting, and some other NP-hard problems in knot theory. *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1496–1507, 2021. doi:10.1137/1.9781611976465.90.

23 Marc Lackenby. A polynomial upper bound on Reidemeister moves. *Annals of Mathematics*, pages 491–564, 2015.

24 Marc Lackenby. The efficient certification of knottedness and Thurston norm. *Advances in Mathematics*, 387:107796, 2021.

25 Marc Lackenby. Links with splitting number one. *Geometriae Dedicata*, 214(1):319–351, 2021.

26 Corentin Lunel and Arnaud de Mesmay. A Structural Approach to Tree Decompositions of Knots and Spatial Graphs. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.50.

27 Kunio Murasugi. On the braid index of alternating links. *Transactions of the American Mathematical Society*, 326(1):237–260, 1991.

28 Dale Rolfsen. *Knots and links*. AMS Chelsea Pub, Providence, R.I, 2003. OCLC: ocm52901393.

29 Martin Scharlemann. Thin position in the theory of classical knots. In *Handbook of knot theory*, pages 429–459. Elsevier, 2005.

30 Horst Schubert. Über eine numerische Knoteninvariante. *Mathematische Zeitschrift*, 61(1):245–288, 1954.

31 Jennifer Schultens. Bridge numbers of torus knots. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 143(3), pages 621–625. Cambridge University Press, 2007.

32 Kurt van Reidemeister. Elementare Begründung der Knotentheorie (in german). *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 5:24–32, 1927.

Persistent (Co)Homology in Matrix Multiplication Time

Dmitriy Morozov
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Primoz Skraba
Queen Mary University of London, UK

Abstract

Most algorithms for computing persistent homology do so by tracking cycles that represent homology classes. There are many choices of such cycles, and specific choices have found different uses in applications. Although it is known that persistence diagrams can be computed in matrix multiplication time for the more general case of zigzag persistent homology [21], it is not clear how to extract cycle representatives, especially if specific representatives are desired. In this paper, we provide the same matrix multiplication bound for computing representatives for the two choices common in applications in the case of ordinary persistent (co)homology. We first provide a fast version of the reduction algorithm, which is simpler than the algorithm in [21], but returns a different set of representatives than the standard algorithm [15]. We then give a fast version of a variant called the row algorithm [10], which returns the same representatives as the standard algorithm.

2012 ACM Subject Classification Computing methodologies → Algebraic algorithms; Mathematics of computing → Algebraic topology; Computing methodologies → Linear algebra algorithms; Theory of computation → Computational geometry

Keywords and phrases persistent homology, matrix multiplication, cycle representatives

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.68

Related Version *Full Version*: `https://arxiv.org/abs/2412.02591` [23]

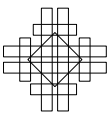
Funding *Dmitriy Morozov*: supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. *Primoz Skraba*: supported by the EPSRC AI Hub on Mathematical Foundations of Intelligence: An "Erlangen Programme" for AI No. EP/Y028872/1.

1 Introduction

Persistent homology [15] is one of the core techniques in topological data analysis. Because of its theoretical importance as well as its wide use in applications, a lot of work has been dedicated to its efficient computation both in theory [21, 10, 7, 4, 20] and in practice [2, 3, 1, 18, 17, 26]. On the theoretical side, the connection between persistent homology and Gaussian elimination was established early on [27], and as a result it has been long accepted that persistence can be computed in matrix multiplication time. Milosavljevic et al. [21] gave an explicit algorithm for the more general case of zigzag persistence [5, 6]. The main difficulty in applying standard linear algebra techniques is the strict constraint on the row and column ordering of the boundary matrix.

As persistent homology has evolved as a field, the significance of its formulation as an $\mathbf{R} = \mathbf{DV}$ decomposition [8] of the input boundary matrix $\mathbf{D}$ has become clear. The cycles and chains recovered from this decomposition found many uses in attributing topological features to the input data [11, 9, 24]. However, it is not immediately clear from the algorithm in [21] how to recover the matrices $\mathbf{R}$ and $\mathbf{V}$ necessary in applications. Rather than deconstructing the algorithm of Milosavljevic et al. [21], we simplify it for the case of ordinary persistence.

© Dmitriy Morozov and Primoz Skraba;
licensed under Creative Commons License CC-BY 4.0



We describe how to compute two different forms of the $\mathbf{R} = \mathbf{DV}$ decomposition that come up in applications – lazy and exhaustive reductions. This allows us to recover specific cycle representatives in matrix multiplication time. As a benefit, due to its simpler setting, the algorithms in this paper are greatly simplified compared to [21]. Finally, using Dey and Hou's fast zigzag construction [13], which reduces computation of zigzag persistence to ordinary persistence; the algorithms in this paper give another way to compute zigzag persistence in matrix multiplication time.

2 Preliminaries

We are concerned with the persistent homology of a filtration, an increasing sequence of simplicial complexes indexed by the natural numbers $\mathbb{N}$:

$$\emptyset = K_0 \subset K_1 \subset \ldots \subset K_N = K$$

While persistent homology can be defined more generally, from an algorithmic perspective this is sufficiently general as nearly all considered cases may be reduced to this setting. We further assume that $|K_i - K_{i-1}| = 1$, or equivalently that the filtration is a total order, i.e., each step in the filtration adds a single cell. If the initial filtration does not satisfy this requirement, and defines only a partial order, we may always extend it to an arbitrary compatible total order, e.g., breaking ties by lexicographical ordering and dimension.

Notation. Throughout, we use the following notation:

- Matrices are denoted by bold capital letters, e.g., $\mathbf{M}$, with sub-matrices indexed by square brackets, e.g., the j-th column is $\mathbf{M}[\cdot, j]$, the i-th row is $\mathbf{M}[i, \cdot]$.
- Indexing starts at 1 and indices may be indexed by sets, e.g., $A = \{1, \ldots, k\}$, $\mathbf{M}[\cdot, A]$ refers to the first k columns. Note that the set may not refer to contiguous columns/rows.
- We consider permutations implicitly, so $(\mathbf{P} \cdot \mathbf{M})[A, \cdot]$ refers to the rows indexed **after** the permutation $\mathbf{P}$ is applied to matrix $\mathbf{M}$. For all the algorithms, performing an explicit permutation, then undoing it, does not affect the asymptotic running time.

2.1 Persistence Algorithm

For completeness, we recount the standard persistence algorithm, see [15], [27], and [16] for a more complete description. Let $\mathbf{D}$ denote the boundary operator, which we represent as an $n \times n$-matrix over some fixed (finite) field $\mathbb{F}$. The rows (top-down) and columns (left-right) are ordered by appearance in the filtration. Hence, the i-th row and column represents the simplex $\sigma = K_i - K_{i-1}$ and $\mathbf{D}[\cdot, i]$ represents the boundary of σ respectively. The goal is to compute the $\mathbf{RV}$-decomposition [8]. That is, find matrices $\mathbf{R}$ and $\mathbf{V}$ such that

$$\mathbf{R} = \mathbf{D} \cdot \mathbf{V},$$

where $\mathbf{R}$ is a matrix in reduced column echelon form and $\mathbf{V}$ is a full-rank upper-triangular matrix; we let $\mathbf{U} = \mathbf{V}^{-1}$. We introduce the helper function

$$\mathbf{low}(j) = \arg\max_i \mathbf{R}[i, j] \neq 0,$$

which is defined whenever $\mathbf{R}[\cdot, j] \neq \mathbf{0}$. When defined, it returns the largest row index i where the column is non-zero; we refer to these lowest non-zero elements as **pivots**. Throughout this paper, **low** always applies to $\mathbf{R}$. The persistence diagram is then given by:

$$\text{dgm} = \{(i, j) \mid i = \mathbf{low}(j)\} \cup \{(i, \infty) \mid \mathbf{R}[\cdot, i] = \mathbf{0} \text{ and } i \neq \mathbf{low}(j)\ \forall j\}. \tag{1}$$

Algorithm 1 Standard Persistence Algorithm (Lazy reduction).

```
R = D
V = I_n, U = I_n
for j = 1 to n do
    while R[·, j] ≠ 0 and ∃j′ < j such that low(j) = low(j′) do
        i ← low(j)
        α ← R[i, j]/R[i, j′]
        R[·, j] ← R[·, j] − α · R[·, j′]
        V[·, j] ← V[·, j] − α · V[·, j′]
        U[j′, ·] ← U[j′, ·] + α · U[j, ·]        (equivalently, U[j′, j] ← α)
```

The standard algorithm is a variant of Gaussian elimination. Algorithm 1, which we call the **lazy reduction**, reduces each column by considering pivots in the columns to the left. In each step of the loop, it removes the lowest non-zero entry until a new pivot is found or the entire column is zeroed out. A straightforward analysis gives a running time bound of $O(n^3)$. In [22], a filtration where this algorithm takes cubic time was given, showing that the bound is tight. The updates of matrix $\mathbf{V}$ follow those of matrix $\mathbf{R}$. The updates in matrix $\mathbf{U}$ undo the updates in $\mathbf{V}$ to maintain $\mathbf{U} \cdot \mathbf{V} = \mathbf{I}$. Because the columns of $\mathbf{R}$ are processed from left to right, during the update of matrix $\mathbf{U}$, the row $\mathbf{U}[j, \cdot]$ has a single (diagonal) element. Therefore, the update in $\mathbf{U}$ is equivalent to just setting the entry $\mathbf{U}[j', j]$ to α.

Algorithm 2 "Look-ahead" Variant of Standard Persistence (Exhaustive reduction).

```
R = D
V = I_n, U = I_n
for j = 1 to n do
    if low(j) is defined then
        i ← low(j)
        for j′ > j and R[low(j), j′] ≠ 0 do
            α ← R[i, j′]/R[i, j]
            R[·, j′] ← R[·, j′] − α · R[·, j]
            V[·, j′] ← V[·, j′] − α · V[·, j]
            U[j, ·] ← U[j, ·] + α · U[j′, ·]        (not the same as U[j, j′] ← α!)
```

We introduce another variant of persistence computation used in applications. This one "looks ahead" and eliminates as many elements from the matrix as it can; so we call it the **exhaustive reduction**. In Algorithm 2, by the time a column is considered, it is already reduced. It is then applied to all columns to the right, zeroing out the entire row. In this way, when a column is processed, all previous pivots have already been applied. Because column $\mathbf{R}[\cdot, j']$ may have already been used to reduce another column $\mathbf{R}[\cdot, j'']$, when an update is applied to it, the row $\mathbf{U}[j', \cdot]$ need not consist of a single diagonal element. Hence, the row update in matrix $\mathbf{U}$ is not equivalent to just setting $\mathbf{U}[j, j']$ to α, like in the lazy version of the algorithm.

Lazy vs. Exhaustive. The algorithms based on the two reductions are in a sense the extreme cases: lazy reduction reduces columns only when necessary; and the exhaustive reduction reduces them as much as possible. To illustrate the difference, in the exhaustive reduction, the entire row to the right of a pivot will be zeroed out, whereas in the case of the lazy

reduction, only those columns will be zeroed out which have a conflicting pivot in the same row. There are, of course, any number of possible other reductions between these – but we do not know of any applications that rely on them.

▶ Remark 1. As noted in [10, Section 3.4], persistent cohomology can be computed by performing the same algorithm on the anti-transpose of $\mathbf{D}$. As such all the algorithms in this paper can compute persistent cohomology.

2.2 Cycle Representatives

Given an $\mathbf{R} = \mathbf{DV}$ decomposition, we distinguish between two types of simplices in the filtration: *positive* simplices create new homology classes; *negative* simplices destroy them. From the matrix decomposition, one can recover a set of cycle representatives of persistent homology. There are two sources of this information, although with different content.

1. $\mathbf{V}$ matrix: The columns of positive simplices in $\mathbf{V}$ correspond to zero columns in the reduced matrix $\mathbf{R}$. These columns in $\mathbf{V}$ are cycles, by definition.
2. $\mathbf{R}$ matrix: The columns of negative simplices in $\mathbf{R}$ are non-zero. They store linear combinations of boundaries, which are cycles (at the time of their birth) that eventually die in the filtration.

The cycles one recovers from matrix $\mathbf{R}$ differ between the lazy and exhaustive reductions. Exhaustive reduction was called total reduction by Cohen-Steiner et al. [9]. They show that these cycles form a lexicographically optimal basis, which can be used to triangulate point cloud data. Nigmetov and Morozov [24] use the lazy reduction – specifically, the columns and rows from matrices $\mathbf{V}$ and $\mathbf{U}$ – to recover "critical sets," subsets of simplicial complexes affected by optimization with the loss formulated in terms of a persistence diagram.

Minimum Spanning Acycle Basis. The negative simplices in the filtration represent a minimal spanning acycle (MSA) [19, 25]. The columns in $\mathbf{V}$ obtained from either lazy or exhaustive reduction – or any reduction that never adds columns of positive simplices to other columns – have the form $\sum_{\tau \in MSA} \lambda_\tau \tau + \lambda_\sigma \sigma$. The support of the chain is one (positive) simplex σ plus a linear combination of simplices in the minimum spanning acycle.

▶ **Lemma 2.** *The representatives from* $\mathbf{V}$ *are the same for lazy and exhaustive reduction.*

Proof. Which columns of $\mathbf{R}$ are non-zero does not depend on the algorithm used; the set of negative simplices only depends on the order of the filtration. We observe that the boundaries of the negative simplices form a basis (since their pivots are in distinct rows). By definition, the boundary of a positive simplex may be expressed as a linear combination of boundaries of negative simplices. Since the boundaries of the negative simplices form a basis, this linear combination is unique. ◀

Death Basis. An alternative is to get cycle representatives directly from the $\mathbf{R}$ matrix. Its non-zero columns represent cycles because multiplying $\mathbf{D} \cdot \mathbf{R} = 0$. This follows directly since $\mathbf{R} = \mathbf{D} \cdot \mathbf{V}$, so $\mathbf{D} \cdot \mathbf{R} = \mathbf{D} \cdot \mathbf{D} \cdot \mathbf{V} = 0$, since $\mathbf{D} \cdot \mathbf{D} = 0$. These columns give cycle representatives for all finite classes, i.e., those homology classes that eventually die.

▶ Remark 3. Following [10], this may be used for cohomology – applying the algorithms to $\mathbf{D}^\perp$, the anti-transpose of $\mathbf{D}$ to obtain the decomposition $\mathbf{R}^\perp = \mathbf{D}^\perp \cdot \mathbf{V}^\perp$. The cocycle representatives are then given by columns in $\mathbf{V}^\perp$. To see that these are cocycles, observe that by construction, they map to zero via the coboundary operator $\mathbf{D}^\perp$ before the corresponding

death time, i.e., below the pivot in the corresponding column of $\mathbf{R}^\perp$, as we have reversed the indexing with the anti-transpose. Likewise, the cocycle representatives of essential cocycles are given by the columns of $\mathbf{V}^\perp$ whose: corresponding columns in $\mathbf{R}^\perp$ are zero *and* they must not be coboundaries, i.e., the corresponding rows in $\mathbf{R}^\perp$ must not contain pivots.

3 Matrix Preliminaries

We recount a few classical results on matrix multiplication. We assume the matrices are over some field $\mathbb{F}$.

▶ **Lemma 4.** *Let $\mathbf{B}$ be an $n \times k$ matrix and $\mathbf{C}$ be a $k \times k$ matrix. The product $\mathbf{B} \cdot \mathbf{C}$ can be computed in $O(nk^{\omega-1})$-time.*

Proof. Divide $\mathbf{B}$ into $\lceil \frac{n}{k} \rceil$ sub-matrices $\mathbf{B}_i$ of size $k \times k$:

$$\mathbf{B} \cdot \mathbf{C} = \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_{\lceil n/k \rceil} \end{bmatrix} \cdot \mathbf{C} = \begin{bmatrix} \mathbf{B}_1 \cdot \mathbf{C} \\ \mathbf{B}_2 \cdot \mathbf{C} \\ \vdots \\ \mathbf{B}_{\lceil n/k \rceil} \cdot \mathbf{C} \end{bmatrix}$$

Each $\mathbf{B}_i \cdot \mathbf{C}$ is a product of two $k \times k$ matrices, which by definition takes $O(k^\omega)$ time. There are $\lceil \frac{n}{k} \rceil$ products to compute yielding a running time of $O\left(\frac{n+1}{k} k^\omega\right) = O(nk^{\omega-1})$. ◀

Column Operations via Matrix Multiplication. Here we relate matrix multiplication with the reduction steps used in the persistence algorithms, i.e., representing column operations via matrix multiplication. Assume that we have the following block matrix,

$$\left[\begin{array}{c|c} \mathbf{B} & \mathbf{C} \end{array}\right] = \left[\begin{array}{ccc|ccc} b(1) & \dots & b(n) & c(1) & \dots & c(m) \end{array}\right]$$

where $b(i)$ and $c(j)$ are (column) vectors. Reducing $c(j)$ which we denote $c'(j)$, using the vectors $b(i)$, is expressed as a linear combination

$$c'(j) = c(j) + \sum_i \lambda_i(j) b(i),$$

where $\lambda_i(j)$ are the coefficients. Reducing only the j-th column of $\mathbf{C}$ may be written as

$$\begin{bmatrix} \mathbf{B} & c'(j) \end{bmatrix} = \begin{bmatrix} \mathbf{B} & c(j) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \lambda(j) \\ 0 & \mathbf{1}_j \end{bmatrix},$$

where $\mathbf{1}_j$ is a column vector where the j-th entry is 1 and zero everywhere else. If we consider $\mathbf{\Lambda}[i,j] := \lambda_i(j)$, we can rewrite the reduction of $\mathbf{C}$ by

$$\begin{bmatrix} \mathbf{B} & \mathbf{C}' \end{bmatrix} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{\Lambda} \\ 0 & \mathbf{I} \end{bmatrix},$$

or equivalently, the reduced matrix is $\mathbf{C}' = \mathbf{C} - \mathbf{B} \cdot \mathbf{\Lambda}$.

▶ **Lemma 5.** *Multiplying a permutation matrix with an $n \times m$ matrix, takes $O(nm)$ time.*

Proof. As a permutation matrix rewrites each element once, the total number of elements gives the bound. ◀

4 Column Algorithms

Here we describe the algorithm in [21] for the special case of classical persistent homology. To aid in exposition, we first recall the **Schur complement**. Given a block matrix

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 \\ \mathbf{B}_2 & \mathbf{C}_2 \end{bmatrix},$$

assuming $\mathbf{B}_2$ is non-singular, we can compute an updated matrix where the rows corresponding to $\mathbf{B}_2$ are zeroed out by the following transformation:

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 \\ \mathbf{B}_2 & \mathbf{C}_2 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{B}_2^{-1}\mathbf{C}_2 \\ 0 & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 - \mathbf{B}_1\mathbf{B}_2^{-1}\mathbf{C}_2 \\ \mathbf{B}_2 & 0 \end{bmatrix} \tag{2}$$

where $\mathbf{C}_1 - \mathbf{B}_1\mathbf{B}_2^{-1}\mathbf{C}_2$ is referred to as the Schur complement.

▶ **Observation 6.** *The matrix $\mathbf{B}_2^{-1}\mathbf{C}_2$ encodes the column operations required to zero out $\mathbf{C_2}$, that is, $\mathbf{B}_2^{-1}\mathbf{C}_2 = \mathbf{\Lambda}$.*

A key fact we will use in the algorithm is that matrix inversion has the same algorithmic complexity as matrix multiplication.

▶ **Theorem 7.** *Inverting a (square) non-singular (upper or lower) triangular matrix can be done in matrix multiplication time.*

Proof. See [23, Appendix] for a simplified proof for the special case of triangular matrices. ◀

4.1 Fast Exhaustive Algorithm

Algorithm 3 Column Algorithm($A = [i, j]$).

```
1:  if i = j then                                                    ▷ leaf
2:      if R[·, i] = 0 then                                          ▷ If no pivot exists
3:          ℓ ← arg min_{ℓ>n} R[·, ℓ] ≠ 0                            ▷ find the next pivot in I_{m+n}
4:          Permute columns i and ℓ
5:          Z ← Z ∪ i
6:  else
7:      B ← [i, (i + j)/2)                                           ▷ left child
8:      C ← [(i + j)/2, j]                                           ▷ right child
9:      Recurse on B
10:     Apply updates from columns B to C:
11:         L ← ⟨low(ℓ) : ℓ ∈ B⟩          ▷ Implicit permutation; L preserves order
12:         Λ ← R[L, B]^{-1} · R[L, C]                               ▷ A_2^{-1} B_2
13:         R[L̄, C] ← R[L̄, C] − R[L̄, B] · Λ   ▷ Apply B to C; Schur update (k × k) · (k × n)
14:         R[L, C] = 0
15:         V[·, C] ← V[·, C] − V[·, B] · Λ                          ▷ Matching update in V
16:     Recurse on C
```

We present a fast version of exhaustive reduction in Algorithm 3. The general structure follows Algorithm 2. However, rather than apply the pivots when we find them, we apply them in batches via the Schur complement through a standard recursion. We first augment

the boundary matrix with an identity matrix, which is of size at least n, for reasons we shall describe below. To simplify notation, we assume n is a power of 2. If it is not, we can set the size of the identity matrix to $m+n$ with $m < n$ such that $m + 2n = 2^x$. Therefore, our initial matrices are:

$$\mathbf{R} = \begin{bmatrix} \mathbf{D} & 0 \\ 0 & \mathbf{I}_{m+n} \end{bmatrix}, \qquad \mathbf{V} = \mathbf{I}_{m+2n}.$$

Throughout the algorithm, we implicitly perform two types of permutations by taking appropriate subsets of rows and columns. The first are column permutations. To ensure that the appropriate submatrices are invertible, we require that after processing k columns of $\mathbf{R}$, there are k pivots, or, equivalently, that the matrix corresponding to the processed columns is full (column) rank. Because cycles reduce to zero columns, the if-statement in Algorithm 3 permutes a zero column from $\mathbf{R}$ with a non-zero column from the identity matrix $\mathbf{I}_{m+n}$, and records the indices of such columns in list Z, to eventually undo the permutation.

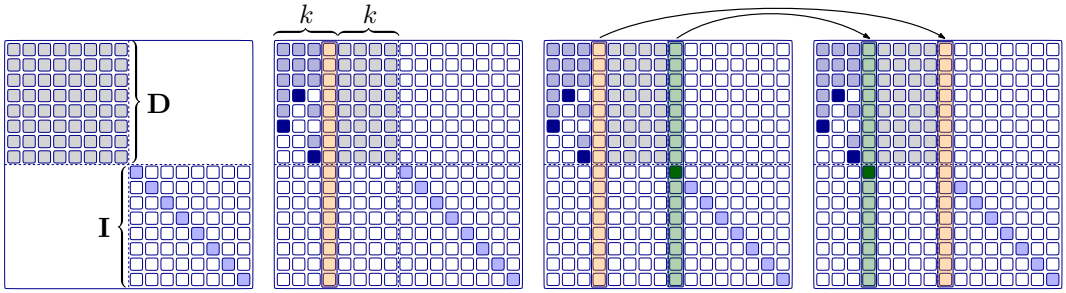


Figure 1 An example of the column permutation in the exhaustive algorithm. (a) Initially, we append an identity matrix to the boundary matrix. (b) Assume we have processed the first k columns. The blue entries represent non-zero entries, and the dark blue boxes represent the pivots. To apply these pivots to next k columns (shown in gray), the first k columns must contain k pivots. (c-d) If there is a a zero column (shown in orange), we transpose it with the first non-zero entry in the appended identity matrix, ensuring the processed k columns all have pivots.

We must also consider row permutations. To apply Theorem 7, we must find the inverse $\mathbf{R}^{-1}[L, B]$ in Algorithm 3, which is used in the Schur complement. We construct a sequence $L = \langle \mathbf{low}(\ell) : \ell \in B \rangle$ of the pivot rows in columns B. Denoting by $\bar{L}$ the sequence of rows outside of L, we denote a permutation that places rows L below $\bar{L}$ as $\mathbf{P}_L$:

$$(\mathbf{P}_L \cdot \mathbf{R}) = \begin{bmatrix} \mathbf{R}[\bar{L}, \cdot] \\ \mathbf{R}[L, \cdot] \end{bmatrix}. \tag{3}$$

See Figure 2(b-c). The permutation is explicitly given by,

$$P_L : \mathbf{low}(B(i)) \mapsto m + n - |B| + i \qquad \text{for } i = 1, \ldots, |B|$$

where $B(i)$ is the i-th entry of the indices in B.

▶ **Lemma 8.** *After permuting the rows by $\mathbf{P}_L$, the bottom-left $k \times k$ matrix is lower triangular, where $k = |L|$.*

We delay the proof until after the description of the algorithm. We now follow a recursion: we split matrix $\mathbf{R}$ into the first half of the columns denoted by B and the second half denoted by C. We proceed to call the algorithm on the first half of the matrix. Once we reach a

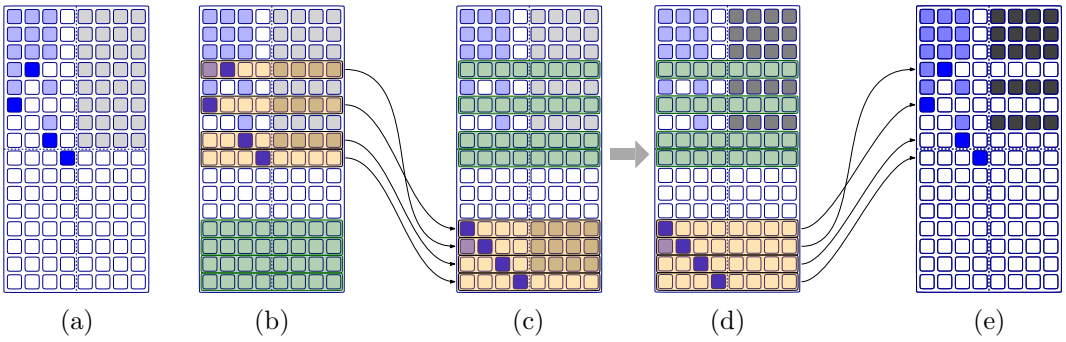


Figure 2 An example of the row permutations. (a) A submatrix consisting of 8 columns where the 4 columns on the left have been reduced, and we need to apply the pivots to the following 4 columns (in gray). (b-c) Each row which has a pivot gets permuted to the bottom of the matrix in the same order as the columns – resulting in a lower triangular matrix. (d) Applying the Schur complement zeros out the gray entries on the bottom right and we update the entries in the upper right (dark gray). (e) With the pivot rows zeroed out on the right, we reverse the permutation.

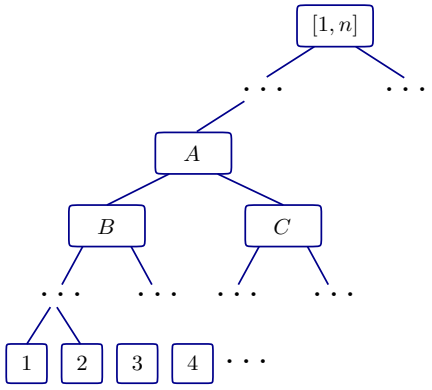


Figure 3 The standard binary recursion tree in how we split the matrix by column. Observe that $|B| = |C| = |L|$ in Algorithm 3 and that $\mathbf{R}[L, B] = (\mathbf{P}_L \cdot \mathbf{R})[\{n - |B| + 1, \ldots, n\}, B]$, the bottom $|B|$ rows in the permuted matrix.

leaf in the recursion tree (shown in Figure 3), the matrix consists of a single column. The key invariant is that whenever we reach a leaf, the corresponding column is reduced with respect to all the columns which come prior to it in the filtration. For the first column, this is tautologically true and we delay the proof for the general case until later.

If we are not in the base case (a single column), we must apply the pivots in the columns indexed by B (left half of the submatrix) to the the columns indexed by C (right half of the submatrix). As described in Equation (3), we construct the permutation $\mathbf{P}_L$ such that

$$(\mathbf{P}_L \cdot \mathbf{R})[\cdot, B \cup C]) = \begin{bmatrix} \mathbf{R}[\bar{L}, B] & \mathbf{R}[\bar{L}, C] \\ \mathbf{R}[L, B] & \mathbf{R}[L, C] \end{bmatrix}$$

By construction of $\mathbf{P}_L$, $\mathbf{R}[L, B]$ is lower-triangular with non-zero diagonal entries and so is full-rank. We can then apply the Schur complement to zero out all the corresponding rows in the columns C, $\mathbf{R}[L, C] \leftarrow 0$ and update the rest of the rows:

$$\underbrace{\mathbf{R}[\bar{L}, C]}_{\mathbf{C}_1} \leftarrow \underbrace{\mathbf{R}[\bar{L}, C]}_{\mathbf{C}_1} - \underbrace{\mathbf{R}[\bar{L}, B]}_{\mathbf{B}_1} \cdot \underbrace{\mathbf{R}[L, B]^{-1}}_{\mathbf{B}_2^{-1}} \cdot \underbrace{\mathbf{R}[L, C]}_{\mathbf{C}_2} = \mathbf{R}[\bar{L}, C] - \mathbf{R}[\bar{L}, B] \cdot \mathbf{\Lambda}$$

To update matrix $\mathbf{V}$, we perform the same operations $\mathbf{\Lambda} = \mathbf{R}[L, B]^{-1} \cdot \mathbf{R}[L, C]$ on $\mathbf{V}$:

$$\mathbf{V}[\bar{L}, C] \leftarrow \mathbf{V}[\bar{L}, C] - \mathbf{V}[\bar{L}, C] \cdot \mathbf{\Lambda}$$

Once the left half of the matrix has been applied to the right half, we recurse on the second half, i.e., C. Here again, because $\mathbf{R}[L, B]$ is lower-triangular full-rank and $\mathbf{R}[L, C] = 0$, the first column of $\mathbf{R}[\cdot, C]$ is reduced. We now prove correctness.

▶ **Lemma 9.** *When the recursion base case is reached for column k, $\mathbf{R}[\mathbf{low}(j), k] = 0$ for all $j < k$.*

Proof. For $k = 1$, this is tautological since $\mathbf{low}(j)$ is empty. For $k > 1$, we observe that entry $\mathbf{R}[\mathbf{low}(i), k]$ was made zero when $i \in B$ and $k \in C$. As the recursion always splits the column range in half, this condition must be satisfied somewhere in the recursion before the base case of k is reached. ◀

Proof of Lemma 8. We observe that when $\mathbf{P}_L$ is applied, the columns corresponding to L have a pivot. By construction, $\mathbf{P}_L$ places the pivots onto the diagonal in the bottom $|L|$ rows. For any of these rows above the diagonal, the entries are 0 by Lemma 9, implying the bottom $k \times k$ matrix is lower triangular. ◀

▶ **Theorem 10.** *Algorithm 3 computes the persistence diagram correctly.*

Proof. By Equation (1), it suffices to show that for all i, $\mathbf{low}(i)$ is computed correctly. For $i = 1$, this is trivially true. For $i > 1$, observe that at any step of the algorithm, we maintain

$$\mathbf{R} = \mathbf{D} \cdot \mathbf{V},$$

as any operations on $\mathbf{R}$ are applied to $\mathbf{V}$. Assuming the pivots are correct for $j < i$, by the above and Lemma 9, $\mathbf{V}$ gives a transformation depending only on columns $j < i$, such that all elements in $\mathbf{R}[\mathbf{low}(j), i] = 0$. If $\mathbf{low}(i)$ does not exist, $D[\cdot, i]$ was in the span of the previous columns and hence a cycle. If $\mathbf{low}(i)$ exists, it is distinct from $\{\mathbf{low}(j)\}_{j<i}$ and so we have a new pairing. The correctness of this pairing follows from the Pairing Uniqueness Lemma [8]. ◀

Running Time Analysis. The base case takes $O(1)$ time. For a general step where $|B| = |C| = k$, applying the update requires a row permutation of a $2n \times 2k$ matrix which takes $O(nk)$ time. We must compute $\mathbf{R}[L, B]^{-1}$ which is inverting a $k \times k$ matrix. Multiplying it with $\mathbf{R}[L, C]$ takes $O(k^\omega)$ time as both are $k \times k$ matrices. Finally, we multiply this product with the $n \times k$ matrix $\mathbf{R}[\bar{L}, C]$, which by Lemma 4, takes $O(nk^{\omega-1})$ time. Solving the recursion, we find that the total running time is $O(n^\omega)$ for $\omega > 2$ or $O(n^2 \log n)$ if $\omega = 2$. For completeness, we include the derivation in the full version [23].

5 Row Algorithm

While Algorithm 3 produces the exhaustive reduction in matrix multiplication time, a natural question is whether the same can be done for computing the lazy reduction (and its representatives). In this section, we present an algorithm which achieves this. We first give the iterative version before describing how it can be computed in $O(n^\omega)$. The idea behind this reduction is to consider rows from the bottom up. Observe that a pivot in the bottom row can be directly identified in $\mathbf{D}$ as the earliest (leftmost) non-zero entry, similarly as the

Algorithm 4 Incremental Row Algorithm.

```
1: R = D
2: V = I_n, U = I_n
3: for i = n to 1 do
4:     if lft(i) is defined then
5:         j ← lft(i)
6:         for j' > j and low(j') = i do
7:             α ← R[i, j']/R[i, j]
8:             R[·, j'] ← R[·, j'] − α · R[·, j]
9:             V[·, j'] ← V[·, j'] − α · V[·, j]
10:            U[j, ·] ← U[j, ·] + α · U[j', ·]        (equivalently, U[j, j'] ← α)
```

pivot in the first column can be directly identified. To formalize the notion of the earliest eligible pivot in a given row, we define

$$\mathbf{lft}(i) = \arg\min_j \{\mathbf{R}[i,j] \neq 0 : \mathbf{low}(j) = i\}$$

As in the case of **low**, we only apply this function to $\mathbf{R}$. The function **lft** returns the earliest column j which is non-zero and is the lowest such non-zero entry in its column. This condition is important as the earliest (left-most) non-zero entry may have pivots below it. This column is used to zero out the i-th row in columns to the right provided they do not already contain a pivot, i.e., if $\mathbf{low}(j) = i$, then the column is applied. This condition is why we obtain equivalence with the lazy reduction – we do not apply the pivot column to later columns which already contain a pivot.

▶ **Lemma 11.** *Algorithm 4 produces the same decomposition as the lazy reduction.*

Proof. This is equivalent to [10, Theorem 3.1]. ◀

5.1 Fast Row Algorithm

The fast version of Algorithm 4 is given in Algorithm 5. As above, it reduces the matrix row by row from the bottom up. This variant does not require matrix inversion and requires less padding. The trade-off is that it requires a more complex sequence of updates. Assuming $\mathbf{D}$ is an $n \times n$ matrix, we set $m < n$ such that $m + n = 2^x$. We initialize our matrices as:

$$\mathbf{R} = \left[\begin{array}{c|c} \mathbf{D} & \\ \mathbf{0}_m & \mathbf{I}_{m+n} \end{array}\right], \qquad \mathbf{\Lambda} = \mathbf{I}_{m+n}.$$

To simplify notation, from this point on, we assume n is a power of 2 (as $m + n < 2n$), so the resulting matrix is $\begin{bmatrix}\mathbf{D} & \mathbf{I}_n\end{bmatrix}$. This ensures that $\mathbf{lft}(i)$ is defined for all i in $\mathbf{R}$, i.e., the identity matrix ensures there are no zero rows.

It will be convenient to again consider permutations, but here they will be column permutations. We build up the permutation incrementally. In the i-th step, we update the permutation with the column transposition

$$\pi(i) : \mathbf{lft}(n - i + 1) \leftrightarrow i.$$

Algorithm 5 Row Algorithm($A = [i, j]$).

if $i = j$ **then** ▷ *leaf*
  $k \leftarrow \mathbf{lft}(n - i + 1)$ ▷ *Identify the pivot in row* $(n - i + 1)$
  Update permutation matrix $\mathbf{P}$ with transposition $i \leftrightarrow k$.
  $(\mathbf{P}\cdot\mathbf{\Lambda}\cdot\mathbf{P})[i, j] \leftarrow -\frac{(\mathbf{R}\cdot\mathbf{P})[n-i+1,j]}{(\mathbf{R}\cdot\mathbf{P})[n-i+1,i]}$ for $j > i$.
  $(\mathbf{R}\cdot\mathbf{P})[n - i + 1, \{i + 1, \ldots, n\}] \leftarrow 0$ ▷ *Zero out* $(n - i + 1)$*-st row; see Remark 12*
else
  $B \leftarrow [i, (i + j)/2)$ ▷ *left child*
  $C \leftarrow [(i + j)/2, j]$ ▷ *right child*
  $C_r \leftarrow n - C + 1$ ▷ *Rows are indexed top-down but the algorithm goes bottom up*
  Recurse on B
  Apply updates from rows B to C: ▷ *full row update;* $(k \times k) \cdot (k \times n)$
    $(\mathbf{R}\cdot\mathbf{P})[C_r, \cdot] \leftarrow (\mathbf{R}\cdot\mathbf{P})[C_r, \cdot] + (\mathbf{R}\cdot\mathbf{P})[C_r, B])(\mathbf{P}\cdot\mathbf{\Lambda}\cdot\mathbf{P})[B, \cdot]$
  Recurse on C
  $\bar{A}_r = [1, i]$ ▷ *everything above A*
  Full column update:
    $(\mathbf{R}\cdot\mathbf{P})[\bar{A}_r, C] \leftarrow (\mathbf{R}\cdot\mathbf{P})[\bar{A}_r, C] + (\mathbf{R}\cdot\mathbf{P})[\bar{A}_r, B] \cdot (\mathbf{P}\cdot\mathbf{\Lambda}\cdot\mathbf{P})[B, C]$ ▷ $(n \times k) \cdot (k \times k)$

which we use a shorthand for the two mappings $\mathbf{lft}(n - i + 1) \mapsto i$ and $i \mapsto \mathbf{lft}(n - i + 1)$. As **lft** is always defined, two columns are permuted in each step. The permutation after i steps is then given by

$$\mathbf{P}_i = \prod_{j \le i} \pi(j).$$

Note that we apply the permutations in order, i.e., for $i = 1, 2, \ldots$ While the columns corresponding to $\mathbf{lft}(n - i + 1)$ are fixed at i for all permutations, the original column at i may be permuted several times. In the algorithm, we omit the subscript; $\mathbf{P}$ refers to the accumulated permutation. As in the case of the column algorithm, we perform the permutations implicitly. We observe that in the permuted order, we arrange pivots on the left along the anti-diagonal, see Figure 4.

We keep track of the operations directly in the matrix $\mathbf{\Lambda}$ which is initialized as the identity matrix. Before delving into the details of the algorithm, assume that the all rows below the i-th row have been reduced. The updates in the i-th row are given by

$$(\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[i, j] = \begin{cases} -\frac{(\mathbf{R}\cdot\mathbf{P})[n-i+1,j]}{(\mathbf{R}\cdot\mathbf{P})[n-i+1,i]} & j > i \\ 1 & n - i + 1 = j \\ 0 & \text{else} \end{cases}$$

We digress here to explain why this works. As mentioned above, if we consider the column-permuted matrix $\mathbf{R} \cdot \mathbf{P}$, it has the pivots in the bottom-left along the anti-diagonal (Figure 4). The i-th column corresponds to the column given by $\mathbf{lft}(n - i + 1)$ (in the original filtration order). Hence, it is the earliest column with a non-zero entry in the i-th row from the bottom such that there is no pivot below. For any non-zero entry to the right of i, it must occur to the right of i in the original filtration order (before permuting), or it would have been returned by $\mathbf{lft}(n - i + 1)$ rather than the current i-th column in the permuted order.

▶ Remark 12. Hence in Algorithm 5, all non-zeros in the row to the right of i are pivots (lowest entries), because every column with a pivot below $n - i + 1$ has been moved to the left of i in the permuted order and so no reduction will be applied, i.e., coefficient will be 0.

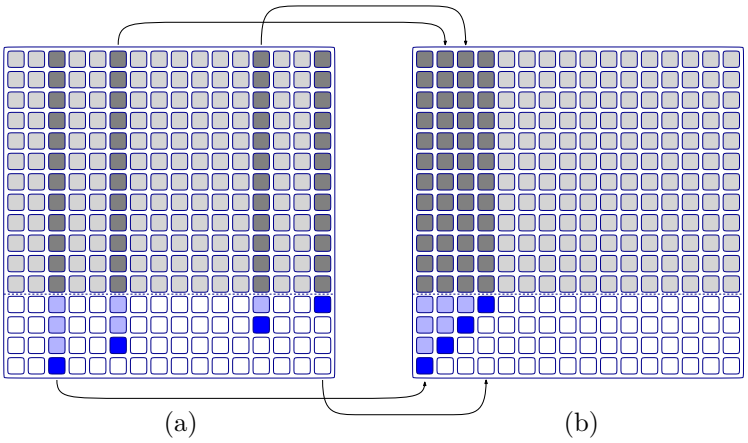


Figure 4 The permutation of the columns so that the pivots in the bottom k rows are on the right. An important observation is that despite this reshuffling of columns, because of the definition of $\mathbf{lft}(\cdot)$, we never "accidentally" reduce columns which occur earlier in the filtration order.

Finally, as we permute the columns we must also permute the rows of the $\mathbf{\Lambda}$ matrix so that the recorded column operations match. This represents the base case of the recursion. As $\mathbf{\Lambda}$ is initialized as the identity matrix, the case $n - i + 1 = j$ is taken care of implicitly.

Returning to the algorithm, we proceed by recursing as in the column algorithm, but on the rows rather than on the columns. We divide the rows of the sub-matrix into the top and bottom halves. We recurse first on the bottom half. Because the pivots are determined by rows below the current one, we can directly identify the pivot with $\mathbf{lft}$.

The base case is a single row. By assumption, rows below the current row have been updated. At row $n - i + 1$, we first find the left-most non-zero entry and perform a column permutation setting the column $\mathbf{lft}(n - i + 1)$ to the i-th column, making $\mathbf{R}[n - i, i]$ a pivot as required. By the reasoning above, we may zero out the row, recording the operations in $\mathbf{\Lambda}$ and we can update $(\mathbf{R} \cdot \mathbf{P})[n - i, \{i + 1, \ldots, n\}] \leftarrow 0$.

Though the recursion proceeds as in Algorithm 3, splitting the matrix into the indices in the "first half" and "second half," a complication is that the columns are processed left to right (same as the indexing), while the rows are processed bottom-up while being indexed top-down. Hence, we introduce:

$$C_r = n - C + 1,$$

by which we mean for any $i \in C$, $i \mapsto n - i + 1$. This indexes the rows from the bottom of the matrix as required by the algorithm. Returning to the recursion, once we processed the columns for B, we first apply the updates to the rows C_r,

$$(\mathbf{R} \cdot \mathbf{P})[C_r, \cdot] \leftarrow (\mathbf{R} \cdot \mathbf{P})[C_r, \cdot] + (\mathbf{R} \cdot \mathbf{P})[C_r, B] \cdot (\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[B, \cdot],$$

shown in orange in Figure 5. Note that though B_r is shown in Figure 5, it is not used explicitly in the algorithm as all the operations in those rows have already been performed. On the other hand, B in $(\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[B, \cdot]$ and in $(\mathbf{R} \cdot \mathbf{P})[C_r, B]$ do not need to be reversed, as they both represent column operations which are in the standard order. After this, the first row in C_r is now up-to-date with all operations from rows below it. This allows us to correctly identify $\mathbf{lft}$ for this row and perform the column transposition accordingly. Hence, when we recurse on C, the base case can be carried out.

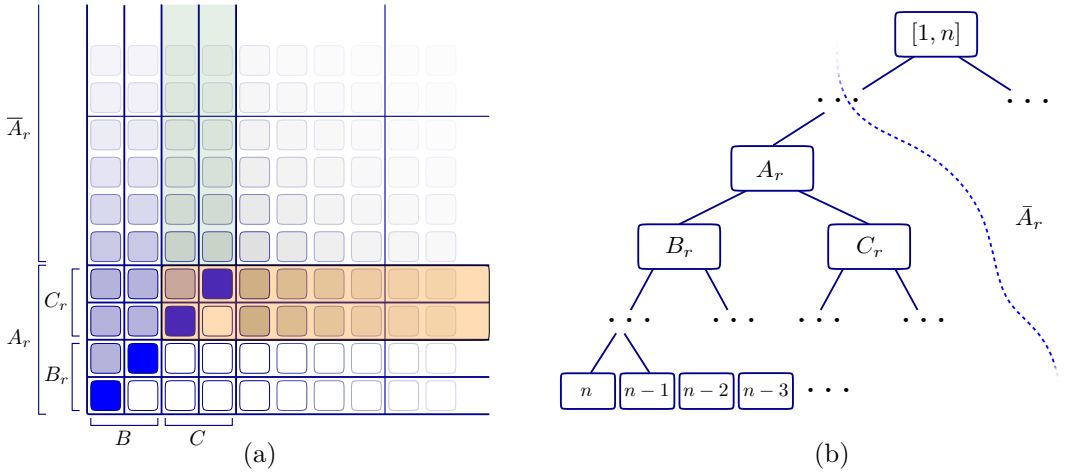


Figure 5 The updating steps in Algorithm 5. On the right we have the recursion tree for the rows, as opposed to the columns which is the same as in Figure 3 – as rows proceed bottom-up and columns proceed left to right. When we return from the recursion on B (or B_r), we have zeroed out all the entries to the right of columns B in rows B_r. We then first need to apply these to the rows in C_r (line 11 in Algorithm 5), shown in orange. We can then identify the correct pivot columns for rows C_r (after recursing), and complete the step by applying the operations from columns B to C above. This is shown in green – and corresponding to the right of $A_r = B_r \cup C_r$ in the recursion tree (the rows above A_r – denoted $\bar{A}_r$) to the right of the dashed line. This makes columns $B \cup C$ and rows A_r completely processed/reduced.

Returning from C, we complete the recursion by performing the update operations on the remaining parts of the columns in C, shown in green in Figure 5. Note, the columns are indexed by C not C_r. Let $\bar{A}_r = \{1, \dots, \min(C_r) - 1\}$. The column update is given as

$$(\mathbf{R} \cdot \mathbf{P})[\bar{A}_r, C] \leftarrow (\mathbf{R} \cdot \mathbf{P})[\bar{A}_r, C] + (\mathbf{R} \cdot \mathbf{P})[\bar{A}_r, B] \cdot (\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[B, C]$$

Now all columns prior to and including C and all rows below and including C_r are up to date. We show the update sequence in Figure 5 (omitting the permutations). Note that in the column update, the rows in $\bar{A}_r$ are correctly indexed, so no reversal is necessary.

▶ **Theorem 13.** *Algorithm 5 computes the lazy reduction.*

Proof. The algorithm is equivalent to Algorithm 4, but it performs the operations by batches. Observe that in the base case for row $n - i + 1$, the column operations recorded in $\mathbf{\Lambda}$ are equal to α in the i-th step of Algorithm 4. If we performed the operations on the entire column, we would, in fact, obtain exactly Algorithm 4 (since there would be no need for any further updates in the recursion).

Hence, to show correctness, we need to ensure that we can correctly compute $\mathbf{lft}(i)$ in each base case. This is trivially true in the first step. In further steps, we need to ensure that all the column operations have been applied to $(\mathbf{R} \cdot \mathbf{P})[C, \cdot]$. We can directly verify that any column operations coming from row $j < i$, fall in exactly one B (with i in C) in the recursion and hence are applied before the base case for i. It only remains to show that the columns $(\mathbf{R} \cdot \mathbf{P})[\cdot, C]$ are up to date before the columns operations from rows in C_r are applied in the appropriate rows of $(\mathbf{R} \cdot \mathbf{P})[\bar{A}_r, \cdot]$, which is done in line 14. ◀

▶ **Theorem 14.** *Matrix* $\mathbf{U} = 2 \cdot \mathbf{I} - \mathbf{\Lambda}$ *satisfies* $\mathbf{D} = \mathbf{R} \cdot \mathbf{U}$*, or equivalently,* $\mathbf{V} = \mathbf{U}^{-1}$*.*

Proof. During a regular reduction, Algorithm 4, updates in matrix $\mathbf{U}$ use row operations to "undo" the column operations in matrices $\mathbf{R}$ and $\mathbf{V}$. When a column j in $\mathbf{R}$ is subtracted from column j', which itself has not been used to reduce any other column, its corresponding row $\mathbf{U}[j', \cdot]$ contains a single diagonal element – compare the situations in Algorithms 1 and 2. In this case, the update in matrix $\mathbf{U}$ is equivalent to setting entry $\mathbf{U}[j', j]$ to the coefficient that is the negation of the coefficient used for the column update. When a column j in $\mathbf{R}$ is subtracted from column j', which itself has not been used to reduce any other column, its corresponding row $\mathbf{U}[j', \cdot]$ contains a single diagonal element.

Algorithm 5 satisfies this property (which is why it produces a lazy reduction): when an update is recorded in matrix $\mathbf{\Lambda}$, the column that is being updated has not yet been used to reduce any other column. Therefore, off the diagonal, the corresponding matrix $\mathbf{U}$ is the negation of matrix $\mathbf{\Lambda}$ – compare the coefficients in Algorithm 5 with α in Algorithm 4. On the diagonal, both $\mathbf{\Lambda}$ and $\mathbf{U}$ are all ones. So adding $2 \cdot \mathbf{I}$ to $-\mathbf{\Lambda}$ recovers $\mathbf{U}$. ◀

▶ Remark 15. As $\mathbf{U}$ is an upper triangular matrix, by Theorem 7, it can be inverted in $O(n^\omega)$-time as required.

Running Time Analysis. The base case takes $O(n)$ time as recording the permutation and zeroing out operations each take linear time. Hence the base case takes $O(n)$ time. For the general case, we need to first apply the column permutations. Since we are permuting at most an $n \times k$ (or $k \times n$) matrix, applying the permutation (and undoing it) takes $O(nk)$ time by Lemma 5. Next, applying the row updates from B_r to C_r, assuming $|B| = |C| = k$, requires multiplying $(\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[B, \cdot]$, a $k \times n$ matrix, with $(\mathbf{R} \cdot \mathbf{P})[C_r, B]$, a $k \times k$ matrix – taking $O(nk^{\omega-1})$ time by Lemma 4. To complete the update, the subtractions take $O(nk)$ time. To compute the column updates, we must multiply $(\mathbf{R} \cdot \mathbf{P})[\bar{A}_r, B]$, an $n \times k$ matrix with $(\mathbf{P} \cdot \mathbf{\Lambda} \cdot \mathbf{P})[B, C]$, a $k \times k$ matrix, which again takes $O(nk^{\omega-1})$ time. Putting these together, we find that the full step can be completed in $O(nk^{\omega-1})$ time and as in the column algorithm case, solving the recursion gives an $O(n^\omega)$ running time for $\omega > 2$ and a $O(n^2 \log n)$ running time for $\omega = 2$.

6 Discussion

In this paper, we have shown that in the case of persistent (co)homology, the standard bases returned by the exhaustive and lazy reductions can be computed in matrix multiplication time. It is worth noting that while both the columns and row algorithms utilize batch updates, they are qualitatively different: (1) Only the column algorithm critically relies on fast matrix inversion; (2) The row algorithm update sequence is substantially more complex but requires only column permutations. While one can use the column algorithm to compute the lazy basis and the row algorithm for the exhaustive basis in the same asymptotic running time, the resulting algorithms are both complex and not particularly illuminating. There are also numerous technical complications to deal with, so we omit them from this paper. Another outstanding issue is whether it is possible to transform the exhaustive representatives to lazy representatives and vice versa without redoing the reduction. Again, we believe this possible, but omit the details due to space and because the resulting asymptotic running time is the same. Finally, we conclude with an open question: what is the optimal running time for computing representatives in zigzag persistence. Recently, this has been partially addressed in [12, 14] but optimality in full generality remains open.

References

1. 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.
2. Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications*, pages 103–117. Springer, 2014. doi:10.1007/978-3-319-04099-8_7.
3. Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat–persistent homology algorithms toolbox. *Journal of symbolic computation*, 78:76–90, 2017. doi:10.1016/J.JSC.2016.03.008.
4. Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian workshop on algorithm theory*, pages 189–200. Springer, 2012. doi:10.1007/978-3-642-31155-0_17.
5. Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, August 2010. doi:10.1007/s10208-010-9066-0.
6. Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 247–256, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542408.
7. Chao Chen and Michael Kerber. An output-sensitive algorithm for persistent homology. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 207–216, 2011. doi:10.1145/1998196.1998228.
8. 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.
9. David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal homologous chains and applications to point cloud triangulations. *Discrete & computational geometry*, 68(4):1155–1174, December 2022. doi:10.1007/s00454-022-00432-6.
10. Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27(12):124003, 2011.
11. Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & computational geometry*, 45(4):737–759, June 2011. doi:10.1007/s00454-011-9344-x.
12. Tamal K Dey and Tao Hou. Updating barcodes and representatives for zigzag persistence. *arXiv preprint arXiv:2112.02352*, 2021.
13. Tamal K Dey and Tao Hou. Fast computation of zigzag persistence. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
14. Tamal K Dey, Tao Hou, and Dmitriy Morozov. A fast algorithm for computing zigzag representatives. *arXiv preprint arXiv:2410.20565*, 2024. doi:10.48550/arXiv.2410.20565.
15. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002. doi:10.1007/S00454-002-2885-2.
16. Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.
17. G. Henselman and R. Ghrist. Matroid Filtrations and Computational Persistent Homology. *ArXiv e-prints*, June 2016. arXiv:1606.00199.
18. Alan Hylton, Greg Henselman-Petrusek, Janche Sang, and Robert Short. Performance enhancement of a computational persistent homology package. In *2017 IEEE 36th international performance computing and communications conference (IPCCC)*, pages 1–8. IEEE, 2017. doi:10.1109/PCCC.2017.8280468.
19. Gil Kalai. Enumeration of q-acyclic simplicial complexes. *Israel Journal of Mathematics*, 45:337–351, 1983.

20 Michael Kerber, Donald R Sheehy, and Primoz Skraba. Persistent homology and nested dissection. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1245. SIAM, 2016. doi:10.1137/1.9781611974331.CH86.

21 Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the twenty-seventh Annual Symposium on Computational Geometry*, pages 216–225, 2011. doi:10.1145/1998196.1998229.

22 Dmitriy Morozov. Persistence algorithm takes cubic time in worst case. *BioGeometry News, Dept. Comput. Sci., Duke Univ*, 2, 2005.

23 Dmitriy Morozov and Primoz Skraba. Persistent (co)homology in matrix multiplication time, 2024. doi:10.48550/arXiv.2412.02591.

24 Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *Discrete & computational geometry*, 72(1):310–344, July 2024. doi:10.1007/s00454-023-00613-x.

25 Primoz Skraba, Gugan Thoppe, and D Yogeshwaran. Randomly weighted d-complexes: Minimal spanning acycles and persistence diagrams. *The Electronic Journal of Combinatorics*, pages P2–11, 2020.

26 Hubert Wagner, Chao Chen, and Erald Vuçini. Efficient computation of persistent homology for cubical data. In *Topological methods in data analysis and visualization II: theory, algorithms, and applications*, pages 91–106. Springer, 2011.

27 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.

Computing Betti Tables and Minimal Presentations of Zero-Dimensional Persistent Homology

Dmitriy Morozov
International Computer Science Institute, Berkeley, CA, USA
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Luis Scoccola
Centre de Recherches Mathématiques et Institut des sciences mathématiques, Laboratoire de combinatoire et d'informatique mathématique de l'Université du Québec à, Montréal, Canada
Université de Sherbrooke, Québec, Canada

Abstract

The Betti tables of a multigraded module encode the grades at which there is an algebraic change in the module. Multigraded modules show up in many areas of pure and applied mathematics, and in particular in topological data analysis, where they are known as persistence modules, and where their Betti tables describe the places at which the homology of filtered simplicial complexes changes. Although Betti tables of singly and bigraded modules are already being used in applications of topological data analysis, their computation in the bigraded case (which relies on an algorithm that is cubic in the size of the filtered simplicial complex) is a bottleneck when working with large datasets. We show that, in the special case of 0-dimensional homology (relevant for clustering and graph classification) Betti tables of bigraded modules can be computed in log-linear time. We also consider the problem of computing minimal presentations, and show that minimal presentations of 0-dimensional persistent homology can be computed in quadratic time, regardless of the grading poset.

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

Keywords and phrases Multiparameter persistence, Zero-dimensional homology, Minimal presentation, Betti table

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.69

Related Version *Full Version*: `https://arxiv.org/abs/2410.22242` [30]

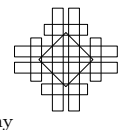
Funding *Dmitriy Morozov*: National Science Foundation, award DMS-2324632.
Luis Scoccola: National Science Foundation through grants CCF-2006661 and CAREER award DMS-1943758, while at Northeastern University; EPSRC grant "New Approaches to Data Science: Application Driven Topological Data Analysis", EP/R018472/1, while at University of Oxford.

Acknowledgements We thank the anonymous reviewers and the SoCG'25 program committee for helpful comments and for spotting an issue with Definition 24, and its corresponding fix.

1 Introduction

Betti tables and persistence. Betti tables are a classical descriptor of a multigraded modules [18, 29, 34], which encode the grades of the generators in a minimal projective resolution of the module (see, e.g., Figure 1 and Example 15). Informally, one can interpret the Betti tables of a graded module as recording the grades at which there is an algebraic change in the module. Graded modules have applications in a wide variety of areas of pure and applied mathematics, including topological data analysis, and more specifically, persistence theory [33, 6], where they are known as *persistence modules*, and where they are used to describe the varying topology of simplicial complexes and other spaces as they are filtered by one or more real parameters.

© Dmitriy Morozov and Luis Scoccola;
licensed under Creative Commons License CC-BY 4.0



Informally, *one-parameter persistence modules* correspond to $\mathbb{Z}$-graded $\Bbbk[x]$-modules, and can thus be classified up to isomorphism effectively. *Multiparameter persistence modules* [11, 6] correspond to $\mathbb{Z}^n$-graded $\Bbbk[x_1, \ldots, x_n]$-modules, and thus do not admit any reasonable classification up to isomorphism (formally, one is dealing with categories of wild representation type [31]; see [11, 2, 4] for manifestations of this phenomenon in persistence theory). For this reason, much of the research in multiparameter persistence is devoted to the study of incomplete descriptors of multiparameter persistence modules. Betti tables (also known as multigraded Betti numbers) provide one of the simplest such descriptors, and various properties of this descriptor from the point of view of persistence theory are well understood, including their effective computation [27, 25, 19, 3], their relationship to discrete Morse theory [22, 1, 21], their optimal transport Lipschitz-continuity with respect to perturbations [32], and their usage in supervised learning [28, 39].

Two-parameter persistent homology. The simplest case beyond the one-parameter case (i.e., the singly graded case) is the two-parameter case (i.e., the bigraded case). Here, one is usually given a finite simplicial complex K together with a function $f : K \longrightarrow \mathbb{R}^2$ mapping the simplices of K to $\mathbb{R}^2$, which is monotonic, i.e., such that $f(\sigma) \leq f(\tau)$ whenever $\sigma \subseteq \tau \in K$ (see Figure 1 for an example). By filtering K using f and taking homology in dimension $i \in \mathbb{N}$ with coefficients in a field $\Bbbk$, one obtains an $\mathbb{R}^2$-graded module, or equivalently, a functor $H_i(K, f; \Bbbk) : \mathbb{R}^2 \longrightarrow \mathbf{Vec}_{\Bbbk}$. Examples include geometric complexes of point clouds filtered by a function on data points, such as a density estimate [8, 12], and graphs representing, say, molecules or networks, filtered by two application-dependent quantities [15]. Applying homology to a bifiltered simplicial complex is justified by the fact that the output is automatically invariant under relabeling, meaning that any operation based on this module, such as computing its Betti tables, will result in a relabeling-invariant descriptor. In this setup, one of the main stability results of [32] implies that, for $f, g : K \longrightarrow \mathbb{R}^2$ any two monotonic functions, we have $\|\mu_f - \mu_g\|_1^{\mathsf{K}} \leq 2 \cdot \|f - g\|_1$, where $\| - \|_1^{\mathsf{K}}$ denotes the Kantorovich–Rubinstein norm between signed measures (also known as the 1-Wasserstein distance), and μ_f and μ_g are the signed measures on $\mathbb{R}^2$ obtained as the alternating sum of Betti tables of $H_i(K, f; \Bbbk)$ and $H_i(K, g; \Bbbk)$, respectively; see [28, Theorem 1] for details. The upshot is that the Betti tables of the homology of bifiltered simplicial complexes form a perturbation-stable, relabeling-invariant descriptor of bifiltered simplicial complexes.

The current standard algorithm for computing the Betti tables of $H_i(K, f; \Bbbk)$ in the two-parameter case is the Lesnick–Wright algorithm [27], which runs in time $O(|K|^3)$. Because of results such as those of [17], one does not expect to find algorithms with better worst-case time complexity than matrix-multiplication time, at least when $i \in \mathbb{N}$ is arbitrary. Current options to speed up practical computations include sparsifying the filtered complex before computing homology [42], as well as computational shortcuts that are known to significantly reduce computational time in practice [25, 3]. Nevertheless, computational cost is still a main bottleneck in real-world applications of persistence, limiting the size of the datasets on which it can be applied.

Zero-dimensional persistent homology. In many applications, persistent homology in dimension zero is all that is required, as it encodes information about the changes in connectivity of filtered simplicial complexes, making it useful for clustering [10, 9, 14, 8, 35, 38] and graph classification [41, 13, 24, 28, 23].

But, if one is only interested in 0-dimensional homology $H_0(K, f; \Bbbk)$, algorithms relying on linear algebra are usually far from the most efficient ones: For example, in the one-parameter case, the Betti tables of the 0-dimensional homology of an $\mathbb{R}$-filtered graph (G, f) can be

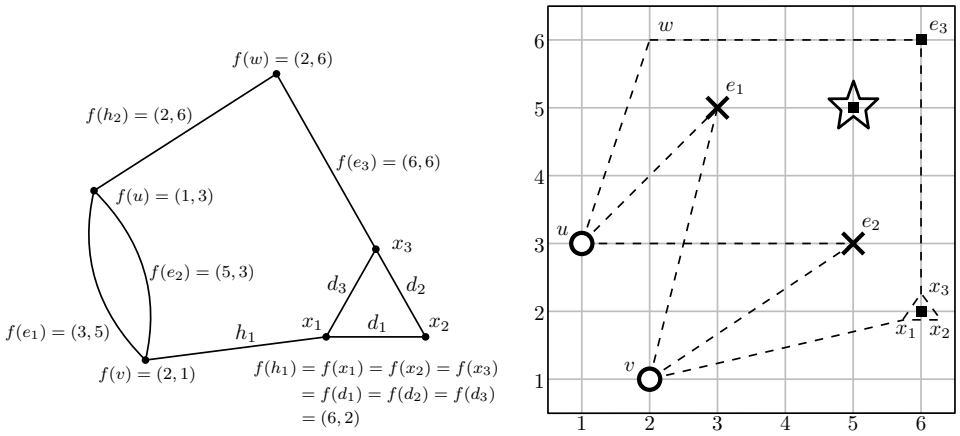


Figure 1 *Left.* A bifiltered graph (G, f) with vertex set $\{u, v, w, x_1, x_2, x_3\}$ and edge set $\{e_1, e_2, e_3, h_1, h_2, d_1, d_2, d_3\}$. *Right.* The bifiltered graph schematically mapped to $\mathbb{R}^2$, together with the Betti tables $\beta_0(H_0(G, f))$ (circles), $\beta_1(H_0(G, f))$ (crosses), $\beta_2(H_0(G, f))$ (stars), and $\beta_0(H_1(G, f))$ (squares).

computed in $O(|G| \log |G|)$ time by first sorting the simplices of G by their f-value, and then doing an ordered pass using a union-find data structure; in the language of barcodes, the Betti tables simply record the endpoints of the barcode, and the barcode can be computed using the elder rule (see [16, pp. 188] or [35, Algorithm 1]).

Contributions. The paper is concerned with the following question: What is the complexity of computing the Betti tables of graphs filtered by posets other than $\mathbb{R}$?

We introduce algorithms for the computation of a minimal set of generators and of a minimal presentation of 0-dimensional persistent homology indexed by an arbitrary poset.

▶ **Theorem A.** *Let $\mathcal{P}$ be any poset, and let (G, f) be a finite $\mathcal{P}$-filtered graph. Algorithm 2 computes the 0th Betti table of $H_0(G, f; \Bbbk)$ in $O(|G|)$ time. Algorithm 3 computes a minimal presentation (and hence the 0th and 1st Betti tables) of $H_0(G, f; \Bbbk)$ in $O(|G|^2)$ time.*

We also introduce a more efficient algorithm specialized to the two-parameter case, which computes all Betti tables, that is, 0th, 1st, and 2nd, by Hilbert's syzygy theorem.

▶ **Theorem B.** *Let (G, f) be a finite $\mathbb{R}^2$-filtered graph. Algorithm 4 computes minimal presentations and the Betti tables of $H_0(G, f; \Bbbk)$ and $H_1(G, f; \Bbbk)$ in $O(|G| \log |G|)$ time.*

Of note is the fact that Betti tables of 0-dimensional two-parameter persistent homology can be computed in log-linear time, as in the one-parameter case.

As another main contribution, we establish a connection between minimal presentations and connectivity properties of filtered graphs (Theorems 23 and 25), which allows us to abstract away the algebraic problem, and focus on a simpler combinatorial problem. The correctness of our algorithms is based on this. Another interesting consequence of these results is that, for arbitrary posets, the 0th and 1st Betti tables of 0-dimensional persistent homology are independent of the field of coefficients, while for the poset $\mathbb{R}^2$, all Betti tables of a filtered graph are independent of the field of coefficients.

Summary of approach and structure of the paper. The main body of the paper has three sections: one on background (Section 2), one on theoretical results (Section 3), and one on algorithms (Section 4). [30, Appendix A] contains proofs.

In order to describe and prove the correctness of our algorithms, we introduce, in the theory section, the notion of a *minimal filtered graph* (Definition 17). Informally, a minimal filtered graph is one whose vertices and edges induce a minimal presentation of its 0-dimensional persistent homology (see Theorem 23). A graph is not minimal if it has some edge that can be either contracted or deleted without changing its 0-dimensional persistent homology (for example, the graph of Figure 1 has both contractible and deletable edges, and is thus not minimal; see Examples 15 and 18). The idea is that, by contracting and deleting edges that do not change 0-dimensional persistent homology, one inevitably ends up with a minimal filtered graph, from which a minimal presentation can be easily extracted. Our main theoretical contributions (Theorems 23 and 25) make this idea precise by giving an explicit minimal presentation of H_0 of any minimal filtered graph, and an explicit minimal resolution of H_0 of any minimal $\mathbb{R}^2$-filtered graph. Our main algorithmic contributions are efficient algorithms for contracting and deleting edges as necessary. Algorithm 4 makes use of a dynamic tree data structure [40], which is the main ingredient that allows us to compute the Betti tables of bifilitered graphs in log-linear time; we give more details in Section 4.

Remark about multi-critical filtrations. The filtrations considered in this paper are 1-*critical*, meaning that each simplex (vertex or edge, in the case of graphs) appears at exactly one grade. In [30, Appendix B], we describe a simple preprocessing step to turn a finite multi-critical filtration by a lattice into a 1-critical filtration with the same 0-dimensional homology. In the two-parameter case, this construction does not change the input complexity, but for other lattices it may increase the input size. Note that the construction does not (necessarily) preserve 1-dimensional homology.

Related work. To the best of our knowledge, the only subcubic algorithm related to Algorithm 4 is that of [8], which, in particular, can be used to compute the Betti tables of H_0 of a function-Rips complex of a finite metric space X in time $O(|X|^2 \log |X|)$. When applied to a function-Rips complex, our Algorithm 4 has the same time complexity; however, our Algorithm 4 applies to arbitrary bifiltered graphs, while function-Rips complexes are very special (and do not include arbitrary filtered graphs). The inner workings of Algorithm 4 are also different from those of [8]: While we rely on dynamic trees, their algorithm relies on a dynamic minimum spanning tree, and, notably, on the fact that, in a function-Rips bifiltration, vertices are filtered exclusively by one of the two filtering functions (so that vertices are linearly ordered in the bifiltration).

The computation of minimal presentations of $\mathbb{R}^n$-filtered complexes is studied in [5]. Since they consider homology in all dimensions, their complexity is significantly worse than the quadratic complexity of Algorithm 3, which only applies to 0-dimensional homology.

Part of the contributions in this paper could be rephrased using the language of discrete Morse theory for multiparameter filtrations [1, 37, 7], specifically, Algorithms 1 and 2 are essentially computing acyclic matchings which respect the filtration. However, this point of view requires extra background, and, to the best of our understanding, cannot be used to describe the more interesting Algorithm 4.

2 Background

As is common in persistence theory, we assume familiarity with very basic notions of category theory, specifically, that of a category and of a functor. We let $\Bbbk$ denote a field, $\mathbf{Vec}_\Bbbk$ denote the category of $\Bbbk$-vector spaces, and $\mathbf{Set}$ denote the category of sets. When the field $\Bbbk$ plays no role, we may denote $\mathbf{Vec}_\Bbbk$ simply by $\mathbf{Vec}$. Proofs can be found in [30, Appendix A.1].

Graphs and filtered graphs. A *graph* $G = (V, E, \partial)$ consists of finite sets V and E, and a function $\partial : E \longrightarrow V \times V$. We refer to the elements of V as *vertices*, typically denoted $v, w, x, y \in V$, and to the elements of E as *edges*, typically denoted $e, d, h \in E$. If $\partial e = (v, w)$, we write $e_0 = v$ and $e_1 = w$. The *size* of a graph G is $|G| = |V| + |E|$.

A *subgraph* of a graph $G = (V, E, \partial)$ is a graph $G' = (V', E', \partial')$, where $V' \subseteq V$, $E' \subseteq E$, and such that $\partial' = \partial|_{E'}$ takes values in $V' \times V' \subseteq V \times V$. If G' is a subgraph of G, we write $G' \subseteq G$.

If $E' \subseteq E$ is a set of edges of G, we let $G \setminus E'$ be the subgraph of G with the same vertices and $E \setminus E'$ as set of edges.

Let $\mathcal{P}$ be a poset. A $\mathcal{P}$-*filtered graph* (G, f^V, f^E) consists of a graph G and functions $f^V : V \longrightarrow \mathcal{P}$ and $f^E : E \longrightarrow \mathcal{P}$ such that $f^V(e_0) \leq f^E(e)$ and $f^V(e_1) \leq f^E(e)$ for all $e \in E$. When there is no risk of confusion, we refer to $\mathcal{P}$-filtered graphs simply as filtered graph, and denote both f^V and f^E by f and the filtered graph (G, f^V, f^E) by (G, f).

If (G, f) is a $\mathcal{P}$-filtered graph and $r \in \mathcal{P}$, we let $(G, f)_r$ be the subgraph of G with vertices $\{v \in V : f(v) \leq r\}$ and edges $\{e \in E : f(e) \leq r\}$.

Persistence modules and persistent sets. Let $\mathcal{P}$ be a poset. A $\mathcal{P}$-*persistence module* is a functor $\mathcal{P} \longrightarrow \mathbf{Vec}$, where $\mathcal{P}$ is the category associated with $\mathcal{P}$. Explicitly, a $\mathcal{P}$-persistence module $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ consists of the following:

- for each $r \in \mathcal{P}$, a vector space $M(r)$;
- for each pair $r \leq s \in \mathcal{P}$, a linear morphism $\varphi^M_{r,s} : M(r) \longrightarrow M(s)$; such that
- for all $r \in \mathcal{P}$, the linear morphism $\varphi^M_{r,r} : M(r) \longrightarrow M(r)$ is the identity;
- for all $r \leq s \leq t \in \mathcal{P}$, we have $\varphi^M_{s,t} \circ \varphi^M_{r,s} = \varphi^M_{r,t} : M(r) \longrightarrow M(t)$.

When there is no risk of confusion, we may refer to a $\mathcal{P}$-persistence module as a persistence module. An *n-parameter persistence module* ($n \geq 1 \in \mathbb{N}$) is an $\mathbb{R}^n$-persistence module.

A *morphism* $g : M \longrightarrow N$ between persistence modules is a natural transformation between functors, that is, a family of linear maps $\{g_r : M(r) \longrightarrow N(r)\}_{r \in \mathcal{P}}$ with the property that $\varphi^N_{r,s} \circ g_r = g_s \circ \varphi^M_{r,s} : M(r) \longrightarrow N(s)$, for all $r \leq s \in \mathcal{P}$. Such a morphism is an *isomorphism* if $g_r : M(r) \longrightarrow N(r)$ is an isomorphism of vector spaces for all $r \in \mathcal{P}$.

If $M, N : \mathcal{P} \longrightarrow \mathbf{Vec}$ are persistence modules, their *direct sum*, denoted $M \oplus N : \mathcal{P} \longrightarrow \mathbf{Vec}$, is the persistence module with $(M \oplus N)(r) \coloneqq M(r) \oplus N(r)$ and with $\varphi^{M \oplus N}_{r,s} \coloneqq \varphi^M_{r,s} \oplus \varphi^N_{r,s} : M(r) \oplus N(r) \longrightarrow M(s) \oplus N(s)$, for all $r \leq s \in \mathcal{P}$.

Similarly, a $\mathcal{P}$-*persistent set* is a functor $\mathcal{P} \longrightarrow \mathbf{Set}$. The concepts of *n-parameter persistent set*, and of *morphism* and *isomorphism* between persistent sets are defined analogously.

Persistent homology and connected components of filtered graphs. If $S \in \mathbf{Set}$, we let $\langle S \rangle_{\Bbbk} \in \mathbf{Vec}$ denote the free vector space generated by S; this defines a functor $\langle - \rangle_{\Bbbk} : \mathbf{Set} \longrightarrow \mathbf{Vec}_{\Bbbk}$. Let $G = (V, E, \partial)$ be a graph. Consider the $\Bbbk$-linear map

$$\begin{aligned} \langle E \rangle_{\Bbbk} &\xrightarrow{d} \langle V \rangle_{\Bbbk} \\ e &\longmapsto e_1 - e_0 \end{aligned} \tag{1}$$

The 0-*dimensional homology* of G, denoted $H_0(G; \Bbbk)$, is the $\Bbbk$-vector space $\operatorname{coker}(d)$, and the 1-*dimensional homology* of G, denoted $H_1(G; \Bbbk)$, is the $\Bbbk$-vector space $\ker(d)$. In particular, every vertex $v \in V$ gives an element $[v] \in H_0(G; \Bbbk)$. When there is no risk of confusion, we omit the field $\Bbbk$ and write $H_i(G)$ instead of $H_i(G; \Bbbk)$.

Homology is functorial, in the following sense. If $G' = (V', E', \partial')$ is a subgraph of G, we have a commutative square

$$\begin{array}{ccc} \langle E' \rangle_{\Bbbk} & \xrightarrow{d'} & \langle V' \rangle_{\Bbbk} \\ \downarrow & & \downarrow \\ \langle E \rangle_{\Bbbk} & \xrightarrow{d} & \langle V \rangle_{\Bbbk} \end{array}$$

induced by the inclusions $V' \subseteq V$ and $E' \subseteq E$. This induces linear maps $H_0(G') \longrightarrow H_0(G)$ and $H_1(G') \longrightarrow H_1(G)$. Moreover, the morphism $H_\bullet(G'') \longrightarrow H_\bullet(G)$ induced by a subgraph $G'' \subseteq G' \subseteq G$ is equal to the composite $H_\bullet(G'') \longrightarrow H_\bullet(G') \longrightarrow H_\bullet(G)$.

If (G, f) is a $\mathcal{P}$-filtered graph, and $i \in \{0, 1\}$, we get a $\mathcal{P}$-persistence module $H_i(G, f) : \mathcal{P} \longrightarrow \mathbf{Vec}$, with $H_i(G, f)(r) = H_i((G, f)_r)$, and with structure morphism $H_i(G, f)(r) \longrightarrow H_i(G, f)(s)$ for $r \leq s \in \mathcal{P}$ induced by the inclusion of graphs $(G, f)_r \subseteq (G, f)_s$.

The set of *connected components* of G, denoted $\pi_0(G)$, is the quotient of V by the equivalence relation $\sim$ where $v \sim w \in V$ if and only if there exists a path in G between v and w. If $v \in V$, we let $[v] \in \pi_0(G)$ denote its connected component, so that $[v] = [w] \in \pi_0(G)$ if and only if v and w belong to the same connected component.

The set of connected components is also functorial with respect to inclusions $G' \subseteq G$, since $[v] = [w] \in \pi_0(G')$ implies $[v] = [w] \in \pi_0(G)$. In particular, if (G, f) is a $\mathcal{P}$-filtered graph, we get a $\mathcal{P}$-persistent set $\pi_0(G, f) : \mathcal{P} \longrightarrow \mathbf{Set}$, with $\pi_0(G, f)(r) = \pi_0((G, f)_r)$, and with the structure morphism $\pi_0(G, f)(r) \longrightarrow \pi_0(G, f)(s)$ for $r \leq s \in \mathcal{P}$ induced by the inclusion of graphs $(G, f)_r \subseteq (G, f)_s$.

The following is straightforward to check.

▶ **Lemma 1.** *If G is a graph, then the map $\langle \pi_0(G) \rangle_{\Bbbk} \longrightarrow H_0(G)$ sending a basis element $[v] \in \pi_0(G)$ to $[v] \in H_0(G)$ is well-defined and an isomorphism of vector spaces. In particular, if (G, f) is a $\mathcal{P}$-filtered graph, composing the persistent set $\pi_0(G, f) : \mathcal{P} \longrightarrow \mathbf{Set}$ with the free vector space functor $\langle - \rangle_{\Bbbk} : \mathbf{Set} \longrightarrow \mathbf{Vec}$ yields a persistence module isomorphic to $H_0(G, f) : \mathcal{P} \longrightarrow \mathbf{Vec}$.* ⌟

Projective persistence modules. Given $r \in \mathcal{P}$, let $\mathsf{P}_r : \mathcal{P} \longrightarrow \mathbf{Vec}$ be the persistence module with $\mathsf{P}_r(s) = \Bbbk$ if $r \leq s$ and $\mathsf{P}_r(s) = 0$ if $r \not\leq s$, with all structure morphisms $\Bbbk \longrightarrow \Bbbk$ being the identity. Equivalently, one can define P_r to be $H_0(\{x\}, \emptyset, \partial, f)$, with $f(x) = r$.

▶ **Notation 2.** *If I is a finite set and $f : I \longrightarrow \mathcal{P}$ is any function, we can consider the direct sum $M = \bigoplus_{i \in I} \mathsf{P}_{f(i)} : \mathcal{P} \longrightarrow \mathbf{Vec}$. When we need to work with elements of such a direct sum, we distinguish summands by writing $M = \bigoplus_{i \in I} \left(\mathsf{P}_{f(i)} \cdot \{i\}\right)$, so that $\left(\mathsf{P}_{f(i)} \cdot \{i\}\right)(r) = 0$ if $r \not\geq f(i)$ and $\left(\mathsf{P}_{f(i)} \cdot \{i\}\right)(r)$ is equal to the free vector space generated by $\{i\}$, for $r \geq f(i)$.*

▶ **Definition 3.** *A persistence module $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ is* projective of finite rank *if there exists a function $\beta^M : \mathcal{P} \longrightarrow \mathbb{N}$ of finite support such that $M \cong \bigoplus_{r \in \mathcal{P}^n} \mathsf{P}_r^{\beta^M(r)}$.*

Note that, drawing inspiration from commutative algebra, projective persistence modules are sometimes also called *free*. The following result justifies the term projective used in Definition 3; see, e.g., [36, Section 3.1] for the usual notion of projective module.

▶ **Lemma 4.** *Let $g : M \longrightarrow N$ be a surjection between $\mathcal{P}$-persistence modules, and let $h : P \longrightarrow N$ with P projective of finite rank. There exists a morphism $h' : P \longrightarrow M$ such that $g \circ h' = h$.*

Resolutions, presentations, and Betti tables. The next result is standard; see, e.g., [26, Proposition 6.24].

▶ **Lemma 5.** *If M is projective of finite rank, then there exists exactly one function (necessarily of finite support) $\beta^M : \mathcal{P} \longrightarrow \mathbb{N}$ such that $M \cong \bigoplus_{r\in\mathcal{P}} \mathsf{P}_r^{\beta^M(r)}$.* ⌟

▶ **Definition 6.** *The* Betti table *of a persistence module $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ that is projective of finite rank is the function $\beta^M : \mathcal{P} \longrightarrow \mathbb{N}$ of Lemma 5.*

The following notation is sometimes convenient.

▶ **Notation 7.** *If $r \in \mathcal{P}$, we let $\delta_r : \mathcal{P} \longrightarrow \mathbb{N}$ be the function defined by $\delta_r(r) = 1$ and $\delta_r(s) = 0$ if $s \neq r$. In particular, $\delta_r = \beta^{\mathsf{P}_r} : \mathcal{P} \longrightarrow \mathbb{N}$.*

▶ **Definition 8.** *Let $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ be a $\mathcal{P}$-persistence module. A* finite projective cover *of M is a surjective morphism $P \longrightarrow M$ where P is projective of finite rank, and $\sum_{r\in\mathcal{P}} \beta^P(r) \in \mathbb{N}$ is minimal.*

▶ **Definition 9.** *Let $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ be a $\mathcal{P}$-persistence module, and let $k \in \mathbb{N}$. A* finite projective k-resolution *(resp.* minimal finite projective k-resolution*) of M, denoted $C_\bullet \longrightarrow M$, is a sequence of morphisms $C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} M$ satisfying*

- *$C_0 \longrightarrow M$ is surjective (resp. a projective cover);*
- *$\partial_i \circ \partial_{i+1} = 0$, for every $0 \leq i \leq k-1$ (so that ∂_{i+1} factors through $\ker \partial_i$);*
- *$C_{i+1} \xrightarrow{\partial_{i+1}} \ker(\partial_i)$ is surjective (resp. a projective cover), for every $0 \leq i \leq k-1$;*
- *C_i is projective of finite rank, for every $0 \leq i \leq k$.*

In particular, a minimal finite projective 0-resolution is simply a projective cover.

▶ **Notation 10.** *Since we only consider finite resolutions, we omit the word "finite" and simply say projective k-resolution. A* (minimal) projective presentation *is a* (minimal) projective 1-resolution*.*

▶ **Definition 11.** *Let $k \in \mathbb{N}$. A persistence module $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ is* finitely k-resolvable *if it admits a finite projective k-resolution.*

▶ **Definition 12.** *Let $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ be finitely k-resolvable and let $0 \leq i \leq k$. The* ith Betti table *of M is the function $\beta_i^M : \mathcal{P} \longrightarrow \mathbb{N}$ (necessarily of finite support) defined as $\beta_i^M := \beta^{C_i}$, where $C_\bullet \longrightarrow M$ is a minimal k-resolution of M.*

▶ **Notation 13.** *When convenient, we write $\beta_i(M)$ instead of β_i^M for the Betti tables of a persistence module M.*

The Betti tables of M are also sometimes called the *(multigraded) Betti numbers* of M. The Betti tables of M, as defined in Definition 12, are independent of the choice of minimal presentation or resolution, thanks to the following standard result.

▶ **Lemma 14.** *Let $k \in \mathbb{N}$. If $M : \mathcal{P} \longrightarrow \mathbf{Vec}$ is finitely k-resolvable, then it admits a minimal projective k-resolution $C_\bullet \longrightarrow M$. Moreover, any other projective k-resolution $C'_\bullet \longrightarrow M$ has the property that $\beta^{C_i} \leq \beta^{C'_i}$ for all $0 \leq i \leq k$.*

▶ **Example 15.** Consider the graph $G = (V, E, \partial)$ with $V = \{x, y\}$, $E = \{a, b\}$, and $\partial(a) = \partial(b) = (x, y)$. Consider the filtration $f : G \longrightarrow \mathbb{R}^2$ with $f(x) = f(y) = (0, 0)$, $f(a) = (0, 1)$, and $f(b) = (1, 0)$. Then, a minimal resolution of $H_0(G, f)$ is given by

$$0 \longrightarrow \mathsf{P}_{(1,1)} \cdot \{\alpha\} \xrightarrow{\partial_2} \mathsf{P}_{(1,0)} \cdot \{a\} \oplus \mathsf{P}_{(0,1)} \cdot \{b\} \xrightarrow{\partial_1} \mathsf{P}_{(0,0)} \cdot \{x\} \oplus \mathsf{P}_{(0,0)} \cdot \{y\} \xrightarrow{\partial_0} H_0(G, f),$$

where $\partial_0(\{x\}) = [x]$, $\partial_0(\{y\}) = [y]$, $\partial_1(\{a\}) = \partial_1(\{b\}) = \{y\} - \{x\}$, and $\partial_2(\{\alpha\}) = \{a\} - \{b\}$. In particular, $\beta_0(H_0(G,f)) = \delta_{(0,0)} + \delta_{(0,0)}$, $\beta_1(H_0(G,f)) = \delta_{(1,0)} + \delta_{(0,1)}$, and $\beta_2(H_0(G,f)) = \delta_{(1,1)}$. This can be easily checked by hand, but it also follows from Theorems 23 and 25, since (G,f) is a minimal filtered graph, in the sense of Definition 17, which we give in the next section.

3 Theory

We start with a simple, standard result which says that a projective presentation of 0-dimensional homology is given by using the grades of vertices as generators, and the grades of edges as relations. Proofs can be found in [30, Appendix A.2].

▶ **Lemma 16.** *Let $\mathcal{P}$ be a poset, let (G,f) be a $\mathcal{P}$-filtered graph, and consider the following morphism of projective modules*

$$\begin{array}{ccc} \bigoplus_{e\in E} \mathsf{P}_{f(e)} \cdot \{e\} & \xrightarrow{\partial_1^{(G,f)}} & \bigoplus_{v\in V} \mathsf{P}_{f(v)} \cdot \{v\} \\ \{e\} & \longmapsto & \{e_1\} - \{e_0\} \end{array}$$

Then $H_0(G,f) \cong \operatorname{coker}\left(\partial_1^{(G,f)}\right)$ and $H_1(G,f) \cong \ker\left(\partial_1^{(G,f)}\right)$. In particular, $H_0(G,f)$ is a finitely presentable persistence module.

The point of minimal filtered graphs, which we now introduce, is that they make the presentation in Lemma 16 minimal (see Theorem 23).

▶ **Definition 17.** *Let (V,E,∂,f) be a filtered graph and let $e \in E$.*
- *The edge e is* collapsible *if $e_0 \neq e_1$ and $f(e) = f(e_i)$ for some $i \in \{0,1\}$.*
- *The edge e is* deletable *if $[e_0] = [e_1] \in \pi_0(V, E\setminus\{e\}, \partial, f)(f(e))$.*

A filtered graph (G,f) is
- vertex-minimal *if it does not contain any collapsible edges.*
- edge-minimal *if it does not contain any deletable edges.*
- minimal *if it is vertex-minimal and edge-minimal.*

▶ **Example 18.** The graph of Example 15 is minimal, as neither of the two edges is collapsible or deletable. On the other hand, the graph of Figure 1 is not minimal since h_1, h_2, d_1, d_2, and d_3 are collapsible, and e_3 is deletable.

As their name suggests, collapsible and deletable edges can be collapsed or deleted:

▶ **Construction 19.** *Let $(G,f) = (V,E,\partial,f)$ be a filtered graph.*
- *If $e \in E$ is collapsible, let $i \in \{0,1\}$ be such that $f(e) = f(e_i)$. Define the* simple collapse *$G \downarrow e \coloneqq (V\setminus\{e_i\}, E\setminus\{e\}, \partial')$, where $\partial' = (\varphi\times\varphi)\circ\partial$ and $\varphi : V \longrightarrow V$ is given by $\varphi(v) = v$ if $v \neq e_i$ and $\varphi(v) = e_{1-i}$ if $v = e_i$.*
- *If $e \in E$ is any edge, define the* simple deletion *$G \setminus e \coloneqq (V, E\setminus\{e\}, \partial)$.*

We now study the effect of collapsing collapsible edges, and of deleting deletable edges. We start with a useful observation, whose proof is straightforward.

▶ **Lemma 20.** *Let (G,f) be a filtered graph and let e and d be two different edges of G.*
1. *Assume that e is collapsible. If d is not collapsible (resp. deletable) in (G,f), then d is not collapsible (resp. deletable) in $(G\downarrow e, f)$.*
2. *If d is not collapsible (resp. deletable) in (G,f), then d is not collapsible (resp. deletable) in $(G\setminus e, f)$.* ⌟

▶ **Lemma 21.** *If $(G, f) = (V, E, \partial, f)$ is a filtered graph and $e \in E$ is collapsible, then $H_i(G, f) \cong H_i(G \downarrow e, f)$ for $i \in \{0, 1\}$.*

▶ **Lemma 22.** *If $(G, f) = (V, E, \partial, f)$ is a filtered graph and $e \in E$ is deletable, then $H_0(G, f) \cong H_0(G \setminus e, f)$ and $H_1(G, f) \cong H_1(G \setminus e, f) \oplus \mathsf{P}_{f(e)}$.*

Next is a construction of a minimal presentation of H_0 of a minimal filtered graph.

▶ **Theorem 23.** *Let $\mathcal{P}$ be a poset, let (G, f) is a $\mathcal{P}$-filtered graph, and let $\partial_1^{(G,f)} : C_1 \longrightarrow C_0$ be the morphism of Lemma 16.*

1. *If (G, f) is vertex-minimal, then the map $C_0 \longrightarrow \operatorname{coker}(\partial_1^{(G,f)})$ is a projective cover. In this case, $\beta_0\big(H_0(G, f)\big) = \sum_{v \in V} \delta_{f(v)}$.*
2. *If (G, f) is a minimal filtered graph, then $C_1 \xrightarrow{\partial_1^{(G,f)}} C_0 \longrightarrow \operatorname{coker}(\partial_1^{(G,f)})$ is a minimal presentation. In this case, we also have $\beta_1\big(H_0(G, f)\big) = \sum_{e \in E} \delta_{f(e)}$.*

We now focus on the case of $\mathbb{R}^2$-filtered graphs. If (G, f) is an $\mathbb{R}^2$-filtered graph, let $f_{\mathbf{x}}, f_{\mathbf{y}} : G \longrightarrow \mathbb{R}$ denote the first and second coordinates of f, respectively.

▶ **Definition 24.** *Let (V, E, ∂, f) be a minimal $\mathbb{R}^2$-filtered graph, and let $\prec$ be a total order on the set of edges E that refines the lexicographic order induced by f (i.e., $e \prec e'$ implies that $f_{\mathbf{x}}(e) < f_{\mathbf{x}}(e')$ or $f_{\mathbf{x}}(e) = f_{\mathbf{x}}(e')$ and $f_{\mathbf{y}}(e) \leq f_{\mathbf{y}}(e')$). An edge $d \in E$ is* cycle-creating *with respect to $\prec$ if $[d_0] = [d_1] \in \pi_0(V, \{e \in E : e \prec d\}, \partial)$.*

Thus, $d \in E$ is cycle-creating if there exists a list of edges $e^1, \ldots, e^k \in E$ with $e^i \prec d$ for $1 \leq i \leq k$, and a list of signs $s^1, \ldots, s^k \in \{+, -\}$, such that $s^1 e^1, \ldots, s^k e^k$ is a directed path from d_0 to d_1. Such a path $w = (s^\bullet, e^\bullet)$ is called a *witness* for d, and we denote $f(w) \coloneqq \big(f_{\mathbf{x}}(d), \max_{1 \leq i \leq k} f_{\mathbf{y}}(e^i)\big)$. A *minimal witness* w of a cycle-creating edge d is one for which $f_{\mathbf{y}}(w) \in \mathbb{R}$ is as small as possible.

Since $\prec$ refines the lexicographic order, we have $f_{\mathbf{x}}(e^i) \leq f_{\mathbf{x}}(d)$ for all i; thus, if $\max_{1 \leq i \leq k} f_{\mathbf{y}}(e^i) \leq f_{\mathbf{y}}(d)$, we would have that $[d_0] = [d_1] \in \pi_0(V, E \setminus \{d\}, \partial, f)(f(d))$, which contradicts the fact that the filtered graph is minimal. This means that if w is a witness for d, then $\max_{1 \leq i \leq k} f_{\mathbf{y}}(e^i) > f_{\mathbf{y}}(d)$, and thus $f(d) < f(w)$.

▶ **Theorem 25.** *Let $(G, f) = (V, E, \partial, f)$ be a minimal $\mathbb{R}^2$-filtered graph, and $\prec$ a total order on E refining the lexicographic order. For each $d \in E$ cycle-creating, let $w_d = (s_d^\bullet, e_d^\bullet)$ be a minimal witness wrt to $\prec$. The kernel of the morphism $\partial_1^{(G,f)}$ of Lemma 16 is given by:*

$$\begin{array}{ccc} \bigoplus_{\substack{d \in E \\ \text{cycle-creating}}} \mathsf{P}_{f(w_d)} \cdot \{d\} & \xrightarrow{\kappa^{(G,f)}} & \bigoplus_{e \in E} \mathsf{P}_{f(e)} \cdot \{e\} \\ \{d\} & \longmapsto & \{d\} - \big(s_d^1 \{e_d^1\} + \cdots + s_d^k \{e_d^k\}\big). \end{array}$$

It follows that

- $\beta_0(H_0(G, f)) = \sum_{v \in V} \delta_{f(v)}$;
- $\beta_1(H_0(G, f)) = \sum_{e \in E} \delta_{f(e)}$;
- $\beta_0(H_1(G, f)) = \beta_2(H_0(G, f)) = \sum_{d \in E, \text{cycle-creating}} \delta_{f(w_d)}$;
- $\beta_i(H_0(G, f)) = 0$ *for* $i \geq 3$, *and* $\beta_i(H_1(G, f)) = 0$ *for* $i \geq 1$.

4 Algorithms

Outline. We first introduce some technical notions and overview the algorithms.

Two $\mathcal{P}$-filtered graph (G, f) and (G', f') are *homology-equivalent* (over $\Bbbk$) if $H_0(G, f; \Bbbk) \cong H_0(G', f'; \Bbbk)$ and $H_1(G, f; \Bbbk) \cong H_1(G', f'; \Bbbk)$.

Algorithm 2 reduces the input filtered graph to a vertex-minimal filtered graph by collapsing edges; it relies on Algorithm 1, which collapses local collapsible edges. A *local collapsible edge* of (G, f) is an edge e such that $e_0 \neq e_1$ and $f(e) = f(e_0) = f(e_1)$. Algorithm 2 then identifies *minimal vertices*, that is, vertices with no adjacent collapsible edge decreasing the grade, and runs a depth-first search from each of these to build and collapse a tree of collapsible edges.

Algorithm 3 first calls Algorithm 2 to perform all collapses and then deletes deletable edges until there are no more deletable edges. It then builds the presentation of Lemma 16.

Algorithm 4 first calls Algorithm 2 to perform all collapses, and then goes through all edges, with respect to the lex order on their grades, and identifies deletable edges, non-deletable edges, and cycle-creating edges.

Dynamic dendrograms. Since this is used in Algorithm 4, we now introduce the dynamic dendrogram data structure, which, informally, represents a dendrogram where elements merge as time goes from $-\infty$ to ∞, and is dynamic in that one is allowed to change the dendrogram by making two elements merge earlier.

▶ **Definition 26.** *Let $(G, f) = (V, E, \partial, f)$ be a $[-\infty, \infty)$-filtered graph. A* dynamic dendrogram *D for (G, f) is a data structure supporting the following operations:*

- *If $v, w \in V$, the operation $D.\mathsf{time_of_merge}(v, w)$ returns the smallest $r \in [-\infty, \infty)$ such that $[v] = [w] \in \pi_0(G, f)(r)$, or ∞ if $[v] \neq [w] \in \pi_0(G, f)(r)$ for all $r \in [-\infty, \infty)$.*
- *If $v, w \in V$ and $t \in [-\infty, \infty)$, the operation $D.\mathsf{merge_at_time}(v, w, t)$ modifies the dendrogram so that it is a dynamic dendrogram for $(V, E \sqcup \{e\}, \partial', f')$, with $f'|_E = f$, $\partial'|_E = \partial$, $f'(e) = t$, and $\partial'(e) = (v, w)$.*

A dynamic dendrogram D can easily and efficiently be implemented using a *mergeable tree* T in the sense of [20]. These trees store heap-ordered forests, i.e., a collection of rooted labeled trees, where the labels decrease (or in our case increase) along paths to the root. The data structures support a wealth of operations for dynamic updates. We need only three of them: $\mathsf{insert}(v, t)$ inserts node v with label t into the forest; $\mathsf{nca}(v, w)$ finds the nearest common ancestor of two nodes v and w; $\mathsf{merge}(v, w)$ merges the paths of v and w to their root(s) while preserving the heap order. We use these operations to implement dynamic dendrograms as follows:

- To implement $D.\mathsf{time_of_merge}(v, w)$, return the label of $T.\mathsf{nca}(v, w)$.
- To implement $D.\mathsf{merge_at_time}(v, w, t)$, let h be a new vertex not already in the mergeable tree T, do $T.\mathsf{insert}(h, t)$, then $T.\mathsf{merge}(v, h)$, and $T.\mathsf{merge}(w, h)$.

Presentation matrices. In the algorithms, to represent a Betti table $\beta_i(M)$ we use a list of elements of the indexing poset. If we have represented the 0th and 1st Betti tables of M with lists β_0 and β_1, we represent a minimal presentation for M with a sparse matrix using coordinate format, that is, with a list of triples (i, j, v) representing the fact that v is the entry at row i and column j, and where i represents the ith element of β_0 and j represents the jth element of β_1.

Algorithm 1 Collapse local collapsible edges.

Input: Filtered graph $(G, f) = (V, E, \partial, f)$
Output: Filtered graph (G', f) homology-equivalent to (G, f) and without local collapsible edges

Initialize dictionary φ with identity map, $\varphi[v] = v$ for $v \in V$
Initialize empty set visited
Initialize $E' \leftarrow E$
for $v \in V$ **do** ▷ Run depth-first search from v on local edges
  Initialize stack with $(v, \emptyset, v)$
  while stack *is not empty* **do**
    $(v, e, u) \leftarrow$ stack.**pop**()
    if $u \notin$ visited **then**
      add u to visited
      if $e \neq \emptyset$ **then**
        $\varphi[u] \leftarrow v$
        remove e from E'
      for $e \in E$ *with* $\{e_0, e_1\} = \{u, x\}$ *and* $f(e) = f(x) = f(u)$ **do** ▷ local collapsible
        push (v, e, x) onto stack
$V' \leftarrow \{v \in V : \varphi[v] = v\}$
return $(G', f) = (V', E', (\varphi \times \varphi) \circ \partial, f)$

Algorithm 2 Collapse to vertex-minimal filtered graph.

Input: Filtered graph $(G, f) = (V, E, \partial, f)$
Output: Vertex-minimal filtered graph (G', f) homology-equivalent to (G, f), and $\beta_0(H_0(G, f))$

$(G_i, f) = (V_i, E_i, \partial_i, f) \leftarrow$ Algorithm 1(G, f)
Initialize dictionary φ with identity map, $\varphi[v] = v$ for $v \in V_i$
Initialize empty set visited
Initialize $E' \leftarrow E_i$
for v *in* V_i **do** ▷ Run depth-first search from minimal vertices
  if $\exists$ *edge* $e \in E_i$ *with* $\{e_0, e_1\} = \{u, v\}$ *and* $f(e) = f(v) > f(u)$ **then** ▷ v is not minimal
    continue
  Initialize stack with $(v, \emptyset, v)$
  while stack *is not empty* **do**
    $v, e, u \leftarrow$ stack.**pop**()
    if $u \notin$ visited **then**
      add u to visited
      if $e \neq \emptyset$ **then**
        $\varphi[u] \leftarrow v$
        remove e from E'
      for *every edge* $e \in E_i$ *with* $\{e_0, e_1\} = \{u, x\}$ *with* $f(e) = f(x) > f(u)$ **do**
        push (v, e, x) onto stack
$V' \leftarrow \{v \in V_i : \varphi[v] = v\}$
return $(G', f) = (V', E', (\varphi \times \varphi) \circ \partial_i, f)$ and $\beta_0 = [f(v) : v \in V']$

Algorithm 3 Minimal presentation of $\mathcal{P}$-filtered graph.

Input: Filtered graph $(G, f) = (V, E, \partial, f)$
Output: Minimal presentation of $H_0(G, f; \Bbbk)$

Initialize β_0, β_1 with empty lists
Initialize empty sparse matrix $\mathcal{M}$ and empty dictionary row_idx
$(V', E', \partial', f) \leftarrow$ Algorithm 2(G, f)
for $e \in E'$ **do** ▷ Check each edge, and delete it if it is deletable
  Define set of vertices $V_e \leftarrow \{v \in V' : f(v) \leq f(e)\}$
  Define set of edges $E_e \leftarrow \{d \in E' : f(d) \leq f(e)\} \setminus e$
  Run breadth-first search on (V_e, E_e, ∂') starting from e_0
  if e_1 *is reachable from* e_0 **then** ▷ $[e_0] = [e_1] \in \pi_0(V_e, E_e, \partial')$ so e is deletable
    $E' \leftarrow E' \setminus \{e\}$
for $v \in V'$ **do**
  β_0.append$(f(v))$
  row_idx$[v] \leftarrow |\beta_0|$
for $e \in E'$ **do** ▷ The morphism $\partial_1^{(V', E', \partial')}$ of Lemma 16
  β_1.append$(f(e))$
  $\mathcal{M}$.append(row_idx$[e_0]$, $|\beta_1|$, -1)
  $\mathcal{M}$.append(row_idx$[e_1]$, $|\beta_1|$, 1)
return $\beta_0, \beta_1, \mathcal{M}$

Algorithm 4 Betti tables and minimal presentation of $\mathbb{R}^2$-filtered graph.

Input: Filtered graph $(G, f) = (V, E, \partial, f)$
Output: Betti tables $\beta_0, \beta_1, \beta_2$ and minimal presentation of $H_0(G, f; \Bbbk)$, and $\beta_0^1 \coloneqq \beta_0(H_1(G, f; \Bbbk))$

Initialize $\beta_0, \beta_1, \beta_2, \beta_0^1$ with empty lists
Initialize empty sparse matrix $\mathcal{M}$ and empty dictionary row_idx
$(V', E', \partial', f) \leftarrow$ Algorithm 2(G, f)
Let D be a dynamic dendrogram on $(V', \emptyset, \partial, g)$, with $g(v) = -\infty$ for all $v \in V'$
for $(x, y) \in \mathbb{R}^2$ *s.t.* $f^{-1}(x, y) \neq \emptyset$ *in lex order* **do** ▷ Visit grades lexicographically
  for $v \in V'$ *with* $f(v) = (x, y)$ **do** ▷ All vertices belong to the projective cover
    β_0.append$(\,(x, y)\,)$
    row_idx$[v] \leftarrow |\beta_0|$
  for $e \in E'$ *with* $f(e) = (x, y)$ **do**
    $s \leftarrow D$.time_of_merge(e_0, e_1)
    if $s \leq y$ **then** ▷ The edge is deletable, so it only affects H_1
      β_0^1.append$(\,(x, y)\,)$
    else ▷ Edge is not deletable, so belongs to relations in resolution
      D.merge_at_time(e_0, e_1, y)
      β_1.append$(\,(x, y)\,)$
      $\mathcal{M}$.append(row_idx$[e_0]$, $|\beta_1|$, -1)
      $\mathcal{M}$.append(row_idx$[e_1]$, $|\beta_1|$, 1)
      if $s < \infty$ **then** ▷ The edge is cycle-creating
        β_2.append$(\,(x, s)\,)$
        β_0^1.append$(\,(x, s)\,)$
return $\beta_0, \beta_1, \beta_2, \beta_0^1, \mathcal{M}$

Complexity and correctness. We conclude the paper by using the theoretical results of Section 3 to prove the main results in the introduction. Proofs for the results in this section are in [30, Appendix A.3].

We start by with a convenient lemma, which gives conditions under which one can collapse an entire subgraph and produce a homology-equivalent filtered graph. The following definition describes the type of subgraph that can be collapsed.

▶ **Definition 27.** *Let $(G, f) = (V, E, \partial, f)$ be a filtered graph. A subset $E' \subseteq E$ is a* monotonic forest *of (G, f) if:*

1. *The subgraph of G spanned by the edges in E' is a forest.*
2. *For every vertex v in the forest, there exists at most one edge $e \in E'$ such that $\{e_0, e_1\} = \{v, w\}$ and $f(w) < f(v)$.*

▶ **Lemma 28.** *Let $(G, f) = (V, E, \partial, f)$ be a filtered graph, and let $E' \subseteq E$ be a set of collapsible edges, which forms a monotonic forest. If $e \in E'$, then all the edges in $E' \setminus \{e\}$ are collapsible in $(G \downarrow e, f)$. In particular, the whole forest E' can be collapsed (in any order) to obtain a filtered graph that is homology-equivalent to (G, f).*

▶ **Lemma 29.** *Let (G, f) be a $\mathcal{P}$-filtered graph. Algorithm 1 outputs a filtered graph homology-equivalent to (G, f) and without local collapsible edges in time $O(|G|)$.*

▶ **Proposition 30.** *Let (G, f) be a $\mathcal{P}$-filtered graph. Algorithm 2 outputs a vertex-minimal filtered graph homology-equivalent to (G, f) and $\beta_0(H_0(G, f))$ in $O(|G|)$ time.*

▶ **Proposition 31.** *Let (G, f) be a finite $\mathcal{P}$-filtered graph. Algorithm 3 outputs a minimal presentation of $H_0(G, f)$ in $O(|G|^2)$ time.*

With these results, we prove Theorem A and Theorem B in [30, Appendix A.4].

References

1 Madjid Allili, Tomasz Kaczynski, and Claudia Landi. Reducing complexes in multidimensional persistent homology theory. *J. Symbolic Comput.*, 78:61–75, 2017. doi:10.1016/j.jsc.2015.11.020.

2 Ulrich Bauer, Magnus B. Botnan, Steffen Oppermann, and Johan Steen. Cotorsion torsion triples and the representation theory of filtered hierarchical clustering. *Adv. Math.*, 369:107171, 51, 2020. doi:10.1016/j.aim.2020.107171.

3 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient two-parameter persistence computation via cohomology. In *39th International Symposium on Computational Geometry*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 15, 17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. 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. Efficient computation of multiparameter persistence, (in preparation).

6 Magnus Bakke Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of algebras and related structures*, EMS Ser. Congr. Rep., pages 77–150. Eur. Math. Soc., Zürich, [2023] ©2023.

7 Guillaume Brouillette, Madjid Allili, and Tomasz Kaczynski. Multiparameter discrete morse theory. *Journal of Applied and Computational Topology*, pages 1–42, 2024.

8 Chen Cai, Woojin Kim, Facundo Mémoli, and Yusu Wang. Elder-rule-staircodes for augmented metric spaces. *SIAM J. Appl. Algebra Geom.*, 5(3):417–454, 2021. doi:10.1137/20M1353605.

9 Gunnar Carlsson and Facundo Mémoli. Multiparameter hierarchical clustering methods. In *Classification as a tool for research*, Stud. Classification Data Anal. Knowledge Organ., pages 63–70. Springer, Berlin, 2010. doi:10.1007/978-3-642-10745-0_6.

10 Gunnar Carlsson and Facundo Mémoli. Classifying clustering schemes. *Found. Comput. Math.*, 13(2):221–252, 2013. doi:10.1007/s10208-012-9141-9.

11 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. In *Computational geometry (SCG'07)*, pages 184–193. ACM, New York, 2007. doi:10.1145/1247069.1247105.

12 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, Red Hook, NY, USA, 2020. Curran Associates Inc.

13 Mathieu Carriere, 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.

14 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *J. ACM*, 60(6):Art. 41, 38, 2013. doi:10.1145/2535927.

15 Andac Demir, Baris Coskunuzer, Ignacio Segovia-Dominguez, Yuzhou Chen, Yulia Gel, and Bulent Kiziltan. Todd: topological compound fingerprinting in computer-aided drug discovery. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc.

16 Herbert Edelsbrunner and John L. Harer. *Computational topology*. American Mathematical Society, Providence, RI, 2010. An introduction. doi:10.1090/mbk/069.

17 Herbert Edelsbrunner and Salman Parsa. On the computational complexity of Betti numbers: reductions from matrix rank. *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*, pages 152–160, 2014. doi:10.1137/1.9781611973402.11.

18 David Eisenbud. *Commutative algebra*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. With a view toward algebraic geometry. doi:10.1007/978-1-4612-5350-1.

19 Ulderico Fugacci, Michael Kerber, and Alexander Rolle. Compression for 2-parameter persistent homology. *Comput. Geom.*, 109:Paper No. 101940, 28, 2023. doi:10.1016/j.comgeo.2022.101940.

20 Loukas Georgiadis, Haim Kaplan, Nira Shafrir, Robert E. Tarjan, and Renato F. Werneck. Data structures for mergeable trees. *ACM Trans. Algorithms*, 7(2), March 2011. doi:10.1145/1921659.1921660.

21 Andrea Guidolin and Claudia Landi. Morse inequalities for the Koszul complex of multipersistence. *J. Pure Appl. Algebra*, 227(7):Paper No. 107319, 26, 2023. doi:10.1016/j.jpaa.2023.107319.

22 Andrea Guidolin and Claudia Landi. On the support of betti tables of multiparameter persistent homology modules. *arXiv preprint arXiv:2303.05294*, 2023.

23 Olympio Hacquard and Vadim Lebovici. Euler characteristic tools for topological data analysis. *Journal of Machine Learning Research*, 25(240):1–39, 2024. URL: http://jmlr.org/papers/v25/23-0353.html.

24 Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4314–4323. PMLR, 13–18 July 2020. URL: https://proceedings.mlr.press/v119/hofer20b.html.

25 Michael Kerber and Alexander Rolle. Fast minimal presentations of bi-graded persistence modules. *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 207–220, 2021. doi:10.1137/1.9781611976472.16.

26 Michael Lesnick. Notes on multiparameter persistence (for amat 840), 2023.

27 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded Betti numbers of 2-parameter persistent homology. *SIAM J. Appl. Algebra Geom.*, 6(2):267–298, 2022. doi:10.1137/20M1388425.

28 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, 2024.

29 Ezra Miller and Bernd Sturmfels. *Combinatorial commutative algebra*, volume 227 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2005.

30 Dmitriy Morozov and Luis Scoccola. Computing betti tables and minimal presentations of zero-dimensional persistent homology, 2024. doi:10.48550/arXiv.2410.22242.

31 L. A. Nazarova. Partially ordered sets of infinite type. *Izv. Akad. Nauk SSSR Ser. Mat.*, 39(5):963–991, 1219, 1975.

32 Steve Oudot and Luis Scoccola. On the Stability of Multigraded Betti Numbers and Hilbert Functions. *SIAM J. Appl. Algebra Geom.*, 8(1):54–88, 2024. doi:10.1137/22M1489150.

33 Steve Y. Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2015. doi:10.1090/surv/209.

34 Irena Peeva. *Graded syzygies*, volume 14 of *Algebra and Applications*. Springer-Verlag London, Ltd., London, 2011. doi:10.1007/978-0-85729-177-6.

35 Alexander Rolle and Luis Scoccola. Stable and consistent density-based clustering via multiparameter persistence. *Journal of Machine Learning Research*, 25(258):1–74, 2024. URL: http://jmlr.org/papers/v25/21-1185.html.

36 Joseph J. Rotman. *An introduction to homological algebra*. Universitext. Springer, New York, second edition, 2009. doi:10.1007/b98977.

37 Sara Scaramuccia, Federico Iuricich, Leila De Floriani, and Claudia Landi. Computing multiparameter persistent homology through a discrete morse-based approach. *Computational Geometry*, 89:101623, 2020. doi:10.1016/j.comgeo.2020.101623.

38 Luis Scoccola and Alexander Rolle. Persistable: persistent and stable clustering. *Journal of Open Source Software*, 8(83):5022, 2023. doi:10.21105/joss.05022.

39 Luis Scoccola, Siddharth Setlur, David Loiseaux, Mathieu Carrière, and Steve Oudot. Differentiability and optimization of multiparameter persistent homology. In *Forty-first International Conference on Machine Learning*, 2024. URL: https://openreview.net/forum?id=ixdfvnO0uy.

40 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. System Sci.*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

41 Qi Zhao and Yusu Wang. *Learning metrics for persistence-based summaries and applications for graph classification*, pages 9859 – 9870. Curran Associates Inc., Red Hook, NY, USA, 2019.

42 Ángel Javier Alonso, Michael Kerber, and Siddharth Pritam. Filtration-domination in bifiltered graphs. *2023 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 27–38, 2023. doi:10.1137/1.9781611977561.ch3.

Computing Geomorphologically Salient Networks via Discrete Morse Theory

Tim Ophelders
Utrecht University, The Netherlands
TU Eindhoven, The Netherlands

Anna Schenfisch
TU Eindhoven, The Netherlands

Willem Sonke
TU Eindhoven, The Netherlands

Bettina Speckmann
TU Eindhoven, The Netherlands

Abstract

Rivers, estuaries, intertidal zones, and other hydrological systems often give rise to complex networks of interconnected channels. Even today, such networks are typically drawn manually by domain experts. Traditional watershed methods for automating this process, where water flows are assumed to follow steepest descent, fail to capture behavior particular to low-relief terrains. At SoCG 2017, Kleinhans et al. proposed a method to construct a network of source-to-sink paths separated by sufficient sediment volume. However, this method is unstable with respect to minor changes of the input terrain, and constructs only channels that flow from one side of the terrain to the other, thereby failing to detect the dead-end channels ("fingers") that characterize intertidal zones.

We show how to compute geomorphologically salient networks that avoid these issues. After extending elevation data to a discrete Morse function on the terrain, we identify channels that flow through saddles and have sufficient volume of sediment on both sides. We then detect fingers, which follow the boundary of "spurs" that have sufficient volume of sediment above a particular height. The main challenge here lies in meaningfully modeling salient spurs and determining suitable heights to measure volume. We implemented our method and applied it to real-world data. Our expert users have validated the mathematical modeling by confirming that the resulting (finger) channels indeed constitute a geomorphologically salient network.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases hydrology, network detection, intertidal zones, braided rivers, discrete Morse theory, volume persistence

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.70

Supplementary Material *Software*: `https://github.com/tue-alga/topotide`
archived at `swh:1:dir:9151f6e4cbe09c40f8cbbdcd71948a8431d5f055`

Funding This work is part of the WadSED (`https://wadsed.nl/`) research program financed by the Dutch Research Council (NWO) under grant P21-13.
Tim Ophelders: supported by the Dutch Research Council (NWO) under the project number VI.Veni.212.260.

1 Introduction

Rivers and estuaries are important natural landscapes. Their channels can effectively be represented by networks. The analysis of such networks is a powerful tool to answer fundamental questions about dynamics of nodes, emergent network properties [18, 17, 24, 25], and the propagation of disturbances. However, geoscientists and engineers struggle with

© Tim Ophelders, Anna Schenfisch, Willem Sonke, and Bettina Speckmann;
licensed under Creative Commons License CC-BY 4.0

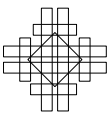


Figure 1 Example of an intertidal zone: part of the Wadden Sea, the Netherlands. *Imagery courtesy of Beeldmateriaal.nl, used under license CC-BY 4.0.*

the creation and analysis of networks from spatial data such as satellite imagery and echo-sounding bathymetries. In general, the number of datasets and models is growing explosively, but the tools to analyze them lag behind considerably. Present practice relies on manual creation and somewhat arbitrary rules: a laborious process with unreliable results.

In this paper we describe an algorithm that constructs geomorphologically salient networks based on the topological properties of the river bed as represented by a digital elevation model (DEM). In computational geometry, algorithms for computing networks from high-relief DEMs are well-studied, for example, in the context of *drainage networks* or *flows* [2, 1, 5, 3, 26]. But most models for water flow in terrains, including drainage networks, generally assume that water follows the direction of steepest descent. On steep terrains this is a reasonable assumption, but channels within a braided river or estuary do not follow steepest descent. Firstly, the inertia of flowing water can result in deviations from the true steepest descent direction. Secondly, water can fill up local minima in the riverbed. Furthermore, if a river always follows steepest descent, it can never split. Yet in practice such splits, called *bifurcations*, form frequently in most braided rivers and multi-channel estuaries; they are key elements for network properties [15].

Kleinhans, van Kreveld, Ophelders, Sonke, Speckmann, and Verbeek [16] were the first to propose a method that computes networks for braided rivers and estuaries based on bathymetry. Their algorithm constructs the network from a set of (partially overlapping) low paths through the DEM (a *striation*), which ensures that bifurcations and confluences can be represented. There are many possible low paths and corresponding channels. The authors choose significant channels based on a morphological concept: sediment volume. Two channels can be considered similar if the volume of terrain between the channels is small,

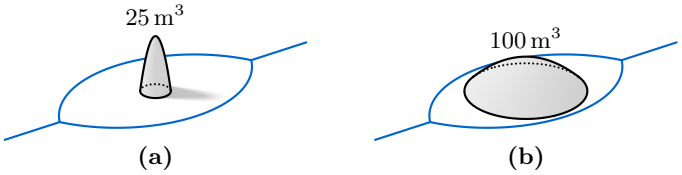


Figure 2 In (a) the channels are separated by a lower volume of sediment than in (b). We hence consider the channels in (a) more similar, even though the maximum in (a) is higher than in (b).

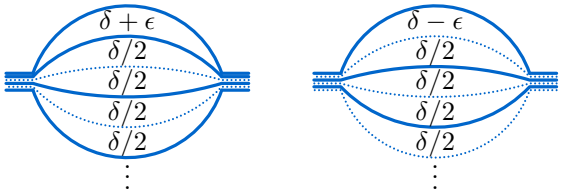


Figure 3 With striation-based methods, if a sediment volume changes by just ε, the set of paths selected from the striation (solid blue lines) may change dramatically even far away from the change.

modeling the fact that a small volume of sediment can easily be eroded in a short time by water flow, merging the two channels into one [14]. The algorithm hence selects a subset of the striation paths which are separated by sufficient volume of sediment (see Figure 2). This method has two drawbacks. First and foremost, it is unstable with respect to minor changes of the terrain (see Figure 3). Secondly, in low-relief terrains, such as intertidal zones and estuaries, there are many geomorphologically significant dead-end channels (*fingers*), caused by tidal in- and outflow (see Figure 1 and its "tree of fingers"). Striation paths by design flow from one side of the terrain to the other and hence the method fails to detect fingers.

In this paper we retain the notion of volume-separated channels, but turn to persistence and discrete Morse theory to avoid instability. Specifically, we use the concept of *volume persistence*, which considers only those topological features of the DEM that correspond to "sand bars" of sufficient volume. That is, we use a *volume-simplified* DEM, which we obtain by *pruning* topological features of the terrain that can be eliminated by removing only a small amount of volume. Topological persistence was introduced by Edelsbrunner, Letscher, and Zomorodian [7], and is a record of homological changes throughout a *filtration* of an underlying space. Such filtrations are often defined with respect to the distance between points, however, other measures can be used. Carr, Snoeyink, and van der Panne [4] describe a method to simplify contour trees using so-called *local geometric measures*, such as (in 2D terrains) the line length of the contour, the area enclosed by the contour, or the volume of the enclosed region. This last type of filtration is exactly the one we use. The remaining channels in our volume-persistent network separate regions of the terrain with high significance.

A couple of years ago, a geomorphologcial paper [23] sketched how to use volume persistence to compute networks from bathymetry; however, the exposition lacks mathematical detail. Height persistence has previously been employed to fill local minima in terrains [5]. De Floriani, Iuricich, Magillo, and Simari [6] compared watershed methods and discrete Morse theory for the construction of watersheds, but only on terrains with high relief and water flow that follows steepest descent.

Contribution and organization. In this paper we give a mathematically sound formalization of volume persistence and its use for network computation. In Section 2 we provide background on discrete Morse theory. Section 3 describes our method to construct a network based on volume persistence. By design these networks are stable under perturbations of small volume, however, like the channels that stem from striations, they connect one side of the terrain to the other. In Section 4 we develop the additional theory necessary to detect salient fingers. The major challenge here is to characterize the area and corresponding terrain volume that defines a finger. It is clear which area lies between two channels that first bifurcate and then merge again. Correspondingly, it is straightforward to compute the volume of terrain between two channels. But what area determines a finger? When are two fingers similar? Which fingers should be retained in a network given a volume threshold

for persistence? To address these challenges, we introduce the notion of "spurs" on the terrain and define salient fingers as those that bound spurs of sufficient volume. Finally, in Section 5 we showcase the results of our method as implemented in our tool TopoTide (`https://github.com/tue-alga/topotide`).

2 Preliminaries

Our network extraction methods are grounded in discrete Morse theory and concepts from topology are useful to describe them. Additionally, various notions of volume play a central role in our methods. Therefore, in this section, we provide definitions related to these topics.

Discrete Morse theory. Traditional Morse theory enables us to study a manifold by defining a Morse function, a differentiable function on the manifold, and studying its features such as critical points and stable and unstable manifolds. As the data describing our terrain is not continuous, but consists of discrete points, we assume the terrain is a two-dimensional *cell complex* T: a surface defined by a set of zero-, one-, and two-cells so that the intersection of any two cells in T is in T itself. We then turn to *discrete Morse theory* [11, 20].

For cells τ and σ, we say that τ is a *face* of σ (or σ is a *coface* of τ) if $\tau \subseteq \sigma$, and we say that τ is a *facet* of σ (or σ is a *cofacet* of τ) if τ is a face of σ and $\dim(\tau) = \dim(\sigma) - 1$. A *discrete Morse function* (DMF) is a map $f\colon T \to \mathbb{R}$ such that every cell has a lower (higher) value than its cofacets (facets) with at most one exception. It may be helpful to visualize f as pairing facets to cofacets where these exceptions occur, in an analog to gradient flow from the smooth setting. These pairings are called *gradient pairs* and we may illustrate them using arrows pointing from facets to cofacets, as in Figure 4. A cell σ is *critical* if σ is not paired with any of its facets or cofacets. A critical zero-cell is called a *minimum*, a critical one-cell is called a *saddle*, and a critical two-cell is called a *maximum*.

Given a DMF f, an *ascending path* is a sequence that alternates between zero- and one-cells, "swimming upstream" along the pairings induced by f. Similarly, a *descending path* is a sequence that alternates between two- and one-cells, following the pairings induced by f. The union of all ascending paths forms a spanning forest, and so does the union of all descending paths; these spanning forests are dual, with the exception of saddles. For each facet of a saddle, there exists a unique ascending path ending at that zero-cell and originating at a minimum; we define the *ascending Morse–Smale complex* (or *MS-complex* for brevity) as a graph $G = (V, E)$ with nodes V corresponding to the minima of f, and arcs E corresponding to the pairs of ascending paths connected via saddles between them. See Figure 4. Similarly, for each cofacet of a saddle, there is a unique descending path ending

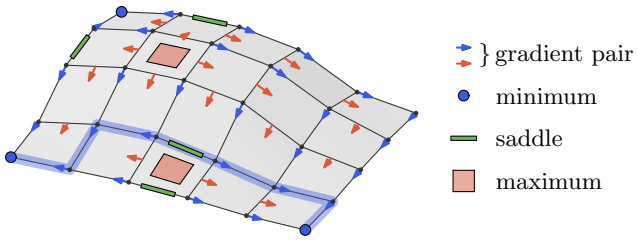


Figure 4 Gradient pairing induced by a DMF on a terrain. Arcs of G are paths of zero- and one-cells that begin and end at minima and pass through a saddle, following gradient pairings. One such arc is highlighted above in blue. We emphasize that gradient pairing arrows do not indicate network flow; because of tides, wind, etc., flow direction is not appropriate for our applications.

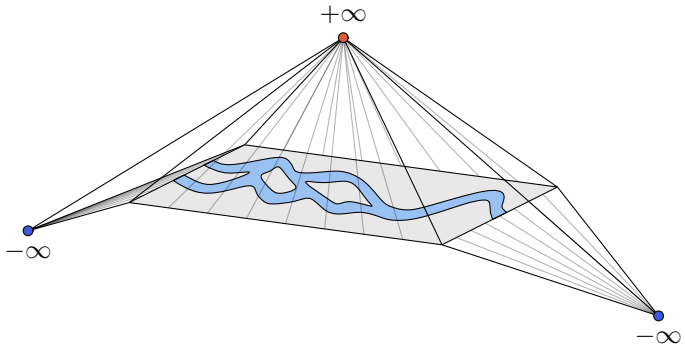


Figure 5 To make T a sphere, we connect three zero-cells at $-\infty$, $-\infty$, and $+\infty$ to the boundary.

at that two-cell and originating at a maximum. Note that, while a node $v \in V$ corresponds to a zero-cell, an arc $e \in E$ represents the union of potentially many zero- and one-cells. Furthermore, the highest zero-cell along e is the highest facet of its saddle. Each face of G contains a single maximum. For a subset $E' \subseteq E$, we write $G - E'$ to denote the arc-induced subgraph on $E \setminus E'$, i.e., the graph containing only the arcs $E \setminus E'$ and their endpoints.

Terrain data. The input to our methods is a *digital elevation model* (DEM), a set of points and associated elevations. These points and elevations correspond to the elevations of geographic locations in a real-world or simulated terrain. DEMs come in many varieties, for instance triangular irregular networks (TINs) or other point configurations. In this paper, we suppose our DEM data consists of a regular rectilinear grid of measurement points. However, we emphasize that discrete Morse theory along with many of the methods we describe, generalizes to a wide variety of DEM types and structures.

We assume that our input DEM corresponds to a region homeomorphic to a disk, i.e., there are no arches or more complex structures. If any two zero-cells have identical elevation, we perform a slight perturbation so that general position is satisfied [8]. We add three additional zero-cells to an input DEM. First, we add two zero-cells at elevation $-\infty$ to represent a global source and sink, and connect to the appropriate regions of the DEM (conceptually, the source and sink along this boundary). We then add a zero-cell at elevation $+\infty$ to represent the land that is not part of the terrain of interest (e.g., the bank of a river), which we assign elevation $+\infty$, and connect to the entire (new) boundary, including the minimum points. Thus, T is topologically a sphere (see Figure 5). Note that T contains non-square two-cells adjacent to the infinite points, so T does not have an entirely grid-like structure; however, the region of interest, namely, points that correspond to data measurements, is endowed with a grid structure, and it suffices to conceptualize T as cubulated. We call a DEM adapted in this way a *terrain*, which is the starting point for our work.

Choosing a discrete Morse function. A discrete Morse function (DMF) $f : T \to \mathbb{R}$ assigns a real value to every cell of T: not only the zero-cells, but also the one- and two-cells. On the other hand, our terrain data consists of elevation values for the zero-cells of T, but not the one- and two-cells. Thus, we have to extend these measurements on the zero-cells to every cell of T in such a way that we obtain a discrete Morse function, whose critical points and gradient pairs reflect the input measurements. Several choices have been proposed in the literature, see for example [21, 12, 13, 9, 22, 19, 10]. In this paper we use the method proposed in [22]; we sketch the relevant gradient pairings in the following construction.

▶ **Construction 1** (From Elevations to Pairings). *The real values associated to every zero-cell in a terrain induce a high-to-low lexicographical order on all cells. First, pair every zero-cell with its lowest lower cofacet, if it exists. Next, for every remaining one-cell w, consider the cofacets of w for which w is the highest facet. Among these, pair w with the two-cell for which the one-cell opposite w is lowest. Finally, repeat the previous step for every remaining one-cell w', considering the unpaired cofacets of w' for which w' is the second-highest facet.*

A main contribution of [22] shows the gradient pairings described above are valid, and do not contradict given elevation data.

▶ **Lemma 2** (Adapted from [22]). *Let T be a terrain. Then the pairings of Construction 1 correspond to a valid DMF f on T, where, for all zero-cells $z \in T$, $f(z)$ is the elevation of z.*

Since a set of valid gradient pairings is equivalent to a DMF inducing those pairings, we generally proceed by discussing gradient pairings rather than an explicit function.

Volume and Paths. Our filtrations depend on a volume-based parameter. Thus, we first need to define the *volume* of a region in T. Conceptually, we think of the elevation of each zero-cell as volume being divided among its four adjacent two-faces, so that each zero-cell contained in a region contributes zero, one, two, three, or four quarters of its volume (height value) to the total, depending on how many of its adjacent two-faces are in the region.

▶ **Definition 3** (Quarter-Pillar Volume). *Consider some height h and two-cell $t \subseteq T$. Denote the zero-cells adjacent to t by z_1, z_2, z_3, and z_4, and for $i \in \{1, 2, 3, 4\}$, let $f(z_i)$ denote the elevation of z_i. Then we define the volume of t above h as*

$$\sum_{i=1}^{4} \frac{\max\{f(z_i) - h, 0\}}{4}.$$

That is, each zero-cell contributes a quarter of its elevation above h to the volume of t. Next, let $H \subseteq T$. We define the volume of H *as the sum of volumes of all two-cells contained in H.*

Note that Definition 3 could easily be adapted for more general settings, e.g., if faces do not necessarily have exactly four nodes, or if the size or shape of the faces is salient. We want to know the volume of regions of the terrain that are above a particular elevation, but as elevation is only defined on zero-cells, we introduce the following concept.

▶ **Definition 4** (Zero-Cell Superlevel Set). *Given a zero-cell z of some terrain T, the* zero-cell superlevel set of z *is the collection of zero-cells of T with equal or higher elevation than z.*

We are generally interested in measuring volumes that correspond to continuous regions of the terrain. This is formalized in the following definition.

▶ **Definition 5** (Zero-Cell Connected). *Let $H_0 \subseteq T$ be a set of zero-cells. Two zero-cells $z_1, z_n \in H_0$ are* (zero-cell) connected *if and only if there is a sequence $z_1, z_2, \ldots, z_n$ such that, for $1 < i < n$, every z_i is contained in H_0 and every adjacent pair, z_i and z_{i+1}, are faces of a common two-cell of T. We say that H_0 is* (zero-cell) connected *if every pair of zero-cells in H_0 is zero-cell connected. A* connected component of H_0 *is a connected set that is not a proper subset of any other connected set in H_0.*

Next, we introduce an idea that allows us to consider lexicographically lowest paths.

▶ **Definition 6** (Lexicographic Order of Paths). *In a terrain T, let P_1 and P_2 be paths of alternating zero- and one-cells. Then we say P_1 is* lexicographically lower than *P_2 if the list of zero-cells in P_1 sorted by elevation from high to low is lexicographically lower than the sorted list of zero-cells in P_2.*

Roughly speaking, Construction 1 defines ascending paths of an MS-complex by introducing zero-/one-cell pairings that are lexicographically low. This has the following consequence.

▶ **Corollary 7** (Lowest Paths are Arcs of G). *Suppose that T is a terrain with a DMF defined as in Construction 1, and consider two minima, z_1 and z_2. Then the lexicographically lowest path through zero- and one-cells of T connecting z_1 and z_2 corresponds to arcs of the corresponding MS-complex, G.*

3 Volume Persistence for Braided Rivers and Estuaries

In this section, we detail our network extraction method when the terrain in question is a braided river or estuary in which no channels truncate (i.e., there are no degree one nodes in the network).

In practice, terrains corresponding to actual data are quite noisy, so a network G that is simply the MS-complex for some corresponding DMF is much more complex than what we might want. Thus, we filter G based on a *volume parameter*. Conceptually, we want to keep channels that run between two "hills" that contain enough volume. Note that if an arc e ends at a minimum in a shallow basin, an endpoint of e may have degree one, i.e., e does not separate two hills. We specify arcs that separate hills more formally below.

▶ **Definition 8** (Splitting Arcs). *Let G be the MS-complex for a DMF on a terrain T. For an arc $e \in E(G)$, let z denote the highest zero-cell of T contained in e. Let H_0 denote the connected component of the zero-cell superlevel set at the height of z that contains z. We say that e is a* splitting arc *if $H_0 \setminus z$ is two connected components.*

Next, we assign a volume to each splitting arc, corresponding to the volume of sediment that would need to be removed in order for the channel to no longer be defined.

▶ **Definition 9** (Volume Persistence of Splitting Arcs). *Let G be the MS-complex for a DMF on a terrain T. For a splitting arc $e \in E(G)$, let z denote the highest zero-cell of T in e. Let H_0 be the connected component of the zero-cell superlevel set at the height of z that contains z. Let H_0^a and H_0^b be the two connected components of $H_0 \setminus z$. Finally, let H^a (and H^b) be the set of two-cells that have at least one incident zero-cell in H_0^a (and H_0^b, respectively). We define the* volume persistence of e *as the minimum volume of H^a or H^b.*

See Figure 6. If the volume persistence of an arc is low, a small change to the terrain would remove the arc. Thus we use the volume persistence of arcs to filter G, as follows.

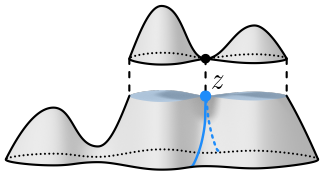


Figure 6 Zero-cell z is the highest zero-cell along a splitting arc of G (shown in blue). The volume persistence of the arc is the smaller volume of the two "hills above z".

▶ **Definition 10** (Volume-Filtered Subnetwork)**.** *For the MS-complex G of a terrain T, and for $\delta \in \mathbb{R}_{\geq 0}$, let G_δ be the graph induced by the set of splitting arcs with volume persistence at least δ, along with all (remaining) lowest paths between nodes in G_δ.*

Note that by Corollary 7, the lowest path between a pair of minima is made up of arcs of G, which may be splitting or non-splitting. The following lemma asserts that using δ as a parameter defines a *filtration* of G, where inclusion occurs as δ decreases.

▶ **Lemma 11.** *Let G be an MS-complex for a terrain T, and let $\delta, \delta' \in \mathbb{R}_{\geq 0}$ such that $\delta \leq \delta'$. Then $G_{\delta'} \subseteq G_\delta$.*

Proof. Consider a splitting arc $e \in E(G_{\delta'})$. Then the volume persistence of e is greater than or equal to $\delta' \geq \delta$, meaning $e \in E(G_\delta)$. Since our choice of e was arbitrary, every splitting arc of G'_δ is a splitting arc of G_δ, and $V(G_{\delta'}) \subseteq V(G_\delta)$. Then the lowest paths between two nodes of $G_{\delta'}$ are also in $G_{\delta'}$. Thus, $G_{\delta'} \subseteq G_\delta$. ◀

We also take the following general position assumptions. Let f be a DMF of a terrain T. We assume f is injective on the zero-cells of T with a finite height, i.e., they each have a unique height. Furthermore, given G, the MS-complex associated to T, each arc of G has a unique volume persistence in the sense of Definition 9.

3.1 Properties of Merging Faces

Given a DMF over a terrain T and the associated MS-complex G, our goal is now to produce the volume-filtered subnetwork G_δ for any $\delta > 0$. The computational challenge is to efficiently compute the volume persistence of each splitting arc $e \in E(G)$. Recall that, for z the highest zero-cell of T along e, the volume persistence of e is the minimum volume of a connected component of the superlevel set above and on either side of z, so we first need to identify such components. A naïve approach might be to begin at a saddle one-cell and use a depth-first search method to explore neighboring two-cells, adding them to the connected components for e until we identify a boundary made up of cells with lower height, i.e., the boundary of the relevant connected components of the superlevel set. However, in doing so, we would potentially revisit the cells in the highest face of G s times (where s is the number of saddles), cells in the second highest face $s-1$ times, and so on. Instead, we propose a faster method to find relevant superlevel sets that makes use of containment properties. By processing arcs "from high to low," we can use information about previously explored regions and avoid the need to analyze the same two-cells more than once. More formally, the following lemma establishes a helpful property that is a result of this strategy.

▶ **Lemma 12.** *Let G be the MS-complex of a DMF on a terrain T with gradient pairings as in Construction 1. Let $e_1, e_2, \ldots, e_m$ be the total ordering of the arcs from high to low via their highest zero-cell. For $1 \leq i \leq m$, let z_i denote the highest zero-cell of e_i. Then the connected component of all zero-cells in the superlevel set above and containing z_i is strictly contained in the faces of the graph $G^i = G - \{e_1, e_2, \ldots, e_{i-1}\}$ that are adjacent to e_i.*

Proof. For some $1 \leq i \leq m$, let z_i denote the highest zero-cell of T along arc e_i, and let H_0 denote the zero-cell connected component of the superlevel set above and containing z_i (i.e., the zero-cells in the *hills above* z_i). Let F denote the union of the faces of G^i adjacent to e_i, considered as a collection of zero-, one-, and two-cells.

First, suppose the boundary of F does not contain e_i. Then arcs in the boundary of F all have a highest zero-cell lower than the height of z_i. In particular, all zero-cells contained in the boundary of F have lower elevation than z_i, and are therefore not contained in H_0.

Then any connected collection of zero-cells of H_0 may not cross the boundary of F. By construction, H_0 intersects e_i at the single point z_i, an interior point of F, so clearly H_0 is not disjoint from F. Thus, $H_0 \cap F = H_0$, as desired.

In the case that the boundary of F contains e_i (for instance, if e_i has a node with degree one), the boundary of F is comprised of a set of arcs whose collective highest zero-cell is z_i. Then the only way for a connected collection of zero-cells in H_0 to leave F is through z_i. However, z_i is not an endpoint of e_i (endpoints of e_i are local minima), so all zero-cells neighboring z_i with higher elevation must be in the interior of F, and the claim follows by similar logic to the preceding case. ◀

Finally, we aim to show that the behavior of zero-cell connectivity of superlevel sets mimics behavior in a continuous terrain; namely, all points in the superlevel set above an arc contained in a face of G^i are part of the same zero-cell connected component.

We first establish a helpful lemma for G, before any arcs have been removed.

▶ **Lemma 13.** *Let G be the MS-complex of a DMF on a terrain T, with gradient pairings as in Construction 1. Let z denote the highest zero-cell along the boundary of some face of G. Then the connected component of the superlevel set of z contains both z and the highest zero-cell of the maximum in that face.*

Proof. Recall that each maximum is contained in a distinct face of G. Let z denote the highest zero-cell along the boundary of some face of G. The saddle containing z is the face of some two-cell, which we denote β. Since the union of descending paths in a face of G forms a spanning tree, there is a unique path P from β to the maximum, following gradient pairs up (i.e., a path of alternating paired one- and two-cells).

For any gradient pair from a one-cell α to a two-cell β, we claim that the highest zero-cell of β is also a zero-cell of α. Recall from Construction 1 that α can be paired with β only if α is lexicographically either the highest or second-highest one-cell of β. In both cases, the highest zero-cell is necessarily a face of α. Thus, by following P, we obtain a non-decreasing path of zero-cells from z to the highest zero-cell of the maximum. ◀

Using Lemma 13, we now show that particular superlevel sets contained in faces of G^i (i.e., G without the first $i-1$ highest arcs) are connected. Importantly, this means intuition about superlevel sets from a continuous terrain also applies in our discretized setting.

▶ **Lemma 14.** *Let G be the MS-complex of a DMF a terrain T, with gradient pairings as in Construction 1. Let $e_1, e_2, \ldots, e_m$ be the total ordering of the arcs from high to low via their highest zero-cell. For $1 \leq i \leq m$, let z_i denote the highest zero-cell of e_i. Then the zero-cells of the superlevel set at z_i contained in a face of the graph $G^i = G - \{e_1, e_2, \ldots, e_{i-1}\}$ adjacent to e_i form a single zero-cell connected component.*

Proof. For some $1 \leq i \leq m$, consider the graph G^i. Let F denote a face of G^i adjacent to e_i, let z_i denote the highest zero-cell along e_i, and let H_0 denote the superlevel set at the height of z_i that is contained in F. First, we observe that any connected component of H_0 has some zero-cell with maximum height, and so is adjacent to at least one maximum. We therefore aim to show all maxima in F that are adjacent to zero-cells of H_0 (i.e., the maxima of F that are "in" the superlevel set) are connected in H_0. By Lemma 13, we know that any two such maxima are in the same connected component of the superlevel set of z_j, where z_j is the highest zero-cell of the arc e_j that separates these maxima in G. However, such a zero-cell z_j is interior to F, which means z_j must be higher than z_i, otherwise, the arc e_j would not have been removed in G^i, and the arc e_j would be a boundary of F. This means maxima in F adjacent to a zero-cell of H_0 are connected in H_0, and we have shown the claim. ◀

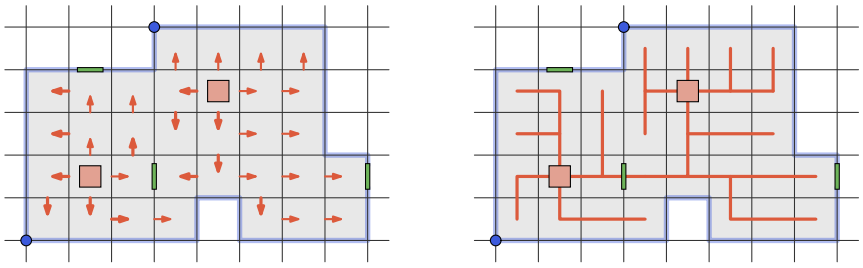


Figure 7 In this single face of G_δ, there are two maxima, so the gradient pairs form a spanning forest consisting of two trees rooted at the maxima (left). By including the saddle, we obtain the red tree for this face, here drawn as the dual (right).

For a single two-cell t, consider the function for volume in t above a particular height; as the height increases/decreases, the volume changes according to a piecewise linear function, with discontinuities every time the height passes over one of the zero-cells of t. Thus, we have a piecewise linear function associated to each two-cell in the terrain.

The function for volume under a face of G above some height is then simply the sum of functions for each two-cell of the face. If the parameters at which the volume function for the face of G changes are sorted (i.e., if the heights of zero-cells in the face are sorted), we can query the function at a specific height by using binary search on the parameters in $O(\log n)$ time. By Lemma 12, by processing arcs from high to low, we know exactly which faces are in the relevant components of the superlevel set above some arc; it is the adjacent merges of faces after we remove all higher arcs from G. The function for volume after the merge of two faces is the sum of functions for volume in each face individually.

4 Volume Persistence for Intertidal Zones

We shift our attention to the detection of fingers, which is primarily relevant when analyzing intertidal zones. As before, we assume our input is a terrain T and use Construction 1 to identify gradient pairings corresponding to a compatible DMF. One major difference between detecting bars separating channels (Section 3) and detecting fingers is that fingers are not necessarily arcs of the MS-complex. Therefore, we have to search for fingers inside each face of the MS-complex. We thus first apply the method from Section 3 for some fixed $\delta > 0$ to identify channels from the MS-complex that separate "sufficient volume." These remaining channels divide the terrain into faces; our goal is to extract fingers within each such face.

First, we define an adaptation of descending paths. Recall that descending paths are alternating sequences of two- and one-cell gradient pairs that collectively form a spanning forest over the terrain, and each maximum corresponds to a tree in this forest. When we filter the MS-complex in Section 3, we remove entire arcs, so that multiple maxima are contained within the same region. Thus, just as we included saddles to connect ascending paths in Section 3, we include saddles to connect descending paths within a particular region. This is illustrated in Figure 7 and defined formally below.

▶ **Definition 15** (Red Tree). *For a terrain, T, and $\delta > 0$, let G_δ denote the associated volume-filtered subnetwork. Within a face of G_δ, add saddles from high to low to the union of descending paths contained within a face of G_δ, stopping once the set of initial trees is joined to form a single (spanning) tree, and omitting any saddles whose inclusion would form a cycle. We call this merged collection of descending paths and saddles a* red tree.

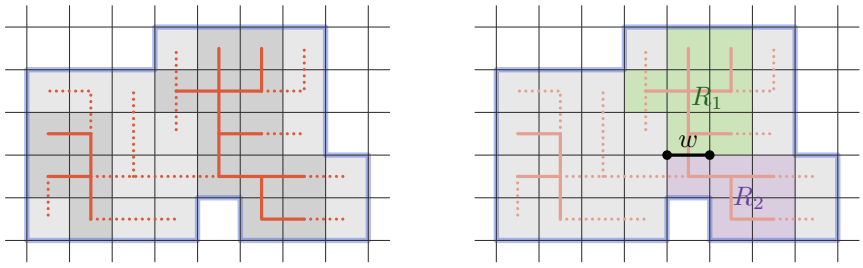


Figure 8 Left: the forest R' for some height h. Right: the volume of a one-cell w is the minimum of the volumes within R_1 and R_2.

Volume and Height Filtration of Descending Paths. In Section 3, we associate volumes to splitting arcs of G, based on the minimum volume of either of the two hills that the arc separates. Here, we take a finer-grained approach and, instead, associate volume to *individual* one-cells in a red tree. While a one-cell generally does not separate hills like a splitting arc of G does, we take inspiration from this sort of behavior and instead, consider the volumes of the two subtrees formed by the one-cell's removal, above a given height. Note that it makes sense to discuss "volume of a subtree" in this setting, as the subtrees are collections of two- and one-cells, and thus, the quarter pillar volume (Definition 3) may be nonzero. More formally, we have the following.

▶ **Definition 16** (Volume of a One-Cell for Height h). *Let R be a red tree with one-cells W. We define the* volume function at any fixed height $h \in \mathbb{R}$, $vol_h \colon W \to \mathbb{R}_{\geq 0}$ *as follows. For a one-cell $w \in W$, let R' be the subforest of R whose one-cells are each incident to some zero-cell with height at least h. Let R_1 and R_2 denote the two subtrees of $R' - \{w\}$ adjacent to w. Let vol_{R_1} and vol_{R_2} be the volumes of the part of the superlevel set of h that corresponds to R_1 and R_2, respectively. Then we define $vol_h(w) = \min\{vol_{R_1}, vol_{R_2}\}$.*

See Figure 8. If the volume of a one-cell at a particular height is low, it does not represent a location in the terrain with significant "separation." Since we eventually use the structure of one-cells that *do* have significant separation in order to identify the locations of our finger channels, we filter the red tree in the following way.

▶ **Definition 17.** *For $\delta \geq 0$, consider G_δ, a volume-filtered subnetwork for some terrain. Suppose that R is a red tree corresponding to G_δ. Let W denote the set of one-cells of R. Then, for $h \geq 0$, we define*

$$R_{\delta,h} = R - \{w \in W \mid vol_h(w) < \delta\}.$$

Note that we now have both a volume and height parameter. If δ varies within a range that does not change the volume filtered subnetwork, G_δ, then decreasing δ or h means more one-cells become part of $R_{\delta,h}$, i.e., we have a bifiltration. However, if δ varies within a range such that G_δ changes, we are no longer guaranteed this property. In particular, if δ increases and an arc of G_δ is removed, two previously disconnected red trees merge, potentially causing the involved one-cells to have a jump in their volumes. In other words, one-cells are monotonically included only within intervals of δ for which G_δ does not change.

Identifying Significant Spurs. First, we define a spur, which is a structure defined for a fixed volume parameter and, conceptually, is determined by noting the order in which various red one-cells have sufficient volume as height varies. It may be helpful to recall that R is a tree made up of one- and two-cells.

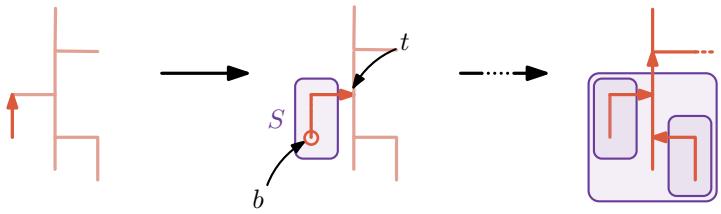


Figure 9 The spur S is defined by beginning at b and stopping once we find two neighbors with higher significance than the previous one-cell. This then defines t and b as the top and bottom of S, respectively. Repeating this process for other leaves, we eventually find all spurs (right).

▶ **Definition 18** (Spurs). *Let R be a red tree. For a fixed $\delta > 0$, each one-cell w of R has a height above which it has volume at least δ; call this the significance of w. Let b be a leaf of R; we check if its adjacent one-cell itself has exactly one adjacent one-cell with higher significance; if so, we move to this new one-cell and repeat this process. Once we encounter a one-cell that does not have exactly one adjacent one-cell with higher significance, we note its last two-cell, t, and define a* spur S with top t and bottom b, *which is the entire subtree of R rooted at t containing the path from b to t. We identify all other spurs by repeating this process for every leaf one-cell of R.*

See Figure 9. The following lemma gives us insight into the structure of spurs of R.

▶ **Lemma 19.** *Let R be a red tree with ℓ leaves, and, for a fixed volume $\delta > 0$, let $(t_i, b_i)_{i=1}^{\ell}$ denote the set of spur top and bottom pairs. Then, for $1 \leq i, j \leq \ell$ if the paths from b_i to t_i and b_j to t_j intersect, the intersection contains either t_i or t_j.*

Proof. When $i = j$, the claim holds trivially. Then consider some $i \neq j$ and suppose that the path from b_i to t_i intersects the path from b_j to t_j, i.e., there is some last two-cell, p, contained in both paths. Towards a contradiction, suppose p is not either t_i or t_j. Then we can denote w_i^1 and w_i^2 as the one-cells on the path from b_i to t_i just before and just after p respectively: define w_j^1 and w_j^2 similarly. Since p is the last two-cell in the intersection, $w_i^2 \neq w_j^2$.

Without loss of generality, suppose the significance of w_i^1 is no higher than that of w_j^1. Note that w_i^2 and w_j^2 are both adjacent to w_i^1, and only w_i^2 can have a higher significance than w_i^2, otherwise, w_i^1 would be adjacent to t_i, contradicting $p \neq t_i$. But then both w_j^2 and w_i^2 would have higher significance than w_j^2, contradicting that w_j^2 is not adjacent to t_j. ◀

Since a spur is a connected collection of one- and two-cells, we measure the volume of a spur using quarter-pillar volume, just as we did previously for other types of superlevel sets.

▶ **Definition 20** (Volume Persistence of Spurs). *Let S be a spur and suppose that the highest zero-cell adjacent to a cell of S has height h_*. We say that the* volume persistence of S *is the volume of the superlevel set at height h_* contained in S.*

A spur with volume persistence of at least δ corresponds to a region of the terrain with enough volume to separate a channel on either side of it, at least up to its flanking height. Specifically, we place fingers along the boundaries of spurs, defined below.

▶ **Definition 21** (Spur Boundaries). *Consider a spur, S. We refer to the collection of zero- and one-cells of S that are adjacent to some two-cell not in S the* boundary of S.

We choose fingers that lie on spur boundaries; our final lemma helps justify this choice.

▶ **Lemma 22.** *Let S be a spur with top t. Then all zero- and one-cells on the boundary of S that are not adjacent to t lie on some collection of ascending paths and saddles.*

Proof. Suppose, towards a contradiction, that w is a one-cell in the boundary of S, not adjacent to t, that does not lie on an ascending path or saddle of the terrain. Recall that every one-cell is either part of an ascending path, is a saddle, or is part of a descending path. I.e., it must be that w is part of a descending path, and thus has a gradient pair to some adjacent two-cell, q. But, since S is the entire subtree below its top, we have $q \in S$, which contradicts our assumption that w is on the boundary of S. ◀

Intertidal Network Computation. We are now ready to compute fingers. We summarize our entire procedure as follows:

1. Extend given elevation data to a DMF on all cells (Construction 1).
2. Compute G_δ, i.e., channels that separate hills of volume no less than δ (Section 3), for an appropriate choice of $\delta > 0$ (in practice, this value is determined by domain experts).
3. In each face of G_δ, create a red tree (Definition 15).
4. For such a red tree, determine its set of spurs (Definition 18).
5. Consider the subset, $\mathcal{S}_\delta$, of spurs with volume persistence no less than δ (Definition 20).
6. For each spur $S \in \mathcal{S}_\delta$, consider the two walks (clockwise and counterclockwise) around the boundary of S starting at the top of S. In each direction, note the first zero-cell we encounter with height no more than the flanking height of S. Such zero-cells define the degree-one end of a finger, and we complete the finger by flowing down via the lowest lexicographical path (along the boundary of S, and then possibly through additional ascending paths and saddles) until we reach a minimum of G_δ.
7. Output G_δ along with all fingers as the resulting network.

We end with a remark that highlights connections between this section and Section 3.

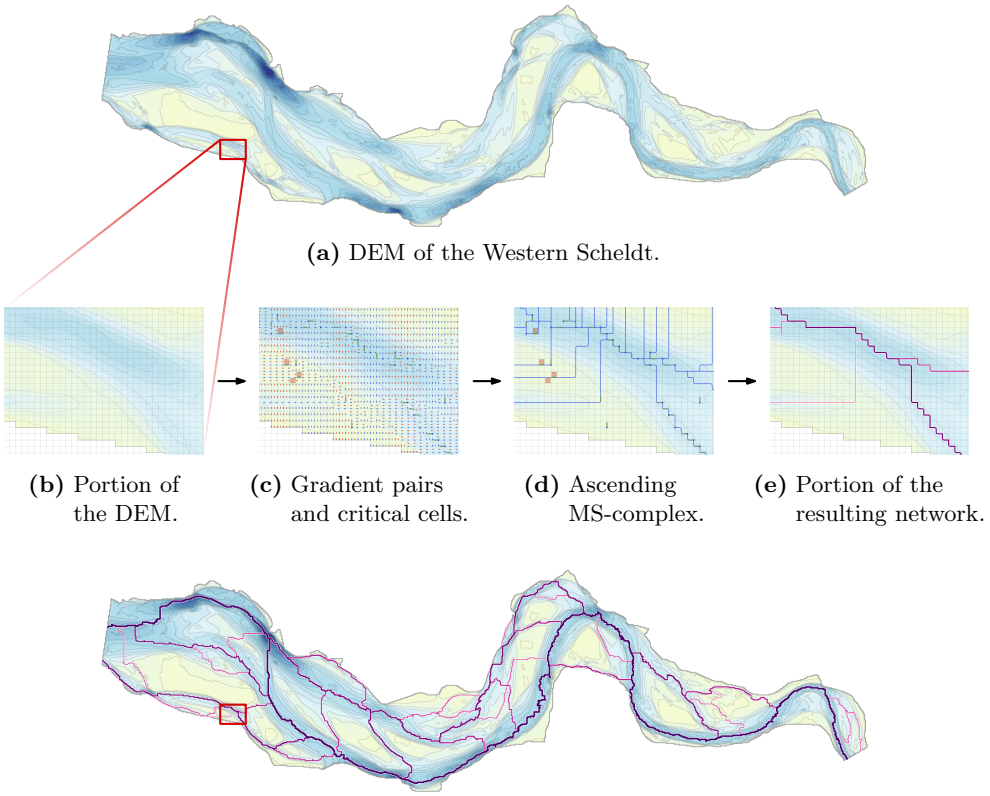
▶ Remark 23. We present intertidal network computation within a face of G_δ and the methods for computing G_δ (Section 3) as two separate operations, but the outputs for each actually share many fundamental behaviors in relation to the terrain. Namely, suppose we consider faces of G_δ as a variant of a spur. The maximum height at which such a face has volume no less than δ is at the highest zero-cell along its boundary, which is necessarily the zero-cell of some saddle. That is, this highest zero-cell height behaves like the flanking height for a spur. Then, just as in Step 6 of the intertidal network computation procedure, by flowing down from this zero-cell (this time in two directions) via the lowest lexicographical path to a minimum of some other face, we recover G_δ as a network. Thus, we could think of network computation in a more streamlined way, and not differentiate between types of ascending paths. However, we chose to present the computation of G_δ as a filtration of channels for clarity, as well as to connect better with related work.

5 Implementation and Validation

To validate our modeling, we implemented our methods and tested the implementation using two real-world datasets: a DEM of the Western Scheldt estuary, and a DEM of a part of the Wadden Sea. The first dataset serves to demonstrate the network extraction from Section 3, while the second dataset demonstrates the finger extraction from Section 4.

Western Scheldt. Figure 10 shows the input and output for the Western Scheldt dataset. In the resulting network (Figure 10f) the color of each arc encodes its volume persistence: from dark purple (high volume) to bright pink (low volume). Arcs with volume persistence less than δ are not shown.

In practice, we found that the method is quite stable under small changes in the DEM. To demonstrate this stability, in Figure 11 we show the networks extracted from historical DEMs from 1999, 2004, 2009, and 2014. Though some areas of the estuary underwent rapid changes during this period, the other parts of the extracted network stay similar.



(a) DEM of the Western Scheldt.

(b) Portion of the DEM.

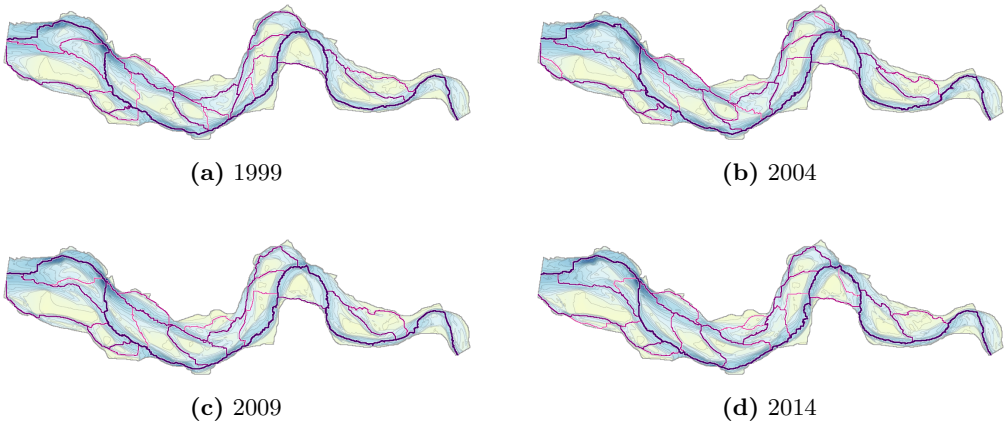
(c) Gradient pairs and critical cells.

(d) Ascending MS-complex.

(e) Portion of the resulting network.

(f) Resulting network.

Figure 10 Our network extraction method (Section 3) applied to a DEM of the Western Scheldt.



(a) 1999

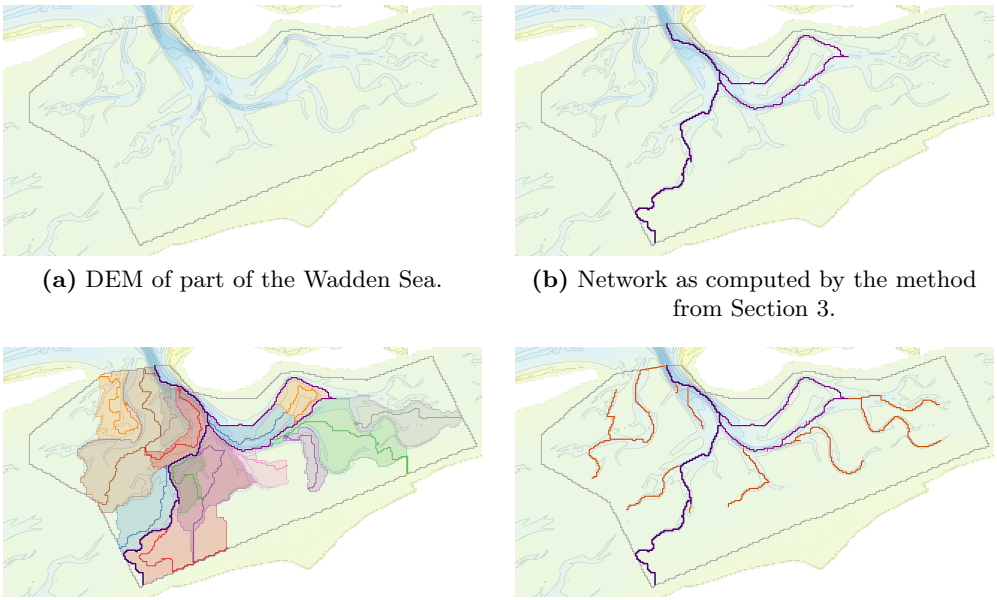
(b) 2004

(c) 2009

(d) 2014

Figure 11 Networks generated for a time series of four historical DEMs of the Western Scheldt.

Wadden Sea. Figure 12 shows the input and output for the Wadden Sea dataset. This dataset spans the same area as depicted in Figure 1. Figure 12b shows the result of our network extraction method. The method detects one significant bar between two channels, but as expected, does not detect the other features, as they are all fingers. Figure 12d then shows the result of our finger extraction method. The spurs (Figure 12c) represent areas of sufficient volume between fingers, and the extracted fingers capture the tree-like structure of the tidal basin well. In most cases, the flanking height of each spur is a good choice for determining how high the starting point of each finger lies. However, some of the extracted fingers are very short because the flanking height is quite low. This may be alleviated by filtering out fingers in their entirety if their length is below a given threshold.



(a) DEM of part of the Wadden Sea.

(b) Network as computed by the method from Section 3.

(c) Spurs shaded with distinct colors.

(d) Network extended with fingers (drawn in orange).

Figure 12 Our finger extraction method (Section 4) applied to a DEM of the Wadden Sea.

Conclusion. The networks found by the implementation show that our algorithm is indeed able to identify geomorphologically salient networks, including fingers, in low-relief terrains. Domain experts have confirmed the high quality of the networks and thereby validated our mathematical modeling.

References

1 L. Arge, J. S. Chase, P. Halpin, L. Toma, J. S. Vitter, D. Urban, and R. Wickremesinghe. Efficient flow computation on massive grid terrain datasets. *GeoInformatica*, 7(4):283–313, 2003. doi:10.1023/A:1025526421410.

2 M. de Berg, P. Bose, K. Dobrint, M. van Kreveld, M. Overmars, M. de Groot, T. Roos, J. Snoeyink, and S. Yu. The complexity of rivers in triangulated terrains. In *Proc. 8th Canadian Conference on Computational Geometry (CCCG)*, pages 325–330, 1996. doi:10.1515/9780773591134-057.

3 M. de Berg, O. Cheong, H. Haverkort, J. Lim, and L. Toma. The complexity of flow on fat terrains and its I/O-efficient computation. *Computational Geometry*, 43(4):331–356, 2010. doi:10.1016/j.comgeo.2008.12.008.

4 H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. 15th IEEE Visualization Conference (VIS)*, pages 497–504, 2004. doi:10.1109/VISUAL.2004.96.

5 A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitasova. TerraStream: From elevation data to watershed hierarchies. In *Proc. 15th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS)*, 2007. doi:10.1145/1341012.1341049.

6 L. De Floriani, F. Iuricich, P. Magillo, and P. Simari. Discrete Morse versus watershed decompositions of tessellated manifolds. In *Proc. 17th Image Analysis and Processing (ICIAP)*, pages 339–348, 2013. doi:10.1007/978-3-642-41184-7_35.

7 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533, 2002. doi:10.1007/s00454-002-2885-2.

8 H. Edelsbrunner and E. P. 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.

9 B. T. Fasy, B. McCoy, and D. L. Millman. If you must choose among your children, pick the right one. In *Proc. 32nd Canadian Conference on Computational Geometry (CCCG)*, pages 336–344, 2020.

10 R. Fellegara, F. Luricich, L. De Floriani, and K. Weiss. Efficient computation and simplification of discrete Morse decompositions on triangulated terrains. In *Proc. 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 223–232, 2014. doi:10.1145/2666310.2666412.

11 R. Forman. A user's guide to discrete Morse theory. *Séminaire Lotharingien de Combinatoire*, 48(B48c):1–35, 2002.

12 A. Gyulassy, P.-T. Bremer, B. Hamann, and V. 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.

13 H. King, K. Knudson, and N. Mramor. Generating discrete Morse functions from point data. *Experimental Mathematics*, 14(4):435–444, 2005. doi:10.1080/10586458.2005.10128941.

14 M. G. Kleinhans. Flow discharge and sediment transport models for estimating a minimum timescale of hydrological activity and channel and delta formation on Mars. *Journal of Geophysical Research*, 110(E12), 2005. doi:10.1029/2005JE002521.

15 M. G. Kleinhans, R. I. Ferguson, S. Lane, and R. J. Hardy. Splitting rivers at their seams: Bifurcations and avulsion. *Earth Surface Processes and Landforms*, 38(1):47–61, 2013. doi:10.1002/esp.3268.

16 M. G. Kleinhans, M. van Kreveld, T. Ophelders, W. Sonke, B. Speckmann, and K. Verbeek. Computing representative networks for braided rivers. *Journal of Computational Geometry*, 10(1):423–443, 2019. doi:10.20382/jocg.v10i1a14.

17 W. A. Marra, M. G. Kleinhans, and E. A. Addink. Network concepts to describe channel importance and change in multichannel systems: Test results for the Jamuna River, Bangladesh. *Earth Surface Processes and Landforms*, 39(6):766–778, 2014. doi:10.1002/esp.3482.

18 P. Passalacqua, S. Lanzoni, C. Paola, and A. Rinaldo. Geomorphic signatures of deltaic processes and vegetation: The Ganges-Brahmaputra-Jamuna case study. *Journal of Geophysical Research*, 118:1838–1849, 2013. doi:10.1002/jgrf.20128.

19 V. Robins, P. J. Wood, and A. 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.

20 N. A. Scoville. *Discrete Morse Theory*, volume 90. American Mathematical Society, 2019.

21 N. Shivashankar, Senthilnathan M., and V. Natarajan. Parallel computation of 2D Morse–Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1757–1770, 2012. doi:10.1109/TVCG.2011.284.

22 N. Shivashankar and V. Natarajan. Parallel computation of 3D Morse–Smale complexes. *Computer Graphics Forum*, 31(3):965–974, 2012. doi:10.1111/j.1467-8659.2012.03089.x.

23 W. Sonke, M. G. Kleinhans, B. Speckmann, W. M. van Dijk, and M. Hiatt. Alluvial connectivity in multi-channel networks in rivers and estuaries. *Earth Surface Processes and Landforms*, 47(2):477–490, 2022. doi:10.1002/esp.5261.

24 A. Tejedor, A. Longjas, I. Zaliapin, and E. Foufoula-Georgiou. Delta channel networks: 1. A graph-theoretic approach for studying connectivity and steady state transport on deltaic surfaces. *Water Resources Research*, 51(6):3998–4018, 2015. doi:10.1002/2014WR016577.

25 A. Tejedor, A. Longjas, I. Zaliapin, and E. Foufoula-Georgiou. Delta channel networks: 2. Metrics of topologic and dynamic complexity for delta comparison, physical inference, and vulnerability assessment. *Water Resources Research*, 51(6):4019–4045, 2015. doi:10.1002/2014WR016604.

26 S. Yu, M. van Kreveld, and J. Snoeyink. Drainage queries in TINs: from local to global and back again. In *Advances in GIS Research II: Proc. 7th International Symposium on Spatial Data Handling*, pages 829–842, 1997.

Banana Trees for the Persistence in Time Series Experimentally

Lara Ost
Faculty of Computer Science, Doctoral School Computer Science, University of Vienna, Austria

Sebastiano Cultrera di Montesano
Eric and Wendy Schmidt Center, Broad Institute of MIT, Cambridge, MA, USA
Harvard, Cambridge, MS, USA

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Abstract

In numerous fields, dynamic time series data require continuous updates, necessitating efficient data processing techniques for accurate analysis. This paper examines the banana tree data structure, specifically designed to efficiently maintain the multi-scale topological descriptor commonly known as *persistent homology* for dynamically changing time series data. We implement this data structure and conduct an experimental study to assess its properties and runtime for update operations. Our findings indicate that banana trees are highly effective with unbiased random data, outperforming state-of-the-art static algorithms in these scenarios. Additionally, our results show that real-world time series share structural properties with unbiased random walks, suggesting potential practical utility for our implementation.

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

Keywords and phrases persistent homology, time series, data structures, computational experiments

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.71

Related Version *Full Version*: `https://arxiv.org/pdf/2405.17920`

Supplementary Material *Software*: `https://github.com/laraost/BananaPersist`
archived at `swh:1:dir:8407d6500f6dbccdb5102f3de3db35a9160b79c8`

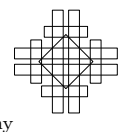
Funding *Lara Ost*: Supported by the Vienna Graduate School on Computational Optimization (VGSCO), FWF project no. W1260-N35.
Sebastiano Cultrera di Montesano: Supported by the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard.
Herbert Edelsbrunner: Partially supported by the Wittgenstein Prize, FWF grant no. Z 342-N31, and by the DFG Collaborative Research Center TRR 109, FWF grant no. I 02979-N35.

1 Introduction

Time series are pervasive across numerous disciplines, ranging from economics and finance to environmental science and healthcare. Often, time series data is dynamic, not static; for example, wearable devices continuously monitor health metrics such as heart rate or blood sugar levels, requiring robust techniques for instant analysis and decision-making. Effective synthesis of the underlying trends in such dynamic data is crucial for predictive modeling and strategic interventions.

Within the field of topological data analysis, persistent homology [4, 11] is recognized as a powerful framework for capturing multi-scale features in complex datasets. Researchers have long considered the challenge of dynamic persistence, and the vineyard algorithm [7]

© Lara Ost, Sebastiano Cultrera di Montesano, and Herbert Edelsbrunner;
licensed under Creative Commons License CC-BY 4.0



was the first developed to address updates in persistence diagrams (defined in Section 2.1) as the input data evolves; see [15] for an implementation. This algorithm, while capable of handling data more general than just time series, requires time linear in the size of the input complex per update, which is restricted to swapping the order of two values. This prompts us to question whether faster processing could be achieved for one-dimensional data. To address this challenge, we implement the banana tree data structure [8], recently introduced and tailored specifically for the persistent homology of dynamic time series. Their theoretical analysis promises the processing of each update in time logarithmic in the number of items plus linear in the number of changes in the persistent diagram; compare this with the linear time require to recompute the diagram [12]. This paper investigates what these results mean in practice. We highlight some of the specific contributions:

- *Efficiency at scale:* we demonstrate that for large datasets, such as those containing over 10^6 items, performing a local or topological update on an unbiased random walk using banana trees is at least 100 times faster than recomputing persistence with `Gudhi` [16], the state of the art static algorithm (see Figure 4).
- *Structure and performance:* we explore how the structure of banana trees, characterized by parameters such as the number of critical items, the nesting depth of bananas, and the lengths of trails, directly impacts the running time of our algorithms. This analysis identifies the types of data best suited for efficient processing (see Figure 2).
- *Worst-case scenarios:* we construct specific examples that challenge our algorithms, and show empirically that these scenarios are rare and highly unstable, reinforcing the robustness of our approach (see Table 2).
- *Real-world data:* through analysis of three specific datasets, we illustrate that real-world data sometimes exhibits structural properties similar to those of unbiased random walks. Preliminary evidence suggests potential for the broad applicability of banana trees in practical scenarios (see Table 3).

In software for topological data analysis, tools like the `Gudhi` library [16], Dionysus [17, 18], and Ripser [2] are established for static datasets but require complete recomputation for updates, making them less suited in dynamic settings. We aim for our implementation to set a new standard in the topological processing of dynamic time series.

Outline. Section 2 introduces persistent homology tailored to time series data and gives an overview of the banana trees data structure. Section 3 evaluates the performance of banana trees through experiments involving time series generated from random walks, with and without a bias. Section 4 explores three distinct types of input time series to assess the performance of banana trees: worst-case scenarios, quasi-periodic signals, and real-world data. Section 5 concludes the paper with a summary of our findings.

2 Banana Trees for Time Series

We start by introducing persistent homology, a method to discern features of data across multiple scales [10]. Forfeiting the generality of this theory, we focus on time series data. We also explain what banana trees are and how they relate to persistent homology; see [8] for details on this data structure and its algorithms.

2.1 Persistent Homology of Time Series

By a *time series* we mean a linear list of real numbers, $c_0, c_2, \ldots, c_{n-1}$, which we view as a piecewise linear map, $f\colon [0, n-1] \to \mathbb{R}$, with $f(i) = c_i$ for $0 \leq i \leq n-1$. We refer to i as an *item* and c_i its *value*. A *critical item* is a local minimum or a local maximum of this map,

and all other items are *non-critical*, e.g. item i if $f(i-1) < f(i) < f(i+1)$. To simplify the discussion, we assume that the map is *generic*, by which we mean that its items have distinct values. In this case, the endpoints (items 0 and $n-1$) are necessarily critical.

The *sublevel set* of f at $t \in \mathbb{R}$, denoted $f_t = f^{-1}(-\infty, t]$, are the points $x \in [0, n-1]$ that satisfy $f(x) \leq t$. When t passes the value of a minimum from below, then the number of connected components of f_t increases by 1, and if t passes the value of a maximum from below, the number of connected components decreases by 1, unless the maximum is an endpoint, in which case the number does not change. Symmetrically, we call $f^t = f^{-1}[t, \infty)$ the *superlevel set* of f at t. Observe that f^t is the sublevel set of $-f$ at $-t$, and that the minima and maxima of $-f$ are the maxima and minima of f. Persistent homology tracks the evolution of the connected components while the sublevel set of f grows, and formally defines when a component is born and when it dies. Complementing this with the same information for the superlevel sets of f, we get what is formally referred to as *extended persistent homology*; see [6] for details. It is best constructed in two phases:

- In *Phase One*, we track the connected components of the sublevel set, f_t, as t increases from $-\infty$ to ∞. A component is *born* at the smallest value of t at which a point of the component belongs to f_t, which is necessarily a minimum. The component *dies* when it merges with another component that was born earlier, which is necessarily at a maximum in the interior of $[0, n-1]$. The *ordinary subdiagram* of f records the birth and death of every component with a point in the plane whose abscissa and ordinate are those values of t at which the component is born and dies, respectively; see Figure 1.
- In *Phase Two*, we track the connected components of the superlevel set, f^t, as t decreases from ∞ to $-\infty$. Birth and death are defined accordingly, and the components are recorded in the *relative subdiagram*.

By construction, the points in the ordinary subdiagram lie above and those of the relative subdiagram lie below the diagonal. The component born at the global minimum of f is special because it does not die during Phase One. Instead, it dies at the global minimum of $-f$, which is the global maximum of f. In topological terms, this happens because the one connected component still alive at the beginning of Phase Two dies in relative homology when its first point enters the superlevel set. This class is represented by the sole point in the *essential subdiagram*. The *extended persistence diagram* is the disjoint union of the three subdiagrams. Hence, the diagram is a multi-set of points in $\mathbb{R}^2$, and so are the three subdiagrams, unless the map is generic, in which case the diagram is a set. See the left panel in Figure 1 for an example but note that it only shows a small number of points in the ordinary subdiagram. An important property of persistence diagrams is their stability with respect to small perturbation of the input data, which was first proved in [5].

The points in the persistence diagram can be characterized using the concept of a *window* introduced in [3]: start with a rectangular frame spanned by a minimum and a maximum in the graph of f such that the minimum lies on the lower edge, and extend the frame horizontally as long as the graph does not intersect the upper edge in its interior. Call the initial frame the *mid-panel*, its extension the *in-panel*, and the symmetric extension for $-f$ the *out-panel*. As proved in [3], the min-max pair defines a point in the persistence diagram iff the graph of f reaches the lower edge in the out-panel. In this case, we call the twice extended frame a *triple-panel window*. The corresponding *double-panel window* consists of the mid-panel and the in-panel but not the out-panel. Any two double-panel windows of f are either disjoint or nested, a property not shared by the triple-panel windows. We use this property to *augment* the persistence diagram by drawing an arrow from a point to another, if the double-panel windows of the first point is nested inside the double-panel window of

the other point, without any other window being nested between them. See Figure 1, which shows four double-panel windows and the corresponding points and arrows in the augmented persistence diagram. The displayed frames are indeed windows because each has an out-panel – next to the mid-panel and thus opposite to the in-panel – which extends as far as the horizontal projection of the minimum (a maximum of $-f$) to the graph of the function.

2.2 Banana Trees

We sketch the banana trees while referring to [8] for the details. Suffice to say that they are based on the Cartesian tree introduced by Vuillemin [21], which stores a list of items in order such that each path from a leaf to the root is ordered by value. This tree has a unique decomposition into paths that correspond to double-panel windows, and the banana tree splits each path into two trails representing the windows nested inside the in-panel and the mid-panel of the corresponding window.

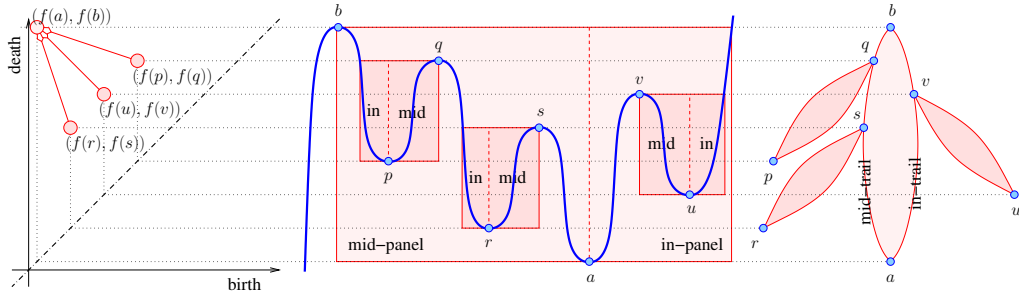


Figure 1 *Middle:* a piece of the graph of f with four double-panel windows of which three are nested inside the fourth. *Left:* the corresponding points in the persistence diagram and the arrows that reflect the nesting relation among the windows. *Right:* the four corresponding bananas in the up-tree of f.

A time series represented by a piecewise linear function, f, is stored in two banana trees together with a linked list and two standard dictionaries. We call the first banana tree the *up-tree* of f because it corresponds to Phase One of the persistence diagram construction. The second banana tree is the *down-tree* of f, which is the up-tree of $-f$. Because of this symmetry, it suffices to describe the up-tree of f. Its *leaves* are the minima of f, and its *internal nodes* are the maxima of f, with the exception of endpoint maxima, if they exist, since they are not critical in Phase One. Each point in the ordinary persistence diagram is a min-max pair, (a, b), and it is represented by a *banana* that connects the leaf a to the internal node b with two parallel trails. The *mid-trail* is a doubly-linked list connecting a to b with the maxima that span windows nested in the mid-panel of the window of a, b, and the *in-trail* is symmetrically defined; see the right panel in Figure 1. The tree structure arises because the maxima belong to the similarly defined bananas of the nested windows. Each trail is sorted in the order of value but also in the order of location along the time series. For example, the mid-trail of the big banana in Figure 1 stores a, s, q, b, which monotonically increases in value and monotonically decreases in location. Compare this with the in-trail storing a, v, b, which monotonically increases in value and in location after we discard b. Indeed, we think of b as part of the mid-trail while only connecting the in-trail to the rest of the tree without properly belonging to it. The global minimum of f (not shown in Figure 1) is special because it is not paired in Phase One. Indeed, it is the extra leaf whose path upward does not end at an internal node. We therefore add a *special root* as the parent of

the root, connected to the global minimum by the *root banana*. While there is only one root banana, there are possibly many *leaf bananas*, which are the bananas with two empty trails (beside the minimum and maximum they connect). Indeed, we call the number of nodes strictly between the minimum and maximum the *length* of the trail, which for trails of leaf bananas is necessarily zero. Another important concept is the *nesting depth* of a banana, which is the number of windows that contain the corresponding window. The root banana has nesting depth zero, and all other bananas have positive nesting depth.

Each update to a time series stored in a banana tree reduces to a sequence of elementary operations, which we name by their impact on the function, f. An *interchange* happens when two maxima or two minima swap the order of their values. Unless their locations are near each other, there is a good chance that an interchange does not have any effect on the banana tree. Otherwise, it is akin a rotation in a binary search tree, if the interchange is between two maxima, and akin a local rearrangement of the path decomposition, if the interchange is between two minima. A *cancellation* removes a min-max pair or, equivalently, the corresponding leaf banana, an *anti-cancellation* is the inverse of a cancellation, and a *slide* swaps the type of a critical item with that of a neighboring non-critical item. The latter three operations appear only a constant number of times whenever we adjust a value. Nevertheless, anti-cancellations present challenges to fast implementation as we have to find out where the new banana is to be inserted, and this search is not supported by any special purpose data structure. On the other hand, multiple interchanges may happen in sequence, so it is important to charge the time for processing each to a change in the persistence diagram. We refer to [8] for details.

3 Experimental Results for Random Walks

This section studies the structural properties of banana trees for random walks, both biased and unbiased. In addition, it compares the running times of local and topological operations with the static algorithm in `Gudhi` [16]. We do not compare it with vineyards [7], which lack optimized software for the one-dimensional case.

We implemented the banana trees data structure in C++20, using the AVL trees and splay trees provided by `boost.intrusive` [14]. The former offer good performance in general, the latter allow for simple splitting and joining. Hence, we use splay trees when performing topological operations and AVL trees everywhere else. The experiments were run on a machine with two AMD EPYC 9534 64 core CPUs and 1.5 terabyte of main memory.

3.1 The Experimental Set-up

We write $\gamma \sim \mathcal{N}(\mu, \sigma)$ for a normally distributed random variable with mean μ and standard deviation σ. Given μ, σ, a *random walk* of length n is a function, $r = r_{\mu,\sigma}$, from the first n non-negative integers to the reals, defined by

$$r(0) = 0 \quad \text{and} \quad r(i) = r(i-1) + \gamma_i \quad \text{for } 1 \leq i < n, \tag{1}$$

in which the $\gamma_i \sim \mathcal{N}(\mu, \sigma)$ are independent. The random walk is *unbiased* if $\mu = 0$ and *biased* if $\mu \neq 0$. We assume $\sigma = 1$ unless stated otherwise.

To evaluate the performance of local updates, we change the value of a single item, and since it depends on the amount, we fix parameters $\Delta \in \mathbb{R}$ and $k \in \mathbb{N}$ and do $2k+1$ parallel updates by changing the value of the i-th item from $r(i)$ to $r(i) + \delta_j$, with $\delta_j = \frac{j}{k}\Delta$ for all integers j with $-k \leq j \leq k$. For the experiments, we pick $\Delta = 5.0$ or 50.0 and $k = 10$, while doing the update on a random walk of length between 10^2 and 10^6 generated for $\mu \in [-1, 1]$

and $\sigma = 1$. Each experiment is repeated one hundred times and averages as well as deviations from the average are reported. For each experiment, we also measure the time to run `Gudhi` on the walk after the value change. This provides the appropriate reference time, since `Gudhi` needs to recompute the persistence diagram from scratch after each update.

To evaluate the performance of topological updates, we pick a fraction, $c \in (0, 1)$, generate a random walk of length n, and then cut it into a *left walk* containing the first $\lfloor cn \rfloor$ items and a *right walk* containing the remaining $\lceil (1-c)n \rceil$ items. There are two such updates, which we now define. To *split*, we first construct the banana tree of the entire random walk, which we then cut into two banana trees, one for the left and the other for the right walk. To *glue*, we first construct the banana trees of the left and right walks, which we then combine to a single banana tree for the walk of length n. Doing this for fractions $c \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, we compare the time to split with the time used by `Gudhi` to construct the persistence diagrams of the left and right walks, and the time to glue with the time used by `Gudhi` to construct the persistence diagram of the entire random walk.

3.2 Structural Properties

We focus on three structural properties of banana trees, which have direct implications for the running time of our algorithms: the *number of critical items*, the *nesting depth of bananas*, and the *lengths of trails*. Since a banana tree does not store non-critical items, its number of nodes is the number of critical items. In an unbiased random walk, the expected number of critical items is 50% of all items. This fraction decreases when the bias increases, with about 27% for $\mu = \pm 1$; see the left panel in Figure 2 for more detailed information.

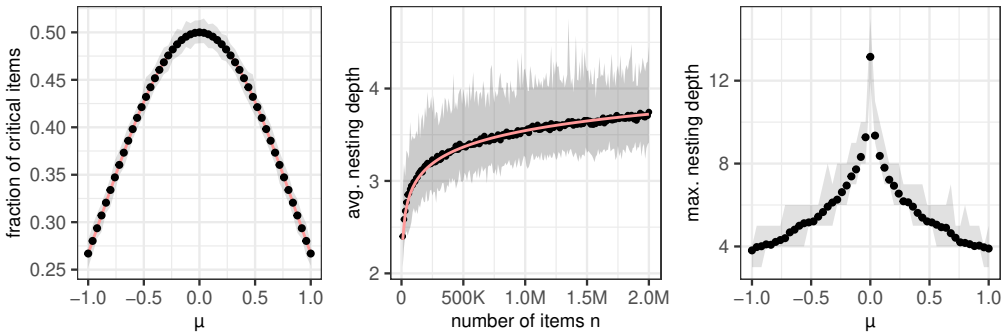


Figure 2 *Left:* the average fraction of items that are critical as a function of the bias μ. *Middle:* the average nesting depth of leaf bananas in banana trees of unbiased random walks with n items. The *ribbon* extends from the minimum to the maximum observed value for each n; the dots mark the mean. The *red line* is the graph of a constant times $\log n$ obtained by linear regression. *Right:* the maximum nesting depth at $n = 10^6$ items as a function of the bias μ.

The middle panel of Figure 2 shows the average nesting depth of a leaf banana for an unbiased random walk, which appears to scale like the logarithm of the number of items. In the unbiased case, even the maximum nesting depth seems to be small (about 13 on average for a random walk of length $2 \cdot 10^6$), and scale logarithmically in the number of items. With increasing bias, the maximum nesting depth decreases, to about 4 at $\mu = \pm 1$; see the right panel of Figure 2. This is because in the biased case, there are fewer critical items but also the bananas tend to arrange in parallel rather than inside each other. We observe that the nesting depth is about the same on average in up-trees and down-trees, both for biased and unbiased random walks, so our results hold for both trees.

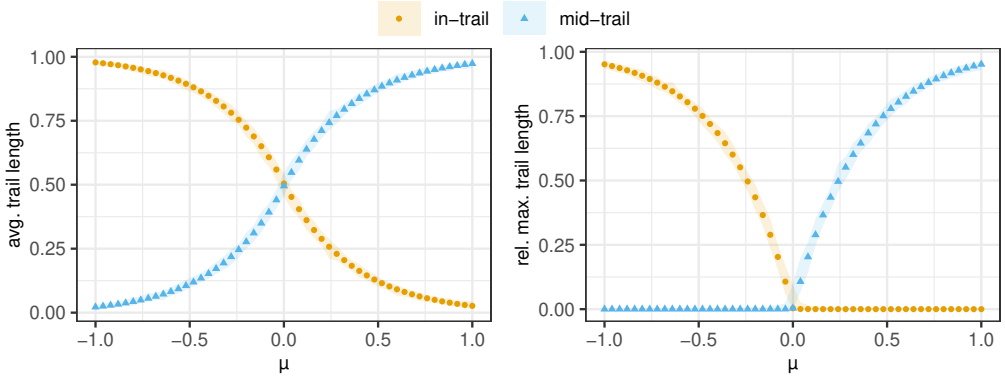


Figure 3 *Left:* the average length of in-trails (*orange dots*) and mid-trails (*blue triangles*) in banana trees of random walks with bias μ, averaged over all input sizes. *Right:* the fraction of nodes on the longest in-trail (*orange dots*) and longest mid-trails (*blue triangles*) in banana trees of random walks with bias μ, averaged over all input sized.

For topological reasons, the mid-trails of the up-tree are the mid-trails of the down-tree, except that they are upside down. In contrast, the in-trails of the two trees are not directly related because the in-panels in one direction become the out-panels in the other. Notwithstanding, we observe that the length of the in-trails in the up-tree and the mid-trails in the down tree are very similar on the average, and by symmetry the same holds for the mid-trails in the up-tree and the in-trails in the down-tree. These two facts do not contradict each other, because for biased data most of the critical items belong to the root banana, which does not adhere to the topological constraints just mentioned. Let us therefore focus on the up-tree. The left panel in Figure 3 shows how the average length of in- and mid-trails depends on the bias. In the unbiased case, both types of trails have the same length on average. For positive μ, the mid-trails tend to be longer than the in-trails, while the reverse happens for negative μ. The main cause for this trend is the longest trail, which starts to dominate the others in length as the bias increases; see the right panel in Figure 3. Assuming $\mu > 0$, the global minimum and maximum tend to lie to the left and right of the middle, respectively, with the majority of the items between them. By convention, the connecting in-trail and mid-trail contain the items to the left and right of the global minimum, respectively. This explains why the mid-trail dominates in this case, and why the in-trail dominates when $\mu < 0$. Indeed, for $\mu = \pm 1$, we observe about 95% of the internal nodes (the local maxima) on the longest trail.

3.3 Local Maintenance

We evaluate the performance of banana trees when the value of a single item changes by δ. Note that the corresponding update is a combination of some number of interchanges and at most one cancellation, at most one anti-cancellation, and at most two slides [8]. Not surprisingly, the time increases with the amount of change. For example, there is no effect at all on the persistence diagram for about 70% of the updates with $\delta = \pm 0.26$, for about 36% of the updates with $\delta = \pm 0.79$, but only about 1% for updates with $\delta = \pm 3.4$. Taking this into account, it is not surprising that banana trees are faster than `Gudhi` when $\delta = \pm 0.26$ for all lengths of random walks we tried.

The left panel in Figure 4 focuses on the case $\delta = \pm 5.0$. As indicated by the vertical dashed and dotted lines, banana trees are faster than `Gudhi` for all random walks of length $n \geq 1059$, and faster in the median for all $n \geq 144$. Not shown in the panel is that the

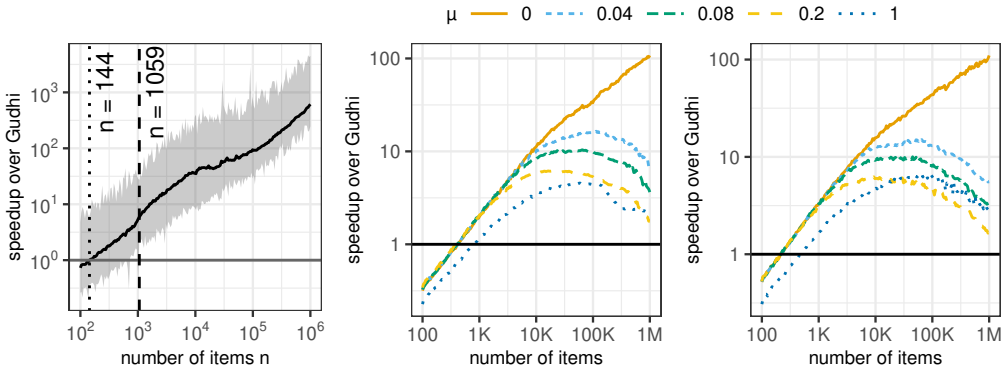


Figure 4 Comparing the maintenance of banana trees with reconstructing the persistence diagram with `Gudhi`, in which the baseline of no speedup ($10^0 = 1$) is marked with a *horizontal gray* line. *Left:* the speedup for updating a value with $\delta = \pm 5.0$ depending on the length of the random walk, n. The ribbon spans the minimum and maximum observed speedup, with the *black curve* tracing the median speedup. *Middle:* the speedup for using banana trees to cut a random walk with bias μ in half. The type and color of the curve encodes the amount of bias, and each curve shows the median speedup over a hundred repeats for each n. *Right:* the speedup for using banana trees to concatenate two equally long random walks with bias μ.

speedup decreases with increasing bias. In particular, for $n = 10^6$, the speedup of 612 at $\mu = 0$ decreases to 258 at $\mu = 1$, and for $n = 1059$, the speedup of 6.0 at $\mu = 0$ decreases to 2.9 at $\mu = 1$. This can be explained by recalling that for large bias the majority of the internal nodes belong to the longest trail, which connects the global minimum to the global maximum. Hence, items with similar values are likely to be near each other and thus require interchanges when an update is performed. However, even for a large change, such as for $\delta = \pm 50.0$, maintaining the banana tree is still faster than reconstructing the persistence diagram with `Gudhi`. For example, the median speedup for $n = 10^4$ exceeds 10 and for $n = 10^6$ it exceeds 100.

3.4 Topological Maintenance

We call the operations of *cutting* a list into two, and *concatenating* two lists to one topological because they change the number of lists in the overall organization of the data. We begin with evaluating the performance of cutting a random walk. The middle panel in Figure 4 shows the speedup over `Gudhi` when we cut a biased or unbiased random walk in half. In the unbiased case, the speedup increases with the length of the walk. Recalling the discussion of structural properties, this can be rationalized by noticing that the maximum nesting depth bounds the number of bananas that need to be split, and the trail length bounds the time to reorganize bananas. The maximum nesting depth scales like $\log n$ and the average trail length is only 0.5, so we anticipate logarithmic time for splitting, which we indeed observe in our experiments. Altering the position of the cut increases the speedup, namely by about 2% for $c = 0.5 \pm 0.2$ and by about 7% for $c = 0.5 \pm 0.4$.

The advantage of the banana tree over `Gudhi` diminishes when the random walks get progressively more biased. The reason is again that the larger the bias, the larger the fraction of internal nodes in the longest trail of the banana tree. Splitting the corresponding banana requires the resetting of many pointers stored in nodes along this trail, which in some cases costs time proportional to the number of critical items, which we indeed observe in our

experiments. Unfortunately, this more than compensates for the low nesting depth, which guarantees that only a small number of bananas need to be split. The upper half of Table 1 shows the running time and the speedup over `Gudhi` for cutting a random walk in half.

Table 1 *Upper half*: the average time for cutting a random walk of length $n = 10^6$ in half, and the speedup over `Gudhi`. *Lower half*; the average time for concatenating two random walks of length $n = 10^6/2$ each, and the speedup over `Gudhi`.

		$\mu = 0$	$\mu = 0.04$	$\mu = 0.08$	$\mu = 0.2$	$\mu = 1$
Cut	time in μs	93	1417	2807	5350	1683
	speedup	105.00	7.24	3.54	1.68	2.15
Concatenate	time in μs	90	1809	3041	5138	1228
	speedup	105.00	5.31	3.12	1.74	3.22

We finally address the reverse operation, that of concatenating two random walks or, correspondingly, of gluing two banana trees. The curves in the right panel of Figure 4 are similar to those in the middle panel, which suggests that the performance is similar to that of cutting, which is confirmed by the numbers in Table 1. For $n \leq 10^4$, gluing appears to be slightly faster than splitting, but the difference is less clear already for $n = 10^6$.

4 Special Time Series

This section considers three types of time series: *worst-case examples* to expose when banana trees under-perform, *quasi-periodic data* to illustrate how banana trees reflect periodicity, and *real-world data* to demonstrate that banana trees are not just theory. While banana trees struggle for data of the first type, they mirror the performance observed in unbiased random walks for the latter two types.

4.1 Worst-Case Examples

The size of the augmented persistence diagram (number of points and arrows) is proportional to the number of critical items. There are configurations in which a single update changes almost the entire diagram and thus takes time at least linear in the number of such items. We construct such *worst-case inputs* and study the performance of banana trees when confronted with such data.

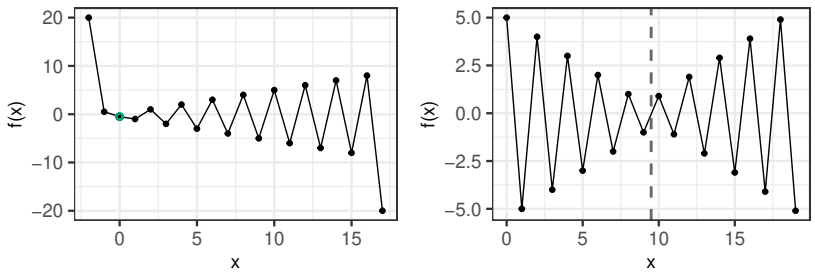


Figure 5 Worst-case examples for local and topological maintenance. *Left:* to increase the value of the marked item triggers a linear number of interchanges. *Right:* to cut the list at the *dashed line* affects every persistent pair. Both operations take time linear in the number of critical items, which for the two time series are all or almost all items.

Consider first the operation that increases the value of the marked item in the left panel of Figure 5. Initially, we have a linear sequence of double-panel windows, each nested inside the next. When the value of the marked item passes that of its left neighbor, an anti-cancellation adds the banana they span to the banana tree, and its window is nested inside the others. As we increase the value further, the marked item interchanges with the maxima to its right, each time moving up one level within the sequence of windows. When the marked item finally becomes the global maximum, it will have interchanged with all other maxima, which takes $\Theta(n)$ time. For $n = 10^2$ and $n = 10^6$, we measured 3.4μs and 4.7ms on average, respectively. This corresponds to a median speedup by a factor of 0.05 and 0.02 if compared with `Gudhi`, which is really a slowdown by a factor of 20 and 50, respectively.

Table 2 Performance of splitting and gluing banana trees for the example time series in the right panel of Figure 5, showing the median slowdown/speedup if compared to `Gudhi`. The parameter interpolates between these time series at $\lambda = 0$ and unbiased random walks at $\lambda = 1$.

	split				glue			
	$\lambda = 0.0$	0.0001	0.01	1.0	$\lambda = 0.0$	0.0001	0.01	1.0
$n = 10^2$	0.06	0.06	0.04	0.35	0.40	0.44	0.68	2.38
$n = 10^4$	0.04	0.03	5.81	11.20	0.46	0.83	33.6	41.40
$n = 10^6$	0.09	4.71	54.70	111.00	0.46	81.0	255	397.00

Consider second the operation that cuts the time series in the right panel of Figure 5 in the middle, which is marked by the vertical dashed line. We observe an average running time between 10μs at $n = 10^2$ and 72ms at $n = 10^6$. Compare this with an average running time of 7μs at $n = 10^2$ and 48ms at $n = 10^6$ for concatenating the two lists back to the original length. This comparison agrees with the earlier observation that gluing banana trees is slightly faster than splitting them. Table 2 lists the speedup if compared to `Gudhi`, which for a factor less than 1 is really a slowdown. To create a more informative experiment, we interpolate between these time series and unbiased walks, with drastic improvements even for small $\lambda > 0$; see Table 2. Indeed, the noise quickly flattens the banana tree by decreasing the length of trails, which explains the improvement.

4.2 Quasi-Periodic Data

Random walks are not periodic, but real-world time series often exhibit some amount of periodicity, at least approximately. To study banana trees under these conditions, we construct what we call *quasi-periodic* inputs. This term does not have a mathematical definition, and for our experiments just means a periodic signal that is randomly perturbed. Specifically, we generate such input by modifying the iterative construction in (1):

$$r(0) = 0 \quad \text{and} \quad r(i) = r(i-1) + \eta_i \quad \text{for } 1 \leq i < n, \tag{2}$$

in which the $\eta_i \sim \mathcal{N}(\mu_i, \sigma)$ are independent and normally distributed, with the mean changing periodically, $\eta_i = \sin(2\pi\omega i)$, as governed by the *frequency parameter* $0 \leq \omega < 1$, which we fix to $\omega = 5/n$. It will be useful to compare the influence of the standard deviation, so we no longer assume $\sigma = 1$.

For large σ, the banana trees of the time series show the features we have seen for unbiased random walks, while for small σ, they locally behave like biased random walks. For example, for $\sigma = 1$, we observe about 37% of the items to be critical, while for $\sigma = 0.5$ and $\sigma = 0.02$ only about 21% and 1% of the items are critical, on average. Like the number of critical

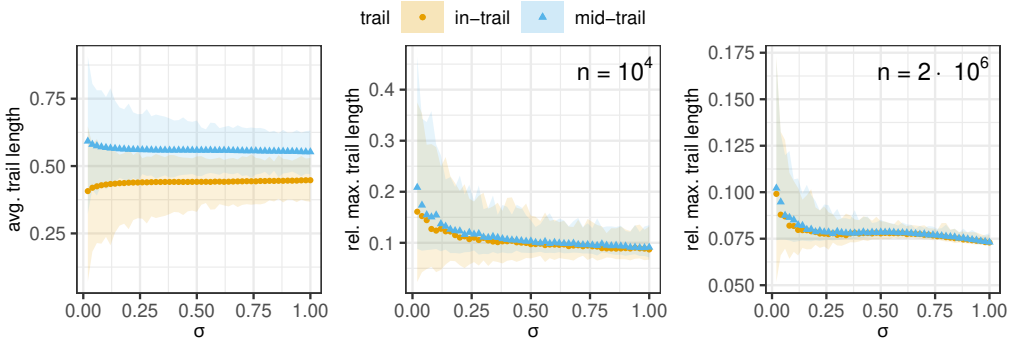


Figure 6 Measuring trail lengths. The in-trails and mid-trails are marked with *orange dots* and *blue triangles*, respectively, and the ribbons extend from the minimum to the maximum observed values. *Left:* the average length of in-trails and mid-trails depending on the standard deviation and averaged over all input sizes. *Middle and right:* the fraction of nodes on the longest in-trail and mid-trail, again depending on the standard deviation but now averaged over inputs of size 10^4 and $2 \cdot 10^6$, respectively.

items, also the maximum nesting depth for quasi-periodic signals is similar to that of random walks: it scales like $\log n$ for large σ and decreases as σ goes to 0. In contrast, the average nesting depth is independent of n, ranging from 2 to 4.7 with mean 2.4. The behavior of the trail length is illustrated in Figure 6. As shown in the left panel, the average trail length is about 0.5, with consistently shorter in-trails than mid-trails, on average. This difference is more pronounced for small σ, when the quasi-periodic signal behaves more like a biased rather than unbiased random walk. The middle and right panels show how the fraction of internal nodes on the longest trail increases as σ goes to 0.

We conclude that banana trees on quasi-periodic signals with large standard deviation resemble unbiased random walks in terms of their structural parameters, and drift toward biased random walks as the standard deviation goes to 0.

4.3 Real World Examples

We analyze structural properties of banana trees constructed on three types of real-world time series:

- electrocardiography (ECG) data from PhysioNet [13, 1, 20]; limiting ourselves to recordings of length at least 10^5, each separated into up to 12 leads, gives 32443 time series in total;
- audio data from EasyCom [9], of which we use 1336 audio files;
- physical activity data from PAMAP2 [19] (temperature, heart rate along with recordings from accelerometers, gyroscopes, and magnetometers), of which we use 560 time series.

We preprocess the data by adding uniformly distributed noise to ensure all values are distinct, making sure the noise is small enough so it breaks ties but does not otherwise alter the ordering.

We observe that the average max nesting depth for the three data sets is only a small factor larger than for unbiased random walks; see the middle three columns in Table 3. The largest maximum nesting depth for PhysioNet data is a bit higher, by a factor 6.3, but 99% of the time series from this data have maximum nesting depth at most 1.68 times that for unbiased random walks. The average trail lengths are again similar to those of

Table 3 Structural properties of the banana trees constructed for time series from three databases. The average, minimum, and maximum of the maximum nesting depth is measured relative to the depth we observe for unbiased random walks with the same fraction of critical items.

	fraction of critical items	max nesting depth			average length	
		avg	min	max	in-trail	mid-trail
PhysioNet	0.24	1.2	0.8	6.3	0.48	0.52
EasyCom	0.22	2.7	1.6	4.9	0.48	0.52
PAMAP2	0.48	1.9	1.2	3.2	0.49	0.51

unbiased random walks – see the last two columns, with standard deviation about 0.05 throughout – and so are the fractions of items in the longest trails, whose geometric means are $0.8\%, 0.008\%, 0.1\%$ of all critical items for the in-trails and $0.8\%, 0.008\%, 0.2\%$ of all critical items for the mid-trails in the three data sets, compared with about 0.3% and 0.4% in unbiased random walks.

5 Discussion

We provide an implementation of the banana tree data structure and supply experimental evidence demonstrating its efficiency. The work reported in this paper confirms the theoretical advantages of banana trees over static alternatives and opens up avenues for further theoretical inquiries and practical applications. One intriguing question that arises in the context of quasi-periodic data is whether we can determine the (possibly varying) period of the data without prior knowledge, or provide meaningful quantification of the extent to which a signal deviates from being periodic.

The potential applications of banana trees extend beyond academic research into practical, real-world scenarios. Comparisons with other methods and identifying impactful use cases represents a promising next step to provide valuable insights in fields such as healthcare, where real-time analysis of patient data is crucial, or finance, where the swift analysis of market data is essential.

References

1 Erick A. Perez Alday, Annie Gu, Amit J. Shah, Chad Robichaux, An-Kwok Ian Wong, Chengyu Liu, Feifei Liu, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Qiao Li, Ashish Sharma, Gari D. Clifford, and Matthew A. Reyna. Classification of 12-lead ECGs: the PhysioNet/Computing in Cardiology Challenge 2020. *Physiological Measurement*, 41(12):124003, December 2020. doi:10.1088/1361-6579/abc960.

2 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, September 2021. doi:10.1007/s41468-021-00071-5.

3 Ranita Biswas, Sebastiano Cultrera di Montesano, Herbert Edelsbrunner, and Morteza Saghafian. Geometric characterization of the persistence of 1D maps. *Journal of Applied and Computational Topology*, June 2023. doi:10.1007/s41468-023-00126-9.

4 Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. doi:10.1090/S0273-0979-09-01249-X.

5 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, January 2007. doi:10.1007/s00454-006-1276-5.

6 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, February 2009. doi:10.1007/s10208-008-9027-z.

7 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, June 2006. doi:10.1145/1137856.1137877.

8 Sebastiano Cultrera di Montesano, Herbert Edelsbrunner, Monika Henzinger, and Lara Ost. Dynamically maintaining the persistent homology of time series. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 243–295, January 2024. doi:10.1137/1.9781611977912.11.

9 Jacob Donley, Vladimir Tourbabin, Jung-Suk Lee, Mark Broyles, Hao Jiang, Jie Shen, Maja Pantic, Vamsi Krishna Ithapu, and Ravish Mehra. EasyCom: An augmented reality dataset to support algorithms for easy communication in noisy environments, 2021. arXiv:2107.04174.

10 Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, January 2022.

11 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.

12 Marc Glisse. Fast persistent homology computation for functions on $\mathbb{R}$, 2023. doi:10.48550/arXiv.2301.04745.

13 Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23):e215–e220, 2000. doi:10.1161/01.CIR.101.23.e215.

14 Olaf Krzikalla and Ion Gaztanaga. Boost.Intrusive C++ library. http://www.boost.org/. (version 1.74).

15 Yuan Luo and Bradley J. Nelson. Accelerating iterated persistent homology computations with warm starts. *Computational Geometry*, 120:102089, June 2024. doi:10.1016/j.comgeo.2024.102089.

16 Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The Gudhi library: Simplicial complexes and persistent homology. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 167–174, 2014. doi:10.1007/978-3-662-44199-2_28.

17 Dmitriy Morozov. Dionysus. https://mrzv.org/software/dionysus/, 2023.

18 Dmitriy Morozov. Dionysus 2. https://mrzv.org/software/dionysus2/, 2023.

19 Attila Reiss. PAMAP2 physical activity monitoring. UCI Machine Learning Repository, 2012. doi:10.24432/C5NW2H.

20 Matthew A. Reyna, Nadi Sadr, Erick A. Perez Alday, Annie Gu, Amit J. Shah, Chad Robichaux, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Sardar Ansari, Hamid Ghanbari, Qiao Li, Ashish Sharma, and Gari D. Clifford. Will two do? varying dimensions in electrocardiography: The PhysioNet/Computing in Cardiology Challenge 2021. In *2021 Computing in Cardiology (CinC)*, volume 48, pages 1–4, September 2021. doi:10.23919/CinC53138.2021.9662687.

21 Jean Vuillemin. A unifying look at data structures. *Communications of the ACM*, 23(4):229–239, April 1980. doi:10.1145/358841.358852.

Geometric Bipartite Matching Based Exact Algorithms for Server Problems

Sharath Raghvendra
North Carolina State University, Raleigh, NC, USA

Pouyan Shirzadian
Virginia Tech, Blacksburg, VA, USA

Rachita Sowle
Virginia Commonwealth University, Richmond, VA, USA

Abstract

For any given metric space, obtaining an offline optimal solution to the classical k-server problem can be reduced to solving a minimum-cost partial bipartite matching between two point sets A and B within that metric space.

For d-dimensional ℓ_p metric space, we present an $\tilde{O}(\min\{nk, n^{2-\frac{1}{2d+1}} \log \Delta\} \cdot \Phi(n))$ time algorithm for solving this instance of minimum-cost partial bipartite matching; here, Δ represents the spread of the point set, and $\Phi(n)$ is the query/update time of a d-dimensional dynamic weighted nearest neighbor data structure. Our algorithm improves upon prior algorithms that require at least $\Omega(nk\Phi(n))$ time. The design of minimum-cost (partial) bipartite matching algorithms that make sub-quadratic queries to a weighted nearest-neighbor data structure, even for bounded spread instances, is a major open problem in computational geometry. We resolve this problem at least for the instances that are generated by the offline version of the k-server problem.

Our algorithm employs a hierarchical partitioning approach, dividing the points of $A \cup B$ into rectangles. It maintains a partial minimum-cost matching where any point $b \in B$ is either matched to another point $a \in A$ or to the boundary of the rectangle it is located in. The algorithm involves iteratively merging pairs of rectangles by erasing the shared boundary between them and recomputing the minimum-cost partial matching. This continues until all boundaries are erased and we obtain the desired minimum-cost partial matching of A and B. We exploit geometry in our analysis to show that each point participates in only $\tilde{O}(n^{1-\frac{1}{2d+1}} \log \Delta)$ number of augmenting paths, leading to a total execution time of $\tilde{O}(n^{2-\frac{1}{2d+1}} \Phi(n) \log \Delta)$.

We also show that, for the ℓ_1 norm and d dimensions, any algorithm that can solve instances of the offline n-server problem with an exponential spread in $T(n)$ time can be used to compute minimum-cost bipartite matching in a complete graph defined on two $(d-1)$-dimensional point sets under the ℓ_1 norm within $T(n)$ time. This suggests that removing spread from the execution time of our algorithm may be difficult as it immediately results in a sub-quadratic algorithm for bipartite matching under the ℓ_1 norm.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Minimum-Cost Bipartite Matching, Server Problems, Primal-Dual Approach

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.72

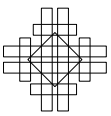
Related Version *Full Version*: https://arxiv.org/abs/2504.06079 [19]

Funding This research was supported by NSF grants CCF 2514753 and CCF 2223871.

1 Introduction

This paper considers two classical optimization problems: the *offline k-server* problem and the *minimum-cost bipartite matching* problem in geometric settings.

© Sharath Raghvendra, Pouyan Shirzadian, and Rachita Sowle;
licensed under Creative Commons License CC-BY 4.0



Offline k-server problem and its variant. Consider a sequence of requests $\varsigma = \langle r_1, \ldots, r_m \rangle$ in a metric space equipped with the cost function $\mathrm{d}(\cdot, \cdot)$. The cost of a single server *servicing* the requests in ς is the sum of the distances between every consecutive pair of points in the sequence, i.e., $\mathfrak{c}(\varsigma) = \sum_{i=1}^{m-1} \mathrm{d}(r_i, r_{i+1})$. Given a sequence $\sigma = \langle r_1, \ldots r_n \rangle$ of n requests and an integer $1 \leq k \leq n$, the *k-sequence partitioning problem* (or simply the k-SP problem) requires partitioning the requests in σ into k subsequences $\varsigma_1, \ldots, \varsigma_k$ so that $\sum_{i=1}^{k} \mathfrak{c}(\varsigma_i)$ is minimized. The optimal solution for the k-sequence partitioning problem is the cheapest way for k servers to service all of the requests in σ. We also consider a variant: the *k-sequence partitioning with initial locations problem* (or the k-SPI problem). Here, in addition to the requests σ, we are also given the initial locations $\eta = \langle s_1, \ldots, s_k \rangle$ for the k servers. The objective of this problem is to partition $\sigma' = \eta\sigma$ into k subsequences $\varsigma_1, \ldots, \varsigma_k$ so that the initial location s_j of server j appears in the first element of the subsequence ς_j and the cost $\sum_{i=1}^{k} \mathfrak{c}(\varsigma_k)$ is minimized. The optimal solution to the k-SPI problem is the offline optimal solution to the well-known k-server problem.

We assume that the requests in σ and the initial locations in η are scaled and translated so that they are contained inside the unit hypercube $[0, 1]^d$. Such scaling and translation do not impact the optimal solutions for the k-SP and k-SPI problems. We define the diameter $\mathrm{Diam}(\sigma)$ to be the largest distance between any two request locations in σ and the *closest pair distance*, denoted by $\mathrm{CP}(\sigma)$, to be the smallest non-zero distance between any two requests in σ. The *spread*, denoted by Δ, is the ratio of the diameter to the closest-pair distance, i.e., $\Delta = \mathrm{Diam}(\sigma)/\mathrm{CP}(\sigma)$.

Minimum-cost bipartite matching. Consider a weighted bipartite graph $G(A \cup B, E \subseteq A \times B)$, where each edge between u and v is assigned a real-valued cost. Let $n := \min\{|A|, |B|\}$. A *matching* of size $t \leq n$, or simply a *t-matching*, is a set of t edges in G that are vertex-disjoint. The *cost* of any matching M is the sum of the costs of its edges. Given a parameter $t \leq n$, the *minimum-cost bipartite t-matching* problem seeks to find a t-matching with a minimum cost. When $|A| = |B| = n$, a matching of size n is called a *perfect matching*. When A and B are d-dimensional point sets and the cost of any edge between $a \in A$ and $b \in B$ is the ℓ_p distance $\|a - b\|_p$, the problem of finding the minimum-cost bipartite t-matching is also called the (partial) *geometric bipartite matching problem*.

Relating the two problems. Chrobak *et al.* [4] established a reduction from the minimum-cost bipartite t-matching problem to the k-SPI problem.

▶ **Lemma 1** ([4, Theorem 11]). *Any algorithm that computes an optimal solution to the n-SPI in an arbitrary metric space in $T(n)$ time can also find, in $T(n) + O(n^2)$ time, a minimum-cost perfect matching in any complete bipartite graph with real-valued costs.*

We strengthen the connection between the two problems by showing a reduction in the reverse direction, i.e., we reduce the k-SP (resp. k-SPI) problem to the minimum-cost bipartite t-matching problem. Given an input sequence σ of requests to the k-SP problem, we construct a bipartite graph $\mathcal{G}_\sigma$ with a vertex set $A \cup B$ and a set of edges $\mathcal{E}$ as follows. *Vertex Set:* For each request r_i, we create a vertex b_i (resp. a_i) in B (resp. A) and designate it as the *entry* (resp. *exit*) gate for request r_i. *Edge Set:* The exit gate a_i of request r_i is connected to the entry gate b_j of every subsequent request r_j with $j > i$ with an edge. The cost of this edge is $\mathrm{d}(a_i, b_j) = \|r_i - r_j\|_p$. Any minimum-cost $(n - k)$-matching M in $\mathcal{G}_\sigma$ can be used to find an optimal solution to the k-SP problem. See Figure 1.

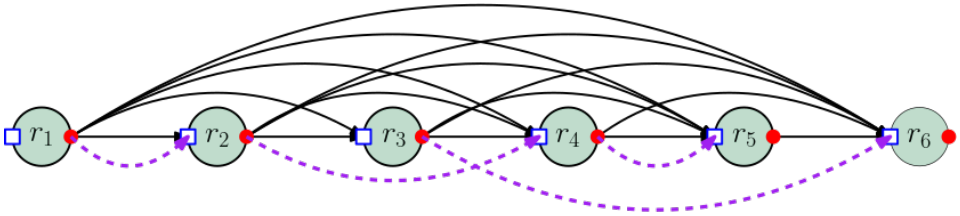


Figure 1 The graph $\mathcal{G}_\sigma$ constructed for $\sigma = \langle r_1, r_2, \ldots, r_6 \rangle$. The vertex set A (red disks) and B (blue squares) represent the exit and entry gates of each request, and the purple dashed lines show an $(n-2)$-partial matching on $\mathcal{G}_\sigma$ representing a 2-partitioning $\langle r_1, r_2, r_4, r_5 \rangle$ and $\langle r_3, r_6 \rangle$.

For the k-SPI problem, for each server j, we add a vertex a^j at the initial location s_j to A and connect a^j to the entry gate b_i of every request r_i in σ. The cost of this edge is $\mathrm{d}(a^j, b_i) = \|s_j - b_i\|_p$. A minimum-cost n-matching in this graph can be converted to an optimal solution for the k-SPI problem.

A different reduction from k-SP and k-SPI problems to the minimum-cost flow problem has been presented in previous works [4, 22, 24], leading to the development of $O(n^2k)$ time algorithm. However, unlike our reduction, their approach generates instances that include edges whose costs are $-\infty$ and fails to maintain the metric properties of costs.

▶ **Lemma 2.** *An optimal solution for an instance σ (resp. σ') of k-SP (resp. k-SPI) problem can be found by computing a minimum-cost matching of size $n-k$ (resp. n) in $\mathcal{G}_\sigma$ (resp. $\mathcal{G}_{\sigma'}$).*

Finally, we extend the reduction Chrobak *et al.* [4] to geometric settings and provide a reduction from the geometric minimum-cost matching problem under the ℓ_1-norm to the geometric version of the n-SPI problem. This reduction, however, creates an instance of the n-SPI problem with a spread of 3^n. Lemma 3 follows directly from this reduction.

▶ **Lemma 3.** *Any algorithm that can solve the d-dimensional n-SPI problem under the ℓ_1 costs in $T(n)$ time can also be used to solve an instance of minimum-cost bipartite matching under the ℓ_1 costs on a complete graph in $T(n)$ time.*

Related work. For a graph with m edges and n vertices, the classical Hungarian algorithm computes a minimum-cost t-matching for all values of $0 \le t \le n$ [11, 16]. Starting with an empty matching, the Hungarian algorithm iteratively builds a minimum-cost i-matching from the maintained minimum-cost $(i-1)$-matching in $O(m + n \log n)$ time by finding a *minimum net-cost augmenting path*, i.e., an augmenting path that increases the matching cost by the smallest value. The overall execution time of the Hungarian algorithm is $O(nm + n^2 \log n)$, or $O(n^3)$ when $m = \Theta(n^2)$. Despite substantial efforts, this remains the most efficient algorithm for the problem. Notable exceptions include specialized cases, such as graphs with small vertex separators [13] or integer edge weights [2, 5, 6].

In geometric settings, Vaidya showed that each iteration of the Hungarian algorithm can be implemented in $\tilde{O}(n\Phi(n))$ time, where $\Phi(n)$ represents the query/update time of a dynamic weighted nearest neighbor (DWNN) data structure. Thus, the minimum-cost t-matching can be computed in $O(n^2\Phi(n))$ time, which is sub-cubic in n provided $\Phi(n)$ is sub-linear. For instance, for the ℓ_1 norm and d dimensions, $\Phi(n) = O(\log^d n)$ [27] and for the ℓ_2 norm and 2 dimensions, $\Phi(n) = \log^{O(1)} n$ [8]. In these cases, the Hungarian algorithm can be implemented in near-quadratic time.

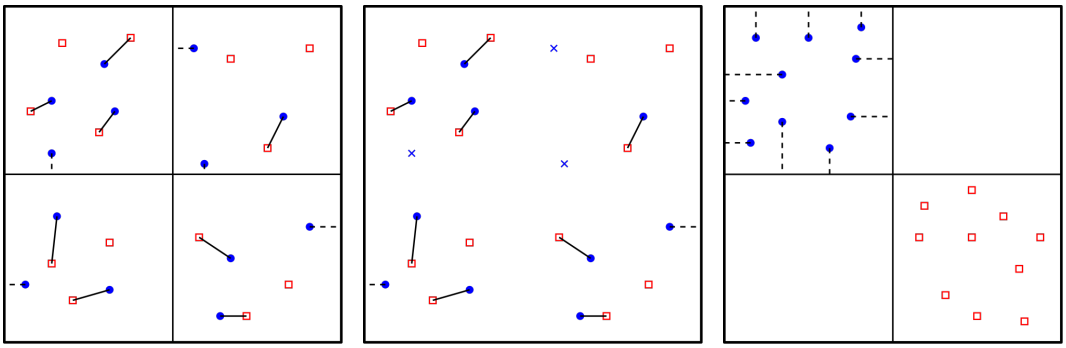


Figure 2 Illustration of the GRS algorithm.

Designing algorithms that compute a minimum-cost matching using sub-quadratic DWNN queries remains a central open problem in computational geometry. There are three notable exceptions. First, for points with integer coordinates and the ℓ_1 and ℓ_∞ norms, Sharathkumar and Agarwal [26] adapted a cost-scaling algorithm [5] and presented an algorithm that executes in $\tilde{O}(n^{3/2}\Phi(n))$ time. Second, Sharathkumar [25] extended this result to 2-dimensional point sets with integer coordinates and the ℓ_2 costs. Their result, however, relies on planarity as well as the edge costs being the square root of integers and does not extend to d-dimensional points with real-valued coordinates. Third, Gattani *et al.* [7] presented a divide-and-conquer algorithm (GRS algorithm) with a worst-case runtime of $\tilde{O}(n^2\Phi(n)\log\Delta)$, where Δ is the spread of the points. For 2-dimensional *stochastic* points drawn from an unknown distribution, however, the expected runtime of the GRS algorithm improves to $\tilde{O}(n^{7/4}\Phi(n)\log\Delta)$.

The GRS algorithm uses a randomly-shifted quadtree to divide the problem into smaller subproblems. Within each square $\square$ of the quadtree, it recursively computes a *minimum-cost extended matching*, allowing each point of B in $\square$ to match either to a point of A in $\square$ or to the boundaries of $\square$ (Figure 2(left)). The algorithm then combines the matchings of the child squares by *erasing* their common boundaries, freeing points of B that were matched to those boundaries (Figure 2(middle)), and then it iteratively matches the freed points.

Gattani *et al.* [7] showed that for stochastic point sets A and B, most points in B will match to a close-by point in A and only $\tilde{O}(n^{3/4})$ points will match to boundaries. Thus, erasing these boundaries creates $\tilde{O}(n^{3/4})$ free points, each again matched in $O(n\Phi(n))$ time, resulting in an overall runtime of $\tilde{O}(n^{7/4}\Phi(n))$. However, this efficiency does not extend to arbitrary point sets. For instance, if all points in A lie in one child $\square_1$ and all points in B lie in another child $\square_2$, then all edges of a minimum-cost matching cross the boundaries. In this case, the minimum-cost extended matching in $\square_2$ matches all points in B to its boundary (Figure 2(right)), and erasing the boundary creates n free points. Hence, the merge step takes $\Omega(n^2\Phi(n))$ time. Furthermore, unlike the Hungarian algorithm, the GRS algorithm does not guarantee optimal intermediate matchings and cannot compute minimum-cost t-matchings.

Our results. The optimal solutions to the k-SP (resp. k-SPI) problems can be computed in $\tilde{O}(nk\Phi(n))$ time by non-trivially adapting the Hungarian algorithm to find a minimum-cost $(n-k)$-matching (resp. n-matching) in $\mathcal{G}_\sigma$ (resp. $\mathcal{G}_{\sigma'}$). This algorithm, however, makes quadratic queries to a DWNN data structure when $k = \Theta(n)$. The main contribution of this paper is the design of a novel algorithm that, for any k, computes the optimal solution to the k-SP and k-SPI problems using only a sub-quadratic number of DWNN queries.

▶ **Theorem 4.** *Given any sequence σ (resp. $\sigma' = \eta\sigma$) of n requests (resp. n requests and k initial server locations) in 2 dimensions with a spread of Δ, and a value $1 \leq k \leq n$, there exists a deterministic algorithm that computes the optimal solution for the instance of k-SP (resp. k-SPI) problem under the ℓ_p norm in $\tilde{O}(\min\{nk, n^{1.8}\log\Delta\}\cdot\Phi(n))$ time.*

Our algorithm also extends to higher dimensions, and for any dimension $d \geq 2$, it computes optimal solutions to the k-SP and k-SPI problems in $\tilde{O}(\min\{nk, n^{2-\frac{1}{2d+1}}\log\Delta\}\cdot\Phi(n))$ time.

Designing geometric bipartite matching algorithms that perform a sub-quadratic number of queries to a DWNN data structure is challenging. Nevertheless, Theorem 4 shows that the bipartite matching instances generated by k-SP and k-SPI problems with bounded spread can be solved using sub-quadratic DWNN queries. For unbounded spread cases, the challenge remains elusive, at least for the k-SPI problem since, by Lemma 3, such an algorithm yields a sub-quadratic solution for the d-dimensional minimum-cost bipartite matching problem under ℓ_1 costs.

Our algorithm uses a hierarchical partitioning tree, whose nodes (referred to as cells) are axis-parallel rectangles with an aspect ratio of at most 3, and each cell splits into two smaller rectangles, forming its children. Our algorithm maintains a set of *current* cells $\mathcal{C}$ that partition the input points and computes a minimum-cost extended $(n-k)$-matching, where the points in B are allowed to match to the cell boundaries. It then replaces a pair of sibling rectangles in $\mathcal{C}$ with their parent, effectively erasing their shared boundary and creating additional free points. These free points are then matched again by finding minimum net-cost augmenting paths. When all the boundaries are erased, the algorithm terminates with the desired minimum-cost $(n-k)$-matching.

Our efficiency analysis faces three main hurdles. First, while finding a minimum net-cost augmenting path typically requires a search on the entire graph, we show that for extended matchings, such paths lie entirely within a current cell, allowing efficient local searches inside each current cell and selecting the global minimum across cells. Second, bounding the time to merge two cells is challenging, especially since some current cells may hold $\Theta(n)$ points, where each point is matched to a far away point in the optimal solution (Figure 3(left)). We address this challenge by showing that any minimum-cost extended matching has only $O(n^{0.8})$ points matched to the shared boundary. To establish this, we critically use the fact that every sub-problem has a hidden low-cost high-cardinality matching with sub-linearly many free points (see Figure 3 (right)). Finally, the k free points in our extended matching M may differ from those in the minimum-cost $(n-k)$-matching M^*. For instance, a point $b \in B$ may be free in M^* but matched to the boundary in M, leaving another point b' unmatched. While correcting such mismatches via alternating and augmenting paths can take $\Theta(n)$ time each (leading to a naïve bound of $O(nk)$), we use geometry to show that only $O(n^{0.8})$ such corrections occur for each cell, yielding an overall runtime to $\tilde{O}(n^{1.8}\Phi(n)\log\Delta)$.

Our proof techniques also refine the analysis of the GRS algorithm when the cost between points $a \in A$ and $b \in B$ is $\|a-b\|_2^q$ for any $q \geq 2$. Prior work showed that the GRS algorithm computes the minimum-cost perfect matching between two 2-dimensional stochastic point sets in $\tilde{O}(n^{2-\frac{1}{2(q+1)}}\Phi(n))$ time [7], which implies a quadratic number of queries to the DWNN data structure as $q \to \infty$. Raghvendra *et al.* [20] observed that, for any $q \geq 1$, there exists a low-cost, high-cardinality matching with sublinear free points. Building on a similar observation and our novel analysis techniques, we show that the GRS algorithm makes only a sub-quadratic number of DWNN queries, regardless of the value of q.

▶ **Theorem 5.** *Suppose U is a set of $2n$ points inside the unit square and A is a subset chosen uniformly at random from all subsets of size n. Let $B = U \setminus A$. Then, there exists an algorithm that computes the minimum-cost perfect matching on the complete bipartite graph on A and B under ℓ_2^q costs in $\tilde{O}(n^{7/4}\Phi(n)\log\Delta)$ expected time.*

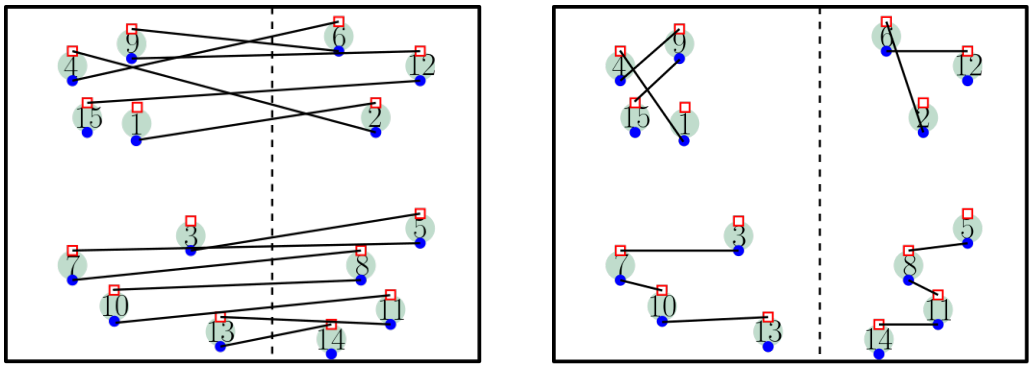


Figure 3 (left) A sub-problem of the k-SP problem, where the optimal solution has a high cost, and (right) there exists a low-cost high-cardinality matching inside the sub-problems.

The setting originally considered in the analysis of the GRS algorithm [7], where A and B are samples from the same distribution, is a special case of randomly partitioning a set of $2n$ points into two sets A and B of n points, considered in Theorem 5.

Due to space limitations, we restrict the description of our algorithm and its analysis to the k-SP problem in 2 dimensions. All other results, including extensions to higher dimensions, are presented in the full version of our paper.

2 Geometric Primal-Dual Framework

Let σ be an input to the k-SP problem, where the distance between two locations a and b is given by $\mathrm{d}(a,b) = \|a-b\|_p$. In this section, we introduce a primal-dual framework based on hierarchical partitioning to compute a minimum-cost $(n-k)$-matching in $\mathcal{G}_\sigma = (A \cup B, E \subset A \times B)$. We begin by describing the hierarchical partitioning scheme.

2.1 Hierarchical Partitioning

Using $\lambda := 9n^{-1/5}$, we construct a hierarchical partitioning $\mathcal{H}$ recursively. Each node of $\mathcal{H}$ is an axis-parallel rectangle, referred to as a *cell*. The root node, $\square^* := [-3n, 3n]^2$, contains all points in $A \cup B$. For each node $\square$, let $A_\square$ and $B_\square$ be the points of A and B inside $\square$ and let $n_\square = |A_\square \cup B_\square|$. If $n_\square \le 2$ (i.e., $\square$ is empty or contains the entry and exit gates of a single request), then $\square$ is marked as a leaf node. Otherwise, we partition $\square$ into two smaller rectangles as follows. Let $\ell_\square$ be the larger of the length and width of rectangle $\square$. Without loss of generality, assume that $\ell_\square$ is the width of $\square$ and let $x_{\min}$ be the x-coordinate of the bottom-left corner of $\square$. For any value $\hat{x} \in [x_{\min} + \frac{\ell_\square}{3}, x_{\min} + \frac{2\ell_\square}{3}]$, define $\Lambda(\hat{x}) := \{u \in A_\square \cup B_\square : |u_x - \hat{x}| \le \ell_\square \lambda\}$; here, u_x denotes the x coordinate of the point u. Let $x^* := \arg\min_{\hat{x} \in [x_{\min} + \frac{\ell_\square}{3}, x_{\min} + \frac{2\ell_\square}{3}]} |\Lambda(\hat{x})|$. We partition $\square$ into two smaller rectangles by using a vertical line defined by $x = x^*$ and add them as the children of $\square$ to $\mathcal{H}$. We refer to the segment partitioning $\square$ into its two children as its *divider* and denote it by $\Gamma_\square$. See Figure 4(left). For any cell $\square$, the four sides of its rectangle are defined by the dividers of its ancestor or the boundaries of the root square. This completes the construction of $\mathcal{H}$. Note that the height of the tree is $O(\log n\Delta)$. A simple sweep-line algorithm can compute x^* in $O(n_\square \log n_\square)$ time. Using this procedure, we construct $\mathcal{H}$ in $\tilde{O}(n \log(n\Delta))$ time.

▶ **Lemma 6.** *For each cell $\square$ of $\mathcal{H}$, the ratio of the largest to the smallest side of $\square$ is at most 3. Furthermore, the number of points of $A_\square \cup B_\square$ with a distance smaller than $\ell_\square\lambda$ to the divider $\Gamma_\square$ is $O(n_\square\lambda)$.*

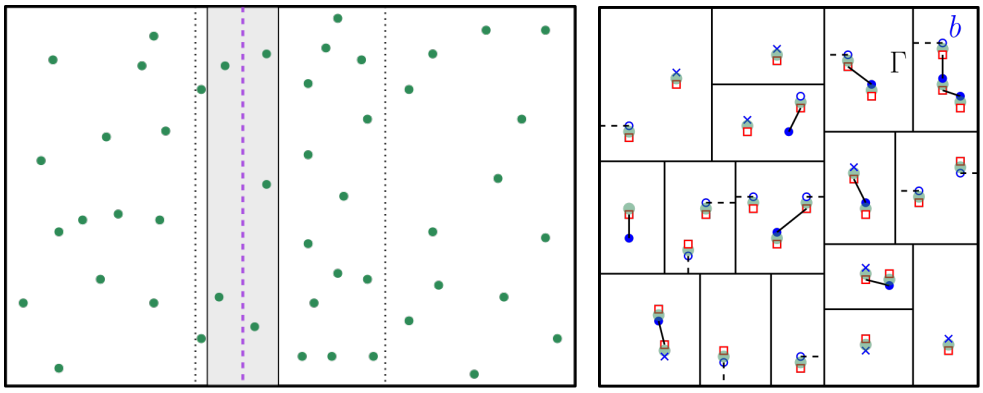


Figure 4 (left) Partitioning a rectangle into two children (the purple dashed vertical line is the divider), and (right) a $\mathcal{C}$-extended 20-matching with 9 matched points (blue discs), 11 boundary-matched points (blue circles), and 8 free points (blue crosses). The matching cost is the total length of the solid and dashed lines. The boundary-matched point b is matched to the divider Γ.

Recollect that a matching M in $\mathcal{G}_\sigma$ is a subset of vertex-disjoint edges. We refer to a point $b \in B$ as *unmatched* in M if it does not have an edge of M incident on it and *matched* otherwise. The following structural property of the k-SP problem will be critical in bounding the efficiency of our algorithm.

▶ **Lemma 7.** *For any cell $\square$ of $\mathcal{H}$, there exists a matching M' between $A_\square \cup B_\square$ that matches all except $O(n_\square^{4/5})$ points of $B_\square$ and has a cost $O(\ell_\square n_\square^{3/5})$.*

2.2 Extended Bipartite Matching

Suppose we are given a subset $\mathcal{C}$ of cells from $\mathcal{H}$ that partitions $\square^*$, i.e., the cells of $\mathcal{C}$ are interior disjoint and these cells cover the root square $\square^*$. Let $\mathrm{d}(u, \square)$ denote the shortest distance from u to the boundaries of the cell $\square$. For any point u inside $\square^*$, let $\square_u$ be the cell of $\mathcal{C}$ that contains u. We define $\mathrm{d}(b, \mathcal{C}) = \mathrm{d}(b, \square_b)$. We extend the definition of matching to allow points in B to match to the boundaries of cells in $\mathcal{C}$. A *$\mathcal{C}$-extended matching* consists of a matching M as well as a subset $B^{\mathcal{C}}$ of points of B that are unmatched in M but instead matched to the boundaries of the cells of $\mathcal{C}$ that contain them. See Figure 4(right). The cost of a $\mathcal{C}$-extended matching $M^{\mathcal{C}}$ is

$$w_{\mathcal{C}}(M^{\mathcal{C}}) := \sum_{(a,b)\in M} \mathrm{d}(a, b) + \sum_{b \in B^{\mathcal{C}}} \mathrm{d}(b, \mathcal{C}). \tag{1}$$

We refer to all points of $B^{\mathcal{C}}$ as *boundary-matched*. All points of B that are neither matched in M nor boundary-matched are considered *free*. All points of A that are not matched in M are also considered free. The size of $M^{\mathcal{C}}$ is equal to $|M| + |B^{\mathcal{C}}|$. When $\mathcal{C}$ is clear from the context, we refer to $M^{\mathcal{C}}$ as an extended matching. An extended matching $M^{\mathcal{C}}$ of size t with the minimum cost is called a *minimum-cost $\mathcal{C}$-extended t-matching*.

Consider the partitioning $\mathcal{C}^* = \{\square^*\}$, where $\square^*$ is the root cell of $\mathcal{H}$. Using the fact that the input points are far from the boundaries of $\square^*$, we show in Lemma 8 below that any minimum-cost $\mathcal{C}^*$-extended matching $M^{\mathcal{C}^*}$ of size t is also a minimum-cost t-matching, i.e., no point of B is boundary-matched in $M^{\mathcal{C}^*}$.

▶ **Lemma 8.** *Suppose $\mathcal{C}^* = \{\square^*\}$, where $\square^*$ is the root cell of $\mathcal{H}$. Let $M^{\mathcal{C}^*} = (M, B^{\mathcal{C}^*})$ be a minimum-cost extended t-matching on $\mathcal{G}_\sigma$, for $t < n$. Then, the matching M is a minimum-cost t-matching.*

Let $M^{\mathcal{C}} = (M, B^{\mathcal{C}})$ denote a $\mathcal{C}$-extended matching. Any path P on the graph $\mathcal{G}_\sigma$ whose edges alternate between matching and non-matching edges in M is called an *alternating path*. An alternating path P is called an *augmenting path* if P starts from a free point $b \in B$ and ends with either (i) a free point $a \in A$, or (ii) a point $b' \in B$. We *augment* the extended matching $M^{\mathcal{C}}$ along an augmenting path P by updating its matching $M \leftarrow M \oplus P$ and for case (ii), we match b' to the boundary and update $B^{\mathcal{C}}$ to include b'. The *net-cost* of P in case (i) is $\phi(P) := \sum_{(a,b)\in P\setminus M} \mathrm{d}(a,b) - \sum_{(a,b)\in P\cap M} \mathrm{d}(a,b)$, and in case (ii) is

$$\phi(P) := \mathrm{d}(b', \mathcal{C}) + \sum_{(a,b)\in P\setminus M} \mathrm{d}(a,b) - \sum_{(a,b)\in P\cap M} \mathrm{d}(a,b).$$

From Lemma 8, given a sequence σ of n requests in the unit square, one can compute an optimal solution to the k-SP problem on σ by computing a minimum-cost $\mathcal{C}^*$-extended $(n-k)$-matching on $\mathcal{G}_\sigma$. Given a partitioning $\mathcal{C}$, to compute a minimum-cost $\mathcal{C}$-extended $(n-k)$-matching, similar to the Hungarian algorithm, one can start from an empty extended matching $M^{\mathcal{C}}$ and iteratively augment $M^{\mathcal{C}}$ along a minimum net-cost augmenting path. In the next section, we present a primal-dual framework that our algorithm uses to efficiently compute minimum net-cost augmenting paths.

2.3 A Constrained Dual Formulation for Extended Matchings

Suppose $\mathcal{C}$ is a set of cells of $\mathcal{H}$ partitioning the root cell $\square^*$. Consider a $\mathcal{C}$-extended matching $M^{\mathcal{C}} = (M, B^{\mathcal{C}})$ on $\mathcal{G}_\sigma$ along with a set of non-negative dual weights $y : A \cup B \to \mathbb{R}_{\geq 0}$. Let A_F be the set of free points of A with respect to $M^{\mathcal{C}}$. We say that $M^{\mathcal{C}}, y(\cdot)$ is *feasible* if,

$$y(b) - y(a) \leq \mathrm{d}(a,b), \qquad \forall (a,b) \in E, \tag{2}$$
$$y(b) - y(a) = \mathrm{d}(a,b), \qquad \forall (a,b) \in M, \tag{3}$$
$$y(b) \leq \mathrm{d}(b, \mathcal{C}), \qquad \forall b \in B, \tag{4}$$
$$y(b) = \mathrm{d}(b, \mathcal{C}), \qquad \forall b \in B^{\mathcal{C}}, \tag{5}$$
$$y(a) = 0, \qquad \forall a \in A_F. \tag{6}$$

For any edge $(a,b) \in E$, the *slack* of (a,b) is defined as $s(a,b) := \mathrm{d}(a,b) - y(b) + y(a)$. The edge (a,b) is *admissible* if $s(a,b) = 0$. For any point $b \in B$, the slack of b is defined as $s(b) := \mathrm{d}(b, \mathcal{C}) - y(b)$. For any feasible extended matching $M^{\mathcal{C}}, y(\cdot)$, the slack of every edge as well as every point $b \in B$ is non-negative. Recall that an augmenting path P starts at a free point $b \in B$ and ends at (i) a free point $a \in A$ or (ii) a point $b' \in B$. The path P is *admissible* if all edges of P are admissible and in case (ii), the slack of the end-point b' is $s(b') = 0$. The following properties of extended feasible matchings are critical in the design of an efficient and correct algorithm.

▶ **Lemma 9.** *Given a feasible $\mathcal{C}$-extended t-matching $M^{\mathcal{C}} = (M, B^{\mathcal{C}})$ and a set of non-negative dual weights $y(\cdot)$ on $A \cup B$, let P be a minimum net-cost augmenting path with respect to $M^{\mathcal{C}}$. Let, for any $\square \in \mathcal{C}$, $y_\square = \max_{b' \in B_\square} y(b')$. Then,*

(a) *all points of P lie inside a single cell of $\mathcal{C}$, and*

(b) *if, for every cell $\square \in \mathcal{C}$, $y_\square \leq \phi(P)$ and for all free points $b \in B_\square$, $y(b) = y_\square$, then $M^{\mathcal{C}}$ is a minimum-cost extended t-matching.*

The property described in Lemma 9(a) is important for the design of an efficient algorithm. Unlike the Hungarian algorithm, which searches the entire graph for the minimum net-cost augmenting path, our algorithm can find the minimum net-cost augmenting path by searching

for the cheapest augmenting path inside each cell $\square \in \mathcal{C}$ and then taking the smallest among them. Thus, we can replace a global search with a search inside each cell of $\mathcal{C}$. The property in Lemma 9(b) is important for the design of a correct algorithm since it provides conditions under which an extended matching is a minimum-cost extended matching. Our algorithm is designed to maintain these conditions as invariants during its execution. Lemma 10 provides a method to update the dual weights, which is essential during the process of merging cells.

▶ **Lemma 10.** *Suppose $M^{\mathcal{C}}, y(\cdot)$ is a feasible $\mathcal{C}$-extended matching and P is a minimum net-cost augmenting path. For any cell $\square \in \mathcal{C}$, define $y_{\square} = \max_{b' \in B_{\square}} y(b')$, and suppose $y_{\square} \leq \phi(P)$ and $y(b_f) = y_{\square}$ for all free points $b_f \in B_{\square}$. Then, one can update the dual weights in $\tilde{O}(n_{\square}\Phi(n_{\square}))$ time such that $M^{\mathcal{C}}, y(\cdot)$ remains feasible, $y(b) \leq \phi(P)$ for all $b \in B_{\square}$, and $y(b_f) = \phi(P)$ for all free points $b_f \in B_{\square}$.*

The next lemma provides critical properties that allow for correctly and efficiently merging cells in $\mathcal{C}$.

▶ **Lemma 11.** *For a cell $\square$ of $\mathcal{H}$, suppose $\square'$ and $\square''$ denote its two children, and let $\mathcal{C}$ denote a partitioning containing $\square'$ and $\square''$. Let $\mathcal{C}' = \mathcal{C} \cup \{\square\} \setminus \{\square', \square''\}$. Given a feasible $\mathcal{C}$-extended matching $(M, B^{\mathcal{C}}), y(\cdot)$, let $B^{\mathcal{C}}_{\square} \subseteq B^{\mathcal{C}}$ denote the subset of boundary-matched points that are matched to the divider $\Gamma_{\square}$ of $\square$. Then,*
(a) *the $\mathcal{C}'$-extended matching $(M, B^{\mathcal{C}} \setminus B^{\mathcal{C}}_{\square}), y(\cdot)$ is also feasible, and,*
(b) $|B^{\mathcal{C}}_{\square}| = O(n^{4/5})$.

From Lemma 11(a), erasing a divider does not cause the feasibility conditions to be violated. The proof of Lemma 11(a) relies on the fact that when we erase the divider of a cell in $\mathcal{C}$, the RHS of (4) will only increase and so it is not violated. Despite preserving the feasibility conditions, erasing the boundary may result in the violation of Lemma 9(b), i.e., the matching may no longer be a minimum-cost extended matching. In our algorithm in Section 3, we describe a process to adjust the matching $M^{\mathcal{C}}$ and dual weights $y(\cdot)$ and obtain a minimum-cost extended matching.

Residual Graph. Similar to the Hungarian algorithm, we define a residual graph that assists in finding the minimum net-cost augmenting path. Consider a feasible $\mathcal{C}$-extended matching $M^{\mathcal{C}} = (M, B^{\mathcal{C}})$ along with a set of dual weights $y(\cdot)$ on points of $A \cup B$. For each cell $\square \in \mathcal{C}$, we define a residual graph $\mathcal{G}_{\square}$. The vertex set of $\mathcal{G}_{\square}$ is a source vertex s and the points in $A_{\square} \cup B_{\square}$. For any edge $(a, b) \in E$ inside $\square$, if $(a, b) \in M$ (resp. $(a, b) \notin M$), there is an edge directed from a to b (resp. from b to a) with a weight $s(a, b)$ in $\mathcal{G}_{\square}$. Furthermore, there is an edge directed from s to every free point $b \in B$ with a weight $y(b)$.

3 A Sub-Quadratic Algorithm for the k-SP Problem

In this section, we describe an algorithm that, given a sequence σ of n requests in 2-dimensions, computes the optimal solution to the k-SP problem in $\tilde{O}(n^{1.8}\Phi(n)\log(n\Delta))$ time.

3.1 Algorithm

Initialize $\mathcal{C}$ to the leaf cells of $\mathcal{H}$. Let $M^{\mathcal{C}} = (M, B^{\mathcal{C}})$ be the extended matching maintained by the algorithm and initialized to $M = \emptyset$ and $B^{\mathcal{C}} = \emptyset$. For each point $v \in A \cup B$, let $y(v)$ denote its dual weight initialized to $y(v) = 0$. Let B_F, initialized to B, be the free points of B with respect to $M^{\mathcal{C}}$. For each cell $\square$ that contains at least one free point $b \in B_F$, let

$P_\square$ denote the minimum net-cost augmenting path inside $\square$. Initially, since $\square$ is a leaf of $\mathcal{H}$, it contains only one point $b \in B_F$, and therefore, $P_\square$ is this point with a net-cost equal to $\mathrm{d}(b, \mathcal{C})$. Our algorithm maintains a priority queue PQ storing every leaf cell $\square \in \mathcal{C}$ with at least one free vertex with a key of $\phi(P_\square)$. At any time during the execution of our algorithm, let $\square_{\min}$ be the cell with the smallest key in PQ and let φ be the key of $\square_{\min}$. Execute the following steps until PQ becomes empty:

- While $|B_F| > k$,
  - *Extended Hungarian search step:* Extract the cell $\square$ with the minimum key of φ from PQ. Augment the matching $M^{\mathcal{C}}$ along $P_\square$ and update the key of $\square$ in PQ (See Section 3.1.1 for details).
- *Merge step:* If $\mathcal{C} = \{\square^*\}$, remove $\square^*$ from PQ and return the matching M of $M^{\mathcal{C}}$. Otherwise, pick a cell $\square' \in \mathcal{C}$ with the smallest perimeter and let $\square$ and $\square''$ be the parent and sibling of $\square'$ in $\mathcal{H}$, respectively. Erase the divider of $\square$, i.e., set $\mathcal{C} = \mathcal{C} \setminus \{\square', \square''\} \cup \{\square\}$. We execute a Merge procedure that updates the matching $M^{\mathcal{C}}$ inside the cell $\square$ so that $\phi(P_\square)$ is at least φ (See Section 3.1.2 for details). At this point, the size of the updated extended matching $M^{\mathcal{C}}$ may not be $(n-k)$, i.e., there may be more than k free points.

Invariants. For each cell $\square \in \mathcal{C}$, let $y_\square = \max_{b \in B_\square} y(b)$. During the execution of our algorithm,

(I1) the extended matching $M^{\mathcal{C}}, y(\cdot)$ is feasible,

(I2) For each cell $\square \in \mathcal{C}$, $y_\square \le \varphi$ and for all free point $b \in B_\square$, $y(b) = y_\square$, and,

(I3) The φ-value is non-decreasing. Furthermore, after each step of the algorithm, φ denotes the smallest net-cost of all augmenting paths with respect to $M^{\mathcal{C}}$.

3.1.1 Details of the Extended Hungarian Search Step

Given a feasible extended matching $M^{\mathcal{C}}, y(\cdot)$ and a cell $\square \in \mathcal{C}$, the extended Hungarian search procedure computes the minimum net-cost augmenting path $P_\square$ and augments $M^{\mathcal{C}}$ along $P_\square$. It then computes the new minimum net-cost augmenting path and updates the key of $\square$ in PQ to be its net-cost. This procedure is similar to the classical Hungarian search procedure executed on $\mathcal{G}_\square$ and is mildly modified to include the augmenting paths that end at the boundary of $\square$. Details of the procedure are as follows.

1. *Update duals:* With s as the source vertex, execute Dijkstra's shortest path algorithm on the residual graph $\mathcal{G}_\square$. Let P_v be the shortest path from s to each $v \in A_\square \cup B_\square$ and let κ_v be the cost of P_v. Let

$$\kappa = \min\{ \min_{a \in A_\square^F} \kappa_a, \min_{b \in B_\square} \kappa_b + s(b)\}. \tag{7}$$

   For any $v \in A_\square \cup B_\square$, if $\kappa_v < \kappa$, set $y(v) \leftarrow y(v) + \kappa - \kappa_v$.
2. *Augment:* Let $u \in A_\square^F \cup B_\square$ be the point realizing the minimum distance in Equation (7). Let P be the augmenting path obtained by removing s. Augment $M^{\mathcal{C}}$ along P.
3. *Update key:* If $B_\square$ has no free points, then remove $\square$ from PQ. Otherwise,
   a. Recompute the residual graph $\mathcal{G}_\square$ with respect to the updated matching,
   b. With s as the source, execute the Dijkstra's shortest path algorithm on $\mathcal{G}_\square$. For each $v \in A_\square \cup B_\square$, let κ_v be the distance from s to v.
   c. Update the key of $\square$ in PQ to $\kappa_\square = \min\{\min_{a \in A_\square^F} \kappa_a, \min_{b \in B_\square} \kappa_b + s(b)\}$.

   Lemma 12 establishes the properties of the extended Hungarian search procedure.

▶ **Lemma 12.** *After the execution of the extended Hungarian search procedure for a cell $\square$, the extended matching $M^{\mathcal{C}}, y(\cdot)$ remains feasible, $y(v) \leq \varphi$ for all points $v \in A_{\square} \cup B_{\square}$, and $y(b_f) = \varphi$ for all free points $b_f \in B_{\square}$. Furthermore, the path P computed by the procedure is a minimum net-cost augmenting path inside $\square$. After augmenting along P, the updated key for $\square$ is the smallest net-cost of all augmenting paths inside $\square$ and is at least φ.*

3.1.2 Details of the Merge Step

Given a feasible extended matching $M^{\mathcal{C}} = (M, B^{\mathcal{C}}), y(\cdot)$, any cell $\square$ of $\mathcal{H}$ where both of its children $\square'$ and $\square''$ are in $\mathcal{C}$, and a value φ, the merge procedure uses the algorithm in Lemma 10 to update the dual weights $y(\cdot)$ inside $\square'$ (resp. $\square''$) so that $M^{\mathcal{C}}, y(\cdot)$ remains feasible, the dual weights of all points in $\square'$ (resp. $\square''$) are at most φ, and the dual weight of all free points $b'_f \in B_{\square'}$ (resp. $b''_f \in B_{\square''}$) is $y(b'_f) = \varphi$ (resp. $y(b''_f) = \varphi$). The procedure then updates $\mathcal{C} \leftarrow \mathcal{C} \cup \{\square\} \setminus \{\square', \square''\}$ and makes the points matched to the divider $\Gamma_{\square}$ free. While there exists a free point $b \in B_{\square}$ with $y(b) < \varphi$,

1. With s as the source, execute Dijkstra's shortest path algorithm on the residual graph $\mathcal{G}_{\square}$. For each $v \in A_{\square} \cup B_{\square}$, let P_v be the shortest path from s to v in $\mathcal{G}_{\square}$ and let κ_v be its cost. Define

$$\kappa = \min\{\min_{a \in A^F_{\square}} \kappa_a, \min_{b \in B_{\square}} \kappa_b + s(b), \min_{b \in B_{\square}} \kappa_b + \varphi - y(b)\}. \tag{8}$$

2. Let $u \in A^F_{\square} \cup B_{\square}$ be the point realizing the minimum value in Equation (8). Let P be the path obtained by removing s from the path P_u.
   a. If $u \in B_{\square}$ and $\kappa = \kappa_u + \varphi - y(u)$ (i.e., κ is determined by the third element in the RHS of Equation (8)), then P is a path from a free point $b \in B_{\square}$ to the matched point u. For each $v \in A_{\square} \cup B_{\square}$, if $\kappa_v < \kappa$, update its dual weight to $y(v) \leftarrow y(v) + \kappa - \kappa_v$. The path P is an admissible alternating path with respect to the updated dual weights. Set $M \leftarrow M \oplus P$. Note that u is now a free point with $y(u) = \varphi$.
   b. Otherwise, P is an augmenting path. For each $v \in A_{\square} \cup B_{\square}$, if $\kappa_v < \kappa$, update its dual weight to $y(v) \leftarrow y(v) + \kappa - \kappa_v$. The path P is an admissible augmenting path with respect to the updated dual weights. Augment M along P.

At the end of this execution, all free points of B in $\square$ have a dual weight of φ. Finally, we update the key for $\square$ by executing steps 3(a)–(c) from the extended Hungarian search step.

The following lemma states the useful properties of the merge step.

▶ **Lemma 13.** *After the execution of the merge procedure on a cell $\square$, the updated extended matching $M^{\mathcal{C}}, y(\cdot)$ is feasible, the dual weights $y(v)$ for every $v \in A_{\square} \cup B_{\square}$ is at most φ, and $y(b_f) = \varphi$ for all free points $b_f \in B_{\square}$. Furthermore, the updated key for $\square$ is the smallest net-cost of all augmenting paths within $\square$ and is at least φ.*

3.2 Analysis

We begin by showing in Section 3.2.1 that the three invariants (I1)–(I3) hold during the execution of our algorithm and use them to show the correctness of our algorithm. We then show in Section 3.2.2 that the running time of our algorithm is $\tilde{O}(n^{9/5}\Phi(n)\log n\Delta)$.

3.2.1 Correctness

Our algorithm initializes $M^{\mathcal{C}}$ with a feasible $\mathcal{C}$-extended matching and sets all dual weights and φ to 0. Therefore, (I1) and (I2) hold at the start of the algorithm. The extended Hungarian search (Lemma 12) as well as the merge step (Lemma 13) do not violate the feasibility of

the extended matching and therefore (I1) holds during the execution of the algorithm. The extended Hungarian search (Lemma 12) and the merge (Lemma 13) procedures keep the dual weight of every point inside $\square$ at or below φ, while ensuring that the dual weight of free points inside $\square$ is φ; hence, (I2) holds. Finally, both the extended Hungarian search (Lemma 12) and the merge (Lemma 13) procedures update the key of $\square$ to the smallest net-cost of all augmenting paths inside $\square$ and do not decrease the key of any cell. From this observation, (I3) follows in a straightforward way.

From (I1) and (I2), our algorithm maintains a feasible $\mathcal{C}$-extended matching where, for each cell $\square \in \mathcal{C}$, $y_{\square} \leq \varphi$. From (I3), φ is equal to the minimum net-cost of all augmenting paths with respect to $M^{\mathcal{C}}$; combining this with Lemma 9(b), we conclude that $M^{\mathcal{C}}$ is a minimum-cost $\mathcal{C}$-extended matching. Upon termination, the algorithm returns a minimum-cost $\mathcal{C}^*$-extended $(n-k)$-matching for $\mathcal{C}^* = \{\square^*\}$, which from Lemma 8 has no boundary-matched points. Therefore, this matching is also a minimum-cost $(n-k)$-matching, as desired.

3.2.2 Efficiency

Both the merge step and the extended Hungarian search step require an execution of Dijkstra's shortest path algorithm on $\mathcal{G}_{\square}$. In the full version of our paper, we show that this execution takes $\tilde{O}(n_{\square}\Phi(n_{\square}))$ time. Using this, we analyze the execution time.

We begin by establishing a bound on the execution time of the merge step, which combines $\square'$ and $\square''$ into a single cell $\square$. At the start of this step, the dual weight of all free points in $\square'$ and $\square''$ are raised to φ (from Lemma 10). As a result, once the divider is removed, the only free points with dual weights below φ are those that are matched to the divider $\Gamma_{\square}$. From Lemma 11(b), the number of points matched to $\Gamma_{\square}$ is $O(n^{4/5})$. Therefore, the while-loop in the merge procedure executes only $O(n^{4/5})$ times. Since each iteration takes $\tilde{O}(n_{\square}\Phi(n_{\square}))$ time, the total execution time of a single execution of the merge step is $\tilde{O}(n^{4/5}n_{\square}\Phi(n_{\square}))$. Given that each point is in only $O(\log n\Delta)$ many cells of $\mathcal{H}$, the total time taken by the merge step across all cells of $\mathcal{H}$ is $\tilde{O}(n^{9/5}\Phi(n)\log n\Delta)$.

Similarly, if the execution time for the extended Hungarian search within a single cell $\square$ is bounded by $O(n^{4/5}n_{\square}\Phi(n_{\square}))$, then the cumulative execution time of the extended Hungarian search across all cells – and consequently, the total runtime of the entire algorithm – can be bounded by $\tilde{O}(n^{9/5}\Phi(n)\log n\Delta)$. In the remainder of our analysis, we establish a bound of $O(n^{4/5}n_{\square}\Phi(n_{\square}))$ for the time taken by the extended Hungarian search within a single cell $\square$. Recall that the algorithm selects a cell $\square$ containing the minimum net-cost augmenting path from the priority queue PQ and performs an extended Hungarian search procedure within $\square$, requiring $\tilde{O}(n_{\square}\Phi(n_{\square}))$ time. To analyze the total execution time of the extended Hungarian search procedure, we show that any cell $\square$ can be selected by the algorithm at most $O(n^{4/5})$ times. In particular, we categorize the selection of $\square$ as a *low-net-cost* case if $\varphi \leq \ell_{\square}n^{-1/5}$ and a *high-net-cost* case otherwise. First, we show that in the high-net-cost case ($\varphi > \ell_{\square}n^{-1/5}$), the number of free points remaining in $\square$ is $O(n^{4/5})$, and as a result, the total number of high-net-cost selections of $\square$ is $O(n^{4/5})$.

▶ **Lemma 14.** *Given a feasible extended matching $M^{\mathcal{C}}, y(\cdot)$ and any cell $\square \in \mathcal{C}$, if the net-cost of the minimum net-cost augmenting path inside $\square$ is greater than $\ell_{\square}n^{-1/5}$, then the number of free points of $M^{\mathcal{C}}$ is $O(n^{4/5})$.*

Next, we bound the number of low-net-cost selections of $\square$. Define $\mathcal{C}_{\square}$ as the set of all cells $\square' \in \mathcal{H}$ that are processed by the merge procedure while $\square \in \mathcal{C}$ and $\varphi \leq \ell_{\square}n^{-1/5}$ (Figure 5). Our algorithm always picks the smallest perimeter cells to merge, and as a result, we obtain the following lemma.

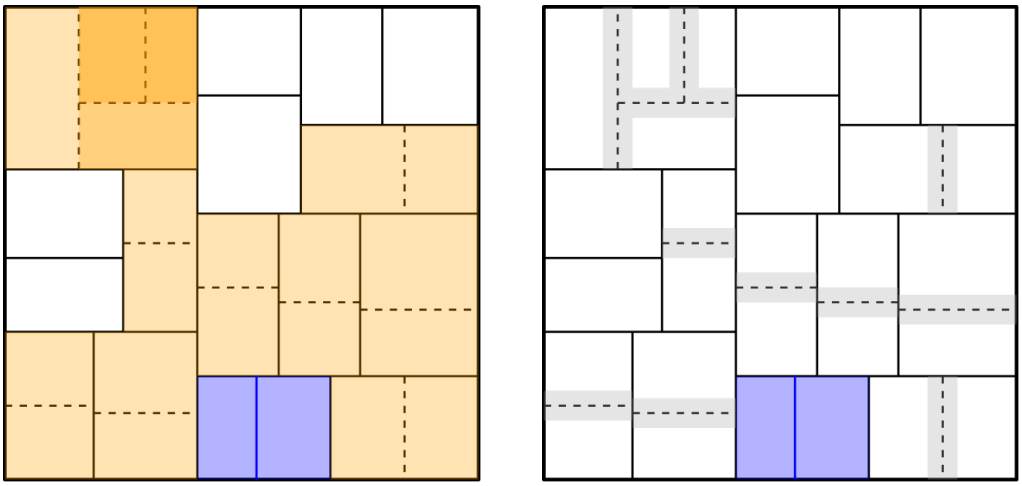


Figure 5 (left) The set $\mathcal{C}_\square$ (the cells shaded in orange with dashed lines as their divider) for a cell $\square$ (shaded in blue), and (right) any boundary-matched point in the gray area might cause a low-net-cost execution of the extended Hungarian search procedure on $\square$.

▶ **Lemma 15.** *For any non-leaf cell $\square$ in $\mathcal{H}$ and any cell $\square' \in \mathcal{C}_\square$, $\frac{1}{3}\ell_{\square'} \leq \ell_\square \leq \frac{9}{4}\ell_{\square'}$.*

For any cell $\square' \in \mathcal{C}_\square$, the merge step erases the divider $\Gamma_{\square'}$ and makes the points that are matched to $\Gamma_{\square'}$ free. Each of these new free points might cause $\square$ to be selected by the algorithm. Therefore, to bound the number of low-net-cost selections of $\square$, we show that the total number of points that are matched to the dividers of the cells in $\mathcal{C}_\square$ is at most $O(n^{4/5})$. For any cell $\square' \in \mathcal{C}_\square$, let $\mathcal{B}_{\square'}$ denote the set of points of $B^{\mathcal{C}}$ that are matched to the divider $\Gamma_{\square'}$ when our algorithm starts the merge step on $\square'$. Thus, we have to bound $\sum_{\square' \in \mathcal{C}_\square} |\mathcal{B}_{\square'}|$ by $O(n^{4/5})$. By invariant (I2), for each point $b \in \mathcal{B}_{\square'}$, $y(b) \leq \varphi \leq \ell_\square n^{-1/5}$. Furthermore, by Condition (5), $y(b) = \mathrm{d}(b, \square') = \mathrm{d}(b, \Gamma_{\square'})$. Therefore, the point b is within a distance $\ell_\square n^{-1/5}$ from the divider $\Gamma_{\square'}$. From Lemma 15, for each cell $\square' \in \mathcal{C}_\square$ and each point $b \in \mathcal{B}_{\square'}$, the point b is within a distance $\ell_\square n^{-1/5} < \ell_{\square'}\lambda$ from the divider $\Gamma_{\square'}$, and from the construction of $\mathcal{H}$ (Lemma 6), $|\mathcal{B}_{\square'}| = O(n_{\square'} n^{-1/5})$. Furthermore, from Lemma 15 and the construction of $\mathcal{H}$, each point $b \in B$ can lie inside a constant number of cells in $\mathcal{C}_\square$; hence, $\sum_{\square' \in \mathcal{C}_\square} |\mathcal{B}_{\square'}| = O(\sum_{\square' \in \mathcal{C}_\square} n_{\square'} n^{-1/5}) = O(n^{4/5})$. Therefore, the total number of low-net-cost selections of $\square$ by the extended Hungarian search procedure is $O(n^{4/5})$.

4 Conclusion and Open Questions

We presented an exact algorithm for the k-SP and k-SPI problems on d-dimensional point sets with bounded spread Δ, achieving sub-quadratic queries to a DWNN data structure and only logarithmic dependence on Δ. We highlight two open directions: First, can the $O(\log \Delta)$ dependence be removed? This would yield sub-quadratic algorithms for geometric bipartite matching, addressing a central open problem in computational geometry. Second, our reductions frame the k-server problem as a mild variant of online metric matching. This connects with extensive work on the online k-server problem [1, 3, 9, 10, 12, 14, 21, 23]. The reduction suggests simplifications and improvements to work function-based implementations, including the scalable algorithm by Raghvendra and Sowle [21]. Moreover, can we adapt the RM algorithm [15, 17, 18], which is the best-known online metric algorithm, to establish tighter bounds on its competitive ratio for k-server?

References

1 Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, James R Lee, and Aleksander Mądry. K-server via multiscale entropic regularization. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 3–16, 2018.

2 Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proc. 63rd IEEE Annual Sympos. Foundations of Computer Science*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

3 M. Chrobak and L. L. Larmore. An optimal on-line algorithm for k servers on trees. *SIAM Journal of Computing*, 20(1):144–148, 1991. doi:10.1137/0220008.

4 Marek Chrobak, H Karloof, Tom Payne, and Sundar Vishwnathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2):172–181, 1991. doi:10.1137/0404017.

5 H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal of Computing*, 18(5):1013–1036, October 1989. doi:10.1137/0218069.

6 Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for general graph-matching problems. *Journal ACM*, 38(4):815–853, 1991. doi:10.1145/115234.115366.

7 Akshaykumar Gattani, Sharath Raghvendra, and Pouyan Shirzadian. A robust exact algorithm for the Euclidean bipartite matching problem. *Advances in Neural Information Processing Systems*, 36, 2024.

8 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha 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.

9 E. Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009. doi:10.1016/J.COSREV.2009.04.002.

10 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995. doi:10.1145/210118.210128.

11 Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

12 James R. Lee. Fusible HSTs and the randomized k-server conjecture. In Mikkel Thorup, editor, *Proc. 59th IEEE Annual Symposium on Foundations of Computer Science*, pages 438–449, 2018. doi:10.1109/FOCS.2018.00049.

13 Richard J Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. doi:10.1137/0209046.

14 Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990. doi:10.1016/0196-6774(90)90003-W.

15 Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science*, pages 505–515, 2017. doi:10.1109/FOCS.2017.53.

16 Abhijeet Phatak, Sharath Raghvendra, Chittaranjan Tripathy, and Kaiyi Zhang. Computing all optimal partial transports. In *Proc. 11th Internat. Conference on Learning Representations*, 2022.

17 Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization*, 2016.

18 Sharath Raghvendra. Optimal analysis of an online algorithm for the bipartite matching problem on a line. In *34th International Symposium on Computational Geometry*, pages 67:1–67:14, 2018. doi:10.4230/LIPICS.SOCG.2018.67.

19 Sharath Raghvendra, Pouyan Shirzadian, and Rachita Sowle. Geometric bipartite matching based exact algorithms for server problems, 2025. arXiv:2504.06079.

20 Sharath Raghvendra, Pouyan Shirzadian, and Kaiyi Zhang. A new robust partial p-Wasserstein-based metric for comparing distributions. In *41st Internat. Conference on Machine Learning*, 2024.

21 Sharath Raghvendra and Rachita Sowle. A scalable work function algorithm for the k-server problem. In *18th Scandinavian Symposium and Workshops on Algorithm Theory*, volume 227, 2022. doi:10.4230/LIPICS.SWAT.2022.30.

22 Tomislav Rudec, Alfonzo Baumgartner, and Robert Manger. A fast implementation of the optimal off-line algorithm for solving the k-server problem. *Mathematical Communications*, 14(1):119–134, 2009.

23 Tomislav Rudec, Alfonzo Baumgartner, and Robert Manger. A fast work function algorithm for solving the k-server problem. *Central European Journal of Operations Research*, 21:187–205, 2013. doi:10.1007/S10100-011-0222-7.

24 Tomislav Rudec and Robert Manger. A new approach to solve the k-server problem based on network flows and flow cost reduction. *Computers & operations research*, 40(4):1004–1013, 2013. doi:10.1016/J.COR.2012.11.006.

25 R. Sharathkumar. A sub-quadratic algorithm for bipartite matching of planar points with bounded integer coordinates. In *Proc. 29th Annual Symposium on Computational Geometry*, pages 9–16, 2013. doi:10.1145/2462356.2480283.

26 R Sharathkumar and Pankaj K Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. 23rd Annual ACM-SIAM Sympos. on Discrete Algorithms*, pages 306–317, 2012. doi:10.1137/1.9781611973099.29.

27 Pravin M Vaidya. Geometry helps in matching. *SIAM Journal of Computing*, 18(6):1201–1225, 1989. doi:10.1137/0218080.

Embedding Graphs as Euclidean kNN-Graphs

Thomas Schibler
University of California, Santa Barbara, CA, USA

Subhash Suri
University of California Santa Barbara, CA, USA

Jie Xue
New York University Shanghai, China

Abstract

Let $G = (V, E)$ be a directed graph on n vertices where each vertex has out-degree k. We say that G is kNN-realizable in d-dimensional Euclidean space if there exists a point set $P = \{p_1, p_2, \ldots, p_n\}$ in $\mathbb{R}^d$ along with a one-to-one mapping $\phi : V \to P$ such that for any $u, v \in V$, u is an out-neighbor of v in G if and only if $\phi(u)$ is one of the k nearest neighbors of $\phi(v)$; we call the map ϕ a *kNN-realization* of G in $\mathbb{R}^d$. The kNN-realization problem, which aims to compute a kNN-realization of an input graph in $\mathbb{R}^d$, is known to be NP-hard already for $d = 2$ and $k = 1$ [Eades and Whitesides, Theoretical Computer Science, 1996], and to the best of our knowledge has not been studied in dimension $d = 1$. The main results of this paper are the following:

- For any fixed dimension $d \geq 2$, we can efficiently compute an embedding realizing at least a $1 - \varepsilon$ fraction of G's edges, or conclude that G is not kNN-realizable in $\mathbb{R}^d$.
- For $d = 1$, we can decide in $O(kn)$ time whether G is kNN-realizable and, if so, compute a realization in $O(n^{2.5}\mathsf{poly}(\log n))$ time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Geometric graphs, k-nearest neighbors, graph embedding, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.73

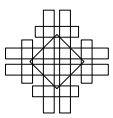
Related Version *Full Version*: `https://arxiv.org/pdf/2504.06503`

1 Introduction

The *kNN-graph* of a set P of points in $\mathbb{R}^d$ is a directed graph with vertex set P and edges defined as follows: there is a directed edge from $a \in P$ to $b \in P$ if and only if b is one of the k-nearest neighbors of a in P (under the Euclidean distance). A directed graph $G = (V, E)$ is *kNN-realizable* in $\mathbb{R}^d$ if it is isomorphic to the kNN-graph of a set of points in $\mathbb{R}^d$. In this paper, we consider the following natural problems regarding kNN-graphs: Can we efficiently check whether a given directed graph is kNN-realizable in $\mathbb{R}^d$? If so, can we efficiently find a kNN-realization of G? More formally, a kNN-*realization* (or kNN-*embedding*) of G in $\mathbb{R}^d$ is a one-to-one mapping $\phi : V \to P$ from the vertex set V of G to a set P of points in $\mathbb{R}^d$ that induces an isomorphism between G and the kNN-graph of P. Throughout the paper, we use the terms embedding and realization interchangeably.

The kNN-graph is a member of the well-known family of *proximity* graphs in computational geometry that includes minimum spanning trees, relative neighborhood graphs, Gabriel graphs, and Delaunay triangulations [11]. Over the past several decades, a substantial research effort has been directed to design efficient algorithms to compute these structures for an input set of points. The *inverse* problem – the focus of our paper – where we want to recover the points that produce a given proximity graph, however, remains less well understood.

© Thomas Schibler, Subhash Suri, and Jie Xue;
licensed under Creative Commons License CC-BY 4.0



An early example of a positive result in this direction is on Euclidean minimum spanning trees. Monma and Suri [22] show that any tree with maximum node degree 5 can be realized as a minimum spanning tree of points in the plane and, additionally, every planar point set admits a minimum spanning tree with degree at most 5. This settles the above problem for *non-degenerate* planar point sets but if we allow co-circularities, then a planar minimum spanning tree can have maximum node degree 6; in that case, the problem of computing a 1NN-realization was proved to be NP-complete by Eades and Whitesides [13]. Dillencourt [12] considers the problem of recognizing triangulations that can be realized as Delaunay triangulations in the plane, and it is also known that all outerplanar triangulations are realizable [24]. (These proofs are non-constructive, however, and only exponential time algorithms are known for computing the coordinates of the points in the realization [1].) Among other related results, Bose et al. [5] give a characterization of *trees* that can be realized in the plane as relative neighborhood or Gabriel graphs, and Eppstein et al. [15] establish some structural properties of kNN graphs of random points in the plane.

The kNN-realization is not directly related to the *metric embedding* problem but there are some obvious similarities. The input to metric embedding is a weighted graph satisfying triangle inequalities and the goal is to find an embedding realizing all pairwise distances. This is not always possible [20] – there are simple metrics that cannot be embedded in any finite dimensional Euclidean space. However, if we allow some *distortion* (i.e. approximation) of distances, then any n-node metric graph can be embedded in $O(\log n)$ dimensional space with polylog distortion [6], or using the celebrated Johnson-Lindenstrauss theorem [18] in dimension $O(\log n/\varepsilon^2)$ with distortion $1+\varepsilon$ for any fixed constant $\varepsilon > 0$.

In metric embedding the goal is to realize all pairwise distances (approximately), while in kNN-realization the goal is only to preserve all *ordinal* neighbor relations in a *directed* graph: a neighbor of each vertex must be closer than any of its non-neighbors but the choice of specific distances does not matter. Although there is some work in embedding ordinal relations as well, the main focus is specialized metric spaces and *bounds* on *ordinal distortion*. In particular, Alon et al. [2] consider embeddings with ordinal relaxations into ultrametric spaces, and Bădoiu et al. [7] improve their bounds for tree and line embeddings. A different line of research concerns *triplets* embedding, where the input is a set of triples (a, b, c) specifying constraints of the form $d(a, b) < d(a, c)$. Even for embedding in one dimension $\mathbb{R}^1$, the general triplet problem as well as the related "betweenness" problem is MAXSNP-hard [9], and a FPTAS is known for maximizing the number of satisfied triplet constraints [17]. (If all $\binom{n}{3}$ triplet constraints are given, then the problem is trivial to solve – indeed, once the leftmost point is determined, the rest of the ordering can be easily decided.) In contrast with the general triplet constraints, the pairwise relations in a kNN-realization problem in $\mathbb{R}^1$ have a richer structure that allows us to solve the problem efficiently.

The kNN-realization is also related to the *sphericity* of graphs [19], where the goal is to embed a (undirected) graph $G = (V, E)$ into an Euclidean space $\mathbb{R}^k$ such that there is a point $p(u)$ for each vertex $u \in V$, and $d(p(u), p(v)) \leq 1$ iff $(u, v) \in E$. The smallest dimension k that admits such an embedding is called the *sphericity* of G [19]. (Another related problem is the *dimension* of a graph, introduced by Erdős, Harary and Tutte [16]: it is the smallest number k such that G can be embedded into Euclidean k-space with every edge of G having length 1.) Along these lines, Chatziafratis and Indyk [8] also show that if we want to preserve the relative distances among the k nearest neighbors of each point, then the embedding dimension must grow linearly with k. Unlike these results, our work is algorithmic – we design efficient algorithms to realize a given graph G in a specified dimension d.

Euclidean embedding of neighbor relations is also studied in social sciences for geometric realization of preference orderings. Given a set V of n voters, a set C of m candidates, and a rank ordering of the candidates by each voter, we say that the preference graph can be realized in Euclidean d-space if each voter's preferences are consistent with its Euclidean distances to all the candidates. In this case, the smallest dimension must satisfy $d \geq \min\{n, m-1\}$ [4].

1.1 Main results

We study the *kNN-realization* problem, which takes a directed graph G as an input and aims to compute a kNN-realization of G in $\mathbb{R}^d$ (or decide the nonexistence of such a realization). Throughout we focus on the *unranked* kNN-realization problem where nearest neighbors are realized as an *unordered set* but our results also hold (with minor caveat) for the ranked version where edges of G specify each of the k neighbors in order (see Concluding Remarks). Our two main results are the following.

1. Assuming that G is kNN-realizable in $\mathbb{R}^d$, we can find a d-dimensional realization in polynomial time that preserves at least a $1-\varepsilon$ fraction of the edges of G, for any fixed $\varepsilon > 0$. If our algorithm fails, we can also conclude that G is *not* kNN-realizable in $\mathbb{R}^d$. In particular, our algorithm is an EPTAS (*efficient polynomial-time approximation scheme*) for the kNN-realization problem for fixed k and d.
2. We give a *linear-time* algorithm to decide wether G is kNN-realizable in $\mathbb{R}^1$. Specifically, the algorithm runs in $O(kn)$ time, which is linear in the size of the input graph. If G is realizable, our algorithm can compute a realization in $O(n^{2.5}\mathsf{poly}(\log n))$ time.

1.2 Basic definitions

For a (directed or undirected) graph G, we use $V(G)$ and $E(G)$ to denote the set of its vertices and edges, respectively. If G is a directed graph, we say G is *k-regular* if the out-degree of every vertex of G is exactly equal to k. We use $\mathsf{in}[v]$ (resp., $\mathsf{out}[v]$) to denote the set consisting of v itself and all in-neighbors (resp., out-neighbors) of v. A *kNN-realization* of $G = (V, E)$ in a metric space $\mathcal{M} = (M, \mathsf{dist})$ is an injective map $\phi : V \to M$ such that $\mathsf{dist}(\phi(v), \phi(u)) < \mathsf{dist}(\phi(v), \phi(u'))$, for any distinct triple $v, u, u' \in V$ where $(v, u) \in E$ and $(v, u') \notin E$. We say G is *kNN-realizable* in $\mathcal{M}$ if there exists a kNN-realization of G in $\mathcal{M}$.

2 Approximate kNN-realization in $\mathbb{R}^d$

We first observe that it is easy to decide in polynomial time whether a given graph G is kNN-realizable in a finite-dimensional Euclidean space. We just have to check the acyclicity of an auxiliary graph defined as follows. Let Λ_G be a directed graph whose vertices correspond to pairs of vertices in $V(G)$ with a directed edge $(\{v, u\}, \{v, u'\})$ for every distinct triple $v, u, u' \in V(G)$ where $(v, u) \in E(G)$ and $(v, u') \notin E(G)$. Intuitively, Λ_G encodes the $\leq$-relation among the pairwise distances of points in any (potential) kNN-realization of G. If Λ_G has a directed cycle, then clearly G is not kNN-realizable. Otherwise, a topological sort of Λ_G gives a total ordering of the vertex pairs in G. With this ordering, we can find a kNN-realization of G in $\mathbb{R}^n$ using the result of Bilu and Linial [3].

On the other hand, deciding whether G is kNN-realizable in $\mathbb{R}^d$, for a specific dimension d, is NP-hard. This is shown by Eades and Whitesides [14] who proved the hardness for $d = 2$ and $k = 1$. Therefore, in this section, we explore an *approximation* algorithm for the realization problem in any fixed dimension d. We first need to define an approximate

solution to our problem. There is a natural way to measure how well a map $\phi : V(G) \to \mathbb{R}^d$ approximates a kNN-realization: randomly sample an edge $(u, v) \in E(G)$ and consider the probability that $\phi(v)$ is among the k nearest neighbors of $\phi(u)$. Formally, we introduce the following definition.

▶ **Definition 1** (approximate kNN-realization)**.** *Let G be a k-regular directed graph. For $c \in [0, 1]$, a map $\phi : V(G) \to \mathbb{R}^d$ is a **c-approximate kNN-realization** of G in $\mathbb{R}^d$ if*

$$\sum_{(u,v)\in E(G)} \sigma_\phi(u, v) \geq c \cdot |E(G)|,$$

where $\sigma_\phi : V(G) \times V(G) \to \{0, 1\}$ is the indicator function defined as $\sigma_\phi(u, v) = 1$ if we have $|\{v' \in V(G)\backslash\{u\} : \|\phi(u) - \phi(v')\|_2 \leq \|\phi(u) - \phi(v)\|_2\}| \leq k$ and $\sigma_\phi(u, v) = 0$ otherwise.

Our main result is an algorithm for computing a $(1 - \varepsilon)$-approximate kNN-realization of G in $\mathbb{R}^d$ with time complexity $f(k, d, \varepsilon) \cdot n^{O(1)}$, for any given $\varepsilon > 0$ (provided that G is kNN-realizable in $\mathbb{R}^d$), where f is some computable function. In other words, for fixed k and d, we obtain an *efficient polynomial-time approximation scheme* (EPTAS) for the kNN-realization problem in $\mathbb{R}^d$.

At a high level, our algorithm consists of two main steps. In the first step, it computes a set $E \subseteq E(G)$ of edges such that $|E| \leq \varepsilon|E(G)|$ and each weakly-connected component of $G - E$ contains $O_\varepsilon(1)$ vertices. The existence of E follows from the result of Miller et al. [21] on graph separators, and the computation of E relies on the approximate minimum cut algorithm of Chuzhoy et al. [10]. The edges in E are the ones we sacrifice in our approximation and so, in the second step, our algorithm computes a map $\phi : V(G) \to \mathbb{R}^d$ satisfying $\sigma_\phi(u, v) = 1$ for all edges $(u, v) \in E(G)\backslash E$. This turns out to be easy, since the weakly-connected components of $G - E$ are of size $O_\varepsilon(1)$ and we can work on these components individually. These two steps will be presented in Sections 2.1 and 2.2, respectively.

2.1 Splitting the graph by removing few edges

A *balanced cut* of an *undirected* graph H is a subset $E \subseteq E(H)$ such that every connected component of $H - E$ contains at most $|V(H)|/2$ vertices. Every directed graph G naturally corresponds to an undirected graph G_0 defined as $V(G_0) = V(G)$ and $E(G_0) = \{\{u, v\} : (u, v) \in E(G)$ or $(v, u) \in E(G)\}$. We need the following important result.

▶ **Lemma 2.** *Let G be a directed graph that is kNN-realizable in $\mathbb{R}^d$, and G_0 be the undirected graph corresponding to G. Then any subgraph H of G_0 with $|V(H)| \geq 2$ admits a balanced cut of size $O(|V(H)|^{1-\frac{1}{d}})$, where the constant hidden in $O(\cdot)$ only depends on k and d.*

Proof sketch. Let $k, d \in \mathbb{N}$ be fixed numbers. The lemma follows from two results in [21]. Specifically, it was shown in [21] that if a graph G is kNN-realizable in $\mathbb{R}^d$, then its corresponding undirected graph G_0 satisfies the following conditions.

1. The degree of every vertex in G_0 is $O(1)$.
2. For every subgraph H of G_0, there exists $S \subseteq V(H)$ such that $|S| = O(|V(H)|^{1-\frac{1}{d}})$ and every connected component of $H - S$ contains at most $\frac{d+1}{d+2} \cdot |V(H)|$ vertices.

We remark that [21] only claimed condition 2 above for the case $H = G_0$, but the argument extends to any subgraph H of G_0. Using the two conditions above, it is fairly easy to construct the desired balanced cut for any subgraph of G_0. ◀

A *minimum* balanced cut refers to a balanced cut consisting of the minimum number of edges. A *c-approximate* minimum balanced cut refers to a balanced cut whose size is at most $c \cdot \mathsf{opt}$, where opt is the size of a minimum balanced cut. The above lemma implies that

if G is kNN-realizable in $\mathbb{R}^d$, then the size of a minimum balanced cut of any subgraph H of G_0 is $O(|V(H)|^{1-\frac{1}{d}})$. We need the following algorithm by Chuzhoy et al. for computing approximate minimum balanced cuts.

▶ **Theorem 3** ([10]). *For any fixed number $\alpha > 0$, there exists an algorithm that, given an undirected graph H of n vertices and m edges, computes a $\log^{O(1/\alpha)} n$-approximate minimum balanced cut of H in $O(m^{1+\alpha})$ time.*

The following lemma is the main result of this section, which states that one can remove a small fraction of edges from G to split G into small components.

▶ **Lemma 4.** *Let $k, d \in \mathbb{N}$ be fixed numbers. Given a k-regular directed graph G and a number $\varepsilon \in (0, 1]$, one can either compute a subset $E \subseteq E(G)$ such that $|E| \leq \varepsilon|E(G)|$ and each weakly-connected component of $G - E$ contains at most $(\frac{1}{\varepsilon})^{O(1)}$ vertices, or conclude that G is not kNN-realizable in $\mathbb{R}^d$, in $O(|V(G)|^{1+\alpha})$ time for any constant $\alpha > 0$. Here the constants hidden in $O(\cdot)$ depend on k, d, and α.*

Proof. Let G_0 be the corresponding undirected graph of G, and $\alpha > 0$ be a constant. It suffices to compute $E \subseteq E(G_0)$ such that $|E| \leq \varepsilon|E(G_0)|$ and each connected component of $G_0 - E$ contains at most $(\frac{1}{\varepsilon})^{O(1)}$ vertices. Indeed, the edges in G corresponding to E form a subset of $E(G)$ of size $O(\varepsilon|E(G)|)$ which satisfies the desired property. This is fine since our algorithm works for an arbitrary $\varepsilon > 0$.

Pick a constant $\alpha' \in (0, \alpha)$ and let ApprxMinCut(H) denote the algorithm in Theorem 3, which returns an approximate minimum balanced cut of H in $O(|V(H)|^{1+\alpha'})$ time. Our algorithm, presented in Algorithm 1, computes $E \subseteq E(G_0)$ by recursively applying ApprxMinCut as a sub-routine. We fix some threshold $\delta = (\frac{1}{\varepsilon})^c$ for a sufficiently large c (depending on k, d, and α). If $|V(G_0)| \leq \delta$, then we simply return $E = \emptyset$. Otherwise, we apply ApprxMinCut(G_0) to obtain an approximate minimum balanced cut of G_0, and add the edges in the cut to E. Then for every connected component C of $G_0 - E$, we recursively apply our algorithm on C, which returns a subset of $E_C \subseteq E(C)$, and we add the edges in E_C to E. Finally, we return E as the output of our algorithm.

Algorithm 1 CutEdge(G_0).

```
1: if |V(G_0)| ≤ δ then return ∅
2: E ← ApprxMinCut(G_0)
3: 𝒞 ← set of connected components of G_0 − E
4: for every C ∈ 𝒞 do
5:     E ← E ∪ CutEdge(C)
6: return E
```

Clearly, every connected component of $G_0 - E$ contains at most $\delta = (\frac{1}{\varepsilon})^{O(1)}$ vertices. So it suffices to show $|E| \leq \varepsilon|E(G_0)|$ and analyze the running time of the algorithm. To bound $|E|$, we observe that when applying ApprxMinCut on any subgraph H of G_0, the size of the edge set obtained is of size $|V(H)|^{1-\frac{1}{d}} \log^{O(1/\alpha')} |V(H)|$. Indeed, a minimum balanced cut of H has size $O(|V(H)|^{1-\frac{1}{d}})$ by Lemma 2, and ApprxMinCut(H) returns a $\log^{O(1/\alpha')} |V(H)|$-approximate minimum balanced cut of H. Since we defined $\delta = (\frac{1}{\varepsilon})^c$ for a sufficiently large c, we may assume that the size of ApprxMinCut(H) is at most $|V(H)|^{1-\frac{1}{2d}}$ for any subgraph H of G_0 such that $|V(H)| > \delta$. Also, we may assume that $\varepsilon(2^{\frac{1}{3d}} - 1)n^{1-\frac{1}{3d}} > n^{1-\frac{1}{2d}}$ for all $n > \delta$. We shall prove that when applying our algorithm on a subgraph H of G_0 with $|V(H)| = n$, the size of $E \subseteq E(H)$ returned is at most $\varepsilon(n - n^{1-\frac{1}{3d}})$.

We apply induction on n. If $V(H) \leq \delta$, then our algorithm returns an empty set and thus the statement trivially holds. Assume the statement holds for $|V(H)| \in [n-1]$, and we consider the case $|V(H)| = n$ (where $n > \delta$). Our algorithm applies ApprxMinCut(H) to obtain an approximate minimum balanced cut $E_0 \subseteq E(H)$ of H. As aforementioned, $|E_0| \leq n^{1-\frac{1}{2d}}$. Let $C_1, \ldots, C_r$ be the connected components of $H - E_0$. Set $n_i = |V(C_i)|$ for $i \in [r]$. We have $n_i \leq n/2$ for all $i \in [r]$, by the definition of a balanced cut. We recursively call our algorithm on each C_i to obtain a subset $E_i \subseteq E(C_i)$. By our induction hypothesis, $|E_i| \leq \varepsilon(n_i - n_i^{1-\frac{1}{3d}})$. Finally, the algorithm returns $E = \bigcup_{i=0}^{r} E_i$. Thus, we have $|E| = |E_0| + \sum_{i=1}^{r} |E_i|$. As $\sum_{i=1}^{r} n_i = n$, $\sum_{i=1}^{r} |E_i| \leq \varepsilon n - \varepsilon \sum_{i=1}^{r} n_i^{1-\frac{1}{3d}}$. Furthermore, since $n_i \leq n/2$ for all $i \in [r]$ and $\sum_{i=1}^{r} n_i = n$, it holds $\sum_{i=1}^{r} n_i^{1-\frac{1}{3d}} \geq 2(\frac{n}{2})^{1-\frac{1}{3d}} = 2^{\frac{1}{3d}} n^{1-\frac{1}{3d}}$. Combining the bounds for $|E_0|$ and $\sum_{i=1}^{r} |E_i|$, we have

$$|E| \leq n^{1-\frac{1}{2d}} + \varepsilon n - \varepsilon \cdot 2^{\frac{1}{3d}} n^{1-\frac{1}{3d}}.$$

Recall our assumption $\varepsilon(2^{\frac{1}{3d}} - 1)n^{1-\frac{1}{3d}} \geq n^{1-\frac{1}{2d}}$. Together with the inequality above, this gives us $|E| \leq \varepsilon(n - n^{1-\frac{1}{3d}})$, so our induction works. In particular, when we apply Algorithm 1 on G_0, the output $E \subseteq E(G_0)$ satisfies $|E| \leq \varepsilon(|V(G_0)| - |V(G_0)|^{1-\frac{1}{3d}}) \leq \varepsilon|V(G_0)|$.

To analyze the time complexity of Algorithm 1, we consider the recursion tree T of our algorithm when applying on G_0. Each node $t \in T$ corresponds to a recursive call, and denote by $G_0(t)$ the corresponding graph handled in that call (which is a subgraph of G_0). The time cost at a node $t \in T$ is $O(|E(G_0(t))|^{1+\alpha'})$, which is just $O(|V(G_0(t))|^{1+\alpha'})$ because $|E(G_0(t))| \leq k|V(G_0(t))|$. Since ApprxMinCut always returns a balanced cut, we know that $|V(G_0(t))| \leq |V(G_0(t'))|/2$ if t' is the parent of t in T. This implies that the depth of T is $O(\log |V(G_0)|)$. Also, the sum of $|V(G_0(t))|$ for all nodes $t \in T$ at one level of T is at most $|V(G_0)|$. Thus $\sum_{t \in T} |V(G_0(t))| \leq |V(G_0)| \log |V(G_0)|$ and $\sum_{t \in T} |V(G_0(t))|^{1+\alpha'} \leq |V(G_0)|^{1+\alpha'} \log |V(G_0)|$. The latter implies that the time complexity of Algorithm 1 is $O(|V(G_0)|^{1+\alpha'} \log |V(G_0)|)$, which is $O(|V(G_0)|^{1+\alpha})$ since $\alpha > \alpha'$. We remark that a more careful analysis can be applied to show that the running time of our algorithm is actually $O(|V(G_0)|^{1+\alpha'})$ instead of $O(|V(G_0)|^{1+\alpha'} \log |V(G_0)|)$. But for simplicity, we chose this looser analysis, which is already sufficient for our purpose. ◀

2.2 Computing the approximate realization

After computing the set $E \subseteq E(G)$ of edges using Lemma 4, we consider the graph $G - E$. Let $C_1, \ldots, C_r$ be the weakly-connected components of $G - E$. We have $|V(C_i)| = (\frac{1}{\varepsilon})^{O(1)}$ by Lemma 4. For each $i \in [r]$, we want to compute a "kNN-realization" of C_i in $\mathbb{R}^d$. Of course, here each C_i is not necessarily k-regular, and thus a kNN-realization of C_i is not defined. But we can slightly generalize the definition of kNN-realization as follows. For a directed graph C, we say a map $\phi : V(C) \to \mathbb{R}^d$ is a *quasi-kNN-realization* of C in $\mathbb{R}^d$ if for every edge $(u, v) \in E(C)$, $|\{v' \in V(C) \backslash \{u\} : \|\phi(u) - \phi(v')\|_2 \leq \|\phi(u) - \phi(v)\|_2\}| \leq k$. By this definition, a kNN-realization is also a quasi-kNN-realization. Also, it is clear that if $\phi : V(H) \to \mathbb{R}^d$ is a quasi-kNN-realization of a directed graph H in $\mathbb{R}^d$, then for any subgraph C of H, the map $\phi_{|V(C)}$ is a quasi-kNN-realization of C in $\mathbb{R}^d$. Therefore, if G is kNN-realizable in $\mathbb{R}^d$, then each of $C_1, \ldots, C_r$ admits a quasi-kNN-realization in $\mathbb{R}^d$.

Next, we discuss how to compute a quasi-kNN-realization of each C_i in $\mathbb{R}^d$ (or conclude it does not exist). To this end, we need the following lemma.

▶ **Lemma 5.** *Let C be a directed graph where $|V(C)| \geq k + 1$. If C admits a quasi-kNN-realization in $\mathbb{R}^d$, then there exists a k-regular supergraph C' of C with $V(C') = V(C)$ such that C' is kNN-realizable in $\mathbb{R}^d$.*

Proof. Assume $\phi : V(C) \to \mathbb{R}^d$ is a quasi-kNN-realization of C in $\mathbb{R}^d$. We can slightly perturb ϕ so that for any distinct $u, v, v' \in V(C)$, $\|\phi(u) - \phi(v')\|_2 \neq \|\phi(u) - \phi(v)\|_2\}$. Now define C' as a directed graph with $V(C') = V(C)$ and $E(C') = \{(u, v) \in V(C') \times V(C') : u \neq v$ and $\mathsf{rank}_\phi(u, v) \leq k\}$, where

$$\mathsf{rank}_\phi(u, v) = |\{v' \in V(C')\backslash\{u\} : \|\phi(u) - \phi(v')\|_2 \leq \|\phi(u) - \phi(v)\|_2\}|.$$

The construction guarantees that C' is k-regular and ϕ is a kNN-realization of C' in $\mathbb{R}^d$. Since ϕ is a quasi-kNN-realization of C, $E(C) \subseteq E(C')$ and thus C' is a supergraph of C. ◀

If $|V(C_i)| \leq k$, then any map from $V(C_i)$ to $\mathbb{R}^d$ is a quasi-kNN-realization of C_i. Otherwise, by the above lemma, to compute a quasi-kNN-realization of C_i in $\mathbb{R}^d$, it suffices to consider every supergraph C_i' of C_i with $V(C_i') = V(C_i)$ and try to compute a kNN-realization of C_i' (or conclude that C_i' is not kNN-realizable). Note that the number of such supergraphs is at most $\exp((\frac{1}{\varepsilon})^{O(1)})$ because $|V(C_i)| = (\frac{1}{\varepsilon})^{O(1)}$. If we obtain a kNN-realization of some C_i', then it is a quasi-kNN-realization of C_i. To compute a kNN-realization $\phi : V(C_i') \to \mathbb{R}^d$ of C_i', we formulate the problem as finding a solution to a system of $(\frac{1}{\varepsilon})^{O(1)}$ degree-2 polynomial inequalities on $(\frac{1}{\varepsilon})^{O(1)}$ variables. For each $v \in V(C_i')$, we represent the coordinates of $\phi(v)$ by d variables $x_1(v), \ldots, x_d(v)$. Then for all distinct $u, v, v' \in V(C_i')$ such that $(u, v) \in E(C_i')$ and $(u, v') \notin E(C_i')$, we introduce a degree-2 polynomial inequality $\sum_{j=1}^d (x_j(u) - x_j(v))^2 \leq \sum_{j=1}^d (x_j(u) - x_j(v'))^2$, which expresses $\|\phi(u) - \phi(v)\|_2 < \|\phi(u) - \phi(v')\|_2$. Clearly, the solutions to this system of inequalities one-to-one correspond to the kNN-realizations of C_i' in $\mathbb{R}^d$. Renegar [23] showed that a system of p degree-2 polynomial inequalities on q variables can be solved in $p^{O(q)}$ time. Therefore, we can compute in $(\frac{1}{\varepsilon})^{(\frac{1}{\varepsilon})^{O(1)}}$ time a kNN-realization of C_i' in $\mathbb{R}^d$ (or decide its non-existence). This further implies that we can compute in $(\frac{1}{\varepsilon})^{(\frac{1}{\varepsilon})^{O(1)}}$ time a quasi-kNN-realization of C_i in $\mathbb{R}^d$ (or decide its non-existence). The total time cost for all C_i is then $(\frac{1}{\varepsilon})^{(\frac{1}{\varepsilon})^{O(1)}} \cdot n$, since $r \leq n$.

If C_i does not admit a quasi-kNN-realization in $\mathbb{R}^d$ for some $i \in [r]$, then we can directly conclude that G is not kNN-realizable in $\mathbb{R}^d$. Otherwise we construct a $(1 - \varepsilon)$-approximate kNN-realization of G in $\mathbb{R}^d$ as follows. Let $\phi_i : V(C_i) \to \mathbb{R}^d$ be the quasi-kNN-realization of C_i in $\mathbb{R}^d$ we compute, for $i \in [r]$. Define a map $\phi : V(G) \to \mathbb{R}^d$ as follows. Pick a vector $\vec{x} \in \mathbb{R}^d$ such that $\|\vec{x}\|_2 >> \max_{i\in[r]} \max_{u,v\in V(C_i)} \|\phi_i(u) - \phi_i(v)\|_2$. Then for each $i \in [r]$ and each $v \in V(C_i)$, set $\phi(v) = i\vec{x} + \phi_i(v)$. The choice of $\vec{x}$ guarantees that for a vertex $v \in V(C_i)$, the ϕ-images of the vertices in C_i are closer to $\phi(v)$ than the ϕ-images of the vertices outside C_i. Also, for any $u, v \in V(C_i)$, $\|\phi(u) - \phi(v)\|_2 = \|\phi_i(u) - \phi_i(v)\|_2$. Therefore, for any $u, v \in V(C_i)$, we have

$$\begin{aligned}
&\{v' \in V(G)\backslash\{u\} : \|\phi(u) - \phi(v')\|_2 \leq \|\phi(u) - \phi(v)\|_2\} \\
&= \{v' \in V(C_i)\backslash\{u\} : \|\phi(u) - \phi(v')\|_2 \leq \|\phi(u) - \phi(v)\|_2\} \\
&= \{v' \in V(C_i)\backslash\{u\} : \|\phi_i(u) - \phi_i(v')\|_2 \leq \|\phi_i(u) - \phi_i(v)\|_2\}.
\end{aligned}$$

Recall the function $\sigma_\phi : V(G) \times V(G) \to \{0, 1\}$ in Definition 1. The above equality shows that for $u, v \in V(C_i)$, if $|\{v' \in V(C_i)\backslash\{u\} : \|\phi_i(u) - \phi_i(v')\|_2 \leq \|\phi_i(u) - \phi_i(v)\|_2\}| \leq k$, then $\sigma_\phi(u, v) = 1$. It follows that $\sigma_\phi(u, v) = 1$ for any $(u, v) \in E(C_i)$ because ϕ_i is a quasi-kNN-realization of C_i, so we have

$$\sum_{(u,v)\in E(G)} \sigma_\phi(u, v) \geq \sum_{i=1}^{r} |E(C_i)| = |E(G)| - |E| \geq (1 - \varepsilon) \cdot |E(G)|.$$

Therefore, ϕ is a $(1-\varepsilon)$-approximate kNN-realization of G in $\mathbb{R}^d$. Combining the time costs for computing E and the maps $\phi_1, \ldots, \phi_r$, the overall time complexity of our algorithm is $O(n^{1+\alpha})$, where $O(\cdot)$ hides a constant depending on k, d, α, and ε.

▶ **Theorem 6.** *Let $\alpha > 0$ be any fixed number. Given a k-regular directed graph G of n vertices and a number $\varepsilon > 0$, one can compute in $f(k, d, \varepsilon) \cdot n^{1+\alpha}$ time a $(1-\varepsilon)$-approximate kNN-realization of G in $\mathbb{R}^d$, or conclude that G is not kNN-realizable in $\mathbb{R}^d$, where f is some computable function depending on α.*

3 kNN-Realization in $\mathbb{R}^1$

To the best of our knowledge, the kNN-realization problem on the line does not seem to have been studied, and it is the focus of this section. A number of line embedding problems are NP-hard as mentioned earlier, including triplet constraints and betweenness problems [9]. In the former, we are given a set of triplet constraints of the form $d(a, b) < d(a, c)$, while in the latter each constraint (a, b, c) requires the ordering to satisfy $a < b < c$. Given the hardness result, the focus in these problems is on approximating the maximum number of satisfied constraints in the linear ordering [17].

Unlike these intractable embedding problems on the line, we show that the partial order constraints implied by a kNN-realization problem have sufficiently rich structure to admit a polynomial time solution. Most of the difficulty is in *deciding* whether a k-regular directed graph G is kNN-realizable on the line; if the answer is yes, then one can easily compute the embedding in polynomial time using linear programming. (It is also worth pointing out that the decision problem for kNN-realization on the line is significantly more complicated than the special case of triplet constraints in [17] when *all* $\binom{n}{3}$ *triples are specified.* Indeed, in that case, one can guess the leftmost point, and then all the remaining points are immediately determined by the triplet constraints. This is not the case in kNN-realization: fixing the leftmost point does not fix its neighbors' order.)

The *decision* version of the kNN-realization problem, of course, is easy *if* we are given a permutation of G's vertices $(v_1, \ldots, v_n)$. In this case, we can easily compute a kNN-realization $\phi : V(G) \to \mathbb{R}^1$ satisfying $\phi(v_1) < \cdots < \phi(v_n)$, or decide that a feasible realization does not exist, by formulating the problem as a linear program (LP) with n variables $x_1, \ldots, x_n$ and the following set of constraints

- $x_i \le x_j$ for all $i, j \in [n]$ with $i \le j$,
- $\Delta_{i,j} < \Delta_{i,k}$ for all distinct $i, j, k \in [n]$ such that $(v_i, v_j) \in E(G)$ and $(v_i, v_k) \notin E(G)$, where $\Delta_{i,j} = x_{\max\{i,j\}} - x_{\min\{i,j\}}$ and $\Delta_{i,k} = x_{\max\{i,k\}} - x_{\min\{i,k\}}$.

Clearly, if $x_1, \ldots, x_n$ satisfy these constraints, then setting $\phi(v_i) = x_i$ gives us the desired realization. On the other hand, if ϕ is the realization we want, then the numbers $x_1, \ldots, x_n$ where $x_i = \phi(v_i)$ must satisfy the constraints. Therefore, we begin with this key problem: efficiently computing an ordering of the images of the vertices on $\mathbb{R}^1$.

3.1 Finding the vertex ordering

We say an ordering $(v_1, \ldots, v_n)$ of $V(G)$ is a *feasible vertex ordering* of G if there exists a kNN-realization $\phi : V(G) \to \mathbb{R}^1$ of G in $\mathbb{R}^1$ satisfying $\phi(v_1) < \cdots < \phi(v_n)$. The goal of this section is to give an $O(kn)$-time algorithm for computing a feasible vertex ordering of G (assuming it exists). We may assume, without loss of generality, that G is weakly-connected; otherwise we can consider each weakly-connected component of G individually.

Our first observation states two key properties of a feasible vertex ordering: (1) for each v_i its k out-neighbors form a contiguous block, and for different v_i's their blocks have the same linear ordering as the vertex ordering, and (2) for each v_i its in-neighbors also form a continuous block, which extends at most k vertices to the left and at most k to the right of v_i. (Recall that both $\mathsf{out}[v]$ and $\mathsf{in}[v]$ include v, for all vertices $v \in V(G)$.) See figure 1.

▶ **Observation 7.** *A feasible vertex ordering $(v_1, \dots, v_n)$ of G satisfies the following.*

1. *There exist $p_1, \dots, p_n \in [n-k]$ such that* **(i)** $\mathsf{out}[v_i] = \{v_{p_i}, \dots, v_{p_i+k}\}$ *for all $i \in [n]$,* **(ii)** $p_1 \leq \cdots \leq p_n$*, and* **(iii)** $p_i \leq i \leq p_i + k$ *for all $i \in n$.*
2. *There exist $q_1, \dots, q_n, q'_1, \dots, q'_n \in [n]$ such that* **(i)** $\mathsf{in}[v_i] = \{v_{q_i}, \dots, v_{q'_i}\}$ *for all $i \in [n]$,* **(ii)** $q_1 \leq \cdots \leq q_n$*,* **(iii)** $q'_1 \leq \cdots \leq q'_n$*, and* **(iv)** $q_i \leq q'_i \leq q_i + 2k$ *for all $i \in [n]$.*

Proof. Let $\phi : V(G) \to \mathbb{R}^1$ be a kNN-realization of G in $\mathbb{R}^1$ satisfying $\phi(v_1) < \cdots < \phi(v_n)$. Clearly, for each $i \in [n]$, the $k+1$ points in $\{\phi(v_1), \dots, \phi(v_n)\}$ closest to $\phi(v_i)$ are $\{\phi(v_{p_i}), \dots, \phi(v_{p_i+k})\}$ for some $p_i \in [n-k]$ such that $p_i \leq i \leq p_i + k$. Furthermore, one can easily verify that $p_1 \leq \cdots \leq p_n$. Assume $p_i > p_j$ for some $i, j \in [n]$ with $i < j$. Then we have $\phi(v_{p_j}) < \phi(v_{p_i}) \leq \phi(v_i) < \phi(v_j) \leq \phi(v_{p_j+k}) < \phi(v_{p_i+k})$. On the other hand, since $v_{p_i+k} \notin \{\phi(v_{p_j}), \dots, \phi(v_{p_j+k})\}$ and $v_{p_j} \notin \{\phi(v_{p_i}), \dots, \phi(v_{p_i+k})\}$,

$$\phi(v_j) - \phi(v_{p_j}) \leq \phi(v_{p_i+k}) - \phi(v_j) \leq \phi(v_{p_i+k}) - \phi(v_i) \leq \phi(v_i) - \phi(v_{p_j}),$$

which contradicts the fact $\phi(v_{p_j}) < \phi(v_{p_i}) \leq \phi(v_i) < \phi(v_j)$. By the definition of kNN-realization, we see $\mathsf{out}[v_i] = \{v_{p_i}, \dots, v_{p_i+k}\}$ for all $i \in [n]$. This proves condition 1.

Condition 2 follows from condition 1. Observe that for each $i \in [n]$, it holds that $\mathsf{in}[v_i] = \{v_j : p_j \leq i \leq p_j + k\} = \{v_j : i - k \leq p_j \leq i\}$. Since $p_1 \leq \cdots \leq p_n$, this implies $\mathsf{in}[v_i] = \{v_{q_i}, \dots, v_{q'_i}\}$ where $q_i = \min\{j : i - k \leq p_j \leq i\}$ and $q'_i = \max\{j : i - k \leq p_j \leq i\}$. The monotoncities $q_1 \leq \cdots \leq q_n$ and $q'_1 \leq \cdots \leq q'_n$ follow directly from the monotoncity $p_1 \leq \cdots \leq p_n$. It suffices to show that $q_i \leq q'_i \leq q_i + 2k$ for all $i \in [n]$. The inequality $q_i \leq q'_i$ is trivial. To see $q'_i \leq q_i + 2k$, assume that $q'_i > q_i + 2k$. Then there exists $j, j' \in [n]$ with $j' > j + 2k$ such that $i - k \leq p_j \leq i$ and $i - k \leq p_{j'} \leq i$. By condition 1, we have $j \geq p_j$ and $p_{j'} \geq j' - k > j + k$. Therefore, $p_{j'} > j + k \geq p_j + k \geq (i - k) + k = i$, which contradicts the fact $i - k \leq p_{j'} \leq i$. This proves condition 2. ◀

Consider the equivalence relation $\sim$ defined on $V(G)$ as $u \sim v$ iff $\mathsf{out}[u] = \mathsf{out}[v]$. Let $\mathcal{C}_G$ be the set of equivalence classes of this relation, which is a partition of $V(G)$ into groups of vertices with the same set of out neighbors. One can easily compute $\mathcal{C}_G$ in $O(k^2 n)$ time as follows. Given a vertex $v \in V(G)$, we check whether $\mathsf{out}[u] = \mathsf{out}[v]$, for all $u \in \mathsf{out}[v]$, which takes $O(k^2)$ time since $|\mathsf{out}[v]| = k + 1$ and each equality test takes $O(k)$ time. Thus, the time for computing all classes is $O(k^2 n)$. Interestingly, we use Observation 7 to compute $\mathcal{C}_G$ in $O(kn)$ time, if G is kNN-realizable in $\mathbb{R}^1$, as shown in the following Lemma. Due to space constraints the proof is included in the full version.

▶ **Lemma 8.** *Let G be a k-regular directed graph of n vertices. One can compute in $O(kn)$ time a partition $\mathcal{C}$ of $V(G)$ such that $\mathcal{C} = \mathcal{C}_G$ if G is kNN-realizable in $\mathbb{R}^1$.*

Write $r = |\mathcal{C}_G|$. Let $(v_1, \dots, v_n)$ be a feasible vertex ordering of G. Observation 7 implies that each class $C \in \mathcal{C}_G$ is a set of consecutive vertices in the sequence $(v_1, \dots, v_n)$, i.e., $C = \{v_\alpha, \dots, v_\beta\}$ for some $\alpha, \beta \in [n]$. Define $\mathsf{out}[C] = \mathsf{out}[v]$ for an arbitrary vertex $v \in C$. Therefore, there is a natural ordering $(C_1, \dots, C_r)$ of the classes in $\mathcal{C}_G$, in which the indices of the vertices in C_i are smaller than the indices of the vertices in C_j for all $i, j \in [r]$ with $i < j$. We call $(C_1, \dots, C_r)$ a *feasible class ordering* of G.

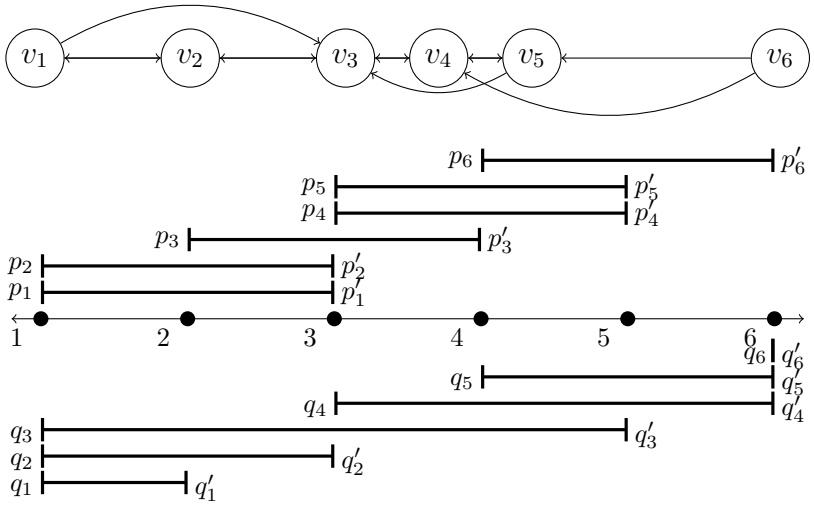


Figure 1 A 2-regular graph on 6 vertices and its realization (top). The values of p, p', q, q' for each vertex illustrate a natural ordering of the in and out-neighborhoods corresponding to the feasible vertex ordering $v_1, \cdots, v_6$ (bottom).

▶ **Observation 9.** *If $(C_1, \ldots, C_r)$ is the feasible class ordering of G corresponding to a feasible vertex ordering $(v_1, \ldots, v_n)$ of G, then there exist $c_1, \ldots, c_r \in [n-k]$ satisfying*

- $\mathsf{out}[C_i] = \{v_{c_i}, \ldots, v_{c_i+k}\}$ *for all* $i \in [r]$,
- $c_1 < \cdots < c_r$,
- *if G is weakly-connected, then $c_{i+1} \leq c_i + k$ for all $i \in [r-1]$.*

Proof. Let $p_1, \ldots, p_n \in [n-k]$ be the indices in condition 1 of Observation 7, for the feasible vertex ordering $(v_1, \ldots, v_n)$. For each $i \in [r]$, set $c_i = p_s$ where $s = 1 + \sum_{j=1}^{i-1} |C_j|$, and we then have $\mathsf{out}[C_i] = \mathsf{out}[v_s] = \{v_{c_i}, \ldots, v_{c_i+k}\}$. Since $p_1 \leq \cdots \leq p_n$, we have $c_1 \leq \cdots \leq c_r$. Furthermore, for all $i \in [r-1]$, we have $c_i \neq c_{i+1}$, simply because $\mathsf{out}[C_i] \neq \mathsf{out}[C_{i+1}]$. Thus, $c_1 < \cdots < c_r$. To see the last property, assume $c_{i+1} > c_i + k$ for some $i \in [r-1]$. Then $(\bigcup_{j=1}^{i} \mathsf{out}[C_j]) \cap (\bigcup_{j=i+1}^{r} \mathsf{out}[C_j]) = \emptyset$. Note that $\bigcup_{j=1}^{i} C_j \subseteq \bigcup_{j=1}^{i} \mathsf{out}[C_j]$ and $\bigcup_{j=i+1}^{r} C_j \subseteq \bigcup_{j=i+1}^{r} \mathsf{out}[C_j]$. So there is no edge between $\bigcup_{j=1}^{i} C_j$ and $\bigcup_{j=i+1}^{r} C_j$, which implies that G is not weakly connected. ◀

To compute a feasible vertex ordering of G, we first compute a feasible class ordering, using the previous observation. Due to space constraints the proof of the following lemma is included in the full version.

▶ **Lemma 10.** *Let G be a weakly-connected k-regular directed graph of n vertices. Given $\mathcal{C}_G$, one can compute in $O(kn)$ time an ordering of $\mathcal{C}_G$, which is a feasible class ordering of G if G is kNN-realizable in $\mathbb{R}^1$.*

Next, we show how to compute a corresponding feasible vertex ordering of G given a feasible class ordering $(C_1, \ldots, C_r)$. By definition, we know that in the feasible vertex ordering, the vertices in C_i must appear before the vertices in C_j for all $i, j \in [r]$ with $i < j$. So it suffices to figure out the "local" ordering of the vertices in each class C_i. We do this by considering the in-neighbors of each vertex. As observed before, for each $v \in V(G)$, $\mathsf{in}[v]$ is the union of several classes in $\mathcal{C}_G$, and in addition, $\mathsf{in}[v] = \bigcup_{i=p}^{q} C_i$ for some $p, q \in [r]$, by condition 2 of Observation 7. It turns out that we can sort the vertices in each class according to the values of p and q. Formally, we prove the following lemma.

▶ **Lemma 11.** *Let G be a weakly-connected k-regular directed graph of n vertices, and $r = |\mathcal{C}_G|$. Given an ordering $(C_1, \ldots, C_r)$ of $\mathcal{C}_G$, one can compute in $O(kn)$ time an ordering $(v_1, \ldots, v_n)$ of $V(G)$ such that if $(C_1, \ldots, C_r)$ is a feasible class ordering of G, then $(v_1, \ldots, v_n)$ is a feasible vertex ordering of G.*

Proof. Our algorithm for computing a feasible vertex ordering of G is presented in Algorithm 2. First, for each $v \in V(G)$, we compute a pair $R(v) = (p, q)$ where $p \in [r]$ (resp., $q \in [r]$) is the minimum (resp., maximum) index such that $C_p \subseteq \mathsf{in}[v]$ (resp., $C_q \subseteq \mathsf{in}[v]$). We define a partial order $\preceq$ on index-pairs by setting $(p, q) \preceq (p', q')$ if $p + q \leq p' + q'$. Then we order the vertices in each class C_i as $(u_1, \ldots, u_{|C_i|})$ such that $R(u_1) \preceq \cdots \preceq R(u_{|C_i|})$. If there exist $u, v \in C_i$ such that $R(u)$ and $R(v)$ are incomparable under the order $\preceq$, then we just arbitrarily order the vertices in C_i. We will see later that in this case, $(C_1, \ldots, C_r)$ is not a feasible class ordering of G. By concatenating the orderings of $C_1, \ldots, C_r$, we obtain an ordering $(v_1, \ldots, v_n)$ of $V(G)$.

We first show that Algorithm 2 can be implemented in $O(kn)$ time. Computing $R(v)$ for all $v \in V(G)$ can be done in $O(kn)$ time, because $|\mathsf{in}[v]| = O(k)$ by (ii) of Observation 7. To see the time cost for ordering the vertices in each C_i, observe that $|C_i| \leq k$. Indeed, $v \in \mathsf{out}[v] = \mathsf{out}[C_i]$ for all $v \in C_i$, which implies $C_i \subseteq \mathsf{out}[C_i]$ and thus $|C_i| \leq |\mathsf{out}[C_i]| = k$. If we sort the vertices in C_i directly, it takes $O(k \log k)$ time. One can improve the time cost to $O(k)$ by observing that for any $v \in C_i$, the pair $R(v) = (p, q)$ satisfies $p \geq i - k$ and $q \leq i + k$, by condition 2 of Observation 7, which implies $2i - 2k \leq p + q \leq 2i + 2k$. In other words, in the sorting task, the keys are all in the range $\{2i - 2k, \ldots, 2i + 2k\}$, whose size is $O(k)$. Thus, the task can be done in $O(k)$ time using bucket sorting. It follows that Algorithm 2 can be implemented in $O(kn)$ time.

To see the correctness of our algorithm, suppose $(C_1, \ldots, C_r)$ is a feasible class ordering of G, and let $(v_1^*, \ldots, v_n^*)$ be a corresponding feasible vertex ordering of G. We do not necessarily have $(v_1, \ldots, v_n) = (v_1^*, \ldots, v_n^*)$. However, as we will see, it holds that $(R(v_1), \ldots, R(v_n)) = (R(v_1^*), \ldots, R(v_n^*))$, which turns out to be sufficient. By (ii) of Observation 7, for each $v \in V(G)$ with $R(v) = (p, q)$, we have $\mathsf{in}[v] = \bigcup_{i=p}^{q} C_i$, and furthermore, $R(v_1^*) \preceq \cdots \preceq R(v_n^*)$. Now consider a class C_i. Note that $C_i = \{v_\alpha, \ldots, v_\beta\} = \{v_\alpha^*, \ldots, v_\beta^*\}$, where $\alpha = 1 + \sum_{j=1}^{i-1} |C_j|$ and $\beta = \sum_{j=1}^{i} |C_j|$. We have $R(v_\alpha^*) \preceq \cdots \preceq R(v_\beta^*)$, and our algorithm guarantees that $R(v_\alpha) \preceq \cdots \preceq R(v_\beta)$. Therefore, $(R(v_\alpha), \ldots, R(v_\beta)) = (R(v_\alpha^*), \ldots, R(v_\beta^*))$. It then follows that $(R(v_1), \ldots, R(v_n)) = (R(v_1^*), \ldots, R(v_n^*))$.

To see $(v_1, \ldots, v_n)$ is a feasible vertex ordering of G, we further observe that the function $\pi : V(G) \to V(G)$ defined as $\pi(v_i) = v_i^*$ for $i \in [n]$ is an automorphism of G. Consider indices $i, j \in [n]$. We have $\mathsf{out}[v_i] = \mathsf{out}[v_i^*]$, since v_i and v_i^* belong to the same class in $\mathcal{C}_G$. Thus, $(v_i, v_j) \in E(G)$ iff $(v_i^*, v_j) \in E(G)$. On the other hand, because $R(v_j) = R(v_j^*)$, we have $\mathsf{in}[v_j] = \mathsf{in}[v_j^*]$ and hence $(v_i^*, v_j) \in E(G)$ iff $(v_i^*, v_j^*) \in E(G)$. As such, $(v_i, v_j) \in E(G)$ iff $(v_i^*, v_j^*) \in E(G)$, which implies that π is an automorphism of G. Now consider a kNN-realization $\phi : V(G) \to \mathbb{R}^1$ of G in $\mathbb{R}^1$ satisfying $\phi(v_1^*) < \cdots < \phi(v_n^*)$. Set $\phi' = \phi \circ \pi$, which is a map from $V(G)$ to $\mathbb{R}^1$. As π is an automorphism of G, ϕ' is also a kNN-realization of G. Furthermore, $\phi'(v_i) = \phi(\pi(v_i)) = \phi(v_i^*)$ for all $i \in [n]$, which implies $\phi'(v_1) < \cdots < \phi'(v_n)$. The existence of ϕ' shows that $(v_1, \ldots, v_n)$ is a feasible vertex ordering of G. ◀

▶ **Theorem 12.** *Given a k-regular directed graph G of n vertices, one can compute in $O(kn)$ time an ordering $(v_1, \ldots, v_n)$ of $V(G)$, which is a feasible vertex ordering of G if G is kNN-realizable in $\mathbb{R}^1$.*

Proof. For the case where G is weakly-connected, the theorem follows directly from Lemmas 8, 10 and 11. Otherwise, we compute the orderings for the weakly-connected components of G individually and concatenate them; this gives us the desired ordering of $V(G)$. ◀

Algorithm 2 VertexOrder$(G, (C_1, \ldots, C_r))$.

for $v \in V(G)$ **do**
  $p \leftarrow \min\{i \in [r] : C_i \subseteq \mathsf{in}[v]\}$
  $q \leftarrow \max\{i \in [r] : C_i \subseteq \mathsf{in}[v]\}$
  $R(v) \leftarrow (p, q)$
for $i = 1, \ldots, r$ **do**
  $L_i \leftarrow$ an ordering $(u_1, \ldots, u_{|C_i|})$ of C_i satisfying $R(u_1) \preceq \cdots \preceq R(u_{|C_i|})$
return $L_1 + \cdots + L_r$

3.2 Deciding the realizability and computing the realization

To decide the kNN-realizability of G, we first run the algorithm of Theorem 12, which returns the vertex ordering $(v_1, \ldots, v_n)$ of $V(G)$, and then run the linear program given at the beginning of Section 3 for this ordering. Either the LP is feasible, in which case the solution gives a kNN-realization of G in $\mathbb{R}^1$, or LP is infeasible, in which case we can conclude that $(v_1, \ldots, v_n)$ is not a feasible vertex ordering and thus G is not kNN-realizable in $\mathbb{R}^1$. Thus, the kNN-realization problem in $\mathbb{R}^1$ can be solved in polynomial time.

Interestingly, we can show that if we are only interested in the *decision* problem (and not actual embedding), then solving the LP is not necessary. Specifically, we can decide whether $(v_1, \ldots, v_n)$ is a feasible vertex by simply checking condition 1 of Observation 7. If the ordering satisfies that condition, then the LP is *always feasible*. The proof of the following lemma is somewhat technical and is presented in the full version.

▶ **Lemma 13.** *Let G be a k-regular directed graph of n vertices. An ordering $(v_1, \ldots, v_n)$ of $V(G)$ is a feasible vertex ordering of G iff it satisfies condition 1 of Observation 7. In particular, one can decide whether a given ordering of $V(G)$ is a feasible vertex ordering of G or not in $O(kn)$ time.*

We can now state the main result of this section.

▶ **Theorem 14.** *Given a k-regular directed graph G of n vertices, one can decide in $O(kn)$ time whether G is kNN-realizable in $\mathbb{R}^1$, and if so, a kNN-realization of G in $\mathbb{R}^1$ can be computed in $O(n^{2.5}\mathsf{poly}(\log n))$ time.*

4 Concluding remarks and extensions

We considered the problem of realizing a directed graph G as an Euclidean kNN graph. Our key results are: (1) for any fixed d, we can efficiently embed at least a $1 - \varepsilon$ fraction of G's edges in $\mathbb{R}^d$ or conclude that G is not realizable, and (2) a linear time algorithm to decide if G is realizable in $\mathbb{R}^1$. Our theorems extend to the case where the neighbors of each vertex in G are given as a *ranked* list, meaning that the embedding must satisfy $||\phi(v) - \phi(u_i)|| < ||\phi(v) - \phi(u_{i+1})||$, for $i = 1, \ldots, k-1$, where u_i is the ith nearest neighbor of v (except in $\mathbb{R}^1$ where we need to solve the LP to decide if it is feasible). Our approximation scheme also applies to other proximity graphs that meet the following conditions: (1) the graph can be partitioned into constant-sized components using a sublinear size separator, and (2) each component's edges can be embedded independently. For example, we can approximately embed Delaunay triangulations in the plane with maximum degree $k = O(n^{\frac{1}{3}})$.

References

1 A. Agrawal, S. Saurabh, and M. Zehavi. A finite algorithm for the realizabilty of a delaunay triangulation. In *17th International Symposium on Parameterized and Exact Computation*, volume 249, pages 1:1–1:16, 2022. doi:10.4230/LIPICS.IPEC.2022.1.

2 N. Alon, M. Bădoiu, E. D. Demaine, M. Farach-Colton, M. Hajiaghayi, and A. Sidiropoulos. Ordinal embeddings of minimum relaxation: General properties, trees, and ultrametrics. *ACM Trans. Algorithms*, 4(4), 2008. doi:10.1145/1383369.1383377.

3 Y. Bilu and N. Linial. Monotone maps, sphericity and bounded second eigenvalue. *Journal of Combinatorial Theory, Series B*, 95(2):283–299, November 2005. doi:10.1016/J.JCTB.2005.04.005.

4 A. Bogomolnaia and J.-F. Laslier. Euclidean preferences. *Journal of Mathematical Economics*, 43(2):87–98, 2007.

5 P. Bose, W. J. Lenhart, and G. Liotta. Characterizing proximity trees. *Algorithmica*, 16:83–110, 1996. doi:10.1007/BF02086609.

6 J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel J. Math*, pages 46–52, 1985.

7 M. Bădoiu, E. D. Demaine, M. Hajiaghayi, A. Sidiropoulos, and M. Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 21–34, 2008.

8 V. Chatziafratis and P. Indyk. Dimension-accuracy tradeoffs in contrastive embeddings for triplets, terminals & top-k nearest neighbors. In *Symposium on Simplicity in Algorithms*, pages 230–243. 2024.

9 B. Chor and M. Sudan. A geometric approach to betweenness. *SIAM J. Discret. Math.*, 11(4):511–523, 1998. doi:10.1137/S0895480195296221.

10 J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167. IEEE, 2020.

11 M. de Berg, O. Cheong, M. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

12 M. Dillencourt. Toughness and delaunay triangulations. In *Proceedings of the Third Annual Symposium on Computational Geometry*, pages 186–194, 1987.

13 P. Eades and S. Whitesides. The realization problem for euclidean minimum spanning trees is np-hard. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 49–56, 1994.

14 P. Eades and S. Whitesides. The logic engine and the realization problem for nearest neighbor graphs. *Theoretical Computer Science*, 169(1):23–37, 1996. doi:10.1016/S0304-3975(97)84223-5.

15 D. Eppstein, M. Paterson, and F. F. Yao. On nearest-neighbor graphs. *Discret. Comput. Geom.*, 17:263–282, 1997. doi:10.1007/PL00009293.

16 P. Erdős, F. Harari, and W. T. Tutte. On the dimension of a graph. *Mathematika*, 12:118–122, 1965.

17 B. Fan, D. Ihara, N. Mohammadi, F Sgherzi, A. Sidiropoulos, and M. Valizadeh. Learning Lines with Ordinal Constraints. *APPROX/RANDOM*, 176:45:1–45:15, 2020. doi:10.4230/LIPICS.APPROX/RANDOM.2020.45.

18 W. Johnson and J. Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

19 H. Maehara. Space graphs and sphericity. *Discrete Applied Mathematics*, 7(1):55–64, 1984. doi:10.1016/0166-218X(84)90113-6.

20 J. Matousek. *Lectures on Discrete Geometry*. Springer, 2002.

21 G. L. Miller, S.H. Teng, W. P. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44:1–29, 1997. doi:10.1145/256292.256294.

22 C. L. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8:265–293, 1992. doi:10.1007/BF02293049.

23 J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13(3):255–299, 1992.

24 K. Sugihara. Simpler proof of a realizability theorem on delaunay triangulations. *Information Processing Letters*, 50(4):173–176, 1994. doi:10.1016/0020-0190(94)00027-1.

The Point-Boundary Art Gallery Problem Is $\exists\mathbb{R}$-Hard

Jack Stade
University of Copenhagen, Denmark

Abstract

We resolve the complexity of the point-boundary variant of the art gallery problem, showing that it is $\exists\mathbb{R}$-complete, meaning that it is equivalent under polynomial time reductions to deciding whether a system of polynomial equations has a real solution.

The art gallery problem asks whether there is a configuration of *guards* that together can see every point inside of an *art gallery* modeled by a simple polygon. The original version of this problem (which we call the point-point variant) was shown to be $\exists\mathbb{R}$-hard [Abrahamsen, Adamaszek, and Miltzow, JACM 2021], but the complexity of the variant where guards only need to guard the walls of the art gallery was left as an open problem. We show that this variant is also $\exists\mathbb{R}$-hard.

Our techniques can also be used to greatly simplify the proof of $\exists\mathbb{R}$-hardness of the point-point art gallery problem. The gadgets in previous work could only be constructed by using a computer to find complicated rational coordinates with specific algebraic properties. All of our gadgets can be constructed by hand and can be verified with simple geometric arguments.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Art Gallery Problem, Complexity, ETR, Polygon

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.74

Related Version *Full Version*: `https://arxiv.org/abs/2210.12817`

Funding Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.

1 Introduction

1.1 Art gallery problem

The original form of the art gallery problem (AGP) presented by Victor Klee (see O'Rourke [11]) asks whether a simple polygonal region P can be *guarded* by n guards. That is, whether there is a set of n points (called the *guards*) in P such that every point in P is *visible* to some guard, meaning that the the line segment between that point and that guard is contained in P. The polygon P is referred to as the *art gallery*.

The vertices of P are usually restricted to rational or integer coordinates, but even so an optimal configuration might require guards with irrational coordinates. Abrahamsen, Adamaszek and Miltzow give explicit examples that require irrational coordinates in [1]. For this reason, we don't expect algorithms to actually output the guarding configurations, only to determine how many guards are necessary.

It is non-trivial to see that the problem is even decidable. The first exact algorithm for the most general variants of the AGP is attributed to Sharir (see Efrat and Har-Peled [7]).

1.2 The complexity class $\exists\mathbb{R}$

The decision problem ETR (Existential Theory of the Reals) asks whether a sentence of form:

$$\exists X_1 \dots \exists X_n \Phi(X_1, \dots, X_n)$$

© Jack Stade;
licensed under Creative Commons License CC-BY 4.0

is true, where the X_i are real variables and Φ is a formula in the X_i involving 0, 1, $+$, $-$, $\cdot$, $=$, $<$, $\leq$, $\neg$, $\wedge$, and $\vee$. The complexity class $\exists\mathbb{R}$ consists of problems that can be reduced to ETR in polynomial time. A number of interesting problems have been shown to be complete for $\exists\mathbb{R}$, including for example packing polygons in a square [3] and the problem of deciding whether there exists a point configuration with a given order type [10, 14].

By a result of Schaefer and Stefankovic [12], the exact inequalities used don't matter; for example we obtain an equivalent definition of $\exists\mathbb{R}$ if we don't allow $=$, $\leq$, or $\neg$ in Φ. The same authors have more recently shown that similar results hold at every level of a *real hierarchy*, of which $\exists\mathbb{R}$ is the first level [13].

It is straightforward to show that NP $\subseteq \exists\mathbb{R}$. It is also known, though considerably more difficult to prove, that $\exists\mathbb{R} \subseteq$ PSPACE (Canny [6]). It is unknown whether either inclusion is strict.

1.3 Art gallery variants

A natural class of variants of the art-gallery problem is given by restricting both the positions that can be occupied by the guards and the points that need to be guarded. We will adopt the notation used in [4]:

▶ **Definition 1.** *(Agrawal, Knudsen, Lokshtanov, Saurabh, and Zehavi [4]) The* X-Y Art Gallery problem*, where $X, Y \in \{Vertex, Point, Boundary\}$, asks whether the polygon P can be guarded with n guards, where if $X = Vertex$ the guards are restricted to lie on the vertices of the polygons, if $X = Boundary$ the guards are restricted to lie on the boundary of the polygon, and if $X = Point$ then the guards can be anywhere inside the polygon. The region that must be guarded is determined by Y analogously.*

Table 1 lists these variants and the known bounds on complexity.

Table 1 Variants of the art gallery problem.

Variant	Complexity Lower Bound	Complexity Upper Bound
Vertex-Y	NP[9]	NP
X-Vertex	NP[9]	NP
Point-Point	∃ℝ[2]	∃ℝ[2]
Boundary-Point	∃ℝ[2]	∃ℝ[2]
Point-Boundary	∃ℝ**[this paper]**	∃ℝ[2]
Boundary-Boundary	NP[9]	∃ℝ[2]

If X or Y is *Vertex*, then the problem is easily seen to be in NP. Lee and Lin [9] showed that all of these variants are NP-hard (the result is stated for all the X-Point variants, but their constructions also work for the other variants. See also [8]). More recently, Abrahamsen, Adamaszek, and Miltzow [2] showed that the point-point and boundary-point variants are $\exists\mathbb{R}$ complete. It is straightforward to extend their proof of membership in $\exists\mathbb{R}$ to any of these variants, but they list $\exists\mathbb{R}$ hardness of the point-boundary variant an open problem.

1.4 Our results

Our main result is that the point-boundary variant of the art gallery problem is, up to polynomial time reductions, as hard as deciding whether a system of polynomial equations has a real solution.

▶ **Theorem 2.** *The point-boundary variant of the art gallery problem is $\exists\mathbb{R}$-complete.*

While the complexity of the boundary-boundary variant is still unsolved, this is is enough to show that the the X-Y art gallery problem is equivalent to the Y-X art gallery problem for $X, Y \in \{\text{Vertex}, \text{Point}, \text{Boundary}\}$.

Our ideas can also be used to considerably simplify the construction in [2]. Both the construction and verification of each of our gadgets is simpler than that of the corresponding gadget in [2]. Indeed, our gadgets can be drawn by hand with little more than compass-and-straightedge-style constructions. Each gadget is specified by a simple set of geometric properties rather than by exact coordinates. Our gadgets are quite flexible and the geometry could probably be adapted to other settings.

Our reduction is from a problem called ETR-INV-REV, which is a slight modification of a problem ETR-INV from [2]. In Section 2, we define this problem and show that it is $\exists\mathbb{R}$-hard.

Given an instance Φ of ETR-INV-REV, we show how to construct a polygon whose boundary can be guarded by some number n guards if and only if Φ has a satisfying assignment. In Section 3, we describe the overall structure of the polygon that we construct and explain how to constrain the positions of guards. In Section 4, we describe copy gadgets that are required in order to use a single variable in multiple constraints. Finally, in Section 5 we describe the gadgets that create constraints and prove Theorem 2.

Throughout, statements marked with ($\star$) have proofs omitted. The proofs can be found in the full version.

2 ETR-INV-REV

The proof of Theorem 2 is by reduction of the problem we call ETR-INV-REV to the point-boundary variant of the art gallery problem.

▶ **Definition 3.** *(ETR-INV-REV) In the problem ETR-INV-REV, we are given a set of real variables $\{x_1, \dots, x_n\}$ and a set of inequalities of the form:*

$$x = 1, \quad xy \geq 1, \quad x\left(\frac{5}{2} - y\right) \leq 1, \quad x + y \leq z, \quad x + y \geq z,$$

for $x, y, z \in \{x_1, \dots, x_n\}$. The problem asks whether there is an assignment of the x_i satisfying these inequalities with each $x_i \in [\frac{1}{2}, 2]$.

Abrahamsen, Adamaszek and Miltzow [2] proved the $\exists\mathbb{R}$-hardness of the point-point and boundary-point variants using a similar problem called ETR-INV.

▶ **Definition 4.** *(Abrahamsen, Adamaszek and Miltzow [2]) (ETR-INV) In the problem ETR-INV, we are given a set of real variables $\{x_1, \dots, x_n\}$ and a set of equations of the form:*

$$x = 1, \quad xy = 1, \quad x + y = z,$$

for $x, y, z \in \{x_1, \dots, x_n\}$. The problem asks whether there is a solution to the system of equations with each $x_i \in [\frac{1}{2}, 2]$.

▶ **Theorem 5.** *(Abrahamsen, Adamaszek, Miltzow [2]) The problem ETR-INV is $\exists\mathbb{R}$-complete.*

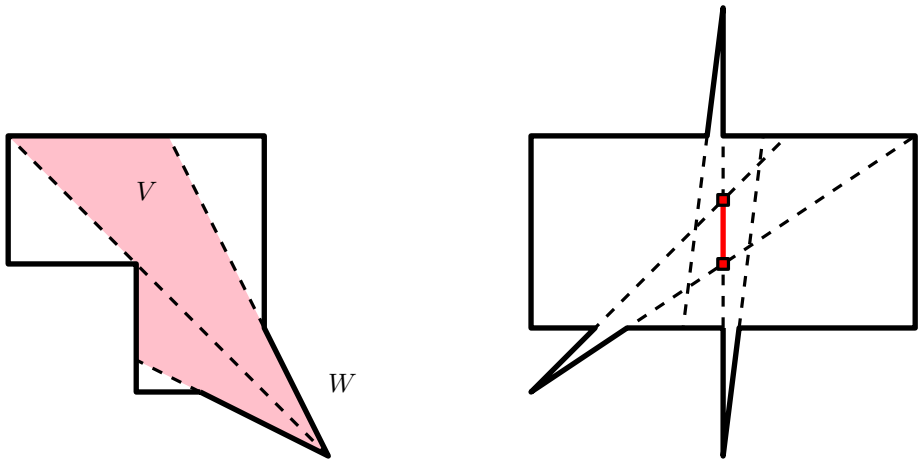


Figure 1 Left: A wedge (W) and its visibility region (V). Right: The intersection of the visibility regions of the three wedges shown forms a guard segment. This method of using 3 wedges to form a guard segment is due to Bertschinger, El Maalouly, Miltzow, Schnider and Weber [5] and is a slight simplification of the method for forming guard segments from [2].

Regardless of the specific art gallery variant, it seems to be difficult to make a construction that admits both $xy \leq 1$ and $xy \geq 1$ constraints. The $xy \leq 1$ inversion gadget in [2] is actually closer to a $x\left(\frac{5}{2} - y\right) \leq 1$ gadget, and another gadget is used to compute a constraint like $y' = \left(\frac{5}{2} - y\right)$. By reducing from ETR-INV-REV instead, our construction avoids the need for a reversing gadget.

▶ **Theorem 6.** ($\star$) *The problem ETR-INV-REV is ∃ℝ-hard.*

3 Arranging the art gallery

Throughout this section, and the rest of the paper, AB refers to a line segment with endpoints A and B, $\overleftrightarrow{AB}$ is the line containing that segment, and $|AB|$ is the length of that segment.

3.1 Creating guard regions

We will designate some number n of disjoint *guard regions* inside the art gallery such that any guarding configuration with n guards must have exactly one guard in each region. Each guard region is determined by *wedges* on the polygon boundary. Each wedge is formed by two edges of the art gallery meeting at a convex corner. The *visibility region* of a wedge is the set of points that can see the tip of the wedge. A wedge and its visibility region is shown in Figure 1 (left).

In a guarding configuration, the visibility region of each wedge must contain a guard. We designate a guard region by specifying some (non-zero) number of wedges, and the guard region is the intersection of the visibility regions of those wedges. These must be chosen that visibility regions of any two wedges *corresponding to different guard regions* do not intersect. A guard region shaped like a line segment is called a *guard segment*, as in Figure 1 (right).

▶ **Lemma 7.** *If we designate n guard regions, then any guarding configuration has at least n guards, and a guarding configuration with exactly n guards has 1 guard in each guard region.*

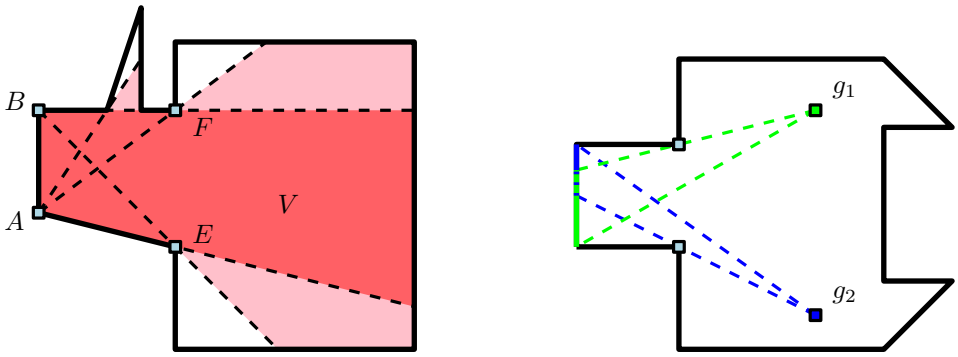


Figure 2 Left: A nook with nook segment AB, opening points E restricting A and F restricting B, and partial visibility region V. The outer parts of V (lighter) can only see part of the nook segment. Right: This art gallery can be guarded by two guards, but there is no such configuration where the nook segment on the left is guarded entirely by a single guard. This creates a continuous constraint between the positions of the two guards.

Proof. Choose wedges $w_1, \ldots, w_n$, one from each of the guard regions. The guard regions are chosen so that the visibility regions of these wedges do not intersect. In a guarding configuration, each of these regions contains at least 1 guard. Since these region do not intersect, this requires at least n guards.

If a guarding configuration contains exactly n guards, then there will be exactly 1 guard in the visibility region of w_i for each i. Consider the guard region containing w_j and suppose v is another wedge forming that guard region. There is a guard in the visibility region of v, and this region doesn't intersect the visibility region of w_i for $i \neq j$, so this guard must be the one in w_j. Repeating this argument for the other wedges forming this guard region, we see that the guard in w_j must be in each of them. So the guard is in the guard region.

So a guarding configuration with exactly n guards has 1 guard in each guard region. ◀

The guard segments in our construction will all be vertical, and the position of a guard on a guard segment will encode a variable in the interval $[\frac{1}{2}, 2]$, with the bottom endpoint of the interval corresponding to $\frac{1}{2}$ and the top endpoint corresponding to 2.

3.2 Constraint nooks

We use constraint *nooks* to enforce constraints involving multiple variables. Each nook consists of a line segment on the boundary of the art gallery, called the *nook segment*, and two polygon vertices called *opening points*. The *region of partial visibility* of a nook is the set of points which can see some part of the nook segment.

If the nook segment has endpoints A and B and the opening points are E and F where E, A, B, F occur in that order on the polygon boundary, then we say that E *restricts* A and F *restricts* B (see Figure 2).

These are slightly more general than the nooks from [2], in that it will sometimes be necessary to intersect a wedge with a nook, as in Figure 2 (left).

A guarding configuration will have some non-zero number of guards in the partial visibility region of a nook, that together must guard the nook segment, as in Figure 2 (right). Multiple guards can collaborate to guard a single nook segment, creating a continuous dependence between the positions of those guards.

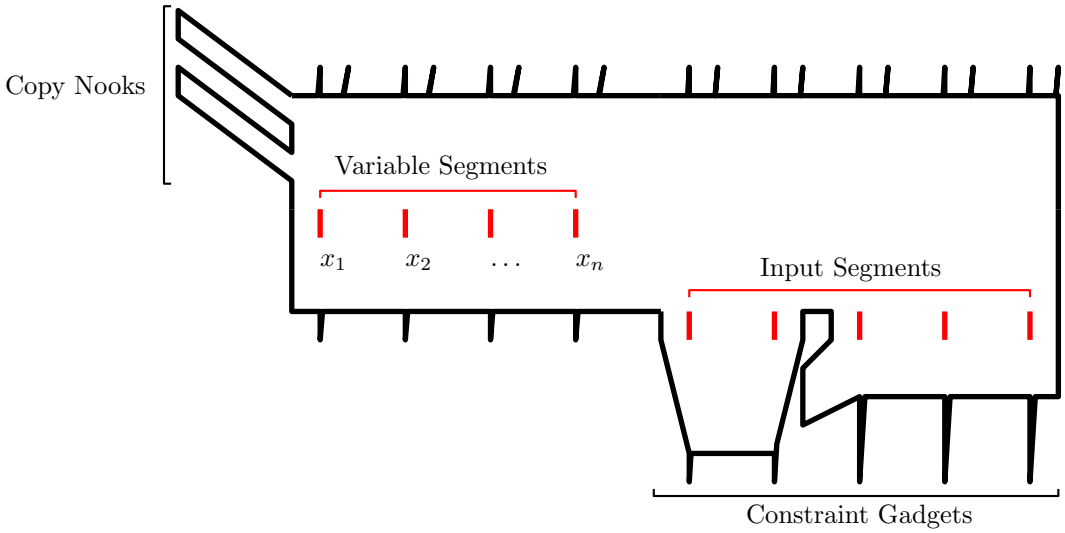


Figure 3 A schematic of the art gallery that we construct (not to scale). The variables are encoded by positions of guards on the variable segments (red, middle). The constraint gadgets on the bottom right enforce constraints like $xy \geq 1$ or $x + y \leq z$ on the variables encoded by the input segments (red, bottom right). The copy nooks copy the values of the appropriate variables onto the input segments. The wedges creating the segment-shaped guard regions can be seen along the top and bottom walls of the art gallery.

3.3 Schematic of the art gallery

Figure 3 shows a schematic of the entire art gallery construction. We start with an instance Φ of ETR-INV-REV with variables $x_1, \ldots, x_n$ and constraints of form:

$$x = 1, \quad xy \geq 1, \quad x\left(\frac{5}{2} - y\right) \leq 1, \quad x + y \leq z, \quad x + y \geq z,$$

Each variable is represented by a guard region called a *variable segment*. The variable segments are arranged in a row near the middle of the art gallery. The height of a guard on a variable segment corresponds linearly to the value of the variable encoded, with the bottom of the variable segment mapping to $\frac{1}{2}$ and the top mapping to 2.

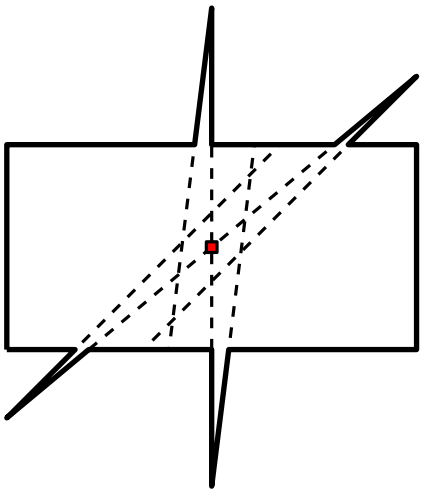


Figure 4 Four wedges create a point-shaped guard region.

The constraints of form $x = 1$ in Φ are created by adding wedges to create a guard region consisting of only a single point, as shown in Figure 4. Each constraint involving more than one variable is created by a *constraint gadget*, which we place in a row on the bottom right of the art gallery. The constraint gadgets are modular in the sense that they (almost) do not depend on Φ. Each constraint gadget operates on either 2 or 3 guard segments called *input segments*. Nooks in the constraint gadget create constraints on the variables represented by the guards on the input segments.

Each input segment can only appear in one constraint gadget, so we need *copy nooks* to relate each input segment to the variable segment for the variable that it represents.

4 Construction of the copy nooks

If a guard on a variable segment encodes a value x and a guard on an input segment encodes a value y, then a copy nook creates a constraint like $x \leq y$ or $x \geq y$. Figure 5 shows an example of a copy nook enforcing a constraint like $x \geq y$. This type of copy nook is due to Stade and Tucker-Foltz [15].

4.1 Verifying a single copy nook

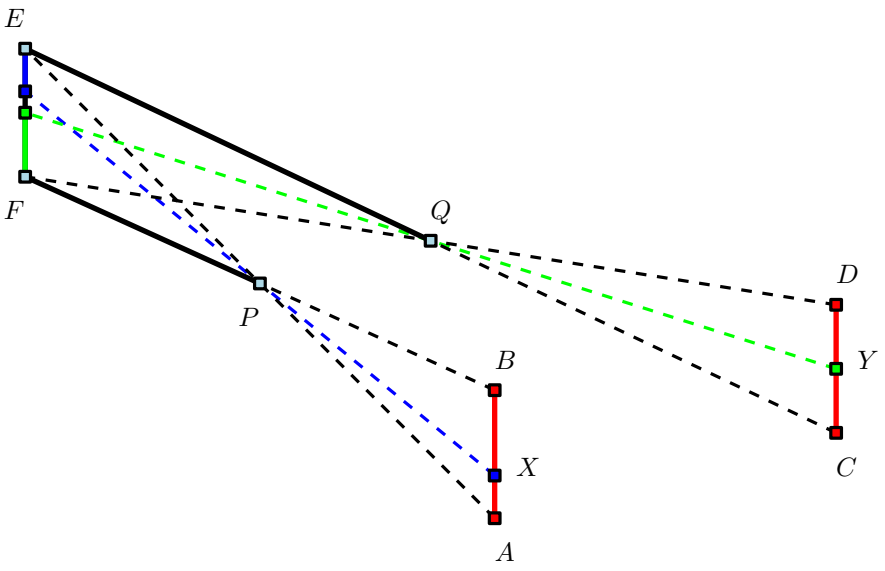


Figure 5 A copy nook, shown in the situation where the guard positions don't fully guard the nook.

▶ **Lemma 8.** *Suppose segments AB and CD are such that $\overleftrightarrow{AB}$ and $\overleftrightarrow{CD}$ are parallel, and suppose $\overleftrightarrow{AC}$ and $\overleftrightarrow{BD}$ intersect at a point P, as in Figure 6. If a line through P intersects AB at a point X and intersects CD at a point Y, then $\frac{|AX|}{|AB|} = \frac{|CY|}{|CD|}$.*

Proof. Triangles APB and CPD are similar, so $\frac{|AP|}{|AB|} = \frac{|CP|}{|CD|}$. Also, the triangles AXP and CYP are similar, so $\frac{|AX|}{|AP|} = \frac{|CY|}{|CP|}$. Multiplying, we obtain $\frac{|AX|}{|AB|} = \frac{|CY|}{|CD|}$. ◀

▶ **Lemma 9.** *Suppose that AB and CD are parallel guard segments and there is a nook with nook segment EF that is parallel to AB and CD. Suppose the nook has openings points P on the intersection point of $\overleftrightarrow{AE}$ and $\overleftrightarrow{BF}$ and Q on the intersection of $\overleftrightarrow{BE}$ and $\overleftrightarrow{CF}$ so that P restricts F and Q restricts G.*

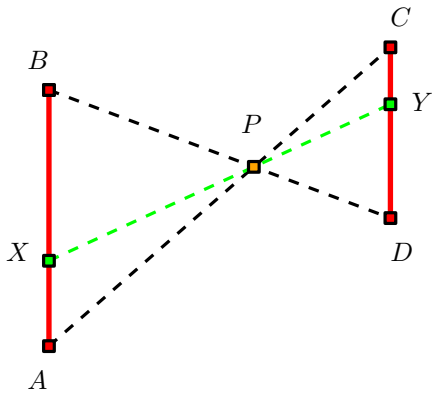


Figure 6 Setup for Lemma 8.

Suppose also that the nook is unobstructed, that is that every segment through P from a point on AB to a point on EF is contained in the art gallery and segments through Q are similarly unobstructed.

Then guards at positions X and Y on AB and CD respectively guard the nook if and only if:

$$\frac{|AX|}{|AB|} \geq \frac{|CY|}{|CD|}$$

The setup for Lemma 9 is shown in Figure 5. The proof is a straightforward application of Lemma 8.

By changing which intersection point restricts which endpoint of the nook segment, we can create a constraint either $x \leq y$ or $x \geq y$. Figure 7 shows a pair of guard segments with both types of copy nook between them.

Enforcing a constraint $x = y$ requires 2 nooks, but all of the constraints allowed by ETR-INV-REV are monotone with respect to any single variable, so we only need a single copy nook for each input segment. For example, $xy \geq 1$ is equivalent to $x'y' \geq 1$, $x \geq x'$ and $y \geq y'$ (when there are no other constraints on x' and y').

In the full version, we show that it is possible to arrange all the copy nooks in such a way that none of them interfere with each other. This construction ensures that the partial visibility region of each copy nook is bounded by lines with slope between $-\frac{1}{2}$ and -1, a fact that will be useful in order to make sure that the constraint gadgets don't interfere with the copy nooks.

5 The constraint gadgets

We start with some specifications that a constraint gadget should adhere to in order to be compatible with the copy nooks. There are several bad cases that we need to prevent:

- The constraint gadget obstructs the visibility between an input segment and a copy nook.
- A guard on a variable segment can see part of a nook segment in the constraint gadget.
- An guard used by the gadget (called an *auxiliary guard*) can see part of the nook segment of a copy nook.

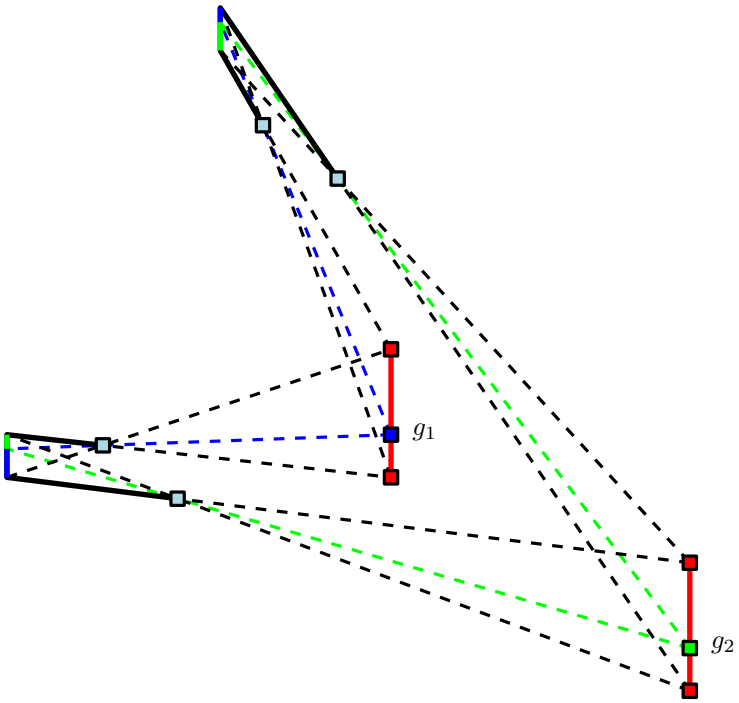


Figure 7 A guard (g_1) on a variable segment and a guard (g_2) on an input segment are related by two copy nooks. The upper nook is guarded when the value encoded by g_1 is at least the value encoded by g_2, and the lower nook is guarded when the value encoded by g_1 is at most the value encoded by g_2.

For each constraint gadget, we designate R_1 and R_2, shown in Figure 8. The region R_1 is bounded by the convex hull of the input segments to that gadget and lines of slope $-\frac{1}{2}$ and -1 through the bottom of the leftmost input segment and the top of the rightmost input segment respectively. R_2 is the set of points that can be connected to a point outside of the constraint gadget by a line segment that is contained in the polygon and has slope between -1 and 0.

▶ **Lemma 10.** *The region R_1 contains all possible sight lines between an input segment and the nook segment of one of the copy nooks, so if the constraint gadget does not have any walls that intersect this region then it will not obstruct the copy gadgets.*

Proof. The construction in Section 4 ensures that all sight lines between an input segment and a nook segment have slope between -1 and $-\frac{1}{2}$, and so are contained in the region R_1. ◀

▶ **Lemma 11.** *The only points in a constraint gadget that can be seen by a variable segment are in R_2. So any nook with a nook segment that doesn't intersect R_2 must be guarded by guards on input segments or auxiliary guards in that gadget.*

Points in a constraint gadget outside of R_2 cannot see any part of the nook segment of any copy nook. So if any auxiliary guard regions don't overlap R_2, then a guard in that region can't help guard any of the copy nooks.

Proof. The construction of the copy nooks ensures that any line between a variable segment and a point at the top of a constraint gadget has slope between -1 and 0. Any sight line between a point on a variable segment and a point in the constraint gadget must pass through the top, so any such line has slope between -1 and 0. So R_2 contains all points in the constraint gadget that can be seen by a point on a variable segment.

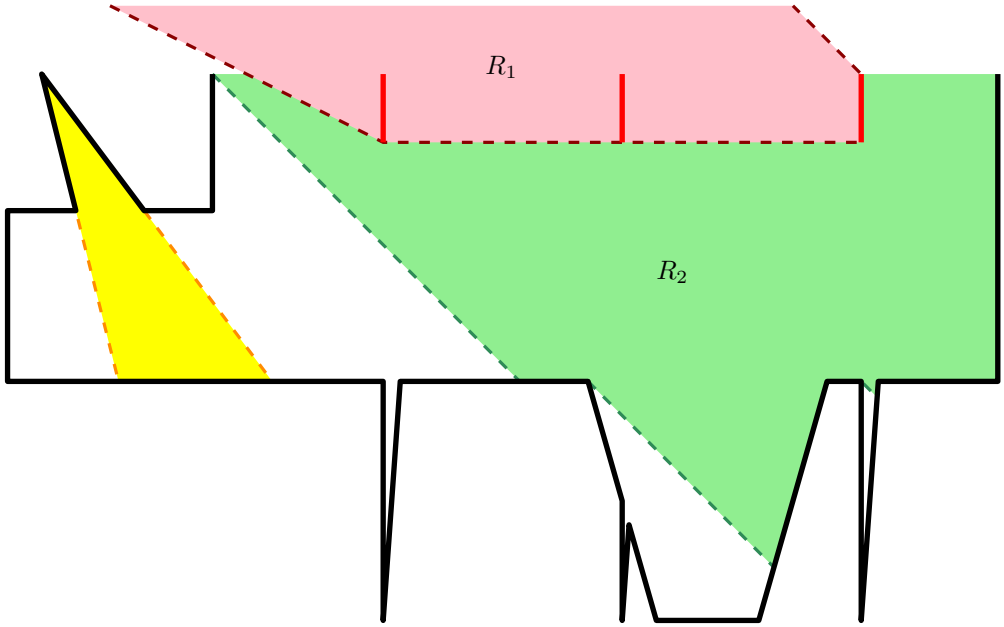


Figure 8 Diagram of a constraint gadget. The region R_1 is bounded by lines with slope -1 and $-\frac{1}{2}$. The region R_2 is bounded by lines with slope -1 and the polygon wall. The yellow region is a an auxiliary guard region, which must not intersect R_2.

The construction of the copy nooks also ensures that the partial visibility region of each copy nook is bounded by lines with slope between -1 and $-\frac{1}{2}$. So the only points in a constraint gadget that can see part of a nook segment of a copy nook are in R_2. ◀

5.1 Inversion gadgets

Each inversion gadget consists of a single constraint nook interacting with the two input variables. Figures 9 and 10 show the $xy \geq 1$ and $x\left(\frac{5}{2} - y\right) \leq 1$ inversion gadgets respectively. Figures 11 and 12 show the geometry of these gadgets in more detail.

In either gadget, we suppose that there is a guard at X on the input segment GH. The nook has opening points P and Q, so the guard at X can see all the points to the left of point Y on the nook segment IJ. For the $xy \geq 1$ gadget, this means that a guard on AB see the rest of IJ if and only if it is above the point W. For the $x\left(\frac{5}{2} - y\right) \leq 1$ gadget, a guard on AB should be below W instead.

In both cases, the guard on AB represents the input x and the guard on GH represents the input y. The segments represent the intervals $[\frac{1}{2}, 2]$, with lower points on the segments corresponding to lower values. So we should show that:

$$\left(\frac{3}{2} \cdot \frac{|AW|}{|AB|} + \frac{1}{2}\right)\left(\frac{3}{2} \cdot \frac{|GX|}{|GH|} + \frac{1}{2}\right) = 1$$

for any X on GH and W on AB as shown.

▶ **Lemma 12.** *If line segments AB and CD are not parallel, as in Figure 13, let P be the intersection of $\overleftrightarrow{AD}$ and $\overleftrightarrow{BC}$. Let E be the point on $\overleftrightarrow{AB}$ such that $\overleftrightarrow{PE}$ and $\overleftrightarrow{CD}$ are parallel, and let F be the point on $\overleftrightarrow{CD}$ such that $\overleftrightarrow{PF}$ and $\overleftrightarrow{AB}$ are parallel. Suppose that a point Y on CD and a point Z on AB are such that P, Y and Z are collinear. Then $\frac{|EA|}{|EB|} = \frac{|FC|}{|FD|}$, and letting $\alpha^2 = |EA||EB|$ and $\beta^2 = |FC||FD|$ we have $\frac{|EZ|}{\alpha} \cdot \frac{|FY|}{\beta} = 1$.*

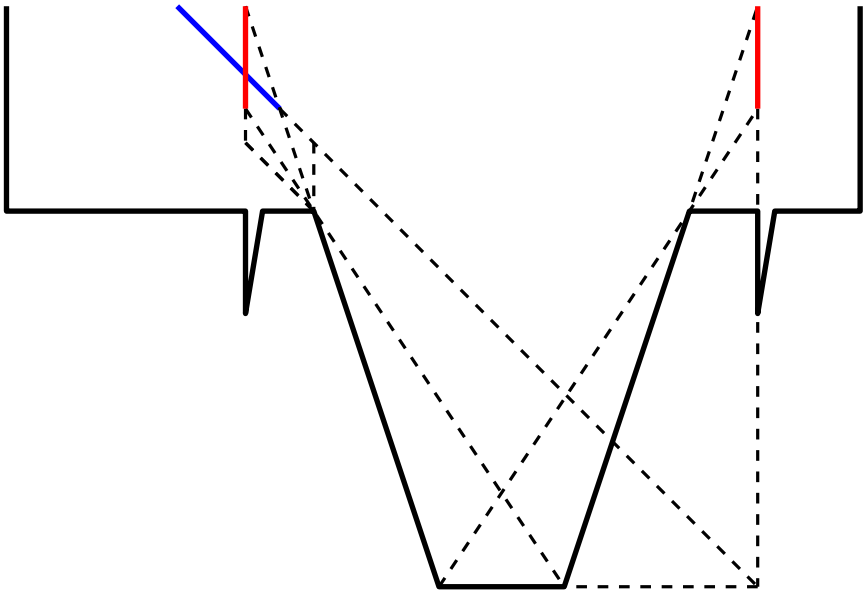


Figure 9 The $xy \geq 1$ inversion gadget. The two red segments are the input segments, with the left segment representing the variable x and the right segment representing y. Both guards see more of the nook segment the higher they are. The blue segment is a "phantom" segment that is used to help verify the gadget.

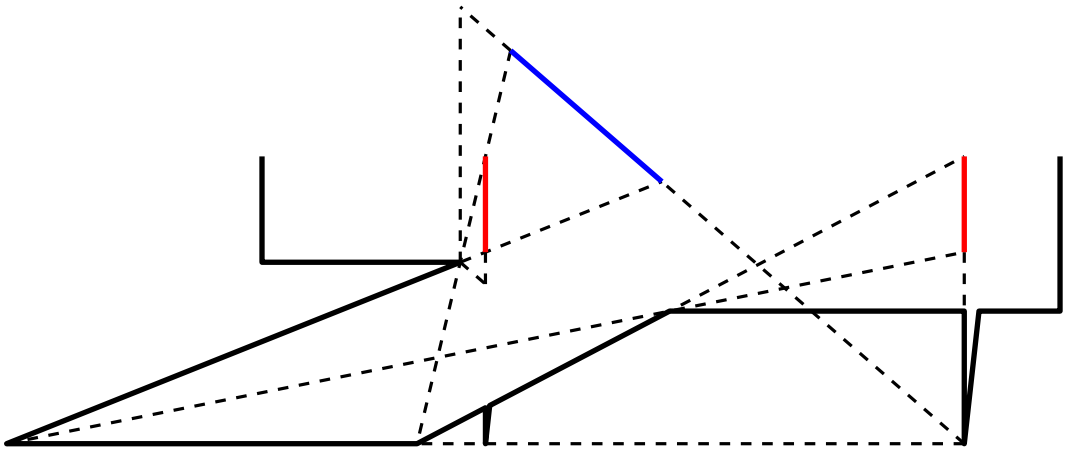


Figure 10 The $x\left(\frac{5}{2} - y\right) \leq 1$ inversion gadget. Compared to the $xy \geq 1$ gadget, the guard on the left segment (representing x) now sees more of the nook segment when it is lower.

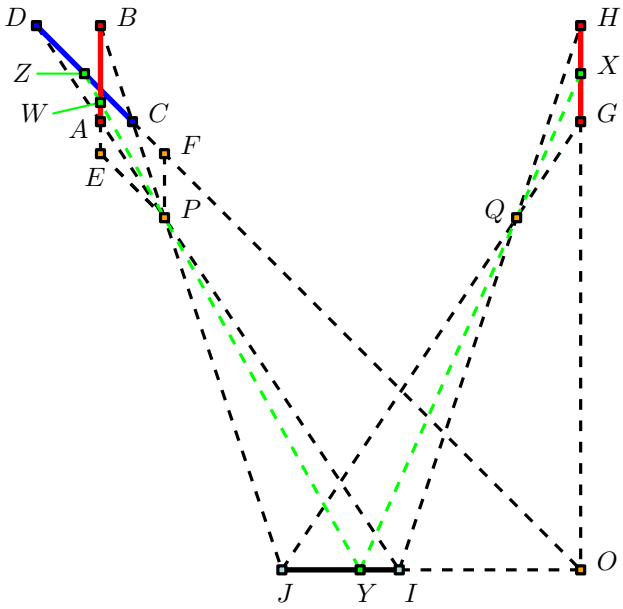


Figure 11 Labeled diagram of the $xy \geq 1$ inversion gadget shown in Figure 10.

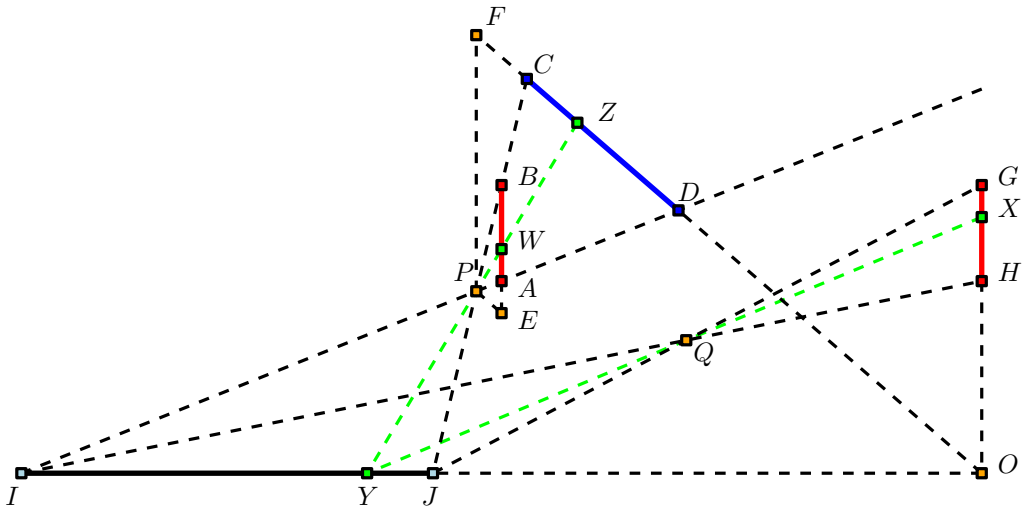


Figure 12 Labeled diagram of the $x\left(\frac{5}{2} - y\right) \leq 1$ inversion gadget shown in Figure 9.

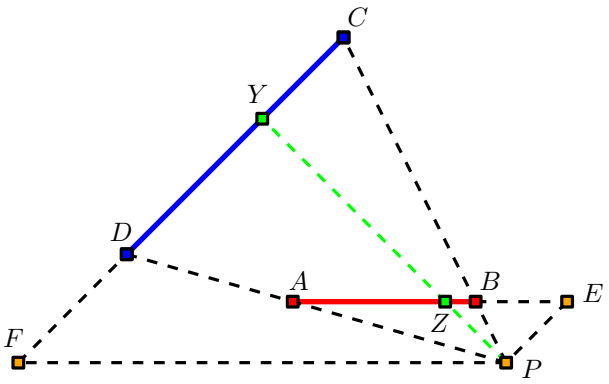


Figure 13 As long as the points P, Y, and Z are colinear, then the value of $|EZ||FY|$ does not depend on the positions of Y and Z, so the positions of Z and Y are related by inversion. The curved relationship occurs because AB and CD are not parallel.

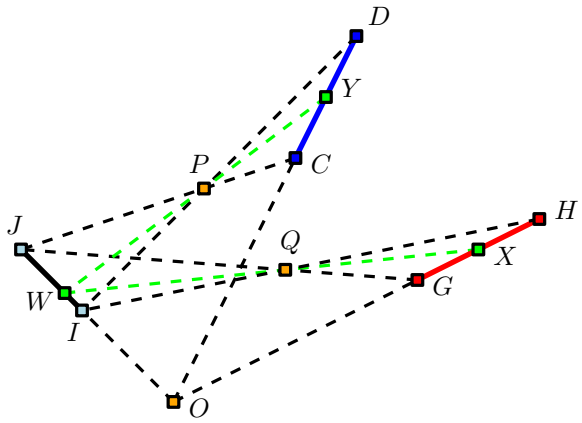


Figure 14 If X is on GH, then define Y by projecting X onto IJ through Q and then projecting that point onto CD through P. The relative position of X on GH is the same as the relative position of Y on CD.

Proof. The triangles EPX and FYP are similar, so $\frac{|EZ|}{|EP|} = \frac{|FP|}{|FY|}$, so $|EZ||FY| = |FP||EP|$. In particular, when $Z = A$, we have $|EA||FD| = |FP||EP|$, and when $Z = B$, we have $|EB||FC| = |FP||EP|$, so $|EA||FD| = |EB||FC|$ and $\frac{|EA|}{|EB|} = \frac{|FC|}{|FD|}$.

Now $|EZ||FY| = |FP||EP| = |EA||FD| = |EB||FC|$, so:

$$|EZ||FY| = \sqrt{|EA||FD||EB||FC|} = \sqrt{\alpha^2\beta^2} = \alpha\beta$$ ◀

The formulas $\frac{|EZ|}{\alpha}$ and $\frac{|EY|}{\beta}$ give mappings from points on the input segments to $\mathbb{R}$. We want the input segment to correspond to the intervals $[\frac{1}{2}, 2]$, so we choose $|EB| = 4|EA|$ (and therefore $|FD| = 4|FC|$), so $\alpha = 2|EA| = \frac{1}{2}|EB|$ and $\beta = 2|FC| = \frac{1}{2}|FD|$. This means that $\frac{|EZ|}{\alpha}$ and $\frac{|FY|}{\beta}$ will map the segments AB and CD respectively onto $[\frac{1}{2}, 2]$.

On its own, Lemma 12 could be used to construct a nook which enforces an inversion constraint on two input segments which aren't parallel. The input segments to a constraint gadget should be parallel, so we will need the result of Lemma 13 to correct the orientation.

▶ **Lemma 13.** *Suppose line segments GH, CD, and IJ are such that $\overleftrightarrow{GH}$, $\overleftrightarrow{CD}$, and $\overleftrightarrow{IJ}$ all intersect at a point O, as in Figure 14. Also suppose that the ratios $\frac{|OG|}{|OH|}$ and $\frac{|OC|}{|OD|}$ are the same. Let P be the point where ID and JC intersect, and Q be the point where IH and JG intersect. Suppose that points X on GH, Y on CD and W on IJ are such that W, P, and Y are collinear and W, Q and X are collinear. Then $\frac{|GX|}{|GH|} = \frac{|CY|}{|CD|}$.*

Proof. Place the figure in the vector space $\mathbb{R}^2$ with the point O at $(0, 0)$. We will show that there is a linear map which sends points G, X and H to C, Y and D respectively.

The pairs of vectors $\{I, G\}$ and $\{I, C\}$ are each bases for $\mathbb{R}^2$. Let θ be the linear isomorphism $\mathbb{R}^2 \to \mathbb{R}^2$ which sends a vector V to the coefficients (t, s) such $V = tI + sG$, and let ψ be a similar map which writes V as $tI + sC$. Now the linear map $\psi^{-1} \circ \theta$ fixes points on the line containing I and J, and sends G to C. Since $\frac{|OA|}{|OB|} = \frac{|OC|}{|OD|}$, this map sends H to D, and so sends Q to P. The point W is fixed, so the line $\overleftrightarrow{WQ}$ is sent to $\overleftrightarrow{WP}$, meaning that X is sent to Y. So points G, X and H are sent to C, Y and D respectively. Since linear maps preserve ratios of distances along a line, we see that:

$$\frac{|GX|}{|GH|} = \frac{|CY|}{|CD|}$$

◀

Note that the mappings from GH to IJ and from IJ to CD are in general *not* linear, only the composition is.

▶ **Lemma 14.** *In either the $xy \geq 1$ or $x\left(\frac{5}{2} - y\right) \leq 1$ gadget, suppose there are points C on $\overleftrightarrow{JP}$ and D on $\overleftrightarrow{IP}$ such that $\overleftrightarrow{GH}$, $\overleftrightarrow{CD}$ and $\overleftrightarrow{IJ}$ all intersect at a point O, with C and D satisfying:*

$$\frac{|OG|}{|OH|} = \frac{|OC|}{|OD|}$$

Suppose also that there are points E on $\overleftrightarrow{AB}$ and F on $\overleftrightarrow{CD}$ so that A is between E and B, C is between F and D, $\overleftrightarrow{PF}$ and $\overleftrightarrow{AB}$ are parallel and $\overleftrightarrow{PE}$ and $\overleftrightarrow{CD}$ are parallel, with $|FD| = 4|FC|$ and $|EB| = 4|EA|$. Then:

$$\left(\frac{3}{2} \cdot \frac{|AW|}{|AB|} + \frac{1}{2}\right)\left(\frac{3}{2} \cdot \frac{|GX|}{|GH|} + \frac{1}{2}\right) = 1$$

so the two inversion gadgets enforce the appropriate constraints.

Proof. Examples of points C, D, E and F with these properties are shown in Figures 11 and 12. Let Z be the intersection of $\overleftrightarrow{YP}$ with CD. By Lemma 13:

$$\frac{|GX|}{|GH|} = \frac{|CZ|}{CD}$$

By Lemma 12, we have that:

$$\frac{|EW|}{2|EA|} \cdot \frac{|FZ|}{2|FC|} = 1$$

Since $4|EA| = |EB| = \frac{4}{3}|AB|$, we have that:

$$\left(\frac{3}{2} \cdot \frac{|AW|}{|AB|} + \frac{1}{2}\right) = \left(\frac{|AW|}{2|EA|} + \frac{|EA|}{2|EA|}\right) = \frac{|EW|}{2|EA|}$$

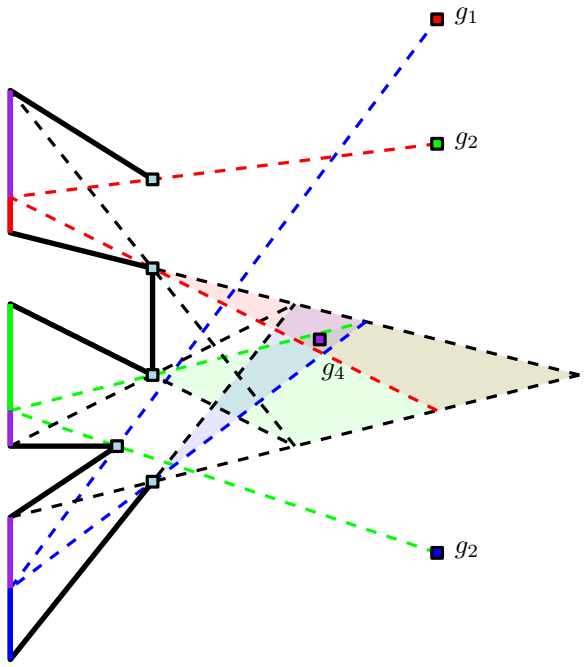


Figure 15 Creating a constraint between 3 variables. The guard g_1, g_2 and g_3 each see part of one of the nook segments. If the three shaded regions intersect, then a guard g_4 can be placed in the intersection so that g_4 guards the parts of the nook segments that aren't guarded by g_1, g_2 and g_3. If the three shaded regions do not intersect, then this isn't possible.

Similarly:

$$\left(\frac{3}{2}\cdot\frac{|GX|}{|GH|}+\frac{1}{2}\right)=\left(\frac{3}{2}\cdot\frac{|CZ|}{|CD|}+\frac{1}{2}\right)=\frac{|FZ|}{2|FC|}$$

So:

$$\left(\frac{3}{2}\cdot\frac{|AW|}{|AB|}+\frac{1}{2}\right)\left(\frac{3}{2}\cdot\frac{|GX|}{|GH|}+\frac{1}{2}\right)=1$$ ◀

It remains to show that gadgets satisfying the conditions of Lemma 14 can actually be created. The most direct approach would require solving a quadratic equation, potentially introducing irrational coordinates. The authors of [2] solved a similar issue with their inversion gadgets by carefully choosing coordinates that yield quadratic equations with rational solutions. However, our gadgets can be constructed more geometrically.

▶ **Lemma 15.** ($\star$) *Geometry for the $xy \geq 1$ and $x\left(\frac{5}{2}-y\right)\leq 1$ gadgets can be constructed in a way that satisfies the conditions of Lemmas 10, 11, and 14. This does not require irrational coordinates.*

5.2 Addition gadgets

Each nook creates a constraint between the positions of only 2 guards, but the addition gadgets should create a constraint involving 3 variables. This can be accomplished by allowing both coordinates of one guard to vary, as illustrated in Figure 15.

The addition gadgets are shown in Figures 16 and 17. Each gadget has 3 input segments, an auxiliary guard region, and three constraint nooks. Each constraint nook creates a constraint between the auxiliary guard and one input segment.

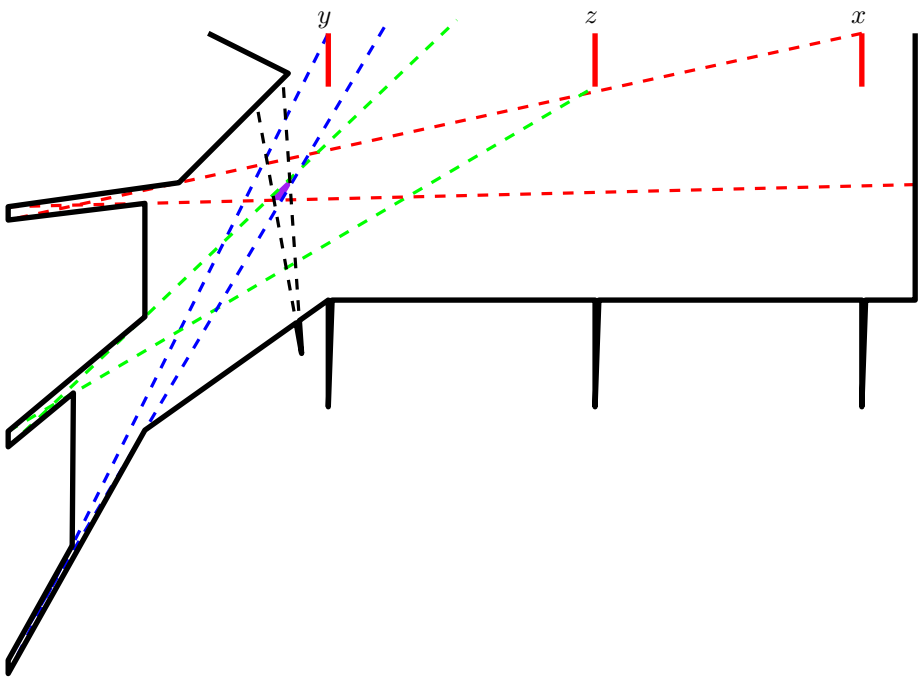


Figure 16 The $x + y \leq z$ addition gadget. The partial visibility regions of the 3 nooks are marked, as is the visibility region of the wedge forming the auxiliary guard region. The auxiliary guard will need to be somewhere in the purple shaded region.

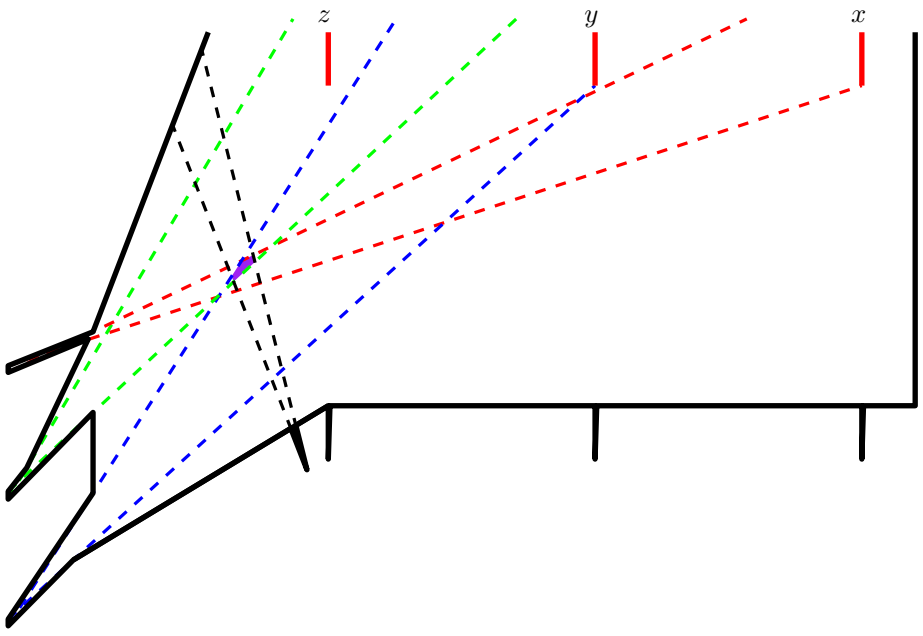


Figure 17 The $x + y \geq z$ addition gadget.

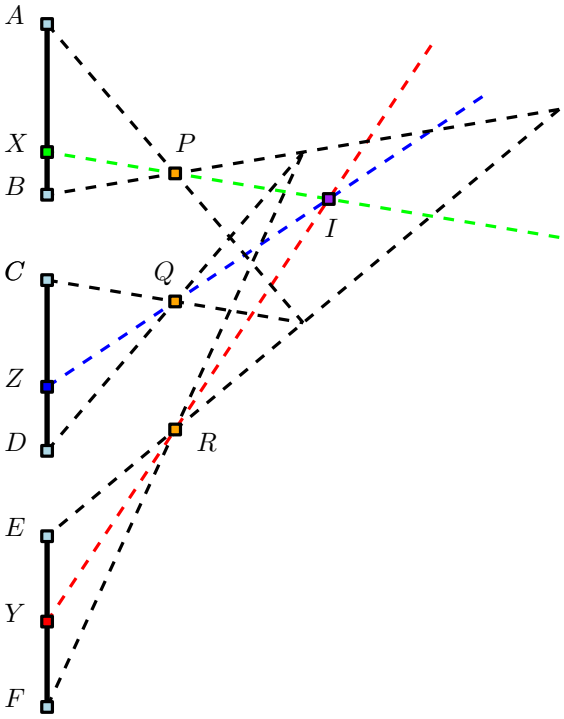


Figure 18 By Lemma 16, $\frac{1}{2}(|AX| + |EY|) = |CZ|$.

The gadget can be guarded when it is possible to place a guard in the auxiliary guard region so that it can see the parts of the nook segments that aren't visible to the input guards. We can use Lemma 8 to control the relationship between the positions of the guards on the input segments and their "shadows" on the nook segments. Lemma 16 allows us to determine what parts of these nook segments can be seen by an auxiliary guard.

▶ **Lemma 16.** *Suppose the line segments AB, CD, and EF have the same length and lie on the same vertical line, as in Figure 18, and suppose $|CB| = |DE|$. Let points P, Q, and R lie on a vertical line, with $|QP| = |QR|$. Note that $\overleftrightarrow{AP}$, $\overleftrightarrow{CQ}$, and $\overleftrightarrow{ER}$ intersect in a single point, and the same is true of $\overleftrightarrow{BP}$, $\overleftrightarrow{DQ}$, and $\overleftrightarrow{FR}$.*

Suppose points X and Y lie on AB and EF respectively. Now $\overleftrightarrow{XP}$ and $\overleftrightarrow{YQ}$ intersect at a point I. Let Z be the point on the intersection of CD and $\overleftrightarrow{IQ}$.

If all is as above, then $\frac{1}{2}(|AX| + |EY|) = |CZ|$.

Proof. The idea is to transform the geometry from Figure 18 to obtain something like Figure 19.

This transformation should send straight lines to straight lines and should fix the line containing points A, B, C, D, E, F, X, Y and Z. Additionally, the points P, Q and R should be sent to "infinity", lines through P should be sent to lines with slope -1, lines through Q should be sent to lines with slope 0, and lines through R should be sent to lines with slope $+1$. If such a transformation exists, then it is clear by straightforward linear algebra that $\frac{1}{2}(|AX| + |EY|) = |CZ|$.

Isomorphisms of the projective space $\mathbb{R}P^2$ send straight lines to straight lines. A degrees-of-freedom argument would be sufficient to find a transformation of projective space with the required properties. Instead we just give it explicitly. Assume that the line containing

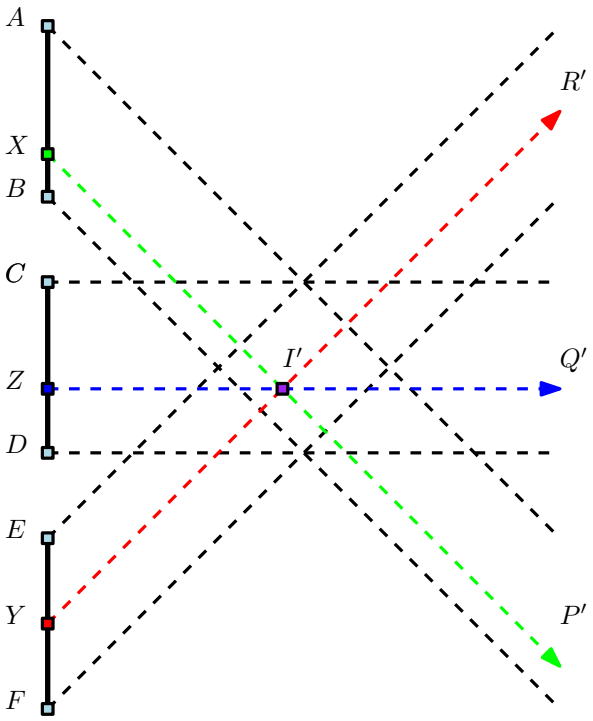


Figure 19 A projective transformation sends points P, Q, and R to points P', Q' and R' at infinity, so the family of lines through any one of these points is sent to a family of parallel lines.

A and B is vertical, and let x_0 be the x-coordinate of this line. Let x_1 be the x-coordinate of P, Q and R, y_0 the y-coordinate of Q, and let $a = |QP| = |QR|$, so P has y-coordinate $y_0 + a$ and R has y-coordinate $y_0 - a$. Then the transformation $(x, y) \to (x', y')$ with the desired properties is defined by:

$$\begin{bmatrix} x_0 + a & 0 & -(x_1 + a)x_0 \\ y_0 & x_0 - x_1 & -y_0 x_0 \\ 1 & 0 & -x_1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Writing the map in this form makes it easy to check what happens to lines through P, Q, and R. In particular, for a 3×3 matrix A, if:

$$A \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ 0 \end{bmatrix}$$

then the map $(x, y) \to (x', y')$ given by:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

sends lines through (p_x, p_y) to lines parallel to $\begin{bmatrix} a \\ b \end{bmatrix}$.

Note that the setup used here is very similar the addition gadgets in [2], although our verification is different. ◀

We want to create gadgets enforcing $x + y = z$, not $\frac{1}{2}(x + y) = z$, so we need to use Lemma 8 to change the scale and offset of z. A schematic of the nooks used by the addition gadgets are shown in Figures 20 and 21.

▶ **Lemma 17.** ($\star$) *It is possible to construct addition gadgets enforcing constraints $x + y \geq z$ or $x + y \leq z$.*

5.3 Final verification

▶ **Lemma 18.** *Given an instance Φ of ETR-INV-REV, it is possible to construct in polynomial time an art gallery P and a number n so that P can be guarded by n guards if and only if Φ is satisfiable.*

Proof. First, create corresponding constraint gadgets for every constraint of form $xy \leq 1$, $x\left(\frac{5}{2} - y\right) \leq 1$, $x + y \leq z$, or $x + y \geq z$ that occurs in Φ. Arrange these in a horizontal row along the bottom of the art gallery so that all the input segments occur in a horizontal row and the distance between any pair of input segments is a multiple of $d = 5$ times the length of a single input segment.

The construction in Section 4 can be used to place the copy nooks and variable segments. Now connect all the pieces of the art gallery and add the wedges forming all the guard segments. These wedges should be chosen to be sufficiently narrow so that visibility regions of wedges from different guard regions do not intersect.

It is straightforward to check that this construction can be done in polynomial time and that the coordinates needed are rational numbers with at most polynomially many bits.

There are n guard regions, consisting of the variable segments, input segments, and auxiliary guards. By Lemma 7, any guarding configuration with n guards must have one guard in each guard region. Conversely, any guarding configuration with one guard in each guard can clearly guard all of these wedges.

There are some walls of the art gallery that aren't part of the wedges, copy nooks, or constraint gadgets. The top, bottom, and right walls of the art gallery can be guarded by any guard on a variable segment. For any two copy nooks, a segment connecting can always be guarded by either of the guards on the variable segments in the visibility regions of those nooks. So any guarding configuration with one guard in each guard region will guard everything expect for possibly the copy nook and constraint gadgets.

By Lemmas 14 and 17, each constraint gadget can be guarded if and only if the guards on its input segments represent variables satisfying the corresponding constraint. Whenever a variable in Φ appears in a constraint, there are copy nooks between that variable segment and an input segment to the gadget representing that constraint. By Lemma 9, these copy nooks are guarded if and only if the guard on the input segment and the guard on the variable segment represent the value in $[\frac{1}{2}, 2]$.

So P can be guarded by n guards if and only if Φ has a satisfying assignment. ◀

The $\exists\mathbb{R}$-hardness of the point-boundary art-gallery problem follows from the $\exists\mathbb{R}$-hardness of ETR-INV-REV, proving Theorem 2.

6 Conclusions

Our result shows that the X-Y and Y-X variants of the art gallery problem are equivalent under polynomial time reductions when $X, Y \in \{\text{Vertex}, \text{Point}, \text{Boundary}\}$. Visibility is reflexive, so this result seems unsurprising. Maybe it could be proven more directly.

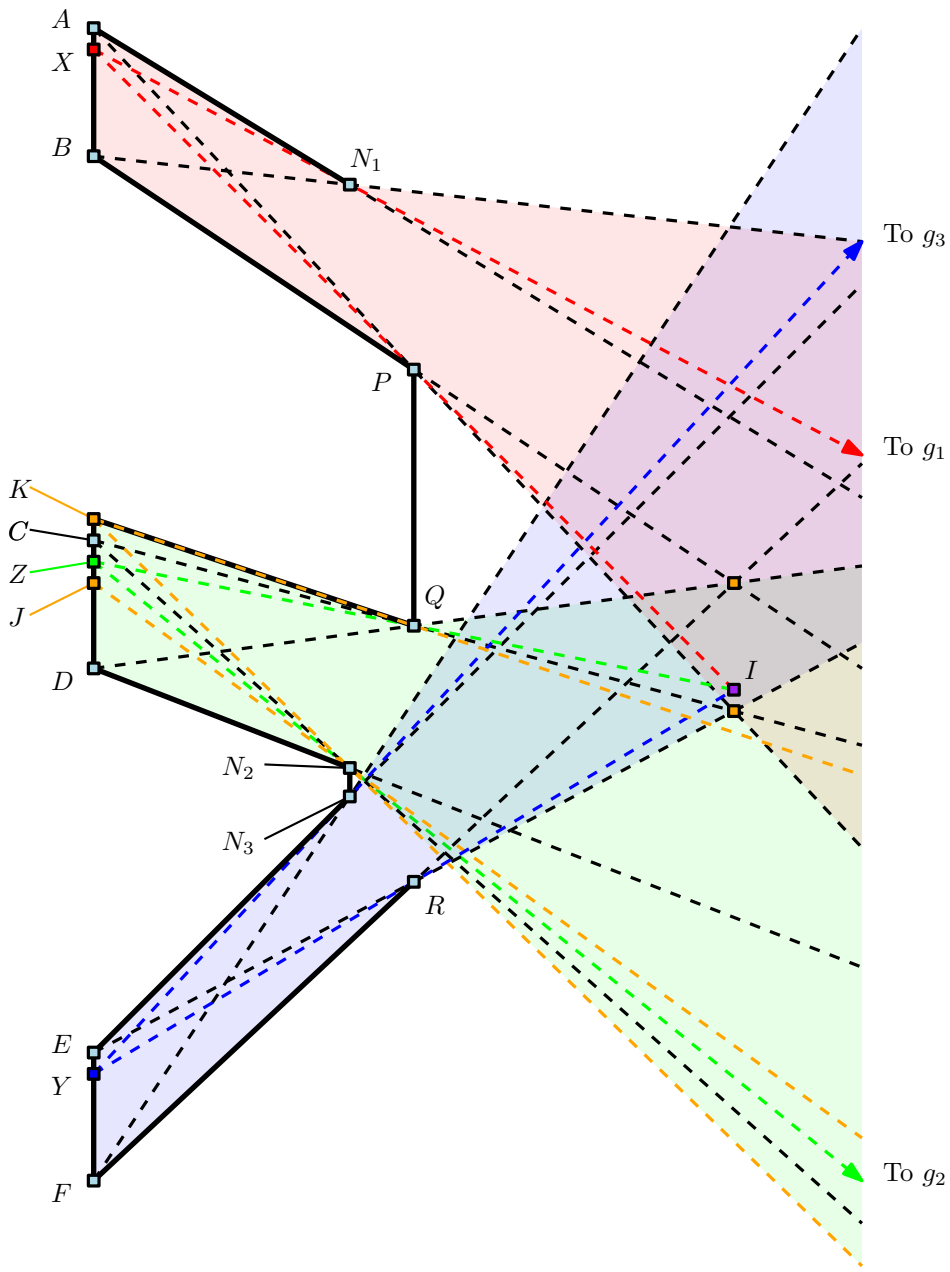


Figure 20 Schematic of the $x + y \leq z$ addition gadget. In the actual gadgets, the lines bounding all the visibility regions have positive slopes as shown in Figures 16 and 17, but it is not possible to draw this faithfully at a readable scale.

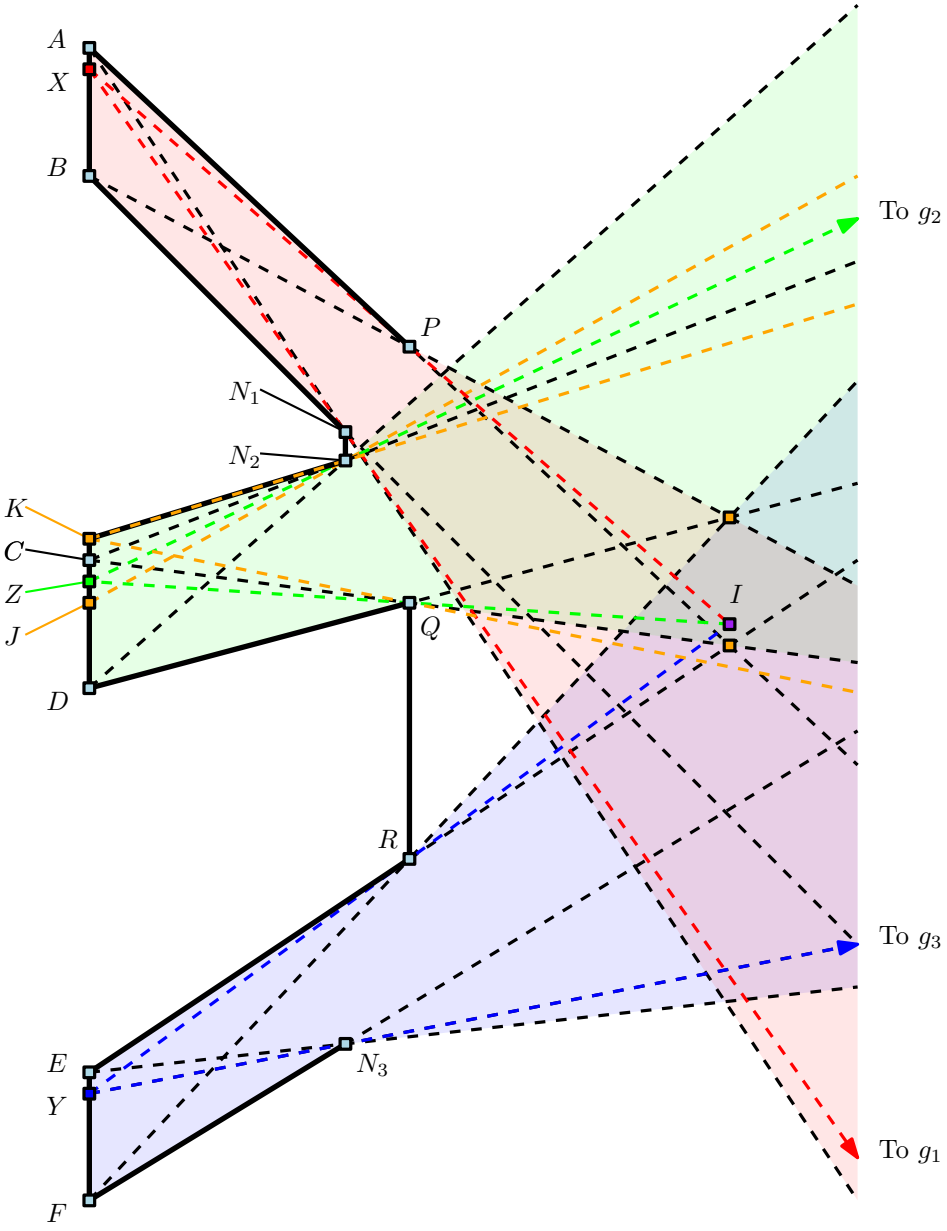


Figure 21 The nooks for the $x + y \geq 1$ gadget.

It is interesting to note that our construction, while intended for the point-boundary variant of the art gallery problem, is also sufficient to show the ∃ℝ-hardness of the standard point-point variant. When it is possible to guard an art gallery produced by our construction in the point-boundary setting, the same guarding configuration is also a point-point guarding configuration. Something similar happens in [2], where the construction for the boundary-point variant is also a construction for the point-point variant. It doesn't seem to be possible to adapt either construction to the boundary-boundary variant.

Unlike the construction from [2], the nook segments in our construction can be chosen to all be axis-parallel. Each other segment is always entirely guarded by a single guard. It may be possible to adapt our construction to show that guarding orthogonal polygons is ∃ℝ-hard.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational Guards are Sometimes Needed. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2017.3.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ∃ℝ-complete. *Journal of the ACM*, 69(1):1–70, 2021. doi:10.1145/3486220.

3 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for er-completeness of two-dimensional packing problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1014–1021, 2020. doi:10.1109/FOCS46700.2020.00098.

4 Akanksha Agrawal, Kristine V. K. Knudsen, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. The Parameterized Complexity of Guarding Almost Convex Polygons. 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 3:1–3:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.3.

5 Daniel Bertschinger, Nicolas El Maalouly, Tillmann Miltzow, Patrick Schnider, and Simon Weber. Topological art in simple galleries. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 87–116. Society for Industrial and Applied Mathematics, January 2022. doi:10.1137/1.9781611977066.8.

6 John Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 460–467, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/62212.62257.

7 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100:238–245, January 2006. doi:10.1007/978-0-387-35608-2_16.

8 Aldo Laurentini. Guarding the walls of an art gallery. *The Visual Computer*, 15:265–278, October 1999. doi:10.1007/s003710050177.

9 D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986. doi:10.1109/TIT.1986.1057165.

10 Nikolai E Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and geometry—Rohlin seminar*, pages 527–543. Springer, 1988.

11 Joseph O'Rourke. *Art gallery theorems and algorithms*, volume 57. Oxford New York, NY, USA, 1987.

12 Marcus Schaefer and Daniel Stefankovic. Fixed points, nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60:172–193, 2017. doi:10.1007/S00224-015-9662-0.

13 Marcus Schaefer and Daniel Štefankovič. Beyond the existential theory of the reals. *Theor. Comp. Sys.*, 68(2):195–226, December 2023. doi:10.1007/s00224-023-10151-x.

14 Peter W. Shor. Stretchability of pseudolines is np-hard. In *Applied Geometry And Discrete Mathematics*, 1990.

15 Jack Stade and Jamie Tucker-Foltz. Topological Universality of the Art Gallery Problem. 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 58:1–58:13, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.58.

Levels in Arrangements: Linear Relations, the g-Matrix, and Applications to Crossing Numbers

Elizaveta Streltsova
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Uli Wagner
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

A long-standing conjecture of Eckhoff, Linhart, and Welzl, which would generalize McMullen's Upper Bound Theorem for polytopes and refine asymptotic bounds due to Clarkson, asserts that for $k \leqslant \lfloor \frac{n-d-2}{2} \rfloor$, the complexity of the $(\leqslant k)$-level in a simple arrangement of n hemispheres in S^d is maximized for arrangements that are polar duals of neighborly d-polytopes. We prove this conjecture in the case $n = d + 4$. By Gale duality, this implies the following result about crossing numbers: In every *spherical arc drawing* of K_n in S^2 (given by a set $V \subset S^2$ of n unit vectors connected by spherical arcs), the number of crossings is at least $\frac{1}{4}\lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor \lfloor \frac{n-2}{2} \rfloor \lfloor \frac{n-3}{2} \rfloor$. This lower bound is attained if every open linear halfspace contains at least $\lfloor (n-2)/2 \rfloor$ of the vectors in V.

Moreover, we determine the space of all linear and affine relations that hold between the face numbers of levels in simple arrangements of n hemispheres in S^d. This completes a long line of research on such relations, answers a question posed by Andrzejak and Welzl in 2003, and generalizes the classical fact that the Dehn–Sommerville relations generate all linear relations between the face numbers of simple polytopes (which correspond to the 0-level).

To prove these results, we introduce the notion of the *g-matrix*, which encodes the face numbers of levels in an arrangement and generalizes the classical *g-vector* of a polytope.

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

Keywords and phrases Levels in arrangements, k-sets, k-facets, convex polytopes, f-vector, h-vector, g-vector, Dehn–Sommerville relations, Radon partitions, Gale duality, g-matrix

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.75

Related Version *Full Version*: `http://arxiv.org/abs/2504.07752` [40]
Full Version: `https://arxiv.org/abs/2504.07770` [41]

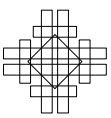
1 Introduction

Levels in arrangements (and the dual notions of k-sets and k-facets) play a fundamental role in discrete and computational geometry and are a natural generalization of convex polytopes (which correspond to the 0-level); we refer to [21, 30, 45] for more background.

It is a classical result in polytope theory that the *Euler–Poincaré relation* is the only linear relation between the face numbers of arbitrary d-dimensional convex polytopes, and that the *Dehn–Sommerville relations* (which we will review below) generate all linear relations between the face numbers of *simple* (or, dually, *simplicial*) polytopes [23, Chs. 8–9]. Another central result in polytope theory is McMullen's *Upper Bound Theorem* [31], which asserts that *cyclic polytopes* have the largest possible number of faces among all d-dimensional convex polytopes with a given number of vertices.

Here, we are interested in generalizations of these results to levels and sublevels in arrangements. To state our results formally, it will be convenient to work with *spherical arrangements* in S^d (which can be seen as a "compactification" of arrangements of affine hyperplanes or halfspaces in $\mathbf{R}^d$ that avoids technical issues related to *unbounded* faces).

© Elizaveta Streltsova and Uli Wagner;
licensed under Creative Commons License CC-BY 4.0



1.1 Levels in Arrangements, Dissection Patterns, and Polytopes

Throughout this paper, let $r = d + 1 \geqslant 1$, and let S^d be the unit sphere in $\mathbf{R}^r$. We denote the standard inner product in $\mathbf{R}^r$ by $\langle \cdot, \cdot \rangle$, and write $\operatorname{sgn}(x) \in \{-1, 0, +1\}$ for the *sign* of a real number $x \in \mathbf{R}$. For a sign vector $F \in \{-1, 0, +1\}^n$, let F_+, F_0, and F_- denote the subsets of coordinates $i \in [n]$ such that $F_i = +1$, $F_i = 0$, and $F_i = -1$, respectively.

Let $V = \{v_1, \ldots, v_n\} \subset \mathbf{R}^r$ be a set of $n \geqslant r$ vectors; fixing the labeling of the vectors, we will also view V as a matrix $V = [v_1 | \ldots | v_n] \in \mathbf{R}^{r \times n}$ with columns v_i. Unless stated otherwise, we assume that V is in general position, i.e., that any r of the vectors are linearly independent. We refer to V as a *vector configuration* of *rank* r.

Every vector $v_i \in V$ defines a great $(d-1)$-sphere $H_i = \{x \in S^d \mid \langle v_i, x \rangle = 0\}$ in S^d and two open hemispheres

$$H_i^+ = \{x \in S^d \mid \langle v_i, x \rangle > 0\}, \qquad H_i^- = \{x \in S^d \mid \langle v_i, x \rangle < 0\}.$$

The resulting *arrangement* $\mathcal{A}(V) = \{H_1^+, \ldots, H_n^+\}$ of hemispheres in S^d determines a decomposition of S^d into *faces* of dimensions 0 through d, where two points $u, u' \in S^d$ lie in the relative interior of the same face of $\mathcal{A}(V)$ iff $\operatorname{sgn}(\langle v_i, u \rangle) = \operatorname{sgn}(\langle v_i, u' \rangle)$ for $1 \leqslant i \leqslant n$. Let $\mathcal{F}(V)$ be the set of all sign vectors $(\operatorname{sgn}(\langle v_1, u \rangle), \ldots, \operatorname{sgn}(\langle v_n, u \rangle)) \in \{-1, 0, +1\}^n$, where u ranges over all non-zero vectors (equivalently, unit vectors) in $\mathbf{R}^r$; we can identify each face of $\mathcal{A}(V)$ with its *signature* $F \in \mathcal{F}(V)$. By general position, the face with signature F has dimension $d - |F_0|$ (the arrangement is *simple*, i.e., there are no faces with $|F_0| > d$). Moreover, we call $|F_-|$ the *level* of the face. Equivalently, the elements of $\mathcal{F}(V)$ correspond bijectively to the partitions of V by oriented linear hyperplanes, and we will also call them the *dissection patterns* of V. In what follows, we will pass freely back and forth between a vector configuration V and the corresponding arrangement $\mathcal{A}(V)$ and refer to this correspondence as *polar duality* (to distinguish it from *Gale duality*, see below).

▶ **Definition 1** (f-matrix and f-polynomial)**.** *For integers s and t, let*[1]

$$f_{s,t} := f_{s,t}(V) := |\{F \in \mathcal{F}(V) \mid |F_0| = s, |F_-| = t\}|.$$

Thus, $f_{s,t}(V)$ counts the $(d-s)$-dimensional faces of level t in $\mathcal{A}(V)$.

Together, these numbers form the f-matrix $f(V) = [f_{s,t}(V)]$*. Equivalently, we can encode this data into the bivariate* f-polynomial $f_V(x, y) \in \mathbf{Z}[x, y]$ *defined by*

$$f_V(x, y) := \sum_{s,t} f_{s,t}(V)\, x^s y^t = \sum_{F \in \mathcal{F}(V)} x^{|F_0|} y^{|F_-|}.$$

We call a vector configuration V *pointed* if it is contained in an open linear halfspace $\{x \in \mathbf{R}^r : \langle u, x \rangle > 0\}$, for some $u \in S^d$, or equivalently, if $\bigcap_{i=1}^n H_i^+ \neq \emptyset$. The closure of this intersection is then a simple (spherical) polytope P, the 0-level of $\mathcal{A}(V)$. By radial projection onto the tangent hyperplane $\{x \in \mathbf{R}^r : \langle u, x \rangle = 1\}$, every pointed configuration $V \subset \mathbf{R}^r$ corresponds to a point set $S \subset \mathbf{R}^d$, see [30, Sec. 5.6]. The convex hull $P^\circ = \operatorname{conv}(S)$ is a simplicial polytope (the polar dual of P), and the elements of $\mathcal{F}(V)$ correspond to the partitions of S by affine hyperplanes; in particular, $f_{s,0}(V)$ counts the $(s-1)$-dimensional faces of P°, and $f_{0,k}(V)$ counts the *k-sets* of S.

[1] By general position, $f_{s,t}(V) = 0$ unless $0 \leqslant s \leqslant d$ and $0 \leqslant t \leqslant n - s$.

1.2 Exact Upper Bounds for Sublevels

The following two special vector configurations will play an important role in this paper.

▶ **Example 2** (Cyclic and Cocyclic Configurations). Let $t_1 < t_2 < \cdots < t_n$ be real numbers and define $v_i := (1, t_i, t_i^2, \ldots, t_i^{r-1}) \in \mathbf{R}^r$. We call $V_{\text{cyclic}}(n,r) := \{v_1, \ldots v_n\}$ and $V_{\text{cocyclic}}(n,r) := \{(-1)^i v_i \colon 1 \leqslant i \leqslant n\}$ the *cyclic* and *cocyclic* configurations of n vectors in $\mathbf{R}^r$, respectively.[2]

Cyclic and cocyclic configurations are examples of neighborly and coneighborly configurations, which we define next. Let $V = \{v_1, \ldots, v_n\} \subseteq \mathbf{R}^r$ be a vector configuration in general position. A subset $W \subseteq V$ is *extremal* if there exists a linear hyperplane H bounding an open halfspace H^+ such that $W \subset H$ and $V \setminus W \subset H^+$. In particular, V is pointed iff $\emptyset \subset V$ is extremal.

▶ **Definition 3** (Neighborly and Coneighborly Configurations). *A vector configuration $V = \{v_1, \ldots, v_n\} \subseteq \mathbf{R}^r$ is* coneighborly *if every open linear halfspace contains at least $\lfloor \frac{n-r+1}{2} \rfloor$ vectors of V. It is* neighborly *if every subset $W \subseteq V$ of size $|W| \leqslant \lfloor \frac{r-1}{2} \rfloor$ is* extremal.

Cyclic configurations are neighborly, cocyclic configurations are coneighborly [48, Cor. 0.8], and these notions are Gale dual to each other (see Section 2). Every neighborly vector configuration $V \subset \mathbf{R}^r$ is pointed, hence corresponds to a point set $S \subset \mathbf{R}^d$, $d = r-1$, and V being neighborly means the simplicial d-polytope $\text{conv}(S)$ is a *neighborly polytope*, i.e., every subset of S of size at most $\lfloor \frac{d}{2} \rfloor$ forms a face. We note that for $r = 1, 2$ ($d = 0, 1$) neighborliness is the same as being pointed, and for $r = 3, 4$ ($d = 2, 3$) V is neighborly iff the point set S is in convex position. By a celebrated result of McMullen [31] neighborly polytopes maximize the number of faces of any dimension:

▶ **Theorem 4** (Upper Bound Theorem for Convex Polytopes). *Let $V \subset \mathbf{R}^r$ be a configuration of n vectors in general position. Then*

$$f_{s,0}(V) \leqslant f_{s,0}(V_{\text{cyclic}}(n,r)) \qquad (0 \leqslant s \leqslant d = r-1)$$

with equality if V is neighborly.

Eckhoff [20, Conj. 9.8], Linhart [26], and Welzl [46], independently of one another (and in slightly different forms) conjectured a far-reaching generalization of Theorem 4 for sublevels of arrangements. To state this conjecture, let $f_{s,\leqslant k}(V) := \sum_{t \leqslant k} f_{s,t}(V)$.

▶ **Conjecture 5** (Generalized Upper Bound Conjecture for Sublevels). *Let $V \subset \mathbf{R}^r$ be a configuration of n vectors in general position, and $0 \leqslant k \leqslant \lfloor \frac{n-r-1}{2} \rfloor$. Then*

$$f_{s,\leqslant k}(V) \leqslant f_{s,\leqslant k}(V_{\text{cyclic}}(n,r)) \qquad (0 \leqslant s \leqslant d = r-1)$$

Equality holds if V is neighborly.

A random sampling argument due to Clarkson (see [18]) shows that Conjecture 5 is true *asymptotically*, for fixed r and $n, k \to \infty$, up to a constant factor depending on r. For the case $s = d$ of vertices at sublevel ($\leqslant k$), Conjecture 5 was proved by Peck [36] and by Alon and Győri [10] for $r \leqslant 3$ and by Welzl [46] for *pointed* vector configurations in rank $r = 4$. The second author [44] proved that it is true up to a factor of 4 for arbitrary rank r. Here, we prove the conjecture for *corank* $n - r = 3$.

[2] The combinatorial types of these configurations are independent of the choice of the parameters t_i.

▶ **Theorem 6.** *Let $V \subset \mathbf{R}^{n-3}$ be a configuration of n vectors in general position. Then*

$$f_{s,\leqslant k}(V) \leqslant f_{s,\leqslant k}(V_{\text{cyclic}}(n,r)) \qquad (0 \leqslant k \leqslant 1, 0 \leqslant s \leqslant d = r-1)$$

Equality holds if V is neighborly.

▶ Remark 7. Bounding the maximum number of faces at level *exactly* k is of a rather different flavor. For coneighborly configurations, all $2\binom{n}{r-1}$ vertices of the dual arrangement are concentrated at one or two consecutive levels $k = \lfloor \frac{n-r+1}{2} \rfloor$ and $k = \lceil \frac{n-r+1}{2} \rceil$. By contrast, determining the maximum number $f_{d,k}$ of vertices at level k for *pointed* vector configurations in $\mathbf{R}^r$ is a difficult open problem, first studied by Lovász [28] and Erdős, Lovász, Simmons, and Straus [22] in the 1970s (see [45] or [30, Ch. 11] for more details and background); even for $r = 3$ (i.e., $d = 2$) there remains a big gap between the best upper and lower bounds to date, which are $O(nk^{1/3})$ and $ne^{\Omega(\sqrt{\log k})}$, respectively (due to Dey [19] and Tóth [43]).

1.3 The Spherical Arc Crossing Number of K_n

By Gale duality (see Section 2), Theorem 6 yields the following result about crossing numbers.

Determining the crossing number $\operatorname{cr}(K_n)$ of the complete graph K_n (the minimum number of crossings in any drawing of K_n in the plane, or equivalently in the sphere S^2, with edges represented by arbitrary Jordan arcs) is one of the foundational unsolved problems in geometric graph theory, first studied by Hill in the 1950's. Hill conjectured the following:

▶ **Conjecture 8** (Hill)**.**

$$\operatorname{cr}(K_n) = X(n) := \frac{1}{4}\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor = \frac{3}{8}\binom{n}{4} + O(n^3) \tag{1}$$

This is known to hold for $n \leqslant 12$, but remains open in general (see [37, Sec. 1.3] or [42] for further background and references). There are several families of drawings showing that $\operatorname{cr}(K_n) \leqslant X(n)$ for all n, but the best lower bound to date [14] is $\operatorname{cr}(K_n) \geqslant 0.985 \cdot X(n)$.

We prove Hill's conjecture for the following class of drawings. Let $V = \{v_1, \ldots, v_n\} \subset S^2$ be a configuration of n *unit* vectors in general position. If we connect every pair of vectors in V by the shortest geodesic arc between them in S^2 (which is unique, since no two vectors are antipodal, by general position) we obtain a drawing of the complete graph K_n in S^2, which we call a *spherical arc drawing*. Let $\operatorname{cr}(V)$ denote the number of crossings in this drawing.

▶ **Theorem 9.** *For every configuration $V \subset S^2$ of n unit vectors in general position,*

$$\operatorname{cr}(V) \geqslant \frac{1}{4}\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor$$

Moreover, the lower bound is attained with equality if V is coneighborly.

The fact that coneighborly configurations yield spherical arc drawings of K_n achieving the number $X(n)$ of crossings in Hill's conjecture, and the connection to the Eckhoff–Linhart–Welzl conjecture were first observed by Wagner [44, 45]. It is known [35] that there are at least $n^{\frac{3}{2}(1-o(1))n}$ combinatorially different coneighborly configurations of n vectors in S^2.

Spherical arc drawings generalize the well-studied class of *rectilinear drawings* of K_n (given by n points in general position in $\mathbf{R}^2$ connected by straight-line segments), which correspond to spherical arc drawings given by pointed vector configurations $V \subset S^2$.

Theorem 9 complements earlier results of Lovász, Vesztergombi, Wagner, and Welzl [29] and Ábrego and Fernández-Merchant [6], who showed that the *rectilinear crossing number* $\overline{\operatorname{cr}}(K_n)$ (the minimum number of crossings in any rectilinear drawing of K_n) is at least

$X(n)$; in fact, $\overline{\mathrm{cr}}(K_n) \geqslant (\frac{3}{8} + \varepsilon + o(1))\binom{n}{4}$ for some constant $\varepsilon > 0$ [29]. Thus, unlike the spherical arc crossing number, the rectilinear crossing number $\overline{\mathrm{cr}}(K_n)$ is strictly larger than $X(n) \geqslant \mathrm{cr}(K_n)$ in the asymptotically leading term. We refer to [8] for a detailed survey, including a series of subsequent improvements [15, 9, 7, 4] leading to the currently best bound [5] $\overline{\mathrm{cr}}(K_n) > 277/729\binom{n}{4} + O(n^3) > 0.37997\binom{n}{4} + O(n^3)$. We remark that the arguments in [29, 6] have been generalized to verify Hill's conjecture for other classes of drawings, including 2-page drawings [2], monotone drawings [13], cylindrical, x-bounded, and shellable drawings [3], bishellable drawings [1], and seq-shellable drawings [34]. Currently we do not know how spherical arc drawings relate to these other classes of drawings.

1.4 The g-Matrix

The central notion of this paper is the *g-matrix* $g(V \to W)$ of a pair $V, W \in \mathbf{R}^{r\times n}$ of vector configurations, which encodes the difference $f(W) - f(V)$ between the f-matrices. The geometric definition of the g-matrix is given in Sec. 3, based on how the f-matrix changes by *mutations* in the course of generic *continuous motion* (an idea with a long history, see, e.g., [11]). The g-matrix is characterized by the following properties:

▶ **Theorem 10.** *Let $V, W \in \mathbf{R}^{r\times n}$ be a pair of vector configurations in general position.*

The g-matrix *$g(V \to W)$ of the pair is an $(r+1)\times(n-r+1)$-matrix with integer entries $g_{j,k} := g_{j,k}(V \to W)$, $0 \leqslant j \leqslant r$, $0 \leqslant k \leqslant n-r$, which has the following properties:*

1. *For $0 \leqslant j \leqslant r$ and $0 \leqslant k \leqslant n-r$, the g-matrix satisfies the skew-symmetries*

$$g_{j,k} = -g_{r-j,k} = -g_{j,n-r-k} = g_{r-j,n-r-k} \tag{2}$$

Thus, the g-matrix is determined by the submatrix $[g_{j,k} : 0 \leqslant j \leqslant \lfloor\frac{r-1}{2}\rfloor, 0 \leqslant k \leqslant \lfloor\frac{n-r-1}{2}\rfloor]$, which we call the small g-matrix. *Equivalently, the* g-polynomial *$g(x,y) := g_{V\to W}(x,y) := \sum_{j,k} g_{j,k}x^j y^k \in \mathbf{Z}[x,y]$ satisfies*

$$g(x,y) = -x^r g(\tfrac{1}{x}, y) = -y^{n-r} g(x, \tfrac{1}{y}) = x^r y^{n-r} g(\tfrac{1}{x}, \tfrac{1}{y}) \tag{3}$$

2. *The g-polynomial determines the difference $f_W(x,y) - f_V(x,y)$ of f-polynomials by*

$$f_W(x,y) - f_V(x,y) = (1+x)^r g(\tfrac{x+y}{1+x}, y) = \textstyle\sum_{j=0}^{r}\sum_{k=0}^{n-r} g_{j,k}\cdot(x+y)^j(1+x)^{r-j}y^k \tag{4}$$

Equivalently (by comparing coefficients), for $0 \leqslant s \leqslant d$ and $0 \leqslant t \leqslant n$,

$$f_{s,t}(W) - f_{s,t}(V) = \sum_{j,k}\binom{j}{t-k}\binom{r-j}{s-j+t-k} g_{j,k}(V \to W), \tag{5}$$

3. *$g(W \to V) = -g(V \to W)$, and $g(U \to W) = g(U \to V) + g(V \to W)$*

▶ Remark 11. The system of equations (5) yields a linear transformation $T = T_{n,r}$ through which the g-matrix $g = g(V \to W)$ of the pair determines the difference $\Delta f = f(W) - f(V)$ of f-matrices by $\Delta f = T(g)$. In the presence of the skew-symmetries (2), the transformation T is injective (Lemma 25), i.e., $g(V \to W)$ is uniquely determined by $\Delta f = f(W) - f(V)$. Thus, Theorem 10 could be taken as a formal definition of the g-matrix.

1.5 Linear Relations

Linear relations between face numbers of levels in simple arrangements have been studied extensively [33, 24, 27, 12, 16].

The first set of linear relations is given by the antipodal symmetry $F \leftrightarrow -F$ of $\mathcal{F}(V)$:

▶ **Observation 12.** *Let $V = \{v_1, \dots, v_n\} \subset \mathbf{R}^r$. Then $f_V(x,y) = y^n f_V(\frac{x}{y}, \frac{1}{y})$; equivalently,*

$$f_{s,t}(V) = f_{s,n-s-t}(V) \quad \text{for all } s \text{ and } t \tag{6}$$

Moreover, it is well-known [23, Sec. 18.1] that the total number of faces of a given dimension $d-s$ (of any level) in a simple arrangement in S^d depends only on n, d, and s:

▶ **Lemma 13.** *Let $\mathcal{A}(V)$ be a simple arrangement of n hemispheres in S^d. Then, for $0 \leqslant s \leqslant d$, the total number of $(d-s)$-dimensional faces (of any level) in $\mathcal{A}(V)$ equals*

$$\sum_t f_{s,t}(V) = 2\binom{n}{s}\sum_{i=0}^{d-s}\binom{n-s-1}{i} = \sum_{i=0}^{d}(1+(-1)^i)\binom{n}{d-i}\binom{d-i}{s} \tag{7}$$

In terms of the f-polynomial, this can be expressed very compactly as

$$f_V(x,1) = \textstyle\sum_{i=0}^{d}\binom{n}{i}\left(1+(-1)^{d-i}\right)(1+x)^i = 2\left(\binom{n}{d}(x+1)^d + \binom{n}{d-2}(x+1)^{d-2} + \dots\right) \tag{8}$$

Linhart, Yang, and Philipp [27] proved the following result, which generalizes the classical *Dehn–Sommerville relations* for simple polytopes:

▶ **Theorem 14** (Dehn–Sommerville Relations for Levels in Simple Arrangements)**.** *Let $V \in \mathbf{R}^{r\times n}$ be a vector configuration in general position. Then*

$$f_V(x,y) = (-1)^d f_V(-(x+y+1), y) \tag{9}$$

Equivalently (by comparing coefficients), for $0 \leqslant s \leqslant d$ and $0 \leqslant t \leqslant n$,

$$f_{s,t}(V) = \sum_j \sum_\ell (-1)^{d-j}\binom{j}{s}\binom{j-s}{t-\ell} f_{j,\ell}(V) \tag{10}$$

▶ Remark 15. The Dehn–Sommerville relations for polytopes correspond to the identity $f_V(x,0) = (-1)^d f_V(-(x+1),0)$. The coefficients on the right-hand side of (10) are zero unless $\ell \leqslant t$ (and $j \geqslant s$). This yields, for every k, a linear system of equations among the numbers $f_{s,t}$, $0 \leqslant s \leqslant d$ and $t \leqslant k$, of face numbers of the $(\leqslant k)$-*sublevel* of the arrangement $\mathcal{A}(V)$. An equivalent system of equations (expressed in terms of an *h-matrix* that generalizes the h-vector of a simple polytope) was proved earlier by Mulmuley [33], under the additional assumption that the $(\leqslant k)$-sublevel is contained in an open hemisphere. Related relations have been rediscovered several times (e.g., in the recent work of Biswas et al. [16]).

▶ Remark 16. The skew-symmetry $g(x,y) = -x^r g(\frac{1}{x}, y)$ of the g-matrix reflects the Dehn–Sommerville relation (9), and the symmetry $g(x,y) = x^r y^{n-r} g(\frac{1}{x}, \frac{1}{y})$ reflects the antipodal symmetry (6).

Let $\mathcal{V}_{n,r}$ denote the set of vector configurations $V \in \mathbf{R}^{r\times n}$ in general position. Let

$$\mathfrak{F}_{n,r} := \operatorname{aff}\{f(V) \colon V \in \mathcal{V}_{n,r}\}, \qquad \mathfrak{G}_{n,r} := \operatorname{lin}\{g(V \to W) \colon V, W \in \mathcal{V}_{n,r}\}$$

be the affine space spanned by all f-matrices, and the linear space spanned by all g-matrices of pairs, respectively. Let $\mathcal{V}^0_{n,r} \subset \mathcal{V}_{n,r}$ be the subset of pointed configurations, and let

$$\mathfrak{F}^0_{n,r} := \operatorname{aff}\{f(V) \colon V \in \mathcal{V}^0_{n,r}\}, \qquad \mathfrak{G}^0_{n,r} := \operatorname{lin}\{g(V \to W) \colon V, W \in \mathcal{V}^0_{n,r}\}$$

be the corresponding subspaces of $\mathfrak{F}_{n,r}$ and $\mathfrak{G}_{r,n}$. Answering a question posed by Andrzejak and Welzl [12], we determine these spaces:

▶ **Theorem 17.** $\dim \mathfrak{F}_{n,r} = \dim \mathfrak{G}_{n,r} = \lfloor \frac{r+1}{2} \rfloor \lfloor \frac{n-r+1}{2} \rfloor$. *More precisely,*

$$\mathfrak{G}_{n,r} = \left\{ g \in \mathbf{R}^{(r+1)\times(n-r+1)} : \begin{array}{c} g_{j,k} = -g_{r-j,k} = -g_{j,n-r-k} = g_{r-j,n-r-k} \\ \text{for } 0 \leqslant j \leqslant r, 0 \leqslant k \leqslant n-r \end{array} \right\} \tag{11}$$

is the space of all real $(r+1) \times (n-r+1)$*-matrices satisfying the skew-symmetries* (2)*, and*

$$\mathfrak{F}_{n,r} = f(V_0) + T(\mathfrak{G}_{n,r})$$

for any fixed $V_0 \in \mathcal{V}_{n,r}$*, where* $T = T_{n,r}$ *is the injective linear transformation given by* (5)*.*

▶ **Theorem 18.** $\dim \mathfrak{F}^0_{n,r} = \dim \mathfrak{G}^0_{n,r} = \lfloor \frac{r-1}{2} \rfloor \lfloor \frac{n-r+1}{2} \rfloor$. *More precisely,*

$$\mathfrak{G}^0_{n,r} = \{ g \in \mathfrak{G}_{n,r} : g_{0,k} = 0, 0 \leqslant k \leqslant n-r \}, \quad \text{and} \quad \mathfrak{F}^0_{n,r} = f(V_0) + T(\mathfrak{G}^0_{n,r})$$

for any $V_0 \in \mathcal{V}^0_{n,r}$*.*

As a specific base configuration V_0 in both theorems, one can take the *cyclic* vector configuration $V_{\text{cyclic}}(n,r)$ (see Example 2), whose f-matrix is known explicitly [12].

2 Gale Duality and Dependency Patterns

Let $V = \{v_1, \dots, v_n\} \subset \mathbf{R}^r$ be a vector configuration in general position, and let $\mathcal{F}^*(V)$ be the set of all sign vectors $(\operatorname{sgn}(\lambda_1), \dots, \operatorname{sgn}(\lambda_n)) \in \{-1, 0, +1\}^n$ given by non-trivial linear dependencies $\sum_{i=1}^n \lambda_i v_i = 0$ (with coefficients $\lambda_i \in \mathbf{R}$, not all of them are zero). We call $\mathcal{F}^*(V)$ the *dependency patterns* of V. Both $\mathcal{F}^*(V)$ and $\mathcal{F}(V)$ are invariant under invertible linear transformations of $\mathbf{R}^r$ and under positive rescaling (multiplying each vector v_i by some positive scalar $\alpha_i > 0$).

If V is a pointed configuration corresponding to a point set $S \subset \mathbf{R}^d$, $d = r-1$, then the elements of $\mathcal{F}^*(V)$ encode the sign patterns of affine dependencies of S, hence they correspond bijectively to (ordered) *Radon partitions* $S = S_- \sqcup S_0 \sqcup S_+$, $\operatorname{conv}(S_+) \cap \operatorname{conv}(S_-) \neq \emptyset$.

▶ **Definition 19** (f^*-matrix and f^*-polynomial)**.** *For integers* s *and* t*, define*[3]

$$f^*_{s,t}(V) := |\{F \in \mathcal{F}^*(V) \mid |F_-| = t, |F_+| = s-t\}|$$

Together, these numbers form the f^*-matrix $f^*(V) = [f^*_{s,t}(V)]$*. Equivalently, we can encode this data into the bivariate* f^*-polynomial $f^*_V(x,y) \in \mathbf{Z}[x,y]$ *defined by*

$$f^*_V(x,y) := \sum_{F \in \mathcal{F}^*(V)} x^{|F_0|} y^{|F_-|} = \sum_{s,t} f^*_{s,t}(V)\, x^{n-s} y^t$$

Dependency patterns and dissection patterns are, in a precise sense, dual to each other:

▶ **Definition 20** (Gale Duality)**.** *Two vector configurations* $V \in \mathbf{R}^{r\times n}$ *and* $W \in \mathbf{R}^{(n-r)\times n}$ *are called* Gale duals *of one another if the rows of* V *and the rows of* W *span subspaces of* $\mathbf{R}^n$ *that are orthogonal complements of one another.*

Since we always assume that V and W are in general position and of full rank, V and W are Gale dual to each other iff $VW^\top = 0$.

[3] Note that $f^*_{s,t}(V) = 0$ unless $r+1 \leqslant s \leqslant n$ and $0 \leqslant t \leqslant s$.

It is well-known that Gale dual configurations determine each other up to linear isomorphisms of their ambient spaces $\mathbf{R}^r$ and $\mathbf{R}^{n-r}$, respectively [30, Sec. 5.6]. Thus, we will speak of *the Gale dual* of V, which we denote by V^*. It follows from the definition that $(V^*)^* = V$.

As another consequence of the definition, $\mathcal{F}^*(V) = \mathcal{F}(V^*)$, hence $f^*_{s,t}(V) = f_{n-s,t}(V^*)$ for all s, t; equivalently, $f^*_V(x, y) = f_{V^*}(x, y)$.

By the well-known separation theorem for convex sets [30, Sec. 1.2], a vector configuration V in general position is either pointed (equivalently, $f_{0,0}(V) = 1$), or there is a linear dependence of the vectors all of whose coefficients are positive (i.e., $f^*_{n,0}(V) = 1$); the same holds for all subsets $W \subseteq V$. It follows from this that $\mathcal{F}(V)$ and $\mathcal{F}^*(V)$ determine each other (see, e.g., [40, Sec. 2] for more details). In particular, a vector configuration $V \subset \mathbf{R}^r$ is neighborly iff $f^*_{s,t}(V) = 0$ for $t \leqslant \lfloor \frac{r-1}{2} \rfloor$, i.e., V is neighborly iff V^* is coneighborly.

Moreover, one can show that the f-matrix and the f^*-matrix of a vector configuration, or equivalently the polynomials $f_V(x, y)$ and $f^*_V(x, y)$ determine each other as well [40, Sec. 2]:

▶ **Theorem 21.** *Let $V \in \mathbf{R}^{r\times n}$ be a vector configuration in general position. Then*

$$f^*_V(x, y) = (x + y + 1)^n - (-1)^r x^n - (x + 1)^n f_V(-\tfrac{x}{x+1}, \tfrac{x+y}{x+1}) \tag{12}$$

and

$$f_V(x, y) = (x + y + 1)^n - (-1)^{n-r} x^n - (x + 1)^n f^*_V(-\tfrac{x}{x+1}, \tfrac{x+y}{x+1}) \tag{13}$$

By Gale duality, Theorem 17 immediately gives a complete description of the affine space $\mathfrak{F}^*_{n,r}$ spanned by the f^*-matrices of vector configurations $V \in \mathbf{R}^{r\times n}$, and there is an analogous characterization for the subspace $(\mathfrak{F}^*_{n,r})^0$ spanned by the f^*-matrices of pointed vector configurations in $\mathbf{R}^r$ (which count the number of Radon partitions of given types for the corresponding point sets in $\mathbf{R}^d$), see [40, Sec. 1.2]

We say that two vector configurations $V, W \in \mathbf{R}^{r\times n}$ have the same *combinatorial type* if (up to a permutation of the vectors) $\mathcal{F}(V) = \mathcal{F}(W)$ (equivalently, $\mathcal{F}^*(V) = \mathcal{F}^*(W)$). Furthermore, we call V and W *weakly equivalent* if they have identical f-matrices (equivalently, identical f^*-matrices).

▶ Remark 22. For readers familiar with *oriented matroids* (see [48, Ch. 6] or [17]), $\mathcal{F}^*(V)$ and $\mathcal{F}(V)$ are precisely the sets of *vectors* and *covectors*, respectively, of the oriented matroid realized by V. However, speaking of "(co)vectors of a vector configuration" seems potentially confusing, and we hope that the terminology of dissection and dependency patterns is more descriptive. The Dehn–Sommerville relations hold for (uniform, not necessarily realizable) oriented matroids. The definition of the g-matrix via continuous motion does not carry over to the oriented matroid setting, but one can still define the g-matrix formally, by the properties described in Theorem 10. We plan to treat this in detail in a future paper.

3 Continuous Motion and the g-Matrix

3.1 The g-Matrix of a Pair

Any two configurations $V = \{v_1, \dots, v_n\}$ and $W = \{w_1, \dots, w_n\}$ of n vectors in general position in $\mathbf{R}^r$ can be deformed into one another through a continuous family $V(t) = \{v_1(t), \dots, v_n(t)\}$ of vector configurations, where $v_i(t)$ describes a continuous path from $v_i(0) = v_i$ to $v_i(1) = w_i$ in $\mathbf{R}^r$. If we choose this continuous motion sufficiently generically, then there is only a finite set of events $0 < t_1 < \cdots < t_N < 1$, called *mutations*, during which the combinatorial type of $V(t)$ (which is encoded by $\mathcal{F}(V(t))$), changes, in a controlled way (see Figures 1 and 2 for an illustration in the case $d = 2$). Specifically, during a mutation,

a unique r-tuple of vectors in $V(t)$, indexed by some $R = \{i_1, \dots, i_r\} \subset [n]$, momentarily becomes linearly dependent and the orientation of this r-tuple (i.e., the sign of $\det[v_{i_1}| \dots |v_{i_r}]$) changes, while the orientations of all other r-tuples of vectors remain unchanged. Thus, any two vector configurations are connected by a finite sequence $V = V_0, V_1, \dots, V_N = W$ such that V_{s-1} and V_s differ by a mutation, $1 \leqslant s \leqslant N$.

We describe the change from $\mathcal{F}(V)$ to $\mathcal{F}(W)$ when V and W differ by a mutation. Let $R \in \binom{[n]}{r}$ index the unique r-tuple of vectors that become linearly dependent. In terms of the polar dual arrangements, this means that the r-tuple of great $(d-1)$-spheres H_i, $i \in R$, momentarily intersect in an antipodal pair $u, -u$ of points in S^d. Immediately before and immediately after the mutation, these r great $(d-1)$-spheres bound an antipodal pair of simplicial d-faces $\sigma, -\sigma$ in $\mathcal{A}(V)$ and a corresponding pair of simplicial d-faces $\tau, -\tau$ in $\mathcal{A}(W)$, respectively. We have $F \in \mathcal{F}(V) \setminus \mathcal{F}(W)$ iff the face of $\mathcal{A}(V)$ with signature F is contained in σ or $-\sigma$, and $F \in \mathcal{F}(W) \setminus \mathcal{F}(V)$ iff the face of $\mathcal{A}(W)$ with signature F is contained in τ or $-\tau$. All other faces are preserved, i.e., they belong to $\mathcal{F}(V) \cap \mathcal{F}(W)$.

Let $Y \in \mathcal{F}(W)$ be the signature of τ. We define a partition $[n] = I \sqcup J \sqcup A \sqcup B$ by

$$I := R \cap Y_+, \quad J := R \cap Y_-, \quad A := ([n] \setminus R) \cap Y_+, \qquad B := ([n] \setminus R) \cap Y_-$$

Define $j := |J|$ and $k := |B|$. We call the pair (j, k) the *type* of the simplicial face τ. The signature $X \in \mathcal{F}(V)$ of the corresponding simplicial face σ of $\mathcal{A}(V)$ satisfies $X_i = -Y_i$ for $i \in R$ and $X_i = Y_i$ for $i \in [n] \setminus R$. Thus, σ is of type $(r-j, k)$. Analogously, $-\tau$ and $-\sigma$ are of type $(r-j, n-r-k)$ and $(j, n-r-k)$, respectively, see Figures 1 and 2.

Let us define $f_\sigma(x, y) := \sum_{F \subseteq \sigma} x^{|F_0|} y^{|F_-|}$, where we use the notation $F \subseteq \sigma$ to indicate that the sum ranges over all $F \in \mathcal{F}(V)$ corresponding to faces of $\mathcal{A}(V)$ contained in σ. The polynomials $f_{-\sigma}(x, y)$, $f_\tau(x, y)$, and $f_{-\tau}(x, y)$ are defined analogously. These four polynomials have a simple form:

$$f_\sigma(x,y) = y^k \left[(x+1)^j (x+y)^{r-j} - x^r\right], \quad f_{-\sigma}(x,y) = y^{n-r-k} \left[(x+1)^{r-j}(x+y)^j - x^r\right]$$
$$f_\tau(x,y) = y^k \left[(x+1)^{r-j} (x+y)^j - x^r\right], \quad f_{-\tau}(x,y) = y^{n-r-k} \left[(x+1)^j (x+y)^{r-j} - x^r\right]$$

We say that the mutation $V \to W$ is of *Type* $(j, k) \equiv (r-j, n-r-k)$. The reverse mutation $W \to V$ is of Type $(r-j, k) \equiv (j, n-r-k)$. We can summarize the discussion as follows:

▶ **Lemma 23.** *Let $V \to W$ be a mutation of Type $(j, k) \equiv (r-j, n-r-k)$ between configurations of n vectors in $\mathbf{R}^r$. Then*

$$f_W(x,y) - f_V(x,y) = \left(y^k - y^{n-r-k}\right) \left[(x+1)^{r-j}(x+y)^j - (x+1)^j (x+y)^{r-j}\right] \tag{14}$$

Note that the right-hand side of (14) is zero if $2j = r$ or $2k = n - r$.

We are now ready to define the g-matrix $g(V \to W)$ of a pair of vector configurations.

▶ **Definition 24** (g-Matrix of a pair)**.** *Let V, W be configurations of n vectors in $\mathbf{R}^r$.*

If $V \to W$ is a single mutation of Type $(i, \ell) \equiv (r-i, n-r-\ell)$ then we define the g-matrix $g(V \to W) = [g_{j,k}(V \to W)]$, $0 \leqslant j \leqslant r$ and $0 \leqslant k \leqslant n-r$, as follows:

If $2i = r$ or $2\ell = n - r$, then $g_{j,k}(V \to W) = 0$ for all j, k. If $2i \neq r$ and $2\ell \neq n-r$, then

$$g_{j,k}(V \to W) := \begin{cases} +1 & \text{if } (j,k) = (i,\ell) \text{ or } (j,k) = (r-i, n-r-\ell) \\ -1 & \text{if } (j,k) = (r-i,\ell) \text{ or } (j,k) = (i, n-r-\ell) \\ 0 & \text{else.} \end{cases}$$

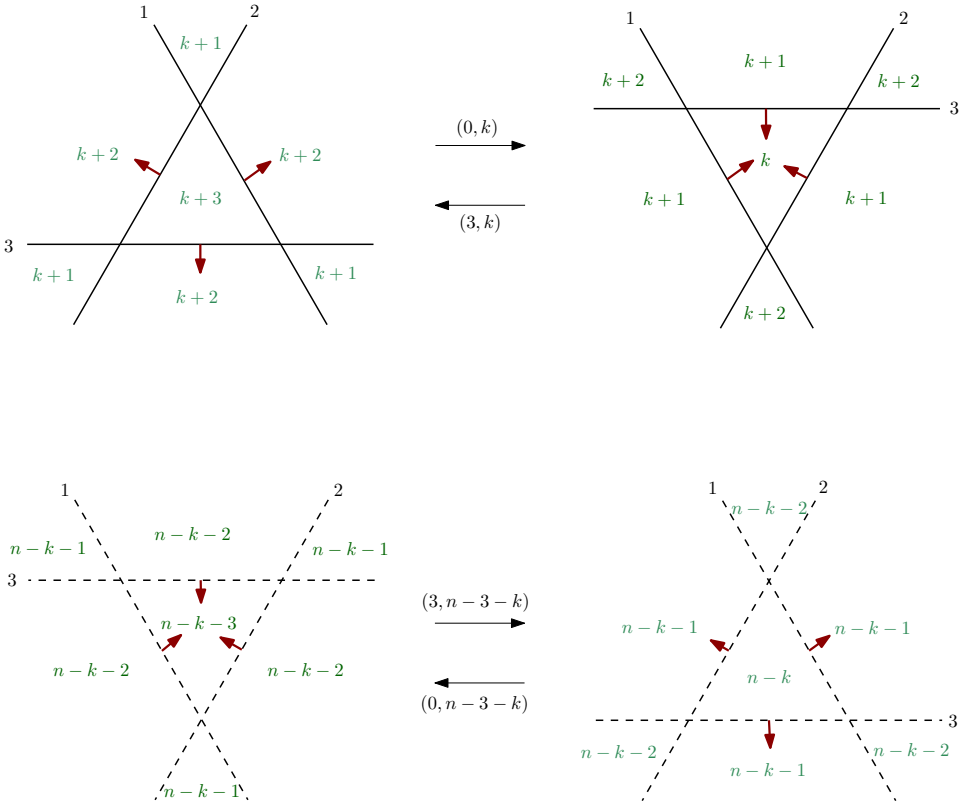


Figure 1 A mutation of Type $(0,k) \equiv (3,n-3-k)$ (from left to right), respectively $(3,k) \equiv (0,n-3-k)$ (from right to left) in S^2. The upper row shows the triangular faces σ and τ before and after the mutation, and the lower row shows the corresponding antipodal faces $-\sigma$ and $-\tau$. The little arrows indicate positive halfspaces, and the labels in full-dimensional faces indicate levels.

More generally, if V and W are connected by a sequence $V = V_0, V_1, \ldots, V_N = W$, where V_{s-1} and V_s differ by a single mutation, then we define

$$g_{j,k}(V \to W) := \sum_{s=1}^{N} g_{j,k}(V_{s-1} \to V_s)$$

Proof of Thm 10. All three properties follow directly from Definition 24 and Lemma 23. ◀

A priori, it may seem that the definition of the g-matrix depends on the choice of a particular sequence of mutations transforming V to W. However, this is not the case:

▶ **Lemma 25.** *Let $f(x,y) = \sum_{s,t} f_{s,t} x^s y^t$ and $g(x,y) = \sum_{j,k} g_{j,k} x^j y^k$ be polynomials (with real coefficients $f_{s,t}$ and $g_{j,k}$ that are zero unless $0 \leqslant s \leqslant d$ and $0 \leqslant t \leqslant n$, respectively $0 \leqslant j \leqslant r$ and $0 \leqslant k \leqslant n-r$). Suppose that $f(x,y)$ and $g(x,y)$ satisfy the identity*

$$f(x,y) = (1+x)^r g(\tfrac{x+y}{1+x}, y) = \textstyle\sum_{j=0}^{r} \sum_{k=0}^{n-r} g_{j,k} \cdot (x+y)^j (1+x)^{r-j} y^k \tag{15}$$

Then, for every fixed t, the numbers $g_{j,t}$, $0 \leqslant j \leqslant r$, are linear combinations of the numbers $f_{s,\ell}$, $0 \leqslant s \leqslant d$ and $0 \leqslant \ell \leqslant t$, with coefficients given inductively by the polynomial equations

$$\sum_j g_{j,t} x^{r-j} = \sum_s f_{s,t} (x-1)^s - \sum_j \sum_{k<t} g_{j,k} \binom{j}{t-k} x^{r-j}$$

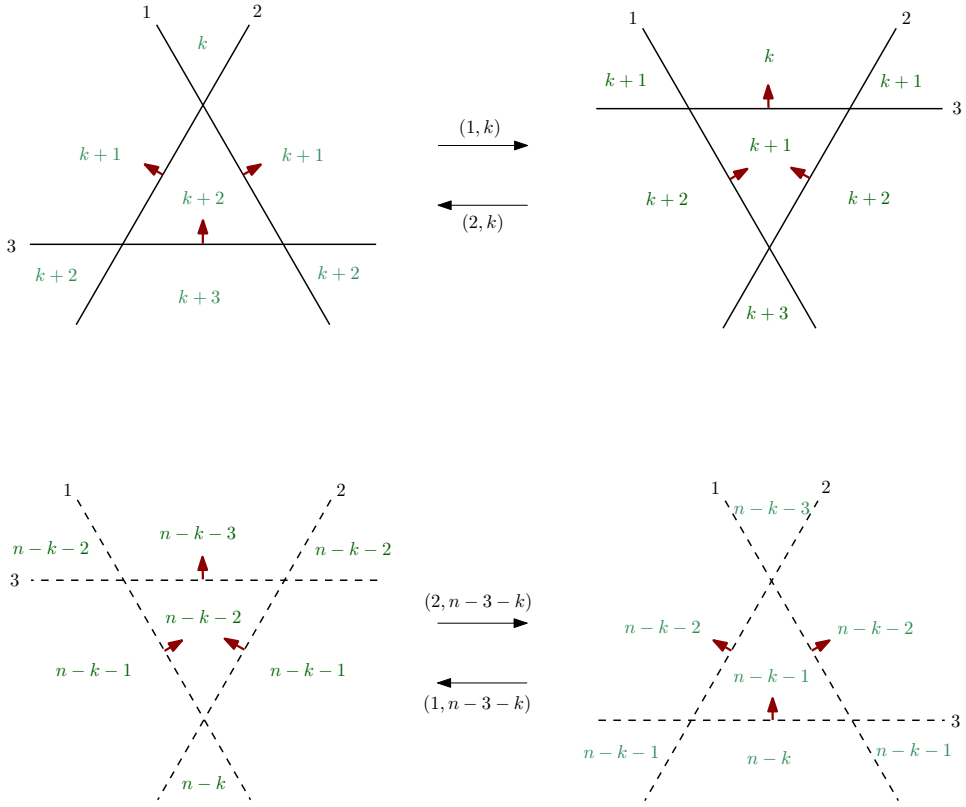


Figure 2 A mutation of Type $(1,k) \equiv (2, n-3-k)$ (from left to right), respectively $(2,k) \equiv (1, n-3-k)$ (from right to left) in S^2.

Proof. The coefficient of y^t in $(x+y)^j(x+1)^{r-j}y^k$ equals $\binom{j}{t-k}(x+1)^{r-j}$ (which is zero unless $0 \leqslant k \leqslant t$). Thus, fixing t and collecting terms in (15) according to y^t, we get

$$\sum_s f_{s,t}x^s = \sum_j \sum_{k \leqslant t} g_{j,k}\binom{j}{t-k}(1+x)^{r-j}$$

Moving the terms with $k < t$ to the other side yields

$$\sum_j g_{j,t}(1+x)^{r-j} = \sum_s f_{s,t}x^s - \sum_j \sum_{k<t} g_{j,k}\binom{j}{t-k}(1+x)^{r-j}$$

The result follows by a change of variable from x to $x-1$ (inductively, the numbers $g_{j,k}$, $k < t$, are determined by the numbers $f_{s,\ell}$, $\ell < t$.) ◀

By Theorem 21, the f-polynomial and the f^*-polynomial of a vector configuration determine each other. This yields the following analogue of Theorem 10 (which can also proved directly, by studying how $\mathcal{F}^*$ changes during mutations):

▶ **Theorem 26.** *Let V, W be configurations of n vectors in $\mathbf{R}^r$. Then*

$$f^*_W(x,y) - f^*_V(x,y) = \sum_{j,k} \underbrace{g_{j,k}(W \to V)}_{=-g_{j,k}(V \to W)}(x+y)^k(x+1)^{n-r-k}y^j \tag{16}$$

Theorems 10 and 26 imply the following:

▶ **Corollary 27.** *Let $V, W \in \mathbf{R}^{r\times n}$ be vector configurations, and let $V^*, W^* \in \mathbf{R}^{(n-r)\times n}$ be their Gale duals. Then $g_{j,k}(V \to W) = -g_{k,j}(V^* \to W^*)$.*

Using the above results, one can show [40, Sec. 3.1] that any pair of neighborly configurations of n vectors in $\mathbf{R}^r$ have the same f-matrix and the same f^*-matrix (equivalently, the g-matrix of the pair is identically zero); analogously for coneighborly configurations.

Moreover, an inductive argument based on *deletions* and *contractions* [40, Sec. 3.2] yields the following result. We say that a vector configuration $V \subset \mathbf{R}^r$ is *j-neighborly* if every subset of V of size j is extremal, and we call V is *k-coneighborly* if $f_{s,t} = 0$ for $t \leqslant k$, i.e., if every open linear halfspace contains at least $k+1$ vectors from V.

▶ **Lemma 28.** *Let $0 \leqslant j \leqslant \frac{r-1}{2}$, $0 \leqslant k \leqslant \frac{n-r-1}{2}$, and let $V, W \in \mathbf{R}^{r\times n}$ be vector configurations such that V is k-coneighborly and W is j-neighborly. Then*

$$g_{j,k}(V \to W) = \tbinom{n-k-r+j}{j}\tbinom{k+r-1-j}{k} - \tbinom{n-k-r+j-1}{j-1}\tbinom{k+r-j}{k} > 0 \tag{17}$$

3.2 The g-Matrix and the g^*-Matrix

We are now ready to define the g-matrix and the g^*-matrix of a vector configuration.

▶ **Definition 29** (g-matrix and g^*-matrix)**.** *Let V be a configuration of n vectors in $\mathbb{R}^r$. Set*

$$g_{j,k}(V) := g_{j,k}(V_{\text{cocyclic}}(n,r) \to V), \qquad \text{and} \qquad g^*_{j,k}(V) := g_{j,k}(V \to V_{\text{cyclic}}(n,r))$$

for $0 \leqslant j \leqslant r$ and $0 \leqslant k \leqslant n-r$. We call $g(V) = [g_{j,k}(V)]$ and $g^*(V) = [g^*_{j,k}(V)]$ the g-matrix *and the* g^*-matrix *of V, respectively. By the skew-symmetries* (2)*, both matrices are determined by their "upper left" quadrants indexed by $0 \leqslant j \leqslant \lfloor\frac{r-1}{2}\rfloor$ and $0 \leqslant k \leqslant \lfloor\frac{n-r-1}{2}\rfloor$, which we call the* small g-matrix *and the* small g^*-matrix, respectively.

By Gale duality, we get $g^*_{j,k}(V) = g_{k,j}(V^*)$, i.e., the g^*-matrix of V is the transpose of the g-matrix of the Gale dual V^*. Moreover, by Lemma 28,

$$g_{j,k}(V) + g^*_{j,k}(V) = \tbinom{n-k-r+j}{j}\tbinom{k+r-1-j}{k} - \tbinom{n-k-r+j-1}{j-1}\tbinom{k+r-j}{k} > 0 \tag{18}$$

for $0 \leqslant j \leqslant \frac{r-1}{2}$ and $0 \leqslant k \leqslant \frac{n-r-1}{2}$.

The 0-th column $[g_{j,0}(V) \mid 0 \leqslant j \leqslant r]$ of the g-matrix corresponds to the classical g-vector of a simple polytope. The *Generalized Lower Bound Theorem* (first proved by Stanley [39], as part of a full characterization of g-vectors, and hence f-vectors of simple polytopes, see [48, Sec. 8.6]), asserts that $g_{j,0}(V) \geqslant 0$ for $0 \leqslant j \leqslant \frac{r-1}{2}$.

The 0-th row $[g^*_{0,k} \mid 0 \leqslant k \leqslant n-r]$ corresponds to a Gale dual version of the g-vector of convex polytopes studied by Lee [25] and Welzl [46]. The following is implicit in [46, Sec. 4] (and implies McMullen's Upper Bound Theorem):

▶ **Theorem 30.** *Let $V \subset \mathbf{R}^r$ be a configuration of n vectors in general position. Then*

$$g^*_{0,k}(V) \leqslant \binom{k+r-1}{r-1} \qquad (0 \leqslant k \leqslant \lfloor\frac{n-r-1}{2}\rfloor)$$

Equivalently, by (18)*, $g_{0,k} \geqslant 0$ for $0 \leqslant k \leqslant \lfloor\frac{n-r-1}{2}\rfloor$. Equality holds if V is coneighborly.*

As a common generalization of the Upper Bound Theorem, the Generalized Lower Bound Theorem, and the Eckhoff–Linhart–Welzl Conjecture (Conj. 5), we propose the following:

▶ **Conjecture 31** (Nonnegativity of the Small g-Matrix)**.** *Let $V \subset \mathbf{R}^r$ be a configuration of n vectors in general position. Then $g_{j,k}(V) \geqslant 0$ for $0 \leqslant j \leqslant \frac{r-1}{2}$ and $0 \leqslant k \leqslant \frac{n-r-1}{2}$.*

Our main result (which implies Theorems 9 and 6) is the following:

▶ **Theorem 32.** *Let $V \subset \mathbf{R}^3$ be a configuration of n vectors in general position. Then*

$$g^*_{1,k}(V) \leqslant (k+1)n - 3\binom{k+2}{2} \qquad (0 \leqslant k \leqslant \lfloor \frac{n-4}{2} \rfloor)$$

Equivalently, $g_{1,k}(V) \geqslant 0$ for $0 \leqslant k \leqslant \lfloor \frac{n-4}{2} \rfloor$. Equality holds if V is coneighborly.

4 The Space $\mathfrak{G}_{n,r}$ Spanned by g-Matrices

Here, we prove Theorem 17. (The proof of the corresponding result for pointed configurations – Theorem 18 – uses similar ideas but is more involved, see [40]). By Theorem 10, the description of the space $\mathfrak{F}_{n,r}$ follows from the description of the space $\mathfrak{G}_{n,r}$, so it remains to prove the latter. Recall that $\mathcal{V}_{n,r}$ is the set of all vector configurations $V \in \mathbf{R}^{r\times n}$ in general position.

By Theorem 10, the g-matrix $g = g(V \to W)$ of any pair $V, W \in \mathcal{V}_{n,r}$ satisfies the skew-symmetries $g_{j,k} = -g_{r-j,k} = -g_{j,n-r-k} = g_{r-j,n-r-k}$ in (2). Thus, in order to prove Theorem 17, it remains to show that $\mathfrak{G}_{n,r} = \operatorname{lin}\{g(V \to W) \colon V, W \in \mathcal{V}_{n,r}\}$ has dimension $\lfloor \frac{r+1}{2} \rfloor \lfloor \frac{n-r+1}{2} \rfloor$. To see this, consider a generic continuous deformation from a coneighborly configuration V_0 to a neighborly configuration V_N, and let V_t, $0 \leqslant i \leqslant N$, be the intermediate vector configurations, i.e., V_t and V_{t-1} differ by a mutation. Thus, the g-matrices $g(V_0 \to V_t)$ and $g(V_0 \to V_{t-1})$ differ by the g-matrix of a mutation, i.e., their first quadrants (small g-matrices) differ in at most one coordinate, by $+1$ or -1. Moreover, $g(V_0 \to V_0)$ is identically zero, and every entry of the first quadrant of $g(V_0 \to V_N)$ is strictly positive by Lemma 28. Thus, the proof of Theorem 17 is completed by the following lemma:

▶ **Lemma 33.** *Let $X_0, X_1, \ldots, X_N$ be vectors in $\mathbf{R}^m$ such that*

1. *$X_0 = 0$;*
2. *X_t and X_{t-1} differ in one coordinate, by $+1$ or -1;*
3. *All coordinates of X_N are non-zero.*

Then there is a subset $X_{t_1}, \ldots, X_{t_m}$ of vectors that form a basis of $\mathbf{R}^m$.

Proof. For $1 \leqslant i \leqslant m$, let t_i be the smallest $t \in \{1, \ldots, N\}$ such that the i-th coordinate of X_{t_i} is non-zero; the index t_i exists by Properties 1 and 3. Moreover, by Property 2, no two coordinates can become non-zero at the same time, i.e., the indices t_i are pairwise distinct. Up to re-labeling the coordinates, we may assume $t_1 < t_2 < \ldots < t_m$. Then, for $1 \leqslant i \leqslant m$, the vector X_{t_i} is linearly independent from the vectors $X_{t_1}, \ldots X_{t_{i-1}}$, since all of the latter vectors have i-th coordinate zero. Thus, the X_{t_i} form a basis. ◀

5 Bounding the Spherical Arc Crossing Number

In this section, we outline the proof of Theorem 32 and explain how it implies Theorems 6 and 9. By Gale duality, Theorem 6 is equivalent to the following:

▶ **Theorem 34.** *Let $V = \{v_1, \ldots, v_n\} \subset \mathbf{R}^3$ be a vector configuration in general position. Then, for all $s \leqslant n$, the numbers $f^*_{s,0}(V)$ and $f^*_{s,\leqslant 1}(V) := f^*_{s,0}(V) + f^*_{s,1}(V)$ are maximized if V is coneighborly.*

Theorem 26 and the skew-symmetries of the g^*-matrix imply the following (see [41]):

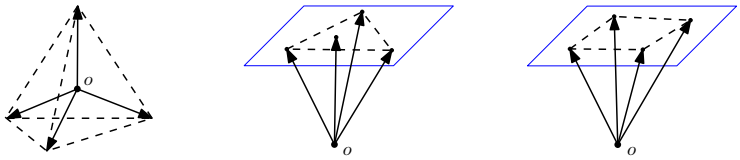


Figure 3 The three combinatorial types of four vectors in general position in $\mathbf{R}^3$.

▶ **Lemma 35.** *Let $V \in \mathbf{R}^{3\times n}$ be a vector configuration in general position and let $s \geqslant 4$. Then there are non-negative integers $\alpha(n,s,k)$ and $\beta(n,s,k)$, $0 \leqslant k \leqslant \lfloor \frac{n-4}{2} \rfloor$, such that*

$$f^*_{s,0}(V) = \sum_{k=0}^{\lfloor \frac{n-4}{2} \rfloor} \alpha(n,k,s) \cdot g^*_{0,k}(V)$$

and

$$f^*_{s,\leqslant 1}(V) = \sum_{k=0}^{\lfloor \frac{n-4}{2} \rfloor} \alpha(n,k,s) \cdot g^*_{1,k}(V) + \sum_{k=0}^{\lfloor \frac{n-4}{2} \rfloor} \beta(n,s,k) \cdot g^*_{0,k}(V)$$

Specifically, $\alpha(n,k,s) = \binom{n-3-k}{s-3} - \binom{k}{s-3}$ and $\beta(n,s,k) = \binom{n-3-k}{s-4}k - \binom{k}{s-4}(n-3-k)$.

Proof of Theorem 34. By Lemma 35, for $s \geqslant 4$, $f^*_{s,0}(V)$ and $f^*_{s,\leqslant 1}(V)$ are non-negative linear combinations of the numbers $g^*_{0,k}(V)$ and $g^*_{1,k}(V)$, $0 \leqslant k \leqslant \lfloor \frac{n-4}{2} \rfloor$. Hence, Theorem 34 follows from Theorems 30 and 32. ◀

Proof of Theorem 9. Let $V \subset \mathbf{R}^3$ be a configuration of n vectors in general position. There are three combinatorial types of quadruples $W \subset V$, which we call Type 0, Type 1, and Type 2, respectively, see Fig. 3. Each quadruple of Type i supports exactly two dependency patterns $F, -F \in \mathcal{F}^*(V)$ with four non-zero signs, one with $|F_-| = i$ negative signs, and the other one with $|F_+| = 4 - i$ negative signs. It follows that

$$f^*_{4,0}(V) + f^*_{4,1}(V) + \frac{1}{2} f^*_{4,2}(V) = \binom{n}{4}$$

Moreover, if all the vectors in V have unit length, then the number of crossings in the induced spherical arc drawing of K_n equals $\mathrm{cr}(V) = \frac{1}{2} f^*_{4,2}(V)$. Thus Theorem 9 is equivalent to the statement that $f^*_{4,2}(V) \geqslant 2X(n)$, equivalently, $f^*_{4,0}(V) + f^*_{4,1}(V) \leqslant \binom{n}{4} - X(n)$, where equality holds in both bounds if V is coneighborly. By the special case $s = 4$ of Theorem 34,

$$f^*_{4,\leqslant 1}(V) \leqslant \sum_{k=0}^{\lfloor \frac{n-4}{2} \rfloor} (n-3-2k)\left((k+1)n - 3\binom{k+2}{2}\right) =: Y(n), \tag{19}$$

with equality if V is coneighborly. By a direct calculation (see [41]), $Y(n) = \binom{n}{4} - X(n)$. ◀

▶ Remark 36. We mention a connection to geometric probability. Let μ be a probability distribution on $\mathbf{R}^3$; we assume that μ is non-degenerate in the sense that every plane through the origin has μ-measure zero. A beautiful argument due to Wendel [47] shows that if μ is *centrally symmetric* and $W = \{w_1, w_2, w_3, w_4\} \subset \mathbf{R}^3$ is a set of four independent μ-random vectors then the probability that W is of Type $i \in \{0,1,2\}$ (Fig. 3) equals q_i, where $q_0 = \frac{\binom{4}{0}+\binom{4}{4}}{2^4} = \frac{1}{8}$, $q_1 = \frac{\binom{4}{1}+\binom{4}{3}}{2^4} = \frac{1}{2}$, and $q_2 = \frac{\binom{4}{2}}{2^4} = \frac{3}{8}$; it follows that for any set $V \subset \mathbf{R}^3$ of

n independent μ-random vectors, the expected number of quadruples of type i equals $q_i\binom{n}{4}$. In particular, if the vectors in V are chosen independently and uniformly at random from S^2 then the expected number of crossings in the spherical arc drawing given by V is $\frac{3}{8}\binom{n}{4}$, which was also independently shown by Moon [32]. Theorem 9, together with a well-known limiting argument [38] implies that for an arbitrary (not necessarily centrally symmetric) non-degenerate probability distribution μ on S^2, the expected number of crossings in the spherical arc drawing given by n independent μ-random points is *at least* $\frac{3}{8}\binom{n}{4}$.

To conclude, we outline some of the main ideas for the proof of Theorem 32 (see [41] for the details). Let $V = \{v_1, \dots, v_n\} \subset \mathbf{R}^3$ be a configuration of n vectors in general position. We will view V as the set $S - o$ of differences $v_i = p_i - o$ between a point set $S = \{p_1, \dots, p_n\} \subset \mathbf{R}^3$ (the *tips* of the vectors) and another point $o \in \mathbf{R}^3$ (the *origin*).

Choose a line ℓ in $\mathbf{R}^3$ through o in general position. By positively rescaling the vectors v_i, we may assume that S is a subset of the cylinder Z consisting of the points in $\mathbf{R}^3$ at Euclidean distance 1 from ℓ; in particular, S is a point set in convex position.

The point set $S \subset \mathbf{R}^3$ corresponds to a neighborly configuration $U = \{u_1, \dots, u_n\}$ of vectors in $\mathbf{R}^4$, and the origin $o \in \mathbf{R}^3$ corresponds to an additional vector $u_0 \in \mathbf{R}^4$. Let $\mathcal{A}(\{u_0\} \sqcup U) = \{H_0^+, H_1^+, \dots, H_n^+\}$ be the polar dual arrangement of hemispheres in S^3. The intersection $\mathcal{A}(U) \cap H_0$ is a spherical arrangment in the equatorial 2-sphere $H_0 \cong S^2$, which is combinatorially isomorphic to the arrangement $\mathcal{A}(V)$.

Let $C = H_i \cap H_j$ be the great circle in S^3 formed by the intersection of two great 2-spheres of the arrangement $\mathcal{A}(U)$, $1 \leqslant i < j \leqslant n$. For $k \leqslant \lfloor \frac{n-4}{2} \rfloor$, consider the subgraph of $\mathcal{A}(U)$ consisting of the vertices and edges of $\mathcal{A}(U)$ contained in C and at sublevel $(\leqslant k)$. This subgraph cannot cover the entire circle C, since C contains at least one pair of antipodal vertices at levels $\lfloor \frac{n-2}{2} \rfloor, \lceil \frac{n-2}{2} \rceil$. Thus, the connected components of this subgraph form closed intervals, which we call *k-arcs* of $\mathcal{A}(U)$. We define $\lambda_k(U, u_0)$ as the number of k-arcs of $\mathcal{A}(U)$ that are completely contained in the negative open hemisphere H_0^-.

Suppose we move the origin o continuously along the line ℓ, while keeping the set S fixed. Consider the corresponding continuous motions of the vector configurations $V = S - o$ in $\mathbf{R}^3$ and $\{u_0\} \sqcup U$ in $\mathbf{R}^4$. A careful analysis of how $\lambda_k(U, u_0)$ and $g_{1,k}(V)$ change during this continuous motion yields

$$g_{1,k}(V) = \lambda_k(U, u_0) \geqslant 0$$

for $0 \leqslant k \leqslant \lfloor \frac{n-4}{2} \rfloor$, which proves Theorem 32.

References

1. Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Dan McQuillan, Bojan Mohar, Petra Mutzel, Pedro Ramos, R. Bruce Richter, and Birgit Vogtenhuber. Bishellable drawings of K_n. *SIAM J. Discrete Math.*, 32(4):2482–2492, 2018. doi:10.1137/17M1147974.
2. Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Gelasio Salazar. The 2-page crossing number of K_n. *Discrete Comput. Geom.*, 49(4):747–777, 2013. doi:10.1007/S00454-013-9514-0.
3. Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Gelasio Salazar. Shellable drawings and the cylindrical crossing number of K_n. *Discrete Comput. Geom.*, 52(4):743–753, 2014. doi:10.1007/S00454-014-9635-0.
4. Bernardo M. Ábrego, József Balogh, Silvia Fernández-Merchant, Jesús Leaños, and Gelasio Salazar. An extended lower bound on the number of $(\leqslant k)$-edges to generalized configurations of points and the pseudolinear crossing number of K_n. *J. Combin. Theory Ser. A*, 115(7):1257–1264, 2008. doi:10.1016/J.JCTA.2007.11.004.

5 Bernardo M. Ábrego, Mario Cetina, Silvia Fernández-Merchant, Jesús Leaños, and Gelasio Salazar. On $\leqslant k$-edges, crossings, and halving lines of geometric drawings of K_n. *Discrete Comput. Geom.*, 48(1):192–215, 2012. doi:10.1007/S00454-012-9403-Y.

6 Bernardo M. Ábrego and Silvia Fernández-Merchant. A lower bound for the rectilinear crossing number. *Graphs Combin.*, 21(3):293–300, 2005. doi:10.1007/S00373-005-0612-5.

7 Bernardo M. Ábrego, Silvia Fernández-Merchant, Jesús Leaños, and Gelasio Salazar. A central approach to bound the number of crossings in a generalized configuration. In *The IV Latin-American Algorithms, Graphs, and Optimization Symposium*, volume 30 of *Electron. Notes Discrete Math.*, pages 273–278. Elsevier Sci. B. V., Amsterdam, 2008. doi:10.1016/J.ENDM.2008.01.047.

8 Bernardo M. Ábrego, Silvia Fernández-Merchant, and Gelasio Salazar. The rectilinear crossing number of K_n: closing in (or are we?). In *Thirty essays on geometric graph theory*, pages 5–18. Springer, New York, 2013.

9 Oswin Aichholzer, Jesús García, David Orden, and Pedro Ramos. New lower bounds for the number of $(\leqslant k)$-edges and the rectilinear crossing number of K_n. *Discrete Comput. Geom.*, 38(1):1–14, 2007. doi:10.1007/S00454-007-1325-8.

10 Noga Alon and E. Győri. The number of small semispaces of a finite set of points in the plane. *J. Combin. Theory Ser. A*, 41(1):154–157, 1986. doi:10.1016/0097-3165(86)90122-6.

11 Artur Andrzejak, Boris Aronov, Sariel Har-Peled, Raimund Seidel, and Emo Welzl. Results on k-sets and j-facets via continuous motion. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pages 192–199, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276884.276906.

12 Artur Andrzejak and Emo Welzl. In between k-sets, j-facets, and i-faces: (i, j)-partitions. *Discrete Comput. Geom.*, 29(1):105–131, 2003.

13 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discrete Comput. Geom.*, 53(1):107–143, 2015.

14 József Balogh, Bernard Lidický, and Gelasio Salazar. Closing in on Hill's conjecture. *SIAM J. Discrete Math.*, 33(3):1261–1276, 2019. doi:10.1137/17M1158859.

15 József Balogh and Gelasio Salazar. k-sets, convex quadrilaterals, and the rectilinear crossing number of K_n. *Discrete Comput. Geom.*, 35(4):671–690, 2006. doi:10.1007/S00454-005-1227-6.

16 Ranita Biswas, Sebastiano Cultrera di Montesano, Herbert Edelsbrunner, and Morteza Saghafian. Depth in arrangements: Dehn-Sommerville-Euler relations with applications. *J. Appl. Comput. Topol.*, 8(3):557–578, 2024. doi:10.1007/S41468-024-00173-W.

17 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1999.

18 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry. II. *Discrete Comput. Geom.*, 4(5):387–421, 1989. doi:10.1007/BF02187740.

19 Tamal K. Dey. Improved bounds for planar k-sets and related problems. *Discrete Comput. Geom.*, 19(3):373–382, 1998. doi:10.1007/PL00009354.

20 Jürgen Eckhoff. Helly, Radon, and Carathéodory type theorems. In *Handbook of convex geometry, Vol. A, B*, pages 389–448. North-Holland, Amsterdam, 1993.

21 Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1987. doi:10.1007/978-3-642-61568-9.

22 Paul Erdős, László. Lovász, A. Simmons, and Ernst G. Straus. Dissection graphs of planar point sets. In *A survey of combinatorial theory (Proc. Internat. Sympos., Colorado State Univ., Fort Collins, Colo., 1971)*, pages 139–149. North-Holland, Amsterdam-London, 1973.

23 Branko Grünbaum. *Convex polytopes*, volume 221 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 2003.

24 T. Gulliksen and A. Hole. On the number of linearly separable subsets of finite sets in $\mathbf{R}^n$. *Discrete Comput. Geom.*, 18(4):463–472, 1997.

25 Carl W. Lee. Winding numbers and the generalized lower-bound conjecture. In *Discrete and computational geometry (New Brunswick, NJ, 1989/1990)*, volume 6 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 209–219. Amer. Math. Soc., Providence, RI, 1991. doi:10.1090/DIMACS/006/13.

26 Johann Linhart. The upper bound conjecture for arrangements of halfspaces. *Beiträge Algebra Geom.*, 35(1):29–35, 1994.

27 Johann Linhart, Yanling Yang, and Martin Philipp. Arrangements of hemispheres and halfspaces. *Discrete Math.*, 223(1-3):217–226, 2000. doi:10.1016/S0012-365X(98)00360-4.

28 László Lovász. On the number of halving lines. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 14:107–108, 1971.

29 László Lovász, Katalin Vesztergombi, Uli Wagner, and Emo Welzl. Convex quadrilaterals and k-sets. In *Towards a theory of geometric graphs*, volume 342 of *Contemp. Math.*, pages 139–148. Amer. Math. Soc., Providence, RI, 2004.

30 Jiří Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

31 Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

32 John W. Moon. On the distribution of crossings in random complete graphs. *J. Soc. Indust. Appl. Math.*, 13:506–510, 1965.

33 Ketan Mulmuley. Dehn-sommerville relations, upper bound theorem, and levels in arrangements. In Chee Yap, editor, *Proceedings of the Ninth Annual Symposium on Computational Geometry (San Diego, CA, USA, May 19-21, 1993)*, pages 240–246, 1993. doi:10.1145/160985.161141.

34 Petra Mutzel and Lutz Oettershagen. The crossing number of seq-shellable drawings of complete graphs. In *Combinatorial algorithms*, volume 10979 of *Lecture Notes in Comput. Sci.*, pages 273–284. Springer, Cham, 2018. doi:10.1007/978-3-319-94667-2_23.

35 Arnau Padrol. Many neighborly polytopes and oriented matroids. *Discrete Comput. Geom.*, 50(4):865–902, 2013. doi:10.1007/S00454-013-9544-7.

36 G. W. Peck. On "k-sets" in the plane. *Discrete Math.*, 56(1):73–74, 1985.

37 Marcus Schaefer. *Crossing numbers of graphs*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, 2018.

38 Edward R. Scheinerman and Herbert S. Wilf. The rectilinear crossing number of a complete graph and Sylvester's "four point problem" of geometric probability. *Amer. Math. Monthly*, 101(10):939–943, 1994.

39 Richard P. Stanley. The number of faces of a simplicial convex polytope. *Adv. in Math.*, 35(3):236–238, 1980.

40 Elizaveta Streltsova and Uli Wagner. Linear relations between face numbers of levels in arrangements. Preprint, 2025. arXiv:2504.07752.

41 Elizaveta Streltsova and Uli Wagner. Sublevels in arrangements and the spherical arc crossing number of complete graphs. Preprint, 2025. arXiv:2504.07770.

42 László A. Székely. Turán's brick factory problem: the status of the conjectures of Zarankiewicz and Hill. In *Graph theory – Favorite conjectures and open problems. 1*, Probl. Books in Math., pages 211–230. Springer, 2016.

43 Géza Tóth. Point sets with many k-sets. *Discrete Comput. Geom.*, 26(2):187–194, 2001. doi:10.1007/S004540010022.

44 Uli Wagner. On a geometric generalization of the upper bound theorem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 635–645, 2006. doi:10.1109/FOCS.2006.53.

45 Uli Wagner. k-Sets and k-facets. In *Surveys on Discrete and Computational Geometry*, volume 453 of *Contemp. Math.*, pages 443–513. Amer. Math. Soc., Providence, RI, 2008.

46 Emo Welzl. Entering and leaving j-facets. *Discrete Comput. Geom.*, 25(3):351–364, 2001. doi:10.1007/S004540010085.

47 James G. Wendel. A problem in geometric probability. *Math. Scand.*, 11:109–111, 1962.

48 Günter M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

A Note on the No-$(d+2)$-On-a-Sphere Problem

Andrew Suk
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Ethan Patrick White
Department of Mathematics, University of Illinois Urbana-Champaign, Urbana, IL, USA

Abstract

For fixed $d \geq 3$, we construct subsets of the d-dimensional lattice cube $[n]^d$ of size $n^{\frac{3}{d+1}-o(1)}$ with no $d+2$ points on a sphere or a hyperplane. This improves the previously best known bound of $\Omega(n^{\frac{1}{d-1}})$ due to Thiele from 1995.

2012 ACM Subject Classification Mathematics of computing → Combinatorics

Keywords and phrases General position, no-four-on-a-cirle, d-dimensional lattice cube

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.76

Related Version *Full Version*: https://arxiv.org/abs/2412.02866

Funding *Andrew Suk*: Supported by NSF CAREER award DMS-1800746, NSF award DMS-1952786, and NSF grant DMS-2246847.
Ethan Patrick White: supported by an NSERC PDF Award.

1 Introduction

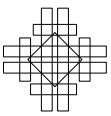
The famous *no-three-in-line problem*, raised by Dudeney [5] in 1917 in the special case $n = 8$, asks if it is possible to select $2n$ points from the lattice square $[n]^2$ such that no three are collinear. Clearly, one cannot do better than $2n$ as $[n]^2$ can be covered by n lines. For $n \leq 52$, many authors have published solutions to this problem obtaining the bound of $2n$ (e.g. see [6]). However, for large values of n, the best known lower bound is due to Hall et al. [9] which contains at least $\left(\frac{3}{2} - \varepsilon\right) n$ points, for any $\varepsilon > 0$ and n sufficiently large.

The similar *no-four-on-a-circle* problem, raised by Erdős and Purdy in 1981 (see [8]), asks to determine the maximum number of points that can be selected from $[n]^2$ such that no four are on a circle. Here, a line is also considered to be a *degenerate* circle. Again, we have the trivial upper bound of $3n$ as any vertical line must contain at most three points. This upper bound was improved by Thiele [19], who showed that at most $\frac{5}{2}n - \frac{3}{2}$ points can be selected, and moreover, he gave a construction of $\frac{n}{4}$ points from $[n]^2$ with no four on a circle (or a line).

In this paper, we study the no-four-on-a-circle problem in higher dimensions (Problem 4 in Chapter 10 of [3]). A *k-sphere* is a k-dimensional sphere. Thus, a 0-sphere is a pair of points, a 1-sphere is a circle, and etc. For simplicity, we will simply use the term *sphere* when referring to a $(d-1)$-dimensional sphere in $\mathbb{R}^d$. Again, the maximum number of points that can be selected from the d-dimensional lattice cube $[n]^d$ with no $d+2$ points on a sphere or a hyperplane is at most $(d+1)n$ since we can cover $[n]^d$ with n hyperplanes. In the other direction, Thiele [18] showed that one can select $\Omega(n^{\frac{1}{d-1}})$ points from $[n]^d$ with no $d+2$ points on a sphere or a hyperplane, providing the first non-trivial construction for this problem. In this note, we prove the following.

▶ **Theorem 1.** *Let $d \geq 3$ be a positive integer. Then there is a subset of the d-dimensional lattice cube $[n]^d$ with $n^{\frac{3}{d+1}-o(1)}$ points with no $d+2$ members on a sphere or a hyperplane.*

© Andrew Suk and Ethan Patrick White;
licensed under Creative Commons License CC-BY 4.0



While it is possible that one can improve this bound to $\Omega(n)$, we make the following more modest conjecture.

▶ **Conjecture 2.** *Let $d \geq 3$ be a positive integer. Then there is a subset of the d-dimensional lattice cube $[n]^d$ with $\Omega(n^{\frac{d}{d+1}})$ points with no $d+2$ members on a sphere or a hyperplane.*

Our paper is organized as follows. In Section 2, we recall several results from VC-dimension theory that will be used in the proof of Theorem 1. In Section 3, we prove Theorem 1. We conclude our paper with several remarks and related problems. For sake of clarity, we systemically remove floor and ceilings whenever they are not crucial. All logarithms are in based 2 unless stated otherwise.

2 VC-dimension theory

In this section, we recall several results from VC-dimension theory that will be used in the proof of Theorem 1. First we introduce some terminology.

Let $G = (P, Q, E)$ be a bipartite graph with independent sets P, Q and edges $E \subset P \times Q$. For $q \in Q$ we define the neighbourhood $N_Q(q) = \{p \in P \colon (p, q) \in E\}$. Define the set-system $\mathcal{F} = \{N_G(q) \colon q \in Q\}$ with ground set P. We say a set $S \subset P$ is *shattered* by $\mathcal{F}$ if for every subset $B \subset S$, there is a set $A \in \mathcal{F}$ such that $A \cap S = B$. The *Vapnik-Chervonenkis dimension* (VC-dimension) of $(P, \mathcal{F})$ is the largest integer d for which there exists a subset $S \subset P$, with $|S| = d$, that is shattered by $\mathcal{F}$.

The *primal shatter function* of $(P, \mathcal{F})$ is defined as

$$\pi_{\mathcal{F}}(z) = \max_{P' \subset P, |P'|=z} |\{A \cap P' \colon A \in \mathcal{F}\}|.$$

The well-known Sauer-Shelah lemma [14, 16] states that if d_0 is the VC-dimension of $\mathcal{F}$, then

$$\pi_{\mathcal{F}}(z) \leq \sum_{i=0}^{d_0} \binom{z}{i}. \tag{1}$$

We will need the following result due to Fox, Pach, Sheffer, Suk, and Zahl.

▶ **Theorem 3** ([7])**.** *Let $G = (P, Q, E)$ be a bipartite graph with $|P| = m$ and $|Q| = n$, such that the set system $\mathcal{F}_1 = \{N(q) \colon q \in Q\}$, with ground set P, satisfies $\pi_{\mathcal{F}_1}(z) \leq cz^{d_0}$ for all z. Then if G is $K_{t,t}$-free, we have*

$$|E(G)| \leq c_1(mn^{1-1/d_0} + n),$$

where $c_1 = c_1(c, d_0, t)$. In particular, $c_1 \leq 10 \cdot ct^{2d_0+1}(d_0 \log d_0)^{d_0}$.

A subset $A \subset [n]^d$ is a *maximal spherical set of* $[n]^d$ if all points in A lie on a single sphere S in $\mathbb{R}^d$, and no point of $[n]^d$ can be added to A while keeping all points on the sphere S. Let $\mathcal{S}_{n,d}$ denote the set-system of all maximal spherical sets of $[n]^d$, with ground set $[n]^d$.

▶ **Lemma 4.** *For $d \geq 2$, the VC-dimension of the set-system $([n]^d, \mathcal{S}_{n,d})$ is at most $d+1$.*

Proof. We proceed by induction on d. The base case $d = 1$ is trivial since a 0-sphere consists of 2 points. For the inductive step, assume the statement holds for $d' < d$. For sake of contradiction, suppose that a set Q of $d+2$ points in $[n]^d$ is shattered by $\mathcal{S}_{n,d}$.

Case 1. Suppose Q is in general position, that is, no $d+1$ members in Q lie on a hyperplane. By linear algebra, any $d+1$ points in Q determines a unique sphere in $\mathbb{R}^d$. However, since Q is shattered by $\mathcal{S}_{n,d}$, there is a sphere that contains Q. Hence, there is no sphere that contains exactly $d+1$ points from Q. Contradiction.

Case 2. Suppose Q contains a $(d+1)$-tuple Q' that lies on a hyperplane h in $\mathbb{R}^d$. Since the intersection of a sphere with h is a $(d-2)$-sphere in h, by induction on d, Q' cannot be shattered by $\mathcal{S}_{n,d}$. Hence, Q cannot be shattered by $\mathcal{S}_{n,d}$, which is a contradiction. ◀

3 Proof of Theorem 1

In this section, we prove Theorem 1. First, let us recall several results. The first is the well-known Chernoff inequality (see [11, Theorem 2.8]).

▶ **Lemma 5** (Chernoff's inequality). *Let $X_1, \dots, X_n$ be independent random variables such that $\mathbb{P}(X_i = 1) = q$ and $\mathbb{P}(X_i = 0) = 1-q$, and let $X = \sum_i X_i$. Then for $0 < \delta < 1$, we have*

$$\mathbb{P}(X \geq (1+\delta)qn) \leq e^{-\frac{\delta^2 qn}{3}},$$

and

$$\mathbb{P}(X \leq (1-\delta)qn) \leq e^{-\frac{\delta^2 qn}{3}}.$$

Next, we estimate the number of points from $[n]^d$ that can lie on a sphere or a hyperplane. For the latter, we will use the following result due to Balogh and White.

▶ **Lemma 6** ([2]). *Let $d \geq 2$ be a positive integer, and $a_1, \dots, a_d \in \mathbb{Z}$ not be all zero with greatest common denominator 1. Let $s = \max_i\{|a_i|\}$, $n \geq s$ be a positive integer, and set*

$$\mathcal{L} = \{(x_1, \dots, x_d) \in \mathbb{Z}^d \colon \sum_{i=1}^{d} a_i x_i = 0\}.$$

If $\mathcal{L} \cap [n]^d$ spans a $(d-1)$-dimensional subspace, then $|\mathcal{L} \cap [n]^d| \leq 3^d n^{d-1}/s$.

▶ **Lemma 7.** *The number of $(d+2)$-tuples in $[n]^d$ that lie on a common hyperplane is at most $O(n^{d^2+d-1})$.*

Proof. By the lemma above, the number of $(d+1)$-tuples in $[n]^d$ that lie on a hyperplane which passes through the origin is at most

$$\sum_s \left(\frac{n^{d-1}}{s}\right)^{d+1} s^{d-1} = O(n^{d^2-1}).$$

By symmetry, for any fixed point $p \in [n]^d$, the number of $(d+1)$-tuples in $[n]^d$ that lies on a hyperplane which passes through p is at most $O(n^{d^2-1})$. After summing over all points in $[n]^d$, the statement follows. Note that Lemma 7 applies to hyperplanes whose intersection with $[n]^d$ spans a $(d-1)$-dimensional subspace. We can assume that any $(d+2)$-tuple in $[n]^d$ lying on a common hyperplane meets this requirement by adding other points from $[n]^d$ if needed. ◀

Finally, we will use the following result due to Sheffer that bounds the number of points from $[n]^d$ that lies on a k-sphere in $\mathbb{R}^d$, for $k \leq d-1$. See also [10, Sec. 11.2, Equation 11.9], [17, Theorem 2].

▶ **Lemma 8** ([15], Lemma 3.2). *Let $d \geq 2$ and $k \leq d-1$. Then there is a positive constant $c = c(d)$ such that every k-sphere in $\mathbb{R}^d$ contains at most $n^{k-1+c/\log\log n}$ points from $[n]^d$.*

Before we prove Theorem 1, let us give a brief outline of the argument. First, we use the probabilistic method and Lemmas 7 and 8 to obtain a large subset $A \subset [n]^d$ such that very few members of A lie on a $(d-2)$-sphere or a hyperplane in $\mathbb{R}^d$. We then apply Theorem 3 and ideas from incidence geometry to estimate the number of r-tuples in A that lie on a common sphere. Finally, we apply the deletion method to find a large subset $A'' \subset A$ such that no $d+2$ members of A'' lie on a sphere or a hyperplane. We now flesh out all of the details of the proof.

Proof of Theorem 1. Without loss of generality, we can assume that n is a power of 2. Indeed, otherwise we can find an integer $n' < n$ that is a power of 2 such that $n' > n/2$, and apply the arguments below to the subcube $[n']^d$. This would only change the hidden constant in the $o(1)$ term in the exponent. Moreover, we will assume that n is sufficiently large.

Fix a positive integer $D < n$ that will be determined later, such that D divides n. We partition the d-dimensional cube $[n]^d$ into D^d smaller cubes $Q_1, \ldots, Q_{D^d}$, where each Q_i is of the form

$$\left\{i_1\frac{n}{D}+1, \ldots, (i_1+1)\frac{n}{D}\right\} \times \cdots \times \left\{i_d\frac{n}{D}+1, \ldots, (i_d+1)\frac{n}{D}\right\}.$$

where $i_1, \ldots, i_d \in \{0, \ldots, D-1\}$. Hence, $|Q_i| = (n/D)^d$.

Consider a random subset $A \subset [n]^d$ obtained by selecting each point in $[n]^d$ independently with probability n^{3-d}. Set $P_i = Q_i \cap A$, for $i = 1, \ldots, D^d$. Let $\mathcal{W}$ be the event that the subset $A \subset [n]^d$ satisfies the following properties.

1. $n^3/2 \leq |A| \leq 2n^3$.
2. $n^3 D^{-d}/2 \leq |P_j| \leq 2n^3 D^{-d}$ for all $1 \leq j \leq D^d$.
3. Every $(d-2)$-sphere contains less than $n^{c_1/\log\log n}$ points of A, where $c_1 = c_1(d)$.
4. The number $(d+2)$-tuples in A that lie on a common hyperplane is at most $c_2 n^{2d+5}$, where $c_2 = c_2(d)$.

By Lemma 5 and Markov's inequality, event $\mathcal{W}$ holds with probability at least $1/2$. Indeed, by Lemma 5, the probability that $|A| > 2n^3$ or $|A| < n^3/2$ is at most $e^{-\frac{n^3}{3}} + e^{-\frac{n^3}{12}}$. Thus, the first property holds with high probability, that is, with probability tending to 1 as n approaches infinity. A similar argument follows for the second property. For the third property, let us fix a $(d-2)$-sphere S in $\mathbb{R}^d$. By Lemma 8, there is a constant $c = c(d)$ such that S contains at most $n^{d-3+c/\log\log n}$ points from $[n]^d$. By Lemma 5,

$$\mathbb{P}[|S \cap A| \geq 2n^{c/\log\log n}] \leq e^{-\frac{n^{c/\log\log n}}{3}}.$$

Since there are at most n^{d^2} $(d-2)$-spheres in $\mathbb{R}^d$ with at least d points from $[n]^d$, the union bound implies that the third property holds with high probability. For the fourth property, let X denote the number $(d+2)$-tuples in A that lie on a common hyperplane. By Lemma 7, we have $\mathbb{E}[X] \leq c' n^{2d+5}$, where $c' = c'(d)$. By Markov's inequality (see [1]), we have

$$\mathbb{P}[X > 10c'n^{2d+5}] < \frac{\mathbb{E}[X]}{10c'n^{2d+5}} < \frac{1}{10}.$$

Putting everything together, and setting c_1, c_2 sufficiently large, event $\mathcal{W}$ holds with probability at least $1/2$.

Thus, let us fix $A \subset [n]^d$ with the four properties described above, and set $t = 2n^{c_3/\log\log n}$, where $c_3 = \max\{c_1, c_2\}$. Let $\mathcal{S}$ be the collection of spheres in $\mathbb{R}^d$, such that each sphere in $\mathcal{S}$ contains at least $d+1$ points from A in general position. Hence, $|\mathcal{S}| = O(|A|^{d+1})$. Let $\mathcal{S}_j \subset \mathcal{S}$ be the set of spheres in $\mathcal{S}$ that contains at least one point of $P_j = Q_j \cap A$. Let $G_j = (P_j, \mathcal{S}_j, E_j)$ be the bipartite incidence graph between P_j and $\mathcal{S}_j$. Since the intersection of two distinct spheres in $\mathbb{R}^d$ is a $(d-2)$-sphere, and every $(d-2)$-sphere contains less than t points from A, each graph G_j is $K_{t,2}$-free.

Let $\mathcal{F}_j$ be the set system whose ground set is P_j, and whose sets are $S \cap P_j$, where $S \in \mathcal{S}_j$ and $|S \cap P_j| \geq t$. That is,

$$\mathcal{F}_j = \{S \cap P_j : S \in \mathcal{S}_j, |S \cap P_j| \geq t\}.$$

By Lemma 4, the VC-dimension of $\mathcal{F}_j$ is at most $d+1$. By inequality (1), we have

$$\pi_{\mathcal{F}_j}(z) = O(z^{d+1}).$$

Hence, we apply Theorem 3 with $t = n^{c_3/\log\log n}$ to conclude that

$$|E_j| \leq n^{c_4/\log\log n}\left(|P_j||\mathcal{S}_j|^{\frac{d}{d+1}} + |\mathcal{S}_j|\right),$$

where $c_4 = c_4(d)$.

Given the partition $[n]^d = Q_1 \cup \cdots \cup Q_{D^d}$ described above, we say that a sphere S *crosses* the subcube Q_i if $S \cap Q_i \neq \emptyset$.

▶ **Observation 9.** *For fixed $d \geq 2$, every sphere S in $\mathbb{R}^d$ crosses at most $c_5 D^{d-1}$ subcubes Q_i, where $c_5 = c_5(d)$.*

Proof. Let $c_5 = c_5(d)$ be a large constant that depends only on d. We will determine c_5 later. For sake of contradiction, suppose there is a sphere S in $\mathbb{R}^d$ the crosses more than $c_5 D^{d-1}$ subcubes Q_i. Note that we will not make any serious attempts to optimize the constant c_5.[1]

Let $T \subset [n]^d \cap S$ be a subset of points $[n]^d \cap S$ obtained by selecting 1 point from each subcube Q_i that S crosses. Hence, $|T| \geq c_5 D^{d-1}$. Let $\mathcal{G} = (T, E)$ be an auxiliary graph on T, where two points in T are adjacent if their distance is less than n/D. Since two points in T have distance less than n/D only if they lie in adjacent subcubes Q_i and Q_j, this implies that $\mathcal{G}$ has maximum degree $3^d - 1$. By Turan's theorem, we can find a subset $T' \subset T$, such that the minimum distance between any two points in T' is at least n/D. Moreover, $|T'| \geq (c_5/3^d)D^{d-1}$.

For each point $p \in T'$, consider the spherical cap C_p obtained by intersecting S with the ball B_p centered at p with radius $n/(2D)$. Since the minimum distance among the points in T' is at least n/D, the collection of spherical caps corresponding to the points in T' will have pairwise disjoint interiors. Moreover, since each spherical cap arises from a ball whose center is on S with radius $n/(2D)$, each spherical cap will have area at least $c_6(n/D)^{d-1}$ where $c_6 = c_6(d)$. Hence, the total area of all of the spherical caps corresponding to the points in T' is at least

$$|T'|c_6(n/D)^{d-1} \geq \frac{c_5 c_6}{3^d} n^{d-1}.$$

[1] A careful calculation shows that one can take $c_5 = 2^{O(d)}$. Another proof follows from a simple inductive argument with $c_5 = d^{O(d)}$.

On the other hand, all of the spherical caps on S will lie inside of a "large" ball B with radius dn. Hence, the area of all of the spherical caps is at most the surface area of the ball B with radius dn. Since the surface area of B is at most $c_7 n^{d-1}$, where $c_7 = c_7(d)$, we have

$$c_7 n^{d-1} \geq \frac{c_5 c_6}{3^d} n^{d-1}.$$

By setting c_5 sufficiently large, we have a contradiction. ◀

By the observation above, $\sum_{j=1}^{D^d} |\mathcal{S}_j| = O(D^{d-1}|\mathcal{S}|)$. Putting all of these bounds together and applying Jensen's inequality, the number of incidences between A and $\mathcal{S}$ is at most

$$\begin{aligned}
\sum_{j=1}^{D^d} |E_j| &\leq n^{c_4/\log\log n} \sum_{j=1}^{D^d} \left(|P_j||\mathcal{S}_j|^{\frac{d}{d+1}} + |\mathcal{S}_j| \right) \\
&\leq 2n^{3+c_4/\log\log n} D^{-d} \sum_{j=1}^{D^d} |\mathcal{S}_j|^{\frac{d}{d+1}} + n^{c_4/\log\log n} \sum_{j=1}^{D^d} |\mathcal{S}_j| \\
&\leq 2n^{3+c_4/\log\log n} D^{-d} D^{\frac{d}{d+1}} \left(\sum_{j=1}^{D^d} |\mathcal{S}_j| \right)^{\frac{d}{d+1}} + c_5 D^{d-1} n^{c_4/\log\log n} |\mathcal{S}| \\
&\leq 2c_5^{\frac{d}{d+1}} n^{3+c_4/\log\log n} D^{-\frac{d}{d+1}} |\mathcal{S}|^{\frac{d}{d+1}} + c_5 D^{d-1} n^{c_4/\log\log n} |\mathcal{S}|.
\end{aligned}$$

Set D to be a power of 2 such that

$$n^{\frac{3(d+1)}{d^2+d-1}} |\mathcal{S}|^{-\frac{1}{d^2+d-1}} \leq D \leq 2n^{\frac{3(d+1)}{d^2+d-1}} |\mathcal{S}|^{-\frac{1}{d^2+d-1}}.$$

Note that $|\mathcal{S}| \leq |A|^{d+1}$. From above, the number of incidences between $\mathcal{S}$ and A is at most

$$n^{\frac{3(d^2-1)}{d^2+d-1} + c_8/\log\log n} |\mathcal{S}|^{\frac{d^2}{d^2+d-1}},$$

where $c_8 = c_8(d)$. Let $\mathcal{S}_r \subset \mathcal{S}$ denote the set of r-rich spheres in $\mathcal{S}$, that is, the set of spheres in $\mathcal{S}$ with at least r points from A. Then we have

$$r|\mathcal{S}_r| \leq n^{\frac{3(d^2-1)}{d^2+d-1} + c_8/\log\log n} |\mathcal{S}_r|^{\frac{d^2}{d^2+d-1}},$$

which implies

$$|\mathcal{S}_r| \leq n^{3(d+1)+c_8/\log\log n} r^{-\frac{d^2+d-1}{d-1}}.$$

Let $\mathcal{T}_i \subset \mathcal{S}$ be the set of spheres in $\mathcal{S}$ with at least 2^i points from A, and at most 2^{i+1} from A. Hence

$$|\mathcal{T}_i| \leq |\mathcal{S}_{2^i}| = n^{3(d+1)+c_8/\log\log n} 2^{-i\frac{d^2+d-1}{d-1}}.$$

Therefore, the number of $(d+2)$-tuples of A that lie on a common sphere $S \in \mathcal{S}$ is at most

$$\sum_{i=1}^{\log|A|} |\mathcal{T}_i| \binom{2^{i+1}}{d+2} \leq n^{3(d+1)+c_8/\log\log n} \sum_{i=1}^{\log|A|} 2^{-i\frac{1}{d-1}} \leq n^{3(d+1)+c_8/\log\log n}.$$

We now consider a random subset $A' \subset A$ obtained by selecting each point in A independently with probability p, where p will be determined later. Let X be the number of $(d+2)$-tuples of A' that lie on a common sphere or a hyperplane. Then, by linearity of expectation, we have

$$\mathbb{E}[X] \leq n^{3(d+1)+c_8/\log\log n}p^{d+2} + O(n^{2d+5})p^{d+2} \leq n^{3(d+1)+c_9/\log\log n}p^{d+2},$$

and

$$\mathbb{E}[|A'|] = p|A| \geq n^3 p/2,$$

where $c_9 = c_9(d)$. Hence, there is a $c_{10} = c_{10}(d)$ such that for $p = n^{-3d/(d+1)-c_{10}/\log\log n}$, we have $\mathbb{E}[X] < \mathbb{E}[|A'|]/4$. By Lemma 5, there is a subset $A' \subset A$ of size at least $p|A|/2$, such that A' contains at most $p|A|/4$ $(d+2)$-tuples on a sphere or a hyperplane. By deleting one point from each $(d+2)$-tuple on a sphere or a hyperplane, we obtain a subset $A'' \subset A'$ of size at least

$$|A|p/4 \geq n^{\frac{3}{d+1}-2c_{10}/\log\log n} = n^{\frac{3}{d+1}-o(1)},$$

such that no $d+2$ members in A'' lie on a sphere or a hyperplane. ◀

4 Concluding remarks

Several authors [13, 12, 4] observed that if n is prime, then by selecting n points from $[n]^d$ on the "modular moment curve" $(x, x^2, \ldots, x^d) \mod n$, for $x = 1, \ldots, n$, no $d+1$ points will lie on a hyperplane, and moreover, no $2d+1$ points lie on a sphere. Unfortunately, this construction may contain $(d+2)$-tuples on a sphere. Nevertheless, one can make the following simple observation.

▶ **Theorem 10.** *Let $d \geq 2$ be a fixed positive integer. Then there is a subset of the d-dimensional lattice cube $[n]^d$ of size $\Omega(n)$ with no $d+1$ points on a hyperplane, and no $2d$ points on a sphere.*

Proof. Let n be prime, and let $A \subset [n]^d$ be points from the d-dimensional cube on the moment curve $(x, x^2, \ldots, x^d) \mod n$, for $1 \leq x \leq \lfloor n/(4d) \rfloor$. Hence, $|A| = \lfloor n/(4d) \rfloor$. For sake of contradiction, suppose there is a sphere S in $\mathbb{R}^d$, such that S contains $2d$ points from A. Note that this means S contains $2d$ of $[n]^d$ when viewed as a sphere in $\mathbb{F}_n^d$. Let the center of $S \subset \mathbb{F}_n^d$ be $(c_1, \ldots, c_d)$ and radius r, such that S contains $2d$ points from A. Then we have $2d$ solutions $s_1, s_2, \ldots, s_{2d}$ to the equation

$$(x-c_1)^2 + (x^2-c_2)^2 + \cdots (x^d - c_d)^2 - r^2 = 0 \mod n.$$

On the other hand, by the division algorithm, we have

$$(x-c_1)^2 + (x^2-c_2)^2 + \cdots (x^d - c_d)^2 - r^2 = (x-s_1)(x-s_2)\cdots(x-s_{2d}) \mod n,$$

which implies that $s_1 + \cdots + s_{2d} = 0 \mod n$ as the coefficient of x^{2d-1} is $0 \mod n$. However, $s_i \leq n/(4d)$, which implies $s_1 + \cdots + s_{2d} \neq 0 \mod n$, contradiction. Hence, no $2d$ points in A lie on a sphere and no $d+1$ points lie on a hyperplane. If n is not prime, we apply Bertrand's postulate to obtain a prime $n' < n$ such that $n' > n/2$, and apply the argument above to $[n']^d$. ◀

Another natural question is to determine the maximum number of points that can be selected from $[n]^d$ with no $d+2$ points on a sphere, but allowing many points on a hyperplane. Since each point from $[n]^d$ lies on a sphere centered at the origin with radius $\sqrt{t}$, where $t = 1, 2, \ldots, dn^2$, we have an upper bound of $d(d+1)n^2$ for this problem. Using Lemma 8 and the probabilistic method, one can show the existence of $n^{2-4/(d+1)-o(1)}$ points from $[n]^2 \subset [n]^d$ with no $d+2$ on a sphere.

References

1 N. Alon and J. Spencer. *The Probabilistic Method, 3rd edn.* Wiley Interscience, 2008.
2 J. Balogh and E. P. White. Grid drawings of graphs in three-dimensions. *arxiv:2404.02369*, 2024.
3 P. Brass, W. Moser, and J. Pach. *Research Problems in Discrete Geometry.* Springer-Verlag, Berlin, Germany, 2005.
4 P. Braß and C. Knauer. On counting point-hyperplane incidences. *Comput. Geom.*, 25:13–20, 2003. doi:10.1016/S0925-7721(02)00127-X.
5 H.E. Dudeney. *Amusements in Mathematics.* Nelson, London, 1917.
6 A. Flammenkamp. Progress in the no-three-in-line problem, II. *J. Combin. Theory Ser. A*, 81:108–113, 1998. doi:10.1006/JCTA.1997.2829.
7 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Eur. Math. Soc.*, 19:1785–1810, 2017.
8 R. K. Guy. *Unsolved Problems in Number Theory.* Springer-Verlag, New York, 2004.
9 R. Hall, T. Jackson, A. Sudbery, and K. Wild. Some advances in the no-three-in-line problem. *J. Combin. Theory Ser. A*, 18:336–341, 1975. doi:10.1016/0097-3165(75)90043-6.
10 H. Iwaniec. *Topics in Classical Automorphic Forms, Grad. Stud. Math.* Amer. Math. Soc., Providence, RI, 1997.
11 S. Janson, T. Luczak, and A. Rucinski. *Random Graphs.* John Wiley & Sons, 2011.
12 H. Lefmann. Extensions of the no-three-in-line problem. *preprint*, 2012.
13 K. Roth. On a problem of Heilbronn. *J. Lond. Math. Soc.*, 1:198–204, 1951.
14 N. Sauer. On the density of families of sets. *J. of Combin. Theory Ser. A*, 12:145–147, 1972.
15 A. Sheffer. Lower bounds for incidences with hypersurfaces. *Discrete Anal.*, 2016.
16 S. Shelah. A combinatorial problem, stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41:247–261, 1972.
17 V. Shelestunova. *Upper bounds for the number of integral points on quadratic curves and surfaces.* Dissertation, Univ. of Waterloo, 2010.
18 T. Thiele. *Geometric Selection Problems and Hypergraphs.* Dissertation, FU Berlin, 1995.
19 T. Thiele. The on-four-on-circle problem. *J. of Combin. Theory Ser. A*, 71:332–334, 1995.

Dynamic Maximum Depth of Geometric Objects

Subhash Suri
Department of Computer Science, University of California, Santa Barbara, CA, USA

Jie Xue
New York University Shanghai, China

Xiongxin Yang
New York University Shanghai, China

Jiumu Zhu
New York University Shanghai, China

Abstract

Given a set of geometric objects in the plane (rectangles, squares, disks etc.), its maximum *depth* (or *geometric clique*) is the largest number of objects with a common intersection. In this paper, we present data structures for dynamically maintaining the maximum depth under insertions and deletions of geometric objects, with sublinear update time. We achieve the following results:

- a $\frac{1}{k}$-approximate dynamic maximum-depth data structure for (axis-parallel) rectangles with $O(n^{1/(k+1)} \log n)$ amortized update time, for any fixed $k \in \mathbb{Z}^+$. In particular, when $k = 1$, this gives an exact data structure for rectangles with $O(\sqrt{n} \log n)$ amortized update time, almost matching the best known bound for the offline version of the problem.
- a $(\frac{1}{2} - \varepsilon)$-approximate dynamic maximum-depth data structure for disks with $n^{2/3} \log^{O(1)} n$ amortized update time, for any constant $\varepsilon > 0$. Having exact data structures for disks with sublinear update time is unlikely, since the static maximum-depth problem for disks is 3SUM-hard and thus does not admit subquadratic-time algorithms.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases dynamic algorithms, maximum depth

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.77

1 Introduction

We consider the problem of *dynamically* maintaining a set $\mathcal{O}$ of geometric objects (rectangles, squares, disks etc.) under insertions and deletions so that at any instant we can find its maximum depth. The *maximum depth* of $\mathcal{O}$ is the largest depth of any point in the plane, where the depth of a point p, denoted $\mathsf{dep}(p, \mathcal{O})$, is defined as the number of objects in $\mathcal{O}$ containing p. The maximum depth of the set $\mathcal{O}$, written as $\mathsf{dep}(\mathcal{O}) = \max_{p \in \mathbb{R}^d} \mathsf{dep}(p, \mathcal{O})$, is equivalent to its *maximum geometric clique*, namely, the largest subset of objects in $\mathcal{O}$ with a nonempty common intersection. It is easy to see that for axis-parallel rectangles and squares in the plane the geometric cliques correspond to graph-theoretic cliques in the *intersection graph* of $\mathcal{O}$ because the Helly number of rectangles is two. This equivalence, however, does not hold for general geometric objects (even if the objects are convex), where the size of the maximum geometric clique can be smaller than the size of the maximum clique in the intersection graph. Nevertheless, in some cases, the maximum depth is closely related to the maximum clique size in the intersection graph. For example, it is well-known that for disks, the maximum depth is at least $\frac{1}{4}$ of the size of the maximum clique [8].

The problem of computing a maximum geometric clique for simple planar shapes such as rectangles, squares, or disks (and their counterparts in d-dimensions) is well-studied in computational geometry, and several efficient algorithms are known [14, 2, 9, 12]. The

© Subhash Suri, Jie Xue, Xiongxin Yang, and Jiumu Zhu;
licensed under Creative Commons License CC-BY 4.0

primary focus of our work is to *dynamically* maintain the maximum depth as well as a point p^* that realizes this depth, i.e., $\mathsf{dep}(p^*, \mathcal{O}) = \mathsf{dep}(\mathcal{O})$. To the best of our knowledge, the fully dynamic version of the maximum-depth problem has not been studied before, except for the case of 1-dimensional intervals on the line, where it can be easily solved optimally with $O(\log n)$ update time [14].

The maximum-depth problem in two dimensions provides an interesting setting for dynamic data structures: unlike the NP-complete problems such as set cover, hitting set or independent set for rectangles, which have been the focus of recent work in dynamic data structures [13, 10, 1, 7, 11, 6], the *static* maximum depth problem for rectangles is polynomial-time solvable [14]. Yet, in attempting to design a dynamic data structure for the maximum depth, we face challenges similar to those in set cover etc, and sometimes have to settle for *approximation* to achieve fast update time. For example, for the maximum depth of (axis-parallel) rectangles, it is very difficult to achieve an $O(n^c)$ update time for $c < 1/2$, since the best known static algorithm requires $n^{1.5}/\log^{O(1)} n$ time [9]. Similarly, for the maximum depth of disks, a sublinear update time is unlikely even in the insertion-only setting (if we want exact depth), because the static problem is known to be 3SUM-hard [3] and thus does not admit a (truly) subquadratic-time algorithm by conjecture.

Another challenge for dynamic maximum depth is the non-decomposability of the problem. Specifically, even if we can partition the set $\mathcal{O}$ of geometric objects as $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$, the value of $\mathsf{dep}(\mathcal{O})$ cannot be computed from $\mathsf{dep}(\mathcal{O}_1)$ and $\mathsf{dep}(\mathcal{O}_2)$. This non-decomposability makes many techniques in data structure designing inapplicable to the maximum-depth problem.

1.1 Our results

In this paper, we study the dynamic maximum-depth problem for (axis-parallel) rectangles and disks, designing exact and approximate dynamic maximum-depth data structures with sublinear update time. Formally, a dynamic maximum-depth data structure maintains a set $\mathcal{O}$ of geometric objects in $\mathbb{R}^2$ (or more generally $\mathbb{R}^d$) under insertions and deletions, and can report a point $p^* \in \mathbb{R}^2$ satisfying $\mathsf{dep}(p^*, \mathcal{O}) = \mathsf{dep}(\mathcal{O})$ together with the value of $\mathsf{dep}(\mathcal{O})$.

Our main result for rectangles is the following theorem.

▶ **Theorem 1.** *For every $k \in \mathbb{Z}^+$, there exists a $\frac{1}{k}$-approximate dynamic maximum-depth data structure for axis-parallel rectangles with $O(n^{1/(k+1)} \log n)$ amortized update time.*

When $k = 1$, the above theorem gives us an exact data structure for dynamic maximum depth of rectangles, with $O(\sqrt{n} \log n)$ update time. Up to logarithmic factors, this bound matches the update time of the best known *offline* dynamic data structure for the problem [9], which is $O((\sqrt{n}/\sqrt{\log n}) \cdot \log^{3/2} \log n)$. Indeed, $\widetilde{O}(\sqrt{n})$ update time may be best one can expect, even in the insertion-only version, because it is a longstanding open problem whether the static maximum-depth problem for rectangles can be solved in $O(n^{1.5-\varepsilon})$ time. The data structure in Theorem 1 can also be used to dynamically maintain the maximum clique of a rectangle graph with the same update time, because the maximum clique size of a rectangle intersection graph equals the maximum depth of the rectangles.

Our main result for disks is the following theorem.

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, there exists a $(\frac{1}{2} - \varepsilon)$-approximate dynamic maximum-depth data structure for disks with $n^{2/3} \log^{O(1)} n$ amortized update time.*

The static maximum-depth problem for disks is known to be 3SUM-hard [3] with a conjectured lower bound of $\widetilde{\Omega}(n^2)$. It is, therefore, unlikely that an exact dynamic maximum-depth data structure for disks with (truly) sublinear update time exists, even in the insertion-only setting.

Theorem 2 also gives a data structure for dynamically maintaining a $(\frac{1}{8}-\varepsilon)$-approximation of the maximum clique size of a disk graph, with the same update time, because the maximum clique size of a disk graph is at most 4 times the maximum depth of the disks [8]. For convenience, we call such a data structure a *dynamic maximum-clique data structure* for disk graphs.

▶ **Corollary 3.** *For any constant $\varepsilon > 0$, there exists a $(\frac{1}{8}-\varepsilon)$-approximate dynamic maximum-clique data structure for disk graphs with $n^{2/3}\log^{O(1)} n$ amortized update time.*

The key idea underlying the data structure of Theorem 2 is to reduce the original problem to dynamic *discrete* maximum depth (for disks), by introducing a $\frac{1}{2}-\alpha$ multiplicative error. In the discrete version of the problem, besides a dynamic set $\mathcal{D}$ of disks, we also have a dynamic set S of points in $\mathbb{R}^2$, and what we want to maintain is $\max_{p\in S}\mathsf{dep}(p,\mathcal{D})$. This kind of reduction works for not only disks, but also general fat objects (where the multiplicative error depends on the fatness of the objects). We believe that this idea is of independent interest and may find applications in other problems as well.

1.2 Related Work

Dynamic data structures have been a focus of research in computational geometry from its very beginning, and there exists a vast literature. Since our work pertains to the dynamic geometric *optimization* problem, we limit our review to the narrow set of papers that have considered related problems. In recent years, a number of papers have addressed the problem of dynamically maintaining such functions as the minimum geometric set cover, minimum geometric hitting set, maximum geometric independent sets, etc., for simple geometric shapes such as rectangles, squares, disks, or their d-dimensional counterpart. In particular, Agarwal et al. [1] were the first to present sub-linear time data structures for maintaining approximate minimum geometric set covers (and hitting sets) of intervals in one dimension and unit squares in two dimensions. The update time bounds were improved and the results generalized to rectangles as well as weighted objects by Chan et al.[10, 11]. Henzinger et al. [13] consider the problem of approximately maintaining the maximum independent set of d-dimensional rectangles. Bhore and Chan [6] present dynamic data structures for a number of other geometric optimization problems such as minimum piercing sets, minimum vertex cover, maximum independent set of rectangles. Yildiz et al. [16] give a data structure to maintain Klee's measure dynamically in a discrete setting.

2 Preliminaries

In this section, we introduce the basic notions used throughout the paper.

Depth. Given a set $\mathcal{O}$ of geometric shapes in $\mathbb{R}^d$, the depth of a point p with respect to $\mathcal{O}$ is the number of objects in $\mathcal{O}$ that contain p. Formally, $\mathsf{dep}(p,\mathcal{O}) = |\{O\in\mathcal{O} : p\in O\}|$. The maximum depth of the set $\mathcal{O}$ is denoted by $\mathsf{dep}(\mathcal{O}) = \max_{p\in\mathbb{R}^d}\mathsf{dep}(p,\mathcal{O})$. We often use p^* to denote a point that maximizes the depth with respect to the set $\mathcal{O}$, i.e., $\mathsf{dep}(p^*,\mathcal{O}) = \mathsf{dep}(\mathcal{O})$, and $\mathcal{O}^*$ to denote a maximum geometric clique for $\mathcal{O}$, i.e., $\mathcal{O}^* = \{O\in\mathcal{O} : p^*\in O\}$.

Disk and fat object. For a disk D, $\mathsf{ctr}(D)$ and $\mathsf{rad}(D)$ denote its center and radius, respectively. This notation also holds for balls. In this paper, the fat object, which generalizes both rectangle and disk, is defined as follows: Let O be a *convex* object in $\mathbb{R}^d$. Let O_{in} be

the largest ball that is contained in O. We say O is α-*fat*, for some parameter $\alpha \geq 1$, if O_{out}, the ball that is concentric with O_{in} and has radius $\mathsf{rad}(O_{\text{out}}) = \alpha \cdot \mathsf{rad}(O_{\text{in}})$, contains O. We call the center of O_{in} the *center* of O and $\mathsf{rad}(O_{\text{in}})$ the *in-radius* of O.[1]

3 Dynamic maximum-depth for rectangles

In this section, we consider the dynamic maximum-depth problem for rectangles. We first describe a data structure for maintaining the exact maximum depth with $O(\sqrt{n}\log n)$ amortized update time and $O(n^{1.5}\log n)$ preprocessing time. This proves the base case of Theorem 1, namely, $k = 1$. We then inductively prove the theorem for all $k \in \mathbb{Z}^+$.

Let $\mathcal{Q}$ be a dynamic set of rectangles in $\mathbb{R}^d$, and denote by n the initial size of $\mathcal{Q}$. Our data structure applies the standard reconstruction technique. Specifically, we will reconstruct the data structure periodically: the $(i+1)$-th reconstruction will be done after handling $n_i/2$ operations following the i-th reconstruction, where n_i is the size of $\mathcal{Q}$ at the time of the i-th reconstruction. Thus, during the i-th period, i.e., the period between the i-th reconstruction and the $(i+1)$-th reconstruction, the size of $\mathcal{Q}$ is always $\Theta(n_i)$. Since our data structure will have an $O(n^{1.5}\log n)$ preprocessing time, the time cost for the $(i+1)$-th reconstruction is $O(n_i^{1.5}\log n_i)$, which can be amortized to the $n_i/2$ operations in the i-th period, which is $O(\sqrt{n_i}\log n_i)$ per operation. Because of the reconstruction, we only need to consider the first period, i.e., the first $n/2$ operations.

Let r be a parameter to be determined later. To construct our data structure, we first partition the plane into r horizontal slabs $L_1, \ldots, L_r$ using $r-1$ horizontal lines such that each slab contains (in its interior) at most $4n/r$ corners of rectangles in $\mathcal{Q}$. For each slab L_i, we construct a dynamic data structure $\mathcal{D}(L_i)$ that can maintain a point $p_i \in L_i$ that maximizes $\mathsf{dep}(p_i, \mathcal{Q})$, as well as the value of $\mathsf{dep}(p_i, \mathcal{Q})$, under insertions and deletions on $\mathcal{Q}$. We call $\mathcal{D}(L_1), \ldots, \mathcal{D}(L_r)$ *slab data structures* for convenience. The design of the slab data structures will be presented shortly. During the update, for each $i \in [r]$ whenever the number of rectangle corners contained in (the interior of) L_i reaches $8n/r$, we split L_i into two horizontal slabs L_i' and L_i'' each of which contains $4n/r$ corners, and replace $\mathcal{D}(L_i)$ with two new slab data structures $\mathcal{D}(L_i')$ and $\mathcal{D}(L_i'')$. In this way, we guarantee that the number of rectangle corners contained in each slab is always $O(n/r)$. We observe that slab splitting can happen at most $O(r)$ times (in the first period).

▶ **Fact 4.** *During the first t operations, the number of slab splittings is at most $2tr/n$.*

Proof. Let $\mathcal{L}$ be the set of all slabs occurred when processing the first t operations. Consider the insertions in the t operations. For each inserted rectangle Q, we charge it to the (at most) two slabs containing the corners of R when Q is inserted. Observe that a slab $L \in \mathcal{L}$ eventually splits only if it gets charged at least n/r times. Indeed, L contains $4n/r$ rectangle corners when it is created and contains $8n/r$ rectangle corners when it splits. Thus, from the time point that L is created to the time point that L splits, there are at least n/r inserted rectangles which have corners in L, which implies that L gets charged at least n/r times. Since there can be at most t inserted rectangles, the total number of charges is bounded by $2t$, which implies that the number of slab splitting is bounded by $2tr/n$. ◀

[1] There are different definitions of fat objects in different papers. One can easily show that our definition is equivalent to the others.

Clearly, using the information stored in the slab data structures, we can compute in $O(r)$ time a point p^* satisfying $\mathsf{dep}(p^*, \mathcal{Q}) = \mathsf{dep}(\mathcal{Q})$, as well as the value of $\mathsf{dep}(\mathcal{Q})$. More precisely, we just consider the points $p_1, \ldots, p_r$ stored in the slab data structures (and their depths with respect to $\mathcal{Q}$) and define p^* as the one that maximizes its depth with respect to $\mathcal{Q}$. Therefore, the remaining task now is to design efficient slab data structures.

The slab data structures. Consider a slab L. We want to maintain a point $p \in L$ that maximizes $\mathsf{dep}(p, \mathcal{Q})$, as well as the value of $\mathsf{dep}(p, \mathcal{Q})$. Clearly, we can ignore the rectangles in $\mathcal{Q}$ that are disjoint from L. Thus, in the following we assume that every rectangle in $\mathcal{Q}$ intersects L. We classify the rectangles in $\mathcal{Q}$ into two types: *long* rectangles and *short* rectangles. A rectangle is *long* if its four corners are all outside L and is *short* if at least one corner is inside (the interior of) L. Let $\mathcal{Q}_{\mathsf{long}} \subseteq \mathcal{Q}$ and $\mathcal{Q}_{\mathsf{short}} \subseteq \mathcal{Q}$ be the sets of long and short rectangles in $\mathcal{Q}$, respectively.

To maintain the maximum depth of $\mathcal{Q}$ inside L, our main idea is to reduce the problem to maintaining the maximum depth of *weighted* intervals. Specifically, we shall construct a set $\mathcal{I}$ of weighted intervals from $\mathcal{Q}$ such that the maximum depth of $\mathcal{Q}$ inside L is equal to the maximum (weighted) depth of $\mathcal{I}$. The construction is done as follows. For each long rectangle $Q = [x^-, x^+] \times [y^-, y^+] \in \mathcal{Q}_{\mathsf{long}}$, we include in $\mathcal{I}$ the interval $[x^-, x^+]$ with weight 1. For the set $\mathcal{Q}_{\mathsf{short}}$ of short rectangles, we first define a *depth function* $f : \mathbb{R} \to \mathbb{N}$ as $f(r) = \max_{p \in \ell_r \cap L} \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}})$ where ℓ_r is the vertical line with equation $x = r$. One can easily observe that f is a piecewise-constant function with $O(|\mathcal{Q}_{\mathsf{short}}|)$ pieces, since the value of f only changes at the x-coordinates of the corners of the short rectangles. Therefore, we can partition the real line $\mathbb{R}$ into $s = O(|\mathcal{Q}_{\mathsf{short}}|)$ intervals $I_1, \ldots, I_s$ such that f is a constant function when restricted to each I_i. Also, for each I_i, there is a y-coordinate $y_i \in \mathbb{R}$ such that $I_i \times \{y_i\} \subseteq L$ and $f(x) = \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}})$ for any $x \in I_i$ and $p \in I_i \times \{y_i\}$. Define a function $g : \mathbb{R} \to \mathbb{R}$ as $g(x) = y_i$ if $x \in I_i$. We include $I_1, \ldots, I_s$ in $\mathcal{I}$ and set the weight of each I_i to be $f(x)$ for an arbitrary $x \in I_i$. (See Figure 1 for an illustration.)

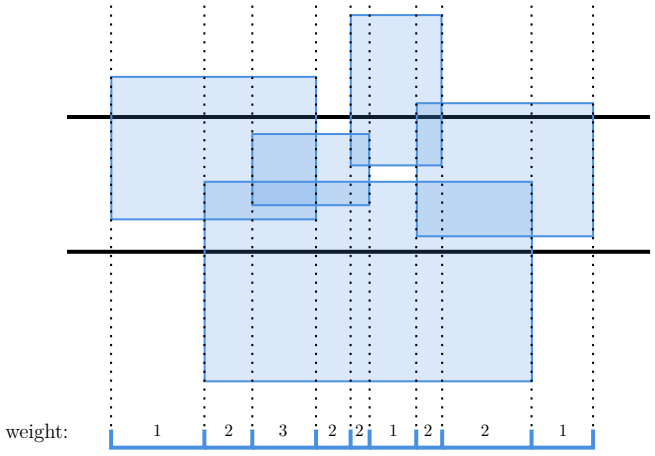


Figure 1 Construct of weighted intervals corresponding to $\mathcal{Q}_{\mathsf{short}}$.

We observe the following simple fact.

▶ **Fact 5.** *If $x^* \in \mathbb{R}$ satisfies that $\mathsf{dep}(x^*, \mathcal{I}) = \mathsf{dep}(\mathcal{I})$, the point $p^* = (x^*, g(x^*)) \in L$ satisfies $\mathsf{dep}(p^*, \mathcal{Q}) = \mathsf{dep}(x^*, \mathcal{I}) = \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$.*

Proof. We first prove that $\mathsf{dep}(p^*, \mathcal{Q}) = \mathsf{dep}(x^*, \mathcal{I})$. By definition, $\mathsf{dep}(p^*, \mathcal{Q}) = \mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{long}}) + \mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{short}})$. Note that a rectangle in $\mathcal{Q}_{\mathsf{long}}$ contains p^* iff its corresponding weight-1 interval in $\mathcal{I}$ contains x^*. So x^* is contained in exactly $\mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{long}})$ weight-1 intervals corresponding to the rectangles in $\mathcal{Q}_{\mathsf{long}}$. Meanwhile, the total weight of the intervals containing x^* which correspond to the rectangles in $\mathcal{Q}_{\mathsf{short}}$ is equal to $f(x^*)$. It follows that $\mathsf{dep}(x^*, \mathcal{I}) = \mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{long}}) + f(x^*)$. As $p^* = (x^*, g(x^*))$, we have $\mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{short}}) = f(x^*)$ by the definition of g. Thus, $\mathsf{dep}(x^*, \mathcal{I}) = \mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{long}}) + \mathsf{dep}(p^*, \mathcal{Q}_{\mathsf{short}}) = \mathsf{dep}(p^*, \mathcal{Q})$.

Since $\mathsf{dep}(p^*, \mathcal{Q}) \leq \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$, it suffices to show that $\mathsf{dep}(x^*, \mathcal{I}) \geq \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$. Consider a point $p \in L$. Again, a rectangle in $\mathcal{Q}_{\mathsf{long}}$ contains p iff its corresponding weight-1 interval in $\mathcal{I}$ contains $x(p)$. Also, $f(x(p)) \geq \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}})$ by the definition of f. Therefore,

$$\mathsf{dep}(x(p), \mathcal{I}) = \mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) + f(x(p)) \geq \mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) + \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}}) = \mathsf{dep}(p, \mathcal{Q}).$$

Since $\mathsf{dep}(x^*, \mathcal{I}) = \mathsf{dep}(\mathcal{I})$, we have $\mathsf{dep}(x^*, \mathcal{I}) \geq \mathsf{dep}(x(p), \mathcal{I}) \geq \mathsf{dep}(p, \mathcal{Q})$. ◀

For our slab data structure $\mathcal{D}(L)$, we can simply use the following result of Imai and Asano [14], which is a byproduct of their (static) algorithm for the maximum-depth problem for weighted rectangles.

▶ **Lemma 6** ([14]). *There exists a dynamic maximum-depth data structure $\mathcal{A}$ for weighted intervals with $O(\log n)$ update time and $O(n \log n)$ preprocessing time.*

Given $\mathcal{Q}$, we first compute the set $\mathcal{I}$ as stated above (as well as the functions f and g), and then build $\mathcal{A}(\mathcal{I})$, i.e., the interval data structure of the above lemma on $\mathcal{I}$). $\mathcal{A}(\mathcal{I})$ maintains $x^* \in \mathbb{R}$ satisfying $\mathsf{dep}(x^*, \mathcal{I}) = \mathsf{dep}(\mathcal{I})$, as well as the value of $\mathsf{dep}(x^*, \mathcal{I})$. The point we maintain is $p^* = (x^*, g(x^*))$, which satisfies $\mathsf{dep}(p^*, \mathcal{Q}) = \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$ by Fact 5. Also, the value of $\mathsf{dep}(p^*, \mathcal{Q})$ is equal to $\mathsf{dep}(x^*, \mathcal{I})$ by Fact 5, and we can maintain the latter from $\mathcal{A}(\mathcal{I})$.

Consider an insertion/deletion on $\mathcal{Q}$, and let Q be the inserted/deleted rectangle. If Q is a long rectangle, we just need to insert/delete its corresponding weight-1 interval on $\mathcal{I}$ and update $\mathcal{A}(\mathcal{I})$. If Q is a short rectangle, the situation is more complicated and what we do is the following. First, we delete from $\mathcal{I}$ all intervals corresponding to the rectangles in the old $\mathcal{Q}_{\mathsf{short}}$. Then we construct the intervals corresponding to the rectangles in the new $\mathcal{Q}_{\mathsf{short}}$ (we shall show how to do this efficiently) and insert them to $\mathcal{I}$. Also, we re-compute the functions f and g corresponding to the new $\mathcal{Q}_{\mathsf{short}}$. For both deletions and insertions on $\mathcal{I}$, we need to update $\mathcal{A}(\mathcal{I})$ accordingly.

Next, we analyze the preprocessing time and update time for our slab data structure. To this end, we first show that one can compute the weighted intervals in $\mathcal{I}$ corresponding to the short rectangles, as well as the functions f and g, in $O(|\mathcal{Q}_{\mathsf{short}}| \log |\mathcal{Q}_{\mathsf{short}}|)$ time. Indeed, we can directly apply the classical (static) rectangle maximum-depth algorithm [14] on $\mathcal{Q}_{\mathsf{short}}$. The algorithm sweeps a vertical line ℓ from left to right, and maintains the maximum depth of $\mathcal{Q}_{\mathsf{short}}$ on ℓ. Therefore, it exactly computes the function f and g, together with their piece intervals $I_1, \ldots, I_s$. The running time of the algorithm is $O(|\mathcal{Q}_{\mathsf{short}}| \log |\mathcal{Q}_{\mathsf{short}}|)$. The intervals in $\mathcal{I}$ corresponding to the long rectangles can be constructed directly. Overall, the time for constructing $\mathcal{I}$ is $O(|\mathcal{Q}| \log |\mathcal{Q}|)$, i.e., $O(n \log n)$. By Lemma 6, $\mathcal{A}(\mathcal{I})$ can also be built in $O(n \log n)$ time as $|\mathcal{I}| = O(|\mathcal{Q}|)$. So the preprocessing time of our slab data structure is $O(n \log n)$. For the update time, we need to distinguish long rectangles and short rectangles. Inserting/deleting a long rectangle corresponds to an insertion/deletion on $\mathcal{A}(\mathcal{I})$ and thus can be done in $O(\log n)$ time. For the insertion/deletion of a short rectangle, we need to

delete $O(|\mathcal{Q}_{\mathsf{short}}|)$ intervals from $\mathcal{I}$, re-compute the intervals corresponding to $\mathcal{Q}_{\mathsf{short}}$ and the functions f and g, and insert $O(|\mathcal{Q}_{\mathsf{short}}|)$ intervals to $\mathcal{I}$. The $O(|\mathcal{Q}_{\mathsf{short}}|)$ insertions/deletions on $\mathcal{A}(\mathcal{I})$ can be done in $O(|\mathcal{Q}_{\mathsf{short}}| \log n)$ time by Lemma 6, and the re-computation can be done in $O(|\mathcal{Q}_{\mathsf{short}}| \log |\mathcal{Q}_{\mathsf{short}}|)$ time as discussed above. As the number of rectangle corners in a slab is always $O(n/r)$, we have $|\mathcal{Q}_{\mathsf{short}}| = O(n/r)$. Thus, the update time for a short rectangle is $O((n/r) \log n)$.

▶ **Lemma 7.** *One can construct each slab data structure in $O(n \log n)$ time such that for insertions/deletions of long (resp., short) rectangles, the data structure can be updated in $O(\log n)$ time (resp., $O((n/r) \log n)$ time).*

Overall time complexity. Using Lemma 7, we can now analyze the overall preprocessing time and update time of our data structure. Partitioning the plane into slabs can be done in $O(n \log n)$ time by sorting the corners of the rectangles in $\mathcal{Q}$. Since there are r slabs and each slab data structure can be constructed in $O(n \log n)$ time, the preprocessing time is $O(rn \log n)$. To analyze the update time, let Q be the inserted/deleted rectangle. Note that there are at most two slabs in which Q is a short rectangle. So among the updates of the slab data structures, there can be at most two updates for short rectangles and $O(r)$ updates for long rectangles. By Lemma 7, these updates take $O((n/r) \log n + r \log n)$ time. Also, rectangle insertions might cause slab splittings and for each slab splitting we need $O(n \log n)$ time to build the new slab data structures. By Fact 4, the time cost for slab splitting during the first t operations is $O(tr \log n)$, and the amortized time cost is then $O(r \log n)$. Therefore, the overall (amortized) update time of our data structure is $O((n/r) \log n + r \log n)$. To balance the two terms, we set $r = \sqrt{n}$, yielding $O(\sqrt{n} \log n)$ update time. The preprocessing time is then $O(n^{1.5} \log n)$. Based on the above discussion, we have the following conclusion, which is the base case $k = 1$ of Theorem 1.

▶ **Lemma 8.** *There exists a dynamic maximum-depth data structure for axis-parallel rectangles with $O(\sqrt{n} \log n)$ amortized update time, which can be built in $O(n^{1.5} \log n)$ time.*

Approximate data structures. Now we are able to prove Theorem 1 for a general $k \in \mathbb{N}$. As aforementioned, we apply induction on k. Lemma 8 gives us the base case $k = 1$. Suppose there exists a $\frac{1}{k}$-approximate dynamic maximum-depth data structure for rectangles with $O(n^{1/(k+1)} \log n)$ (amortized) update time and $O(n^{1+1/(k+1)} \log n)$ preprocessing time. We want to show the existence of a $\frac{1}{k+1}$-approximate data structure with $O(n^{1/(k+2)} \log n)$ update time and $O(n^{1+1/(k+2)} \log n)$ preprocessing time.

As before, our data structure partitions the plane into r slabs. The only difference occurs in the design of the slab data structures. Consider a slab L. We want our slab data structure for L to maintain a point $p^* \in L$ satisfying that $\mathsf{dep}(p^*, \mathcal{Q}) \geq \frac{1}{k+1} \cdot \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$; this guarantees that our overall data structure achieves the approximation ratio $\frac{1}{k+1}$. Again, let $\mathcal{Q}_{\mathsf{long}}$ and $\mathcal{Q}_{\mathsf{short}}$ denote the set of long and short rectangles in $\mathcal{Q}$ with respect to L, respectively. Unlike in our exact data structure, we now handle $\mathcal{Q}_{\mathsf{long}}$ and $\mathcal{Q}_{\mathsf{short}}$ separately. For long rectangles, we maintain a point $p_1^* \in L$ that maximizes $\mathsf{dep}(p_1^*, \mathcal{Q}_{\mathsf{long}})$. This can be done using exactly the same idea as before: map each long rectangle $Q = [x^-, x^+] \times [y^-, y^+] \in \mathcal{Q}_{\mathsf{long}}$ to the interval $[x^-, x^+]$, and build $\mathcal{A}$ on these intervals. For short rectangles, we maintain a point $p_2^* \in L$ satisfying the condition that $\mathsf{dep}(p_2^*, \mathcal{Q}_{\mathsf{short}}) \geq \frac{1}{k} \cdot \max_{p \in L} \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}})$. To this end, we build a $\frac{1}{k}$-approximate dynamic maximum-depth data structure on $\mathcal{Q}_{\mathsf{short}}$, which can give us the desired point p_2^*. By our induction hypothesis, there exists such a data structure with $O(n^{1/(k+1)} \log n)$ update time and $O(n^{1+1/(k+1)} \log n)$ preprocessing time. Define $p^* = p_1^*$ if $\mathsf{dep}(p_1^*, \mathcal{Q}_{\mathsf{long}}) \geq \mathsf{dep}(p_2^*, \mathcal{Q}_{\mathsf{short}})$ and $p^* = p_2^*$ if $\mathsf{dep}(p_1^*, \mathcal{Q}_{\mathsf{long}}) < \mathsf{dep}(p_2^*, \mathcal{Q}_{\mathsf{short}})$. We observe the following.

▶ **Lemma 9.** $\mathsf{dep}(p^*, \mathcal{Q}) \geq \frac{1}{k+1} \cdot \max_{p \in L} \mathsf{dep}(p, \mathcal{Q})$.

Proof. Let $p \in L$ be a point that maximizes $\mathsf{dep}(p, \mathcal{Q})$. By definition, we have $\mathsf{dep}(p, \mathcal{Q}) = \mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) + \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}})$. We consider two cases: $\mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) \geq \frac{1}{k+1}\mathsf{dep}(p, \mathcal{Q})$ and $\mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) < \frac{1}{k+1}\mathsf{dep}(p, \mathcal{Q})$. If $\mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) \geq \frac{1}{k+1}\mathsf{dep}(p, \mathcal{Q})$, we have

$$\mathsf{dep}(p^*, \mathcal{Q}) \geq \mathsf{dep}(p_1^*, \mathcal{Q}) \geq \mathsf{dep}(p_1^*, \mathcal{Q}_{\mathsf{long}}) \geq \mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) \geq \frac{1}{k+1}\mathsf{dep}(p, \mathcal{Q}).$$

On the other hand, if $\mathsf{dep}(p, \mathcal{Q}_{\mathsf{long}}) < \frac{1}{k+1}\mathsf{dep}(p, \mathcal{Q})$, we must have $\mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}}) > \frac{k}{k+1}\mathsf{dep}(p, \mathcal{Q})$. In this case, we have

$$\mathsf{dep}(p^*, \mathcal{Q}) \geq \mathsf{dep}(p_2^*, \mathcal{Q}) \geq \mathsf{dep}(p_2^*, \mathcal{Q}_{\mathsf{short}}) \geq \frac{1}{k} \cdot \mathsf{dep}(p, \mathcal{Q}_{\mathsf{short}}) > \frac{1}{k+1} \cdot \mathsf{dep}(p, \mathcal{Q}).$$

Therefore, we always have $\mathsf{dep}(p^*, \mathcal{Q}) \geq \frac{1}{k+1} \cdot \mathsf{dep}(p, \mathcal{Q})$. ◀

The data structure for maintaining p_1^* can be constructed in $O(|\mathcal{Q}_{\mathsf{long}}| \log |\mathcal{Q}_{\mathsf{long}}|)$ time and updated in $O(\log |\mathcal{Q}_{\mathsf{long}}|)$ time, by Lemma 6. Furthermore, the data structure for maintaining p_2^* can be constructed in $O(|\mathcal{Q}_{\mathsf{short}}|^{1+1/(k+1)} \log |\mathcal{Q}_{\mathsf{short}}|)$ time and updated in $O(|\mathcal{Q}_{\mathsf{short}}|^{1/(k+1)} \log |\mathcal{Q}_{\mathsf{short}}|)$ time. Since $|\mathcal{Q}_{\mathsf{long}}| = O(n)$ and $|\mathcal{Q}_{\mathsf{short}}| = O(n/r)$, for each slab data structure, the preprocessing time is $O(n \log n + (n/r)^{1+1/(k+1)} \log(n/r))$ and the update time is $O(\log n)$ for long rectangles and $O((n/r)^{1/(k+1)} \log(n/r))$ for short rectangles. This implies that our overall data structure has preprocessing time $O(rn \log n + r(n/r)^{1+1/(k+1)} \log(n/r))$ and update time $O(r \log n + (n/r)^{1/(k+1)} \log(n/r))$. By setting $r = n^{1/(k+2)}$, we have the desired $\frac{1}{k+1}$-approximation data structure with $O(n^{1/(k+2)} \log n)$ amortized update time and $O(n^{1+1/(k+2)} \log n)$ preprocessing time.

▶ **Theorem 1.** *For every $k \in \mathbb{Z}^+$, there exists a $\frac{1}{k}$-approximate dynamic maximum-depth data structure for axis-parallel rectangles with $O(n^{1/(k+1)} \log n)$ amortized update time.*

4 Dynamic maximum-depth for disks

We now describe our dynamic maximum-depth data structure for disks. The main idea underlying our result is to reduce the maximum-depth problem to *discrete* maximum depth.

Suppose we want to design a $(\frac{1}{2} - \varepsilon)$-approximate maximum-depth data structure for disks, where $\varepsilon > 0$ is a constant. Without loss of generality, we assume that ε is sufficiently small, say, $\varepsilon < 0.1$. Let δ be a sufficiently large integer such that $\frac{1}{\delta} \leq 1 - \cos \varepsilon\pi$. For a disk D in $\mathbb{R}^2$ with center (x, y) and radius r, we define

$$\Gamma_\delta(D) = \{(x - 3r + ir/\delta, y - 3r + jr/\delta) \in \mathbb{R}^2 : i, j \in [6\delta]\},$$

and call the points in $\Gamma_\delta(D)$ the *companion points* of D. For a set $\mathcal{D}$ of disks, write $\Gamma_\delta(\mathcal{D}) = \bigcup_{D \in \mathcal{D}} \Gamma_\delta(D)$. The set of companion points is the discrete point set for which our data structure dynamically maintains depth. The following fact is straightforward.

▶ **Fact 10.** *Let D be a disk and S be a set of points in D. There exists a circular sector X of D with angle $(1 - 2\varepsilon)\pi$ such that $|S \cap X| \geq (\frac{1}{2} - \varepsilon) \cdot |S|$.*

Proof. Suppose we randomly sample a circular sector X of D with angle $(1 - 2\varepsilon)\pi$, from the uniform distribution over the space of all such sectors. The probability that a specific point $p \in S$ is contained in X is 1 if p is the center of D and is $\frac{1}{2} - \varepsilon$ otherwise. Therefore, $\mathbb{E}[|S \cap X|] \geq (\frac{1}{2} - \varepsilon) \cdot |S|$, which in turn implies that there exists a sector X of D with angle $(1 - 2\varepsilon)\pi$ for which $|S \cap X| \geq (\frac{1}{2} - \varepsilon) \cdot |S|$. ◀

▶ **Lemma 11.** *Let $\mathcal{D}$ be a set of congruent disks such that $\bigcap_{D\in\mathcal{D}} D \neq \emptyset$. Then for every $D \in \mathcal{D}$, there exists $p \in \Gamma_\delta(D)$ such that $\mathsf{dep}(p, \mathcal{D}) \geq \left(\frac{1}{2} - \varepsilon\right) \cdot |\mathcal{D}|$.*

Proof. Without loss of generality, we assume that the disks in $\mathcal{D}$ are unit disks. Let $p^* \in \bigcap_{D\in\mathcal{D}} D$ and P^* be the unit disk centered at p^*. Define $\mathsf{ctr}(\mathcal{D})$ as the set of the centers of the disks in $\mathcal{D}$. Since $p^* \in \bigcap_{D\in\mathcal{D}} D$, we have $\mathsf{ctr}(\mathcal{D}) \subseteq P^*$. By Fact 10, there is a circular sector X of P^* with angle $(1-2\varepsilon)\pi$ such that $|\mathsf{ctr}(\mathcal{D}) \cap X| \geq (\frac{1}{2} - \varepsilon) \cdot |\mathsf{ctr}(\mathcal{D})| = (\frac{1}{2} - \varepsilon) \cdot |\mathcal{D}|$. Suppose u and v are the two vertices of X on the boundary of P^*. Define o as the middle point of the segment connecting u and v, and D_o as the disk centered at o with radius $1 - \cos \varepsilon\pi$. We claim that $\mathsf{dep}(p, \mathcal{D}) \geq \left(\frac{1}{2} - \varepsilon\right) \cdot |\mathcal{D}|$ for any point $p \in D_o$. Consider an arbitrary point $p \in D_o$. Since the angle of X is $(1-2\varepsilon)\pi$, the distance between o and p^* is $\sin \varepsilon\pi$. Also, the distance between o and u (resp., v) is $\cos \varepsilon\pi$. Recall that $\varepsilon < 0.1$ by our assumption, which implies that $\sin \varepsilon\pi < \cos \varepsilon\pi$. Therefore, the distance between p and p^* is at most $\sin \varepsilon\pi + (1 - \cos \varepsilon\pi) < 1$, and the distance between p and u (resp., v) is at most $\cos \varepsilon\pi + (1 - \cos \varepsilon\pi) = 1$. Therefore, the unit disk centered at p contains points p^*, u, v, which further implies that it contains X. Consequently, the unit disks in $\mathcal{D}$ whose centers lie in X contains p. We have $|\mathsf{ctr}(\mathcal{D}) \cap X| \geq (\frac{1}{2} - \varepsilon) \cdot |\mathcal{D}|$, which then implies that $\mathsf{dep}(p, \mathcal{D}) \geq \left(\frac{1}{2} - \varepsilon\right) \cdot |\mathcal{D}|$. See Figure 2 for an illustration.

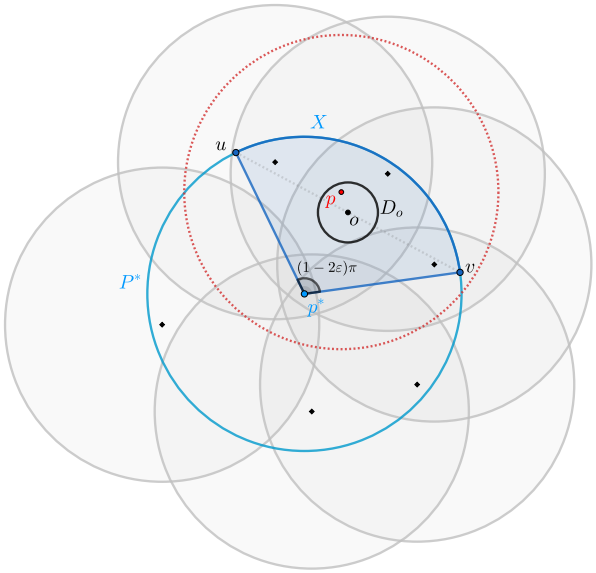


Figure 2 Illustrating the proof of Lemma 11.

Now it suffices to show $\Gamma_\delta(D) \cap D_o \neq \emptyset$ for every $D \in \mathcal{D}$. Note that the distance between o and the center of D is at most 2, since $p^* \in D$ and $o \in P^*$. Furthermore, D_o contains an axis-parallel square centered at o with side-length $1 - \cos \varepsilon\pi$, as its radius is $1 - \cos \varepsilon\pi$. Since $1 - \cos \varepsilon\pi \geq \frac{1}{\delta}$, this square then contains a point in $\Gamma_\delta(D)$, by our construction of $\Gamma_\delta(D)$. Therefore, $\Gamma_\delta(D) \cap D_o \neq \emptyset$. ◀

▶ **Corollary 12.** *For any set $\mathcal{D}$ of disks in $\mathbb{R}^2$, the following holds:*

$$\max_{p\in\Gamma_\delta(\mathcal{D})} \mathsf{dep}(p, \mathcal{D}) \geq \left(\frac{1}{2} - \varepsilon\right) \cdot \mathsf{dep}(\mathcal{D}).$$

Proof. Let $p^* \in \mathbb{R}^2$ such that $\mathsf{dep}(p^*, \mathcal{D}) = \mathsf{dep}(\mathcal{D})$, and $\mathcal{D}^* = \{D \in \mathcal{D} : p^* \in D\}$. Suppose $D_0 \in \mathcal{D}^*$ is the smallest disk in $\mathcal{D}^*$. For each disk $D \in \mathcal{D}^*$, we can find a smaller disk $D' \subseteq D$ with the same radius as D_0 such that $p^* \in D'$. Let $\mathcal{D}'$ be the set of all these disks. Then $\mathcal{D}'$

is a set of congruent disks containing p^*, and $\mathsf{dep}(p, \mathcal{D}') \leq \mathsf{dep}(p, \mathcal{D}^*)$ for any $p \in \mathbb{R}^2$. The former implies that $\bigcap_{D \in \mathcal{D}'} D \neq \emptyset$. Furthermore, we have $D_0 \in \mathcal{D}'$. By Lemma 11, there exists $p \in \Gamma_\delta(D_0)$ such that $\mathsf{dep}(p, \mathcal{D}') \geq (\frac{1}{2} - \varepsilon) \cdot |\mathcal{D}'| = (\frac{1}{2} - \varepsilon) \cdot |\mathcal{D}^*| = (\frac{1}{2} - \varepsilon) \cdot \mathsf{dep}(\mathcal{D})$. This proves the corollary. ◀

Using this corollary, we can reduce the task of maintaining a $(\frac{1}{2} - \varepsilon)$-approximation of the maximum depth of disks to the dynamic *discrete* maximum-depth problem for disks. Let $\mathcal{D}$ be a dynamic set of disks. Corollary 12 implies that $\max_{p \in S} \mathsf{dep}(p, \mathcal{D}) \geq (\frac{1}{2} - \varepsilon) \cdot \mathsf{dep}(\mathcal{D})$, where $S = \Gamma_\delta(\mathcal{D})$. Therefore, it suffices to consider the dynamic discrete maximum-depth instance with point set S and disk set $\mathcal{D}$.

4.1 Dynamic discrete maximum depth for disks

In this section, we design a dynamic discrete maximum-depth data structure for disks with $n^{2/3} \log^{O(1)} n$ amortized update time. Via the standard lifting argument, the dynamic discrete maximum-depth problem for disks can be reduced to the same problem for 3D halfspaces. To construct the discrete maximum-depth data structure, we first need a dynamic *depth-query* data structure for halfspaces. Specifically, such a data structure maintains a dynamic set $\mathcal{H}$ of hyperplanes and can return, for a query point $q \in \mathbb{R}^3$, the value of $\mathsf{dep}(q, \mathcal{H})$.

▶ **Lemma 13.** *There exists an dynamic depth-query data structure for 3D halfspaces with $n^{2/3} \log^{O(1)} n$ query time and $\log^{O(1)} n$ amortized update time. The data structure can be built in $n \log^{O(1)} n$ time.*

Proof. By duality between points and halfspaces, the depth-query problem is equivalent to the range-counting problem: store a dynamic set S of points in $\mathbb{R}^3$ such that for a query halfspace H, one can return the value $|S \cap H|$. A dynamic 3D halfspace range-counting data structure with $n^{2/3} \log^{O(1)} n$ query time and $\log^{O(1)} n$ amortized update time can be directly obtained using the dynamic partition tree given by Matoušek [15]. ◀

Let S be the dynamic set of points and $\mathcal{H}$ be the dynamic set of halfspaces, and we want to maintain $\max_{p \in S} \mathsf{dep}(p, \mathcal{H})$. We build the dynamic depth-query data structure in Lemma 13 for $\mathcal{H}$. We store the points in S in a standard partition tree $\mathcal{T}$ [15]. The tree $\mathcal{T}$ has $|S| \log^{O(1)} |S|$ nodes and can be built in $|S| \log^{O(1)} |S|$ time. The depth of $\mathcal{T}$ is $\log^{O(1)} |S|$ and the leaves of $\mathcal{T}$ one-to-one correspond to the points in S. Each node v of $\mathcal{T}$ corresponds to a subset $S(v) \subseteq S$, which consists of the points on the leaves of the subtree rooted at v. Furthermore, given any halfspace H in $\mathbb{R}^3$, one can find $k = |S|^{2/3} \log^{O(1)} |S|$ nodes $v_1, \ldots, v_k$ of $\mathcal{T}$ in $|S|^{2/3} \log^{O(1)} |S|$ time such that $S \cap H = \bigcup_{i=1}^{k} S(v_i)$ and the sets $S(v_1), \ldots, S(v_k)$ are pairwise disjoint; these nodes are called the *canonical nodes* for H. For each node v, we maintain two fields, $\sigma(v)$ and $\mu(v)$. The fields $\sigma(v)$ satisfies the following invariant: for any point $p \in S$, the sum of $\sigma(v)$ over all nodes v satisfying $p \in S(v)$ is equal to $\mathsf{dep}(p, \mathcal{H})$. The field $\mu(v)$ is defined as

$$\mu(v) = \begin{cases} \sigma(v) & \text{if } v \text{ is a leaf,} \\ \sigma(v) + \max_{u \in \mathsf{ch}(v)} \mu(u) & \text{otherwise,} \end{cases}$$

where $\mathsf{ch}(v)$ is the set of children of v. By the invariant for the σ-fields, we see that the μ-field of the root of $\mathcal{T}$ is just equal to $\max_{p \in S} \mathsf{dep}(p, \mathcal{H})$.

Halfspace insertion/deletion. For a halfspace insertion/deletion, we do not need to change the partition tree $\mathcal{T}$ itself; instead, we need to update the σ-fields and μ-fields. Let H be a halfspace to be inserted. We find the canonical nodes $v_1, \ldots, v_k$ for H. Then we increase the fields $\sigma(v_1), \ldots, \sigma(v_k)$ by 1. We also update the μ-fields of all ancestors of $v_1, \ldots, v_k$. Note that if a node is not an ancestor of any of $v_1, \ldots, v_k$, then its μ-field is not changed. The deletion of a halfspace can be handled in the same way, with the only difference that we decrease the fields $\sigma(v_1), \ldots, \sigma(v_k)$ by 1. The update time is $|S|^{2/3} \log^{O(1)} |S| = n^{2/3} \log^{O(1)} n$. Besides, we also need to update the depth-query data structure for $\mathcal{H}$, which takes $|\mathcal{H}|^{2/3} \log^{O(1)} |\mathcal{H}| = n^{2/3} \log^{O(1)} n$ time as well.

Point insertion/deletion. To delete a point p from S, we can directly remove the leaf v of $\mathcal{T}$ corresponding to p, and update the μ-fields of all ancestors of v. This takes $\log^{O(1)} n$ time. To handle point insertions, we apply the classical logarithmic method [4]. This method, instead of using one partition tree, stores S in $O(\log |S|)$ partition trees each of which corresponds to a subset of S whose size is a power of 2. The only problem we need to solve here is how to efficiently merge two partition trees storing subsets of S with size 2^i into a larger partition tree $\mathcal{T}$, with the σ-fields and μ-fields. The tree $\mathcal{T}$ itself can be built in $2^i \cdot i^{O(1)}$ time. To compute the σ-fields and μ-fields for the nodes of $\mathcal{T}$, what we do is the following. Let $S' \subseteq S$ be the subset $\mathcal{T}$ stores. We use the depth-query data structure to obtain $\mathsf{dep}(p, \mathcal{H})$ for all $p \in S'$. For each leaf v of $\mathcal{T}$, we set $\sigma(v) = \mathsf{dep}(p, \mathcal{H})$, where $p \in S'$ is the point corresponding to v. We set $\sigma(v) = 0$ for all internal nodes v of $\mathcal{T}$. Clearly, the invariant for the σ-fields holds. Using the σ-fields, we then compute the μ-field for every node of $\mathcal{T}$ in a bottom-up order. The most time-consuming part in this procedure is the depth queries, which takes $|S'| \cdot |\mathcal{H}|^{2/3} \log^{O(1)} |\mathcal{H}| = 2^i n^{2/3} \log^{O(1)} n$ time. As such, the amortized update time for a point insertion is $n^{2/3} \log^{O(1)} n$.

▶ **Lemma 14.** *There exists a dynamic discrete maximum-depth data structure for disks with $n^{2/3} \log^{O(1)} n$ amortized update time for insertions/deletions of points and disks; here n is the total number of points and disks in the instance.*

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, there exists a $(\frac{1}{2} - \varepsilon)$-approximate dynamic maximum-depth data structure for disks with $n^{2/3} \log^{O(1)} n$ amortized update time.*

Proof. We just build the data structure of Lemma 14 on the point set $S = \Gamma_\delta(\mathcal{D})$ and the disk set $\mathcal{D}$. The data structure maintains a point $p^* \in S$ such that $\mathsf{dep}(p^*, \mathcal{D}) = \max_{p \in S} \mathsf{dep}(p, \mathcal{D})$ and thus $\mathsf{dep}(p^*, \mathcal{D}) \geq (\frac{1}{2} - \varepsilon) \cdot \mathsf{dep}(\mathcal{D})$ by Corollary 12. Each insertion (resp., deletion) on $\mathcal{D}$ corresponds to $O(1)$ insertions (resp., deletions) on S and one insertion (resp., deletion) on $\mathcal{D}$. Therefore, the update time is $O(n^{2/3} \log^{O(1)} n)$ in total. ◀

We remark that our data structure for disks can be generalized to balls in higher dimensions. Indeed, the proof of Fact 10 holds for balls in any fixed dimension. To solve dynamic discrete maximum depth for balls in $\mathbb{R}^d$, one can reduce to the problem for halfspaces in $\mathbb{R}^{d+1}$ and use partition trees in $\mathbb{R}^{d+1}$. The update time becomes $n^{d/(d+1)} \log^{O(1)} n$.

4.2 Generalization to fat objects

In this section, we discuss how our reduction from the dynamic maximum-depth problem to its discrete counterpart works for general fat objects (where the approximation ratio depends on the fatness of the objects).

For a α-fat object O in $\mathbb{R}^d$ with whose center $(x_1 \cdots, x_d)$ and in-radius r, we define its *companion points* as

$$\Gamma_\alpha(O) = \left\{(x_1 \pm \sqrt{2} i_1 r, \cdots, x_d \pm \sqrt{2} i_d r) : i_1, \cdots, i_d \in \left[\lceil \sqrt{2}\alpha \rceil\right]\right\}.$$

The key observation of the reduction for general fat objects is the following fact.

▶ **Fact 15.** *A d-dimensional ball with center $(x_1 \cdots, x_d)$ and radius $2\alpha r$ can be covered by balls with centers in $\Gamma_\alpha(O)$ and radii r.*

▶ **Corollary 16.** *For any set $\mathcal{O}$ of α-fat objects in $\mathbb{R}^d$, the following holds:*

$$\max_{p \in \Gamma_\alpha(\mathcal{O})} \mathsf{dep}(p, \mathcal{O}) \geq \frac{1}{\left(2\lceil \sqrt{2}\alpha \rceil\right)^d} \cdot \mathsf{dep}(\mathcal{O}).$$

Proof. Let $p^* \in \mathbb{R}^d$ such that $\mathsf{dep}(p^*, \mathcal{O}) = \mathsf{dep}(\mathcal{O})$, and $\mathcal{O}^* = \{O \in \mathcal{O} : p^* \in O\}$. Suppose $O_0 \in \mathcal{O}^*$ is the smallest object in $\mathcal{O}^*$, i.e., O_0 has the smallest in-radius, which is assumed to be 1 without loss of generality. For each object $O \in \mathcal{O}^*$, we can find a unit ball $O' \subseteq O$ as following: the unit ball centered on the line segment between p^* and $\mathsf{ctr}(O)$, which is tangent to all tangent hyperplanes of O at p^*. (See Figure 3 for an illustration.)

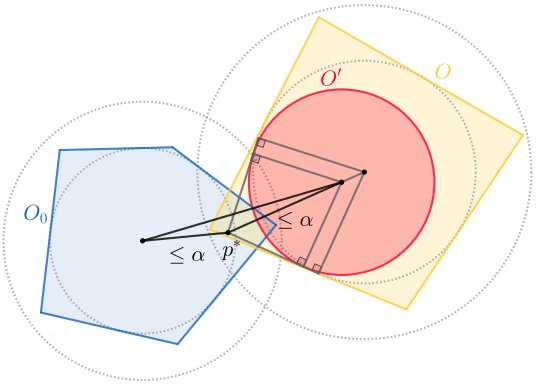


Figure 3 Illustrating the proof of Corollary 16 in $\mathbb{R}^2$.

Since O_0 contains p^*, we have the distance between p^* and $\mathsf{ctr}(O_0)$ is at most α. Moreover, it is clear that the distance between p^* and $\mathsf{ctr}(O')$ is at most α. Therefore, the distance between $\mathsf{ctr}(O')$ and $\mathsf{ctr}(O_0)$ is at most 2α, which means that the centers of these unit balls lie within a ball with center $\mathsf{ctr}(O_0)$ and radius 2α. By Fact 15, we can cover this ball by unit balls with centers in $\Gamma_\alpha(O_0)$. Therefore, by the pigeonhole principle, we have

$$\max_{p \in \Gamma_\alpha(\mathcal{O})} \mathsf{dep}(p, \mathcal{O}) \geq \max_{p \in \Gamma_\alpha(\mathcal{O}_0)} \mathsf{dep}(p, \mathcal{O}) \geq \frac{1}{|\Gamma_\alpha(\mathcal{O}_0)|} \cdot \mathsf{dep}(\mathcal{O}) = \frac{1}{\left(2\lceil \sqrt{2}\alpha \rceil\right)^d} \cdot \mathsf{dep}(\mathcal{O}),$$

which completes the proof. ◀

As a concluding remark, we note that our reduction for general fat objects relies on the following question: Given a ball of radius 2α in $\mathbb{R}^d$, how many unit balls are needed to cover it? We employ a rather general method in the construction of $\Gamma_\alpha(O)$, to illustrate the main idea, and the constant $\left(2\lceil \sqrt{2}\alpha \rceil\right)^d$ in Corollary 16 can be improved for specific values of α and d. For example, in the case of disk ($d = 2$ and $\alpha = 1$), as proved in [5], only 7 unit disks are needed, instead of 16 in Corollary 16. This optimization problem for particular values of α and d may have independent interest.

References

1 Pankaj K. Agarwal, Hsien-Chih Chang, Subhash Suri, Allen Xiao, and Jie Xue. Dynamic geometric set cover and hitting set. *ACM Trans. Algorithms*, 18(4):40:1–40:37, 2022. doi:10.1145/3551639.

2 Helmut Alt and Ludmila Scharf. Computing the depth of an arrangement of axis-aligned rectangles in parallel. In *26th European Workshop on Computational Geometry*, pages 33–36, 2010.

3 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008. doi:10.1137/060669474.

4 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

5 Károly Bezdek. *Körök optimális fedései (Optimal Covering of Circles)*. PhD thesis, Eötvös Lorand University, 1979.

6 Sujoy Bhore and Timothy M. Chan. Fast static and dynamic approximation algorithms for geometric optimization problems: Piercing, independent set, vertex cover, and matching. *CoRR*, abs/2407.20659, 2024. doi:10.48550/arXiv.2407.20659.

7 Sujoy Bhore, Guangping Li, and Martin Nöllenburg. An algorithmic study of fully dynamic independent sets for map labeling. *ACM J. Exp. Algorithmics*, 27:1.8:1–1.8:36, 2022. doi:10.1145/3514240.

8 Paz Carmi, Matthew J. Katz, and Pat Morin. Stabbing pairwise intersecting disks by four points. *Discret. Comput. Geom.*, 70(4):1751–1784, 2023. doi:10.1007/S00454-023-00567-0.

9 Timothy M. Chan. A (slightly) faster algorithm for klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. doi:10.1016/J.COMGEO.2009.01.007.

10 Timothy M. Chan and Qizheng He. More dynamic data structures for geometric set cover with sublinear update time. *J. Comput. Geom.*, 13(2):90–114, 2021. doi:10.20382/JOCG.V13I2A6.

11 Timothy M. Chan, Qizheng He, Subhash Suri, and Jie Xue. Dynamic geometric set cover, revisited. 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 3496–3528. SIAM, 2022. doi:10.1137/1.9781611977073.139.

12 Bernard Marie Chazelle and D. T. Lee. On a circle placement problem. *Computing*, 36(1-2):1–16, 1986. doi:10.1007/BF02238188.

13 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 51:1–51:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.SOCG.2020.51.

14 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, December 1983. doi:10.1016/0196-6774(83)90012-3.

15 Jirí Matoušek. Efficient partition trees. *Discret. Comput. Geom.*, 8:315–334, 1992. doi:10.1007/BF02293051.

16 Hakan Yildiz, John Hershberger, and Subhash Suri. A discrete and dynamic version of klee's measure problem. 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/paper28.pdf.

Efficient Greedy Discrete Subtrajectory Clustering

Ivor van der Hoog
Technical University of Denmark, Lyngby, Denmark

Lara Ost
University of Vienna, Austria

Eva Rotenberg
Technical University of Denmark, Lyngby, Denmark

Daniel Rutschmann
Technical University of Denmark, Lyngby, Denmark

Abstract

We cluster a set of trajectories $\mathcal{T}$ using subtrajectories of $\mathcal{T}$. We require for a clustering C that any two subtrajectories $(\mathcal{T}[a,b], \mathcal{T}[c,d])$ in a cluster have disjoint intervals $[a,b]$ and $[c,d]$. Clustering quality may be measured by the number of clusters, the number of vertices of $\mathcal{T}$ that are absent from the clustering, and by the Fréchet distance between subtrajectories in a cluster.

A Δ-cluster of $\mathcal{T}$ is a cluster $\mathcal{P}$ of subtrajectories of $\mathcal{T}$ with a centre $P \in \mathcal{P}$, where all subtrajectories in $\mathcal{P}$ have Fréchet distance at most Δ to P. Buchin, Buchin, Gudmundsson, Löffler and Luo present two $O(n^2 + nm\ell)$-time algorithms: $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ computes a *single* Δ-cluster where P has at least ℓ vertices and maximises the cardinality m of $\mathcal{P}$. $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ computes a *single* Δ-cluster where $\mathcal{P}$ has cardinality m and maximises the complexity ℓ of P.

In this paper, which is a mixture of algorithms engineering and theoretical insights, we use such maximum-cardinality clusters in a greedy clustering algorithm. We first provide an efficient implementation of $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ that significantly outperforms previous implementations. Next, we use these functions as a subroutine in a greedy clustering algorithm, which performs well when compared to existing subtrajectory clustering algorithms on real-world data. Finally, we observe that, for fixed Δ and $\mathcal{T}$, these two functions always output a point on the Pareto front of some bivariate function $\theta(\ell, m)$. We design a new algorithm $\mathbf{PSC}(\Delta, \mathcal{T})$ that in $O(n^2 \log^4 n)$ time computes a 2-approximation of this Pareto front. This yields a broader set of candidate clusters, with comparable quality to the output of the previous functions. We show that using $\mathbf{PSC}(\Delta, \mathcal{T})$ as a subroutine improves the clustering quality and performance even further.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Algorithms engineering, Fréchet distance, subtrajectory clustering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.78

Related Version *Full Version*: `http://arxiv.org/abs/2503.14115`

Supplementary Material
Software (Source code): `https://github.com/laraost/DiscreteSubtrajectoryClustering` archived at `swh:1:dir:d00278b75c11b80bce00c51dbf8ac904972d8586`
Software (Test Suite): `https://github.com/dacin21/discrete-subtrajectory-clustering-test-suite`, archived at `swh:1:dir:e8f5c54ffb22bda19e4ca7858b7a6ea5b5ab2090`

Funding *Ivor van der Hoog*, *Eva Rotenberg*, and *Daniel Rutschmann* are grateful to the Carlsberg Foundation for supporting this research via Eva Rotenberg's Young Researcher Fellowship CF21-0302 "Graph Algorithms with Geometric Applications".
Ivor van der Hoog: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987
Lara Ost: Funded by the Vienna Graduate School on Computational Optimization (VGSCO), FWF-Project No. W1260-N35

© Ivor van der Hoog, Lara Ost, Eva Rotenberg, and Daniel Rutschmann;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Trajectories describe the movement of objects. The moving objects are modeled as a sequence of locations. Trajectory data often comes from GPS samples, which are being generated on an incredible scale through cars, cellphones, and other trackers. We focus on *subtrajectory clustering* where clusters are sets of subtrajectories. The goal is to create few clusters and to ensure that subtrajectories within a cluster have low pairwise discrete Fréchet distance. This algorithmic problem has been studied extensively [1, 8, 10, 12, 13, 15, 16, 19, 21, 22, 20, 24, 27, 30] in both theoretical and applied settings. For surveys, see [7] and [28].

Subtrajectory clustering. Comparing whole trajectories gives little information about shared structures. Consider two trajectories T_1 and T_2 that represent individuals who drive the same route to work, but live in different locations. For almost all distance metrics, the distance between T_1 and T_2 is large because their endpoints are (relatively) far apart. However, T_1 and T_2 share similar subtrajectories. Subtrajectory clustering allows us to model a wide range of behaviours: from detecting commuting patterns, to flocking behaviour, to congestion areas [12]. In the special case where trajectories correspond to traffic on a known road network, trajectories can be mapped to the road network. In many cases, the underlying road network might not be known (or it may be too dense to represent). In such cases, it is interesting to construct a sparse representation of the underlying road network based on the trajectories. The latter algorithmic problem is called *map construction*, and it has also been studied extensively [2, 6, 3, 14, 17, 25, 26]. We note that subtrajectory clustering algorithms may be used for map construction: if each cluster is assigned one centre, then the collection of centres can be used to construct an underlying road network [9, 10].

Related work. BBGLL [12] introduce the following problem. Rather than clustering $\mathcal{T}$, their goal is to compute a *single* cluster. Given $\mathcal{T}$, a Δ-*cluster* is any pair $(P, \mathcal{P})$ where:

- P is a subtrajectory of a trajectory in $\mathcal{T}$,
- $\mathcal{P}$ is a set of subtrajectories, where for all $\mathcal{T}[a,b], \mathcal{T}[c,d] \in \mathcal{P}$, $[a,b] \cap [c,d] = \emptyset$,
- and, for all $P' \in \mathcal{P}$, the Fréchet distance between P and P' is at most Δ.

See Figure 1. They define two functions that each output a single Δ-cluster $(P, \mathcal{P})$:

- $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ requires that $|P| = \ell$ and maximises $|\mathcal{P}|$.
- $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ requires that $|\mathcal{P}| = m$ and maximises $|P|$.

If $|P| = \ell$ and $|\mathcal{P}| = m$, their algorithms use $O(n^2 + nm\ell)$ time and $O(n\ell)$ space.

AFMNPT [1] propose a clustering framework. They define a clustering C as any set of clusters. They propose a scoring function with constants (c_1, c_2, c_3) that respectively weigh: the number of clusters, the radius of each cluster, and the number of uncovered vertices. Their implement a heuristic algorithm that uses this scoring function.

BBDFJSSSW [9] do map construction. They compute a set of Δ-clusters C for various Δ. For each $(P, \mathcal{P}) \in C$, they use P as a road on the map they are constructing. Thus, it may be argued that [9] does subtrajectory clustering. As a preprocessing step, they implement $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$. Choosing various ℓ and Δ, they compute a collection S of "candidate" clusters. They greedily select a cluster $(P, \mathcal{P}) \in S$, add it to the clustering, and remove all subtrajectories in $\mathcal{P}$ from all remaining clusters in S. For a cluster $(Q, \mathcal{Q}) \in S$, this may split a $Q' \in \mathcal{Q}$ into two subtrajectories. BBGHSSSSSW [10] give improved implementations.

ABCD [4] do not require that for all $\mathcal{T}[a,b], \mathcal{T}[c,d] \in \mathcal{P}$, $[a,b] \cap [c,d] = \emptyset$. They do require that the final clustering C has no uncovered vertices, and that for each cluster $(P, \mathcal{P})$, $|P| \leq \ell$ for some given ℓ. For any Δ, denote by k_Δ the minimum-sized clustering that meets

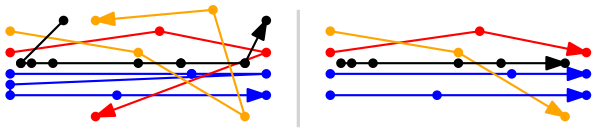


Figure 1 (left) Four trajectories. (right) Five subtrajectories $\mathcal{P}$. Depending on the choice of $P \in \mathcal{P}$, we may get five different clusters $(P, \mathcal{P})$.

their requirements. Their randomised polynomial-time algorithm computes a clustering of $O(\Delta)$-clusters of size $\tilde{O}(k_\Delta \ell)$. BCD [8] give polynomial-time clustering algorithm of $O(\Delta)$-clusters of size $\tilde{O}(k_\Delta)$. Conradi and Driemel [16] implement an $\tilde{O}(\ell^2 n^4 + k_\Delta \ell n^4)$-time algorithm that computes a set of $O(\Delta)$-clusters of size $O(k_\Delta \log n)$.

Contribution. Our contribution is threefold. First, in Section 5, we describe an efficient implementation of $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$. We compare our implementation to all existing single-core implementations.

In Section 5 we propose that $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ produces high-quality clusters across all clustering metrics. We create a greedy clustering algorithm that fixes (ℓ, Δ) and iteratively adds the solution $(P, \mathcal{P}) = \mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ to the clustering (removing all vertices in $\mathcal{P}$ from $\mathcal{T}$). An analogous approach for $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ gives two clustering algorithms.

In Section 6 we note that the functions $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ search a highly restricted solution space, having fixed either the minimum length of the centre of a cluster or the minimum cardinality. We observe that these functions always output a cluster on the Pareto front of some bivariate function $\theta(m, \ell)$. Since points on this Pareto front give high-quality clusters, we argue that it may be beneficial to approximate all vertices of this Pareto front directly – sacrificing some cluster quality for cluster flexibility. We create a new algorithm $\mathbf{PSC}(\Delta, \mathcal{T})$ that computes in $O(n^2 \log^4 n)$ time and $O(n)$ space a set of clusters that is a 2-approximation of the Pareto font of $\theta(m, \ell)$. We show that using $\mathbf{PSC}(\Delta, \mathcal{T})$ as a subroutine for greedy clustering gives even better results.

In Section 7 we do experimental analysis. We compare our clustering algorithms to AFMNPT [1] and BBGHSSSSSW [10]. We adapt the algorithm in [10] so that it terminates immediately after computing the subtrajectory clustering. We do not compare to Conradi and Driemel [16] as their setting is so different that it produces incomparable clusters. We compare the algorithms based on their overall run time and space usage. We evaluate the clustering quality along various metrics such as the number of clusters, the average cardinality of each cluster, and the Fréchet distance between subtrajectories in a cluster. Finally, we consider the scoring function by AFMNPT [1].

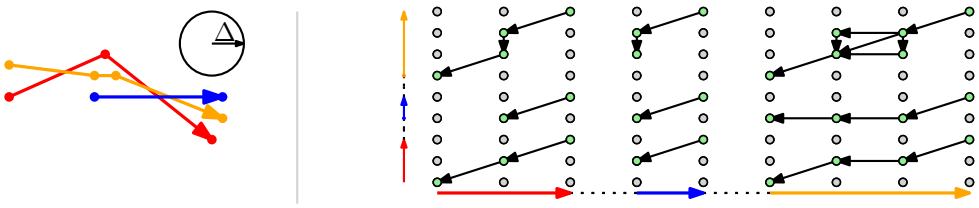


Figure 2 A set of trajectories $\mathcal{T}$ and the matrix $M_\Delta(\mathcal{T}, \mathcal{T})$ as a graph.

2 Preliminaries

The input is a set $\mathcal{T}$ of trajectories where $n = \sum_{T \in \mathcal{T}} |T|$. We denote by $\mathcal{X}$ all vertices of $T \in \mathcal{T}$. We view any trajectory T with n_T vertices as a map from $[n_T]$ to $\mathbb{R}^2$ where $T(i)$ is the i'th vertex on T. For brevity, we refer to $\mathcal{T}$ as a single trajectory obtained by concatenating all trajectories in $\mathcal{T}$ arbitrarily. A *boundary edge* in $\mathcal{T}$ is any edge in $\mathcal{T}$ between the endpoints of two consecutive trajectories. A *subtrajectory* $\mathcal{T}[a, b]$ of $\mathcal{T}$ is defined by $a, b \in [n]$ with $a \leq b$, where we require that the subcurve from $\mathcal{T}(a)$ to $\mathcal{T}(b)$ contains no boundary edge.

Fréchet distance. We first define discrete walks for pair of trajectories $P = (p_1, \ldots, p_x)$ and $T = (t_1, \ldots, t_y)$ in $\mathbb{R}^2$. We say that an ordered sequence F of points in $[x] \times [y]$ is a *discrete walk* if consecutive pair $(i, j), (s, l) \in F$ have $s \in \{i, i-1\}$ and $l \in \{j, j-1\}$. The *cost* of a discrete walk F is the maximum distance $d(P(i), Q(j))$ over $(i, j) \in F$. The discrete Fréchet distance is the minimum-cost discrete walk from (x, y) to $(1, 1)$:

$$\mathcal{D}_F(P, Q) := \min_F \text{cost}(F) = \min_F \max_{(i,j) \in F} d(P(i), Q(j)).$$

Free-space matrix. Given $\mathcal{T}$ and a value Δ, the *Free-space matrix* $M_\Delta(\mathcal{T}, \mathcal{T})$ is an $n \times n$ 01-matrix where for all $(i, j) \in [n] \times [n]$ the matrix has a zero at position i, j if $d(\mathcal{T}(i), \mathcal{T}(j)) \leq \Delta$. We use matrix notation where (i, j) denotes row i and column j. In the plane, $(0, 0)$ denotes the top left corner. We also use $M_\Delta(\mathcal{T}, \mathcal{T})$ to denote the digraph where the vertices are all $(i, j) \in [n] \times [n]$ and there exists an edge from (i, j) to (s, k) whenever (see Figure 2):

- $s \in \{i, i-1\}$ and $k \in \{j, j-1\}$, and
- $M_\Delta(\mathcal{T}, \mathcal{T})[i, j] = M_\Delta(\mathcal{T}, \mathcal{T})[s, k] = 0$.

Alt and Godeau [5] show that for any two subtrajectories $\mathcal{T}[a, b]$ and $\mathcal{T}[c, d]$: $\mathcal{D}_F(T[a, b], T[c, d]) \leq \Delta$ if and only if there is a directed path from (b, d) to (a, c).

▶ **Definition 1.** *Given Δ and $\mathcal{T}$, we say that a vertex (b, x) in $M_\Delta(\mathcal{T}, \mathcal{T})$ can* reach *row a if there exists an integer $x' \leq x$ such that there is a directed path from (b, x) to (a, x'). We denote by $\texttt{row}(b, x)$ the minimum integer a' such that (b, x) can reach row a'.*

▶ **Definition 2** (Figure 1)**.** *A cluster is any pair $(P, \mathcal{P})$ where P is a subtrajectory and $\mathcal{P}$ is a set subtrajectories with $P \in \mathcal{P}$. We require that for all $\mathcal{T}[a, b], \mathcal{T}[c, d] \in \mathcal{T}$, $[a, b] \cap [c, d] = \emptyset$. We define its* length $|P|$*,* cardinality $|\mathcal{P}|$*, and* coverage $\texttt{cov}(P, \mathcal{P}) = \bigcup_{S \in \mathcal{P}} S$.

Note that we follow [1, 9, 10, 12] and consider only *disjoint* clusterings, meaning $\forall (P, \mathcal{P}') \in C$, $\forall \mathcal{T}[a, b], \mathcal{T}[c, d] \in \mathcal{P}$, the intervals $[a, b]$ and $[c, d]$ are disjoint. For brevity, we say that a Δ-cluster is any cluster $(P, \mathcal{P})$ where $\forall P' \in \mathcal{P} : \mathcal{D}_F(P, P') \leq \Delta$.

▶ **Definition 3.** *A* clustering C *is a set of clusters. Its* coverage *is* $\texttt{cov}(C) := \bigcup_{(P,\mathcal{P}) \in C} \texttt{cov}(P, \mathcal{P})$.

AFMNPT [1] propose a cost function to determine the quality of a clustering C. We distinguish between *k-centre* and *k-means* clustering:

▶ **Definition 4.** *Given constants* (c_1, c_2, c_3) *,* $\texttt{Score}(C)$ *is a sum that weights three terms:*

1. *The number of clusters in* C*:* $c_1|C|$.
2. *The chosen cost function for the clustering. I.e.,*
   $c_2 \max_{(P,\mathcal{P})\in C} \text{cen}(P,\mathcal{P}) = c_2 \max_{(P,\mathcal{P})\in C} \max_{P'\in\mathcal{P}} \mathcal{D}_F(P,P')$ *when doing* k*-centre clustering, or*
   $c_2 \sum_{(P,\mathcal{P})\in C} \text{mean}(P,\mathcal{P}) = c_2 \sum_{(P,\mathcal{P})\in C} \sum_{P'\in\mathcal{P}} \mathcal{D}_F(P,P')$ *when doing* k*-means clustering.*
3. *The fraction of uncovered points:* $c_3 \frac{|\mathcal{X}\setminus \texttt{cov}(C)|}{|\mathcal{X}|}$.

We also consider unweighted metrics: the maximal and average Fréchet distance between subtrajectories in a cluster, and the maximal and average cardinality of the clusters.

3 Data

We use five real-world data sets (see Table 1). For some of these data sets we include a subsampled version that only retains a $\frac{1}{c}$ fraction of the trajectories.

- The first three data sets are Athens-small, Chicago and Berlin, consisting of GPS samples from car traffic across these cities. These data sets are widely used benchmark tests for road map construction [2, 9, 31, 23].
- The fourth data set is the *Drifter* data set that was used for subtrajectory clustering by Driemel and Conradi [16]. It consists of GPS data from ocean drifters.
- The fifth data set is the UnID data set, which is the smallest data set used in [29].

Unfortunately, we cannot use any of the data sets in [1]. Most of these are no longer available. Those that are available are far too large. It is unknown how [1] preprocessed these.

Synthetic data. All competitor algorithms do not terminate on the larger real-world data sets, even when the time limit is 24 hours. Thus, we primarily use these sets to compare algorithmic efficiency. Our qualitative analysis uses the subsampled data sets and synthetic data. We generate data by creating a random domain X. Then, for some parameter c, we randomly select $c\%$ of the domain and invoke an approximate TSP solver. We add the result as a trajectory. This results in a collection of trajectories with many similar subtrajectories. Synthetic-A has a random domain of 200 vertices. Synthetic-B has a random domain of 100 vertices. For both of these, we consider $c \in \{50, 90, 95\}$.

Table 1 Our data sets, their sizes, and whether we subsampled a fraction of their trajectories.

Name	Real world	# of trajectories	# of vertices	Subsampled
Athens-small	yes	128	2840	no
Chicago-4	yes	222	29344	$\frac{1}{4}$'th
Chicago	yes	888	118360	no
Berlin-10	yes	2717	19130	$\frac{1}{10}$'th
Berlin	yes	27188	192223	no
Drifter	yes	2167	1931692	no
UniD	yes	362	214077	no
Synthetic-A	no	50	5000 - 9500	no
Synthetic-B	no	100	5000 - 9500	no

4 Introducing and solving SC(m, ℓ, Δ, $\mathcal{T}$)

BBGLL [12] introduce the problem **SC**(m, ℓ, Δ, $\mathcal{T}$). The input are integers m, ℓ, Δ and a set of trajectories $\mathcal{T}$. The goal is to find a Δ-cluster $(P, \mathcal{P})$ where $|\mathcal{P}| \geq m$ and $|P| = \ell$. The problem statement may be altered by maximising m or ℓ instead. This yields the problems **SC**(max, ℓ, Δ, $\mathcal{T}$) and **SC**(m, max, Δ, $\mathcal{T}$). There exist three single-core implementations of these algorithms (the *MOVETK library* [18], *map-construct* [9] and *map-construct-rtrees* [10], a version of *map-construct* that uses r-trees). [18] uses $O(n^2)$ space and $O(n^2 + nm\ell)$ time. Both [9, 10] use $O(n\ell)$ space and $O(n^2 + nm\ell)$ time, and [10] offers running time improvements over [10]. The latter uses the semi-continuous instead of discrete Fréchet distance. On the experimental input, switching between these metrics does not change the output. We do not compare to the existing GPU-based implementation [21].

Contribution. We sketch how we optimise the BBGLL algorithm in the full version Considers two integers a, b with $\ell = b - a$ (Figure 3). It alternates between `StepFirst` which increments a, `StepSecond` which increments b, and `Query` that given $(\Delta, \ell = b - a, m)$ aims to find a Δ-cluster $(P, \mathcal{P})$ with $P = \mathcal{T}[a, b]$ and $|\mathcal{P}| = m$. We apply two ideas:

1. **Windowing.** By [12], it suffices to keep only $M_\Delta(\mathcal{T}[a, b], \mathcal{T})$ in memory. Furthermore, one may store only the $z' \in O(n\ell)$ zeroes in this sub-matrix. This is implemented in [9, 10]. By cleverly abandoning memory, our `StepFirst` deletes a row in constant time. The `StepSecond` function is then the bottleneck, as it adds a new row.
2. **Row generation.** In [12], the `StepSecond` and `Query` functions iterate over all n cells in the bottommost row of $M_\Delta(\mathcal{T}, \mathcal{T})$. By applying a range searching data structure, we instead only iterate over all zeroes in this row.

We note that map-construct-rtree [10] uses r-trees for row generation (we refer to the journal version [11]). However, we believe that the size of their r-tree implementation dominates the time and space used by their algorithm in our experiments. In the full version, we describe how to use a light-weight range tree to implement this principle.

Experiments. We compare the implementations across our data sets for a variety of choices of Δ and m. We use the implementations of **SC**(m, max, Δ, $\mathcal{T}$) so that our experiments have to fix fewer parameters. Figures 4 and 5 shows the time and space usage across a subset of our experiments. The data clearly shows that our implementation is significantly more efficient than the previous single-core implementations. It is often more than a factor 1000 more efficient in terms of runtime and memory usage. We do not provide a more extensive analysis of these results since this is not our main contribution.

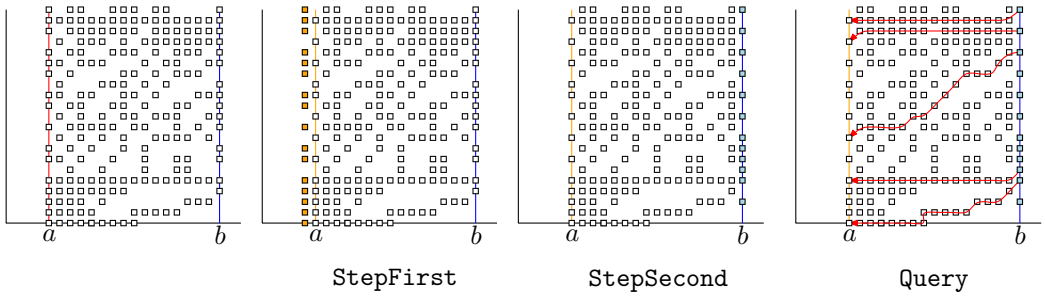


Figure 3 An overview of the algorithm in [12]. We illustrate zeroes in $M_\Delta(\mathcal{T}, \mathcal{T})$ with a square.

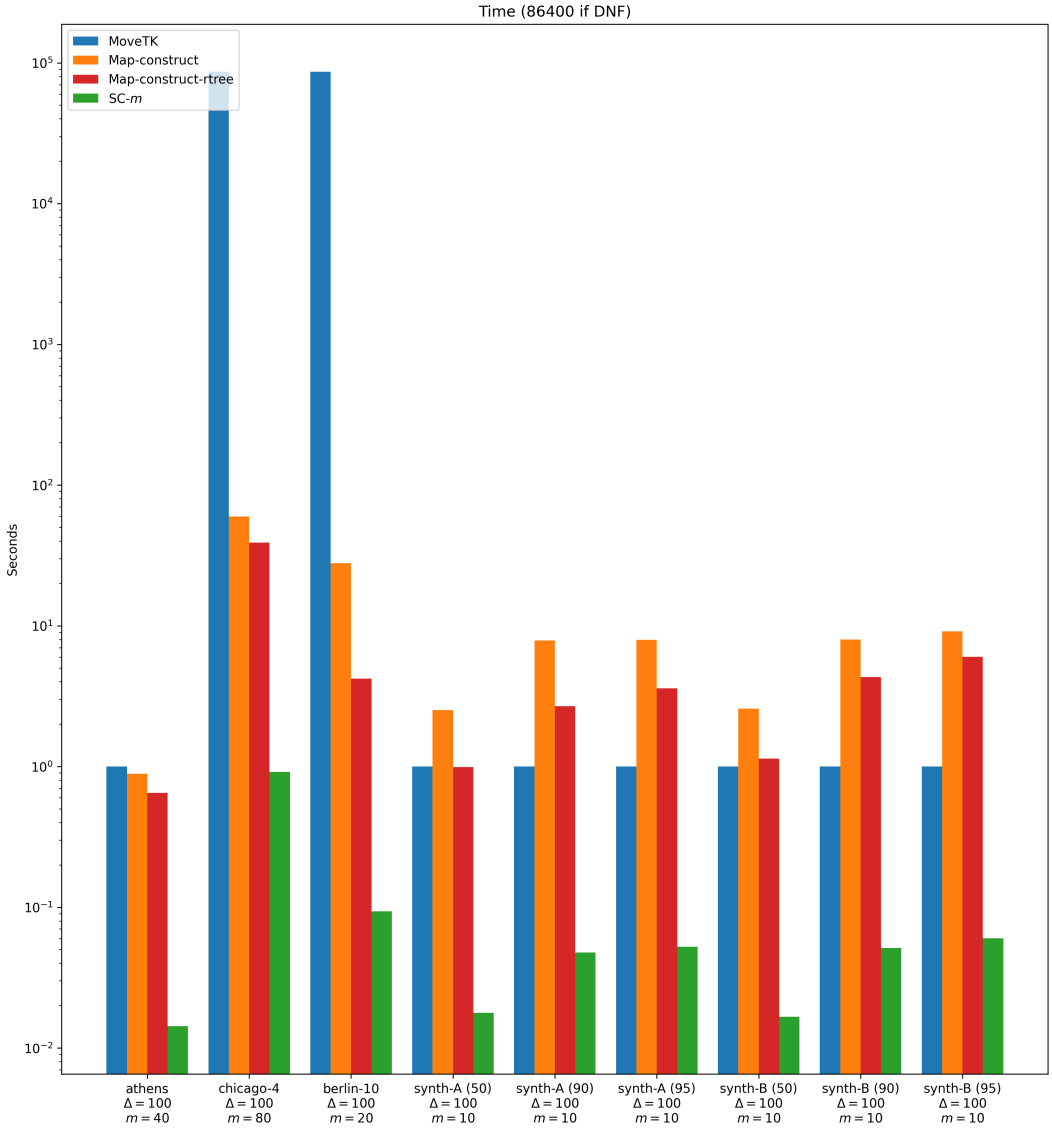


Figure 4 ■ BBGLL implementation comparison ■ Running time ■ Logarithmic scaling.

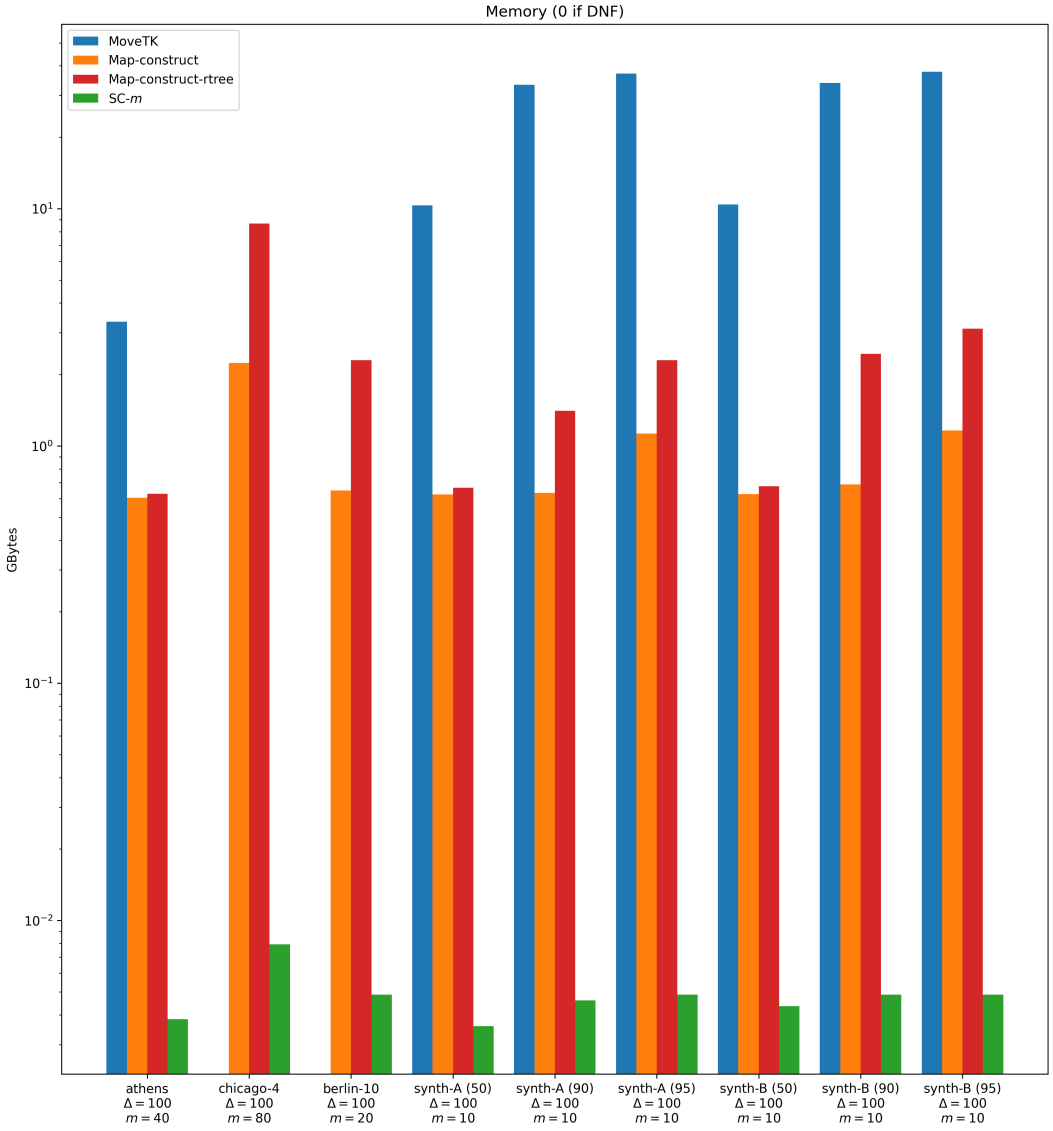


Figure 5 ■ BBGLL implementation comparison ■ Memory usage ■ Logarithmic scaling.

5 Greedy clustering using SC(max, ℓ, Δ, $\mathcal{T}$)

Our primary contribution is that we propose to use **SC**(max, ℓ, Δ, $\mathcal{T}$) in a greedy clustering algorithm. On a high level, our clustering algorithm does the following:

- We first choose the scoring function and constants (see Definition 4).
- Our input is the set of trajectories $\mathcal{T}$, some Δ, some ℓ, and an empty clustering C.
- We obtain a cluster $(P, \mathcal{P}) = \mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$, and add it to C.
- We then remove all subtrajectories in $\mathcal{P}$ from $\mathcal{T}$, and recurse.
- We continue this process until the addition of $(P, \mathcal{P})$ to C no longer increases $\texttt{Score}(C)$.

The implementation details depend on whether we are doing k-centre or k-median clustering.

5.1 k-centre clustering.

Given are (c_1, c_2, c_3) and a set of trajectories $\mathcal{T}$. We choose some integer ℓ. The k-centre clustering scoring function weighs the value: $\max_{(P,\mathcal{P})\in C} \text{cen}(P, \mathcal{P}) = \max_{(P,\mathcal{P})\in C} \max_{P'\in\mathcal{P}} \mathcal{D}_F(P, P')$.
If C is a set of Δ-clusters (we assume there exists a cluster in C whose radius is Δ) then:

$$\texttt{Score}(C) = c_1 \cdot |C| + c_2 \cdot \Delta + c_3 \cdot \frac{|\mathcal{X} \setminus \texttt{cov}(C)|}{|\mathcal{X}|}.$$

Let C be a set of Δ-clusters and $(P, \mathcal{P}) = \mathbf{SC}(\max, \ell, \Delta, \mathcal{T} - \texttt{cov}(C))$. Per construction of our algorithm, $\texttt{cov}(P, \mathcal{P}) \cap \texttt{cov}(C) = \emptyset$ and so the contribution of $(P, \mathcal{P})$ to $\texttt{Score}(C \cup \{(P, \mathcal{P})\})$ equals $c_1 - c_3 \cdot \frac{|\texttt{cov}(P,\mathcal{P})|}{|\mathcal{X}|}$. We obtain $|\texttt{cov}(P, \mathcal{P})|$ using constant additional time throughout our algorithm. This leads to the following algorithm:

▶ **Clustering 1** (SC-$\ell((c_1, c_2, c_3), \mathcal{T})$ under k-centre clustering). *We run Algorithm 1 (c_1, c_2, c_3) for exponentially increasing $\Delta \in (2, 4, \ldots)$ where the subroutine f is **SC**(max, ℓ, Δ, $\mathcal{T}$). We then output the maximum-score clustering C over all runs.*

Algorithm 1 `k_centre`(`vector` (c_1, c_2, c_3), `value` Δ, `trajectory` $\mathcal{T}$, `subroutine` f).

```
C ← ∅, T ← 𝒯.
while |T| > 0 do
    (P, 𝒫) ← the result of applying f to (Δ, T)
    if Score(C ∪ {(P, 𝒫)}) > Score(C) then
        C.add((P, 𝒫)).
        T ← T − 𝒫.
    else
        Break.
    end if
end while
return C
```

We may also fix m instead of ℓ. Or, formally:

▶ **Clustering 2** (SC-$m((c_1, c_2, c_3), \mathcal{T})$ under k-centre clustering). *We run Algorithm 1 (c_1, c_2, c_3) for exponentially increasing $\Delta \in (2, 4, \ldots)$ where the subroutine f is **SC**(m, max, Δ, $\mathcal{T}$). We then output the maximum-score clustering C over all runs.*

5.2 k-means clustering.

Our algorithm for k-means clustering is less straightforward. To get good performance we must iterate over different choices of Δ across the algorithm. Consider a clustering C and $(P, \mathcal{P}) = \mathbf{SC}(\max, \ell, \Delta, \mathcal{T} - \texttt{cov}(C))$ for some Δ. Under k-centre clustering, the more points a Δ-cluster covers, the better its scoring contribution. For k-means clustering this is no longer true as the contribution of $(P, \mathcal{P})$ to $\texttt{Score}(C \cup \{(P, \mathcal{P})\})$ equals:

$$\text{Contribution of } (P, \mathcal{P}) \text{ to } \texttt{Score}(C) = c_1 - c_3 \cdot \frac{|\texttt{cov}(P, \mathcal{P})|}{|\mathcal{X}|} + \sum_{P' \in \mathcal{P}} c_2 \cdot \mathcal{D}_F(P, P').$$

We abuse the fact that subtrajectories share no vertices to simplify the above expression to:

$$\text{Contribution of } (P, \mathcal{P}) \text{ to } \texttt{Score}(C) = c_1 + \sum_{P' \in \mathcal{P}} \left(c_2 \cdot \mathcal{D}_F(P, P') - c_3 \frac{|P'|}{|\mathcal{X}|} \right).$$

Consider a set of "candidate" clusters S. To select the best cluster from S, we use the same greedy set cover technique as [1]. We assign to every cluster a evaluation value. This is the ratio between how much it reduces the score, and how much it increases the score. I.e.,

$$\texttt{evaluation}(P, \mathcal{P}) = \frac{\sum_{P' \in \mathcal{P}} c_3 \frac{|P'|}{|\mathcal{X}|}}{c_1 + \sum_{P' \in \mathcal{P}} c_2 \cdot \mathcal{D}_F(P, P')}$$

We then select the cluster with the best evaluation among all $(P, \mathcal{P}) \in S$.

▶ **Clustering 3** (SC-$\ell((c_1, c_2, c_3), \mathcal{T})$ under k-means clustering). *We run Algorithm 2 where the subroutine f is $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$.*

▶ **Clustering 4** (SC-$m((c_1, c_2, c_3), \mathcal{T})$ under k-means clustering). *We run Algorithm 2 where the subroutine f is $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$.*

Algorithm 2 `k_means`(`vector` (c_1, c_2, c_3), `trajectory` $\mathcal{T}$, `subroutine` f).

```
C ← ∅, T ← 𝒯.
while |T| > 0 do
    (P, 𝒫) ← null
    for Δ ∈ (2, 4, ...) do
        (Q, 𝒬) ← the result of applying f to (Δ, 𝒯)
        if evaluation(Q, 𝒬) > evaluation(P, 𝒫) then
            (P, 𝒫) ← (Q, 𝒬).
        end if
    end for
    if Score(C ∪ {(P, 𝒫)}) > Score(C)) then
        C.Add((P, 𝒫)).
        T ← T − 𝒫.
    else
        Break.
    end if
end while
return C
```

6 A new algorithm as a subroutine

Our previous clusterings 1-4 fix a parameter ℓ (or m) and subsequently *only* produce clusters $(P, \mathcal{P})$ of maximal cardinality with $|P| = \ell$ (or of maximum length with $|\mathcal{P}| = m$).

This approach has three clear downsides:

1. To find a suitable ℓ (or m) you have to train the algorithm on a smaller data set.
2. Using multiple values of ℓ (or m) to broaden the search space is computationally expensive.
3. Even if we fix constantly many choices for ℓ and m, the corresponding algorithm has a heavily restricted solution space which affects the quality of the output.

To alleviate these downsides, we design a new algorithm based on the following observation:

▶ **Definition 5.** *For fixed $(\Delta, \mathcal{T})$, we define a bivariate Boolean function $\theta(m, \ell)$ where $\theta(m, \ell)$ is 1 if and only if there exists a Δ-cluster $(P, \mathcal{P})$ of $\mathcal{T}$ with $m = |\mathcal{P}|$ and $\ell = |P|$.*

The functions $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ always output a Δ-cluster $(P, \mathcal{P})$ where $\theta(|P|, |\mathcal{P}|)$ lies on the Pareto Front of $\theta(\cdot, \cdot)$. If clusters on the Pareto front of $\theta(\cdot, \cdot)$ are good candidates to include in a clustering, then it may be worthwhile to directly compute an approximation of this Pareto front as a candidate set:

▶ **Definition 6.** *Given $(\Delta, \mathcal{T})$, a set of Δ-clusters S is a 2-approximate Pareto front if for every Δ-cluster $(P, \mathcal{P})$ there exists a $(Q, \mathcal{Q}) \in S$ with: $|\mathcal{P}| \leq |\mathcal{Q}|$ and $|P| \leq 2|Q|$.*

Our new algorithm, $\mathbf{PSC}(\Delta, \mathcal{T})$, *iterates* over a 2-approximate Pareto front. Storing this front explicitly uses $O(n^2 \log n)$ space. Instead, we generate these clusters on the fly.

High-level description. Let $B(\mathcal{T})$ be a balanced binary tree over $\mathcal{T}$. Let $S(\Delta, \mathcal{T})$ start out as empty. Our algorithm $\mathbf{PSC}(\Delta, \mathcal{T})$ (defined by Algorithm 3) does the following:

- For each inner node in B, denote by $\mathcal{T}[a, b]$ the corresponding subtrajectory of $\mathcal{T}$.
- For every prefix or suffix P of $\mathcal{T}[a, b]$, let $(P, \mathcal{P})$ be a Δ-cluster maximising $|\mathcal{P}|$.
- We add $(P, \mathcal{P})$ to $S(\Delta, \mathcal{T})$.

Computing all prefix clusters for $\mathcal{T}[a, b]$. Given a trajectory $\mathcal{T}[a, b]$ and our current set $S(\Delta, \mathcal{T})$, we simultaneously compute for all prefixes P of $\mathcal{T}[a, b]$ a maximum-cardinality cluster $(P, \mathcal{P})$ and add this cluster to $S(\Delta, \mathcal{T})$. We define an analogue of Definition 1:

▶ **Definition 7.** *Consider row a in $M_\Delta(\mathcal{T}, \mathcal{T})$. For $(x, y) \in M_\Delta(\mathcal{T}, \mathcal{T})$ let $\texttt{col}_a(x, y)$ be the maximum column $d \leq y$ such that there exists a directed path from (x, y) to (a, d).*

Given $\mathcal{T}[a, b]$ we initialize our algorithm by storing integers a and $c \leftarrow a$. We increment c and maintain for all $y \in [n]$ the value $\texttt{col}_a(c, y)$. This uses $O(n)$ space, as does storing row c. Given this data structure, we alternate between a `StepSecond` and `Query` subroutine:

`StepSecond`: We increment c by 1. For each $y \in [n]$, we check if $d(\mathcal{T}(c), \mathcal{T}(y)) > \Delta$. If it is then (c, y) is an isolated vertex in the graph $M_\Delta(\mathcal{T}, \mathcal{T})$. Thus, there exists no directed path from (c, y) to any vertex in row a and we set $\texttt{col}_a(c, y) = -\infty$. Otherwise, $\texttt{col}_a(c, y) = \max\{\texttt{col}_a(\alpha, \beta) \mid \alpha \in \{c, c-1\}, \beta \in \{y, y-1\}\}$ which we compute in $O(1)$ time.

`Query`: For a prefix $P = \mathcal{T}[a, c]$ of $\mathcal{T}[a, b]$ we compute a maximum-cardinality Δ-cluster $(P, \mathcal{P})$ using the values $\texttt{col}_a(c, y)$ (see Algorithm 4).

Note that Algorithm 4 is essentially the same as the algorithm in [12] that we described in Section 5. Thus, it computes a maximum-cardinality cluster. The key difference is that we, through $\texttt{col}_a(b, j_2)$, avoid walking through the free-space matrix in $O(\ell)$ time to compute j_1. We denote by `Query`'$(T[c, b], \Delta, \mathcal{T}, S(\Delta, \mathcal{T}))$ the function which computes for a suffix $P = \mathcal{T}[c, b]$ of $\mathcal{T}[a, b]$ a maximum-cardinality cluster $(P, \mathcal{P})$ in an analogous manner.

▶ **Lemma 8.** *The set $S(\Delta, \mathcal{T})$ is a 2-approximate Pareto front.*

Proof. Consider any Δ-cluster $(P, \mathcal{P})$. Let $P = \mathcal{T}[a, b]$. Let $\mathcal{T}[i, k]$ be the minimal subtrajectory in our tree that has $\mathcal{T}[a, b]$ as a subtrajectory. Let the children of $\mathcal{T}[i, k]$ in our tree be $\mathcal{T}[i, j]$ and $\mathcal{T}[j + 1, k]$. Then, $a \leq j < b$ by minimality of $\mathcal{T}[i, k]$. Hence, $\mathcal{T}[a, j]$ is a suffix of $\mathcal{T}[i, j]$, and $\mathcal{T}[j + 1, b]$ is a prefix of $\mathcal{T}[j + 1, k]$. There must exist a trajectory $P' \in \{\mathcal{T}[a, j], \mathcal{T}[j + 1, b])\}$ which contains at least half as many vertices as $T[a, b]$. Since P' is either a prefix or suffix of a subtrajectory in our tree, we must have added some maximum-cardinality cluster $(P', \mathcal{P}')$ to $S(\Delta, \mathcal{T})$. Moreover, since P' is a subtrajectory of P, $|\mathcal{P}'| \geq |\mathcal{P}|$ (since any discrete walk from row b to row a is also a discrete walk in the Free-space matrix from row d to row c for $[c, d] \subseteq [a, b]$). ◀

▶ **Theorem 9.** *The algorithm PSC(Δ, $\mathcal{T}$) iterates over a 2-approximate Pareto front of $\mathcal{T}$ in $O(z \log^4 n)$ time using $O(n \log n$ space. Here, z denotes the number of zeroes in $M_\Delta(\mathcal{T}, \mathcal{T})$.*

Proof. Consider the balanced binary tree $B(\mathcal{T})$. Each level of depth of $B(\mathcal{T})$ corresponds to a set of subtrajectories that together cover $\mathcal{T}$. For every such subtrajectory $\mathcal{T}[a, b]$, both `StepSecond` and `Query` get invoked for each row $c \in [a, b]$. Since the tree has depth $\log n$, it follows that we invoke these functions $O(\log n)$ times per row c of $M_\Delta(\mathcal{T}, \mathcal{T})$ (i.e. $O(n \log n)$ times in total). Naively, both the `StepSecond` and `Query` functions iterate over all columns in their given row c. However, just as in Section 5, we note that our functions skip over all (j_2, c) for which $d(\mathcal{T}(j_2), \mathcal{T}(c)) > \Delta$. It follows that we may use our row generation trick to spend $O(Z_c \log^3 n)$ time instead (Z_c denotes the number of zeroes in row c). Since each row c gets processed $O(\log n)$ times, this upper bounds our running time by $O(z \log^4 n)$.

For space, we note that invoking `StepSecond` and `Query` for a row c need only rows c and $c - 1$ in memory. So the total memory usage is $O(n)$. Explicitly storing $S(\Delta, \mathcal{T})$ takes $O(n^2 \log n)$ space. However, since we only iterate over $S(\Delta, \mathcal{T})$, the total space usage remains $O(n)$. The range tree therefore dominates the space with its $O(n \log n)$ space usage. ◀

6.1 Greedy clustering using PSC(Δ, $\mathcal{T}$)

We reconsider Algorithms 1 and 2 in Section 5 for k-centre and k-means clustering. As written, these algorithms invoke a subroutine f that returns a single cluster $(P, \mathcal{P})$. Then, the algorithm computes whether it wants to add $(P, \mathcal{P})$ to the current clustering. We can

Algorithm 3 **PSC**(Δ, $\mathcal{T}$).

```
S(Δ, T) ← ∅.
B(T) ← balanced binary tree over T.
for all T[a, b] ∈ B(T) do
    for all c ∈ [a, b] do
        StepSecond(Δ, T, c).
        Query(T[a, c], Δ, T, S(Δ, T)).
    end for
    for all c ∈ [a, b] in decreasing order do
        StepSecond(Δ, T, c).
        Query'(T[c, b], Δ, T, S(Δ, T)).
    end for
end for
return S(Δ, T)
```

Algorithm 4 Query($P = \mathcal{T}[a, c]$, Δ, $\mathcal{T}$, $S(\Delta, \mathcal{T})$).

$(c, j_2) \leftarrow M_\Delta(\mathcal{T}, \mathcal{T})[c, n]$, $\mathcal{P} \leftarrow \emptyset$
while $j_2 > 0$ **do**
  if $\texttt{col}_a(c, j_2) \neq -\infty$ **then**
    $j_1 \leftarrow \texttt{col}_a(b, j_2)$.
    Add $\mathcal{T}[j_1, j_2]$ to $\mathcal{P}$.
    $(c, j_2) \leftarrow (c, j_1 - 1)$.
  else
    $(c, j_2) \leftarrow (c, j_2 - 1)$.
  end if
end while
Add $(P, \mathcal{P})$ to $S(\Delta, \mathcal{T})$.

adapt these algorithms to this new function. For f, we invoke $\mathbf{PSC}(\Delta, \mathcal{T})$. This function iterates over a set of clusters $(P, \mathcal{P})$. For each cluster $(P, \mathcal{P})$ that we encounter, we let either Algorithm 1 or 2 compute whether it wants to add $(P, \mathcal{P})$ to the current clustering.

7 Experiments

We propose three new greedy clustering algorithms for k-centre clustering:

1. `SC-`ℓ runs Algorithm 1, for exponentially scaling Δ, with $f \leftarrow \mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$,
2. `SC-`m runs Algorithm 1, for exponentially scaling Δ, with $f \leftarrow \mathbf{SC}(m, \max, \Delta, \mathcal{T})$, and
3. `PSC` runs Algorithm 1, for exponentially scaling Δ, with $f \leftarrow \mathbf{PSC}(\Delta, \mathcal{T})$.

Our algorithms `SC-`ℓ and `SC-`m need to fix a parameter ℓ and m respectively. We choose $\ell \in \{4, 8, 16, 32, 64\}$ and $m \in \{2, 4, 8, 16\}$ and do a separate run for each choice. Thus, together with `PSC`, we run ten different algorithm configurations. We additionally compare to the subtrajectory clustering algorithm from [10], which is oblivious to the scoring function.

- `Map-construction` runs the algorithm in [10]. We terminate this map construction algorithm as soon as it has constructed a clustering of $\mathcal{T}$.

***k*-means clustering.** For k-means clustering, we obtain three algorithms by running Algorithm 2 instead. Note that our clustering algorithms invoked Algorithm 1 across several runs for exponentially scaling Δ. Our clustering algorithms do not invoke Algorithm 2 in this manner. Instead, Algorithm 2 considers exponentially scaling Δ's internally. Our experiments show that this heavily impacts the clustering's performance on large input.

We again obtain ten algorithm configurations. In addition, we run the map construction algorithm from [10] and the k-means subtrajectory clustering algorithms from [1]:

- `Envelope` runs the algorithm from [1] with the scoring parameters (c_1, c_2, c_3).

Experimental setup. We design different experiments to measure performance and the scoring function. For performance, we run fourteen experiments across the five real-world data sets. Seven experiments for k-centre clustering, and seven for k-means clustering. Each experiment ran three times, using a different vector (c_1, c_2, c_3). Each experiment runs 11 programs for k-centre clustering, or 12 for k-means clustering. If a program fails to terminate after 24 hours, we do not extract any output or memory usage.

We restrict scoring measurements to the three smaller or subsampled real-world data sets. This is necessary to make `Map-construct` and `Envelope` terminate. We add six synthetic data sets. We only compare under k-means clustering since only our algorithms can optimise for k-centre clustering. The experiments were conducted on a machine with a 4.2GHz AMD Ryzen 9 7950X3D and 128GB memory. Over 90 runs hit the timeout. Running the total set of experiments sequentially takes over 100 days on this machine.

The full results per experiment are organised in tables in the full version. Here, we restrict our attention to only three plots.

Choosing (c_1, c_2, c_3). Recall that c_1 weighs the number of clusters in the clustering and that c_3 weighs the fraction of uncovered points. We follow the precedent set in [1] and choose $c_1 = 1$, and c_3 equal to the number of trajectories in $\mathcal{T}$. Finally, c_2 weighs the Fréchet distance across clusters. For each data set, we first find a choice for c_2 such the algorithms lead to a non-trivial clustering. We then also run the algorithms using the scoring vectors $(c_1, 10 \cdot c_2, c_3)$ and $(c_1, 0.1 \cdot c_2, c_3)$, for a total of three scoring vectors per data set.

Comparing quality. To compare solution quality, we measure the scoring function and other qualities such as: the total number of clusters in the clustering, the maximum Fréchet distance from a subtrajectory to its centre, and the average Fréchet distance across clusters. Our competitor algorithms frequently time out on the real-world data sets. Hence, the qualitative comparison also makes use of our synthetic data set. We use three different configurations of our synthetic set to create input of different size. We choose to compare only under k-means clustering for a more fair comparison to other algorithms. Indeed, the algorithm by [1] was designed for the k-means metric ([10] is scoring function oblivious). Consequently, comparing under k-centre would only give us an unfair advantage.

7.1 Results for k-means clustering

The bar plot in Figure 6 visualizes how the algorithms score. Figures 7 and 8 show, respectively, a logarithmic-scale plot of the time and space usage. These plots use the "medium" choice for c_2 for each data set. Note that our choice of c_2 is very similar across the data sets in the figure. This is because *Athens*, *Berlin* and *Chicago* come from the same source, and thereby have the same scaling. We subsequently designed our synthetic data to match this scaling. Plots for the other scoring vector choices are contained in the full version.

Clustering quality. Figure 6 shows that `Map-construct` (which is oblivious to the scoring function) scores poorly on all inputs. One reasons for this is that it always outputs a large number of clusters. The `Envelope` algorithm, which was specifically designed for this scoring function, almost always obtains the best score.

For any data set, for any scoring vector, we always find a choice for ℓ (or m) where `SC-`ℓ and `SC-`m obtain a comparable score to `Envelope`. The algorithm `PSC` scores best out of all newly proposed clustering algorithms. Across all data sets and scoring functions it frequently competes with, and sometimes even improves upon, the score of `Envelope`.

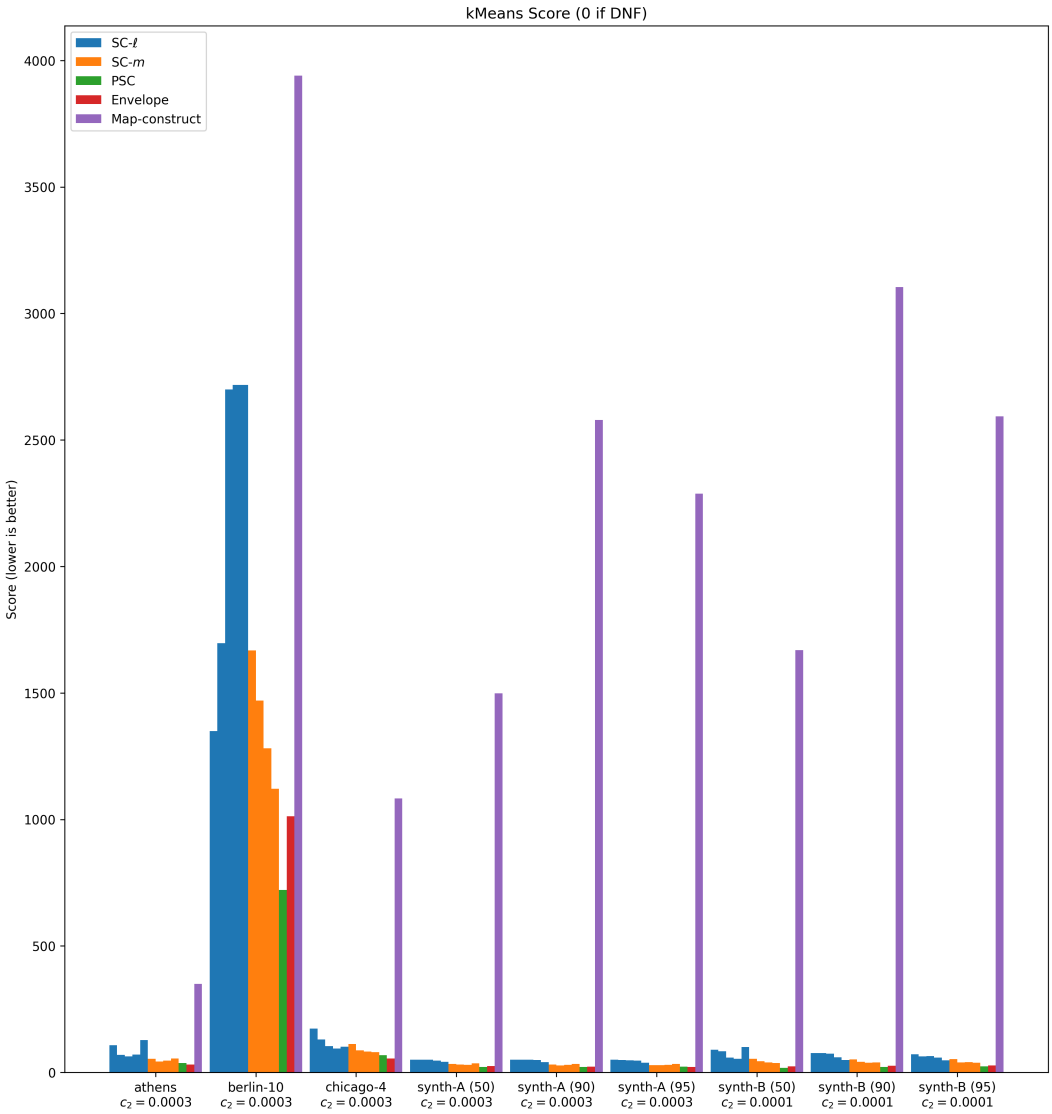


Figure 6 ■ k-means clustering ■ k-means score ■.

Other quality metrics. We briefly note that the full data in the appendix also considers other quality metrics: the number of clusters and the maximum and average Fréchet distance within a cluster. Perhaps surprisingly, `Map-construct` does not perform better on the latter two metrics even though it outputs many clusters. `Envelope` scores worse on these metrics across nearly all data sets and scoring vectors. However, it stays within a competitive range.

Runtime comparison. Figure 7 shows the runtime performance in logarithmic scale. On the *Berlin* data set, no algorithm terminates in time. On *Drifter* and *UniD*, only `PSC` terminates within the time limit. In the figure, we consider the synthetic data sets their place.

`Envelope` may produce high-quality clusterings, but takes significant time to do so. Despite using considerably less memory than `Map-construct`, it is consistently more than a factor two slower. Our algorithms `SC-`ℓ and `SC-`m can always find a choice of ℓ or m such that previous algorithms are orders of magnitude slower. Our `PSC` algorithm is always orders of magnitude faster than the existing clustering algorithms. This is especially true on larger data sets, which illustrates that our algorithms have better scaling.

When we compare our own algorithms, we note that we can frequently find a choice for ℓ such that SC-ℓ is significantly faster than SC-m and PSC. However, when considering the runtime to quality ratio, PSC appears to be a clear winner.

Memory consumption. Figure 8 shows memory performance on a logarithmic scale. The figure excludes *Drifter*, *UniD* and *Berlin* for the same reason as when comparing running times. We record zero memory usage whenever an algorithm times out.

We observe that `Map-construct` uses an order of magnitude more memory than its competitors. This is because it stores up to $O(n^2)$ candidate clusters in memory, together with the entire free-space matrix $M_\Delta(\mathcal{T}, \mathcal{T})$ for various choices of Δ. `Envelope` also has a large memory footprint and uses at least a factor 100 more memory on all of the inputs.

When comparing our own implementations, we observe that SC-ℓ and PSC have a very small memory footprint. These algorithms use $O(n\ell)$ and $O(n)$ space. Since we fix ℓ to be a small constant, both algorithms use near linear space. The space usage of SC-m is $O(n\ell)$ where ℓ is the length of the longest subtrajectory P such that there exists a Δ-cluster $(P, \mathcal{P})$ with $|\mathcal{P}| \geq m$. The data shows that for many choices of (m, Δ), this $O(n\ell)$ space is considerable. Yet, it is still considerably lower than pre-existing algorithms.

7.2 Results for k-centre clustering

When comparing score, runtime, or memory usage under k-centre clustering, we reach the same conclusions. The only difference is that the performance of SC-m becomes more competitive to SC-ℓ and PSC on large data sets. This is because $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ uses much time and space whenever both the data set and Δ are large. Our k-means clustering meta-algorithm considers a large Δ each round, and invokes $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ every round.

Our k-centre clustering meta-algorithm has a separate run for each Δ. If Δ is large, it terminates quickly because it adds very high-cardinality clusters to the clustering. This speeds up all our implementations, but SC-m the most.

8 Conclusion

We proposed new subtrajectory clustering algorithms. Our first two approaches use the algorithms $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$ by BBGLL [12]. We use these algorithms as a subroutine in a greedy clustering algorithm to cluster a set of trajectories $\mathcal{T}$.

Consider the Boolean function $\theta(\ell, m)$ that outputs `true` if there exists a Δ-cluster $(P, \mathcal{P})$ of $\mathcal{T}$ with $|P| = \ell$ and $|\mathcal{P}| = m$. We observe that these functions always output a point on the Pareto front of this function. We show a new algorithm $\mathbf{PSC}(\Delta, \mathcal{T})$ that, instead of returning a single cluster on this Pareto front, iterates over a 2-approximation of this Pareto front instead. Intuitively, this function iterates over a broader set of candidate clusters that have comparable quality to any output of $\mathbf{SC}(\max, \ell, \Delta, \mathcal{T})$ and $\mathbf{SC}(m, \max, \Delta, \mathcal{T})$. We create a greedy clustering algorithm PSC using this algorithm as a subroutine.

Our analysis shows that our new clustering algorithms significantly outperform previous algorithms in running time and space usage. We primarily use the scoring function from [1] to measure clustering quality and observe that our algorithms give competitive scores. For other quality metrics, our algorithms often perform better than their competitors. We observe that PSC is the best algorithm when comparing the ratio between score and performance.

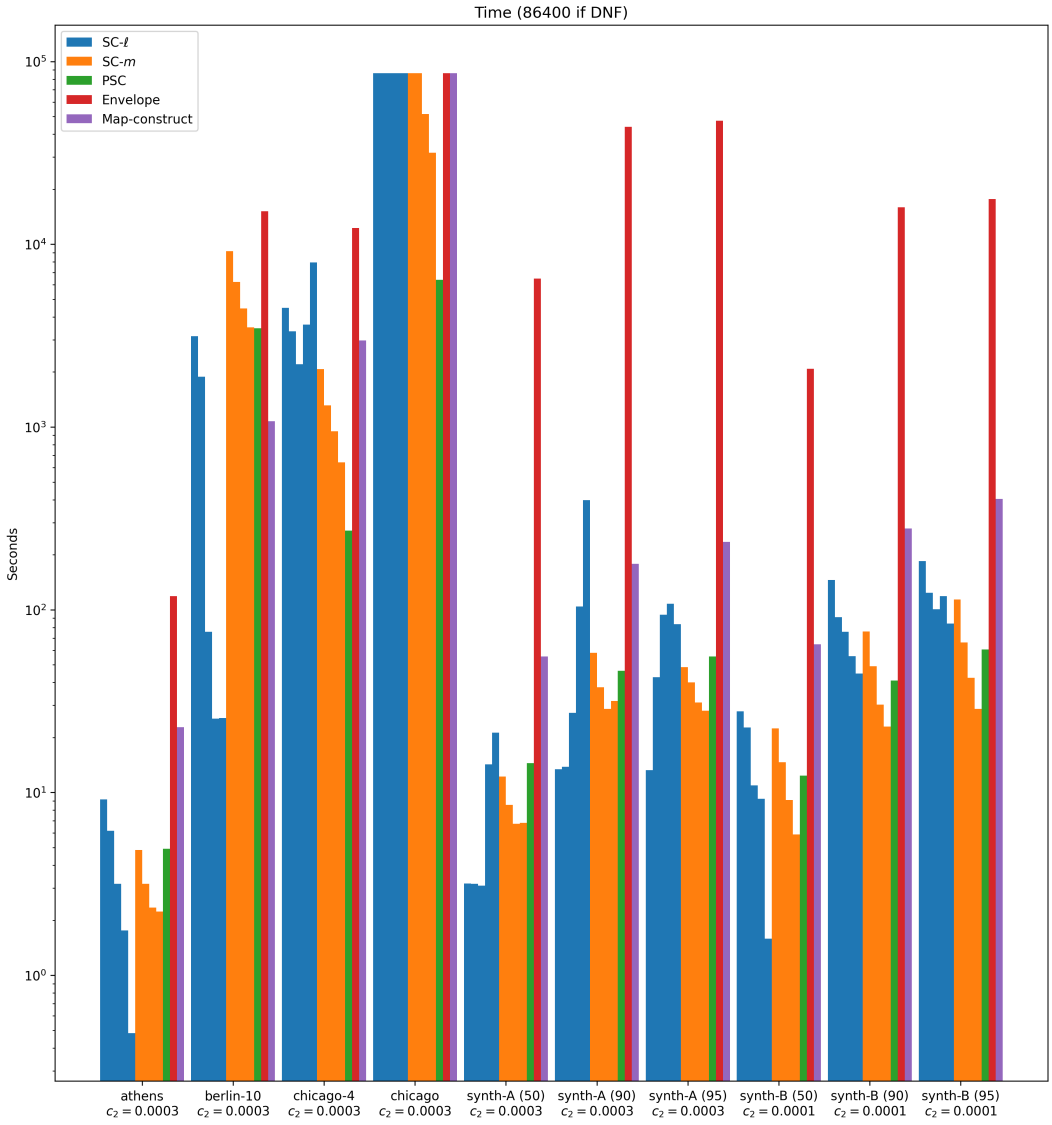


Figure 7 ■ k-means clustering ■ Running time ■ Logarithmic scaling.

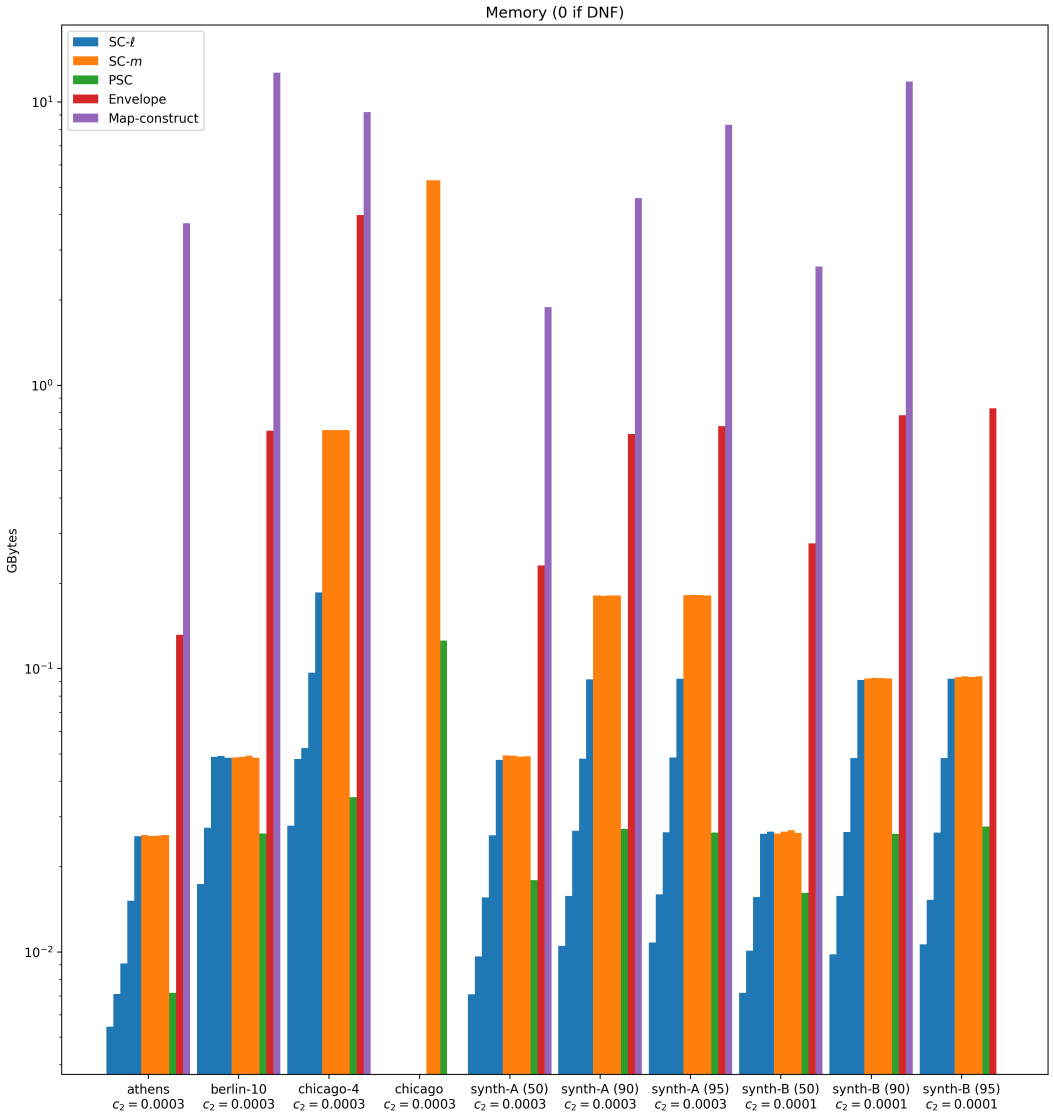


Figure 8 ■ k-means clustering ■ Memory usage ■ Logarithmic scaling
Whenever a run did not terminate, we display a memory usage of zero.

References

1 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.

2 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.

3 Mahmuda Ahmed and Carola Wenk. Constructing street networks from gps trajectories. *European Symposium on Algorithms (ESA)*, pages 60–71, 2012. doi:10.1007/978-3-642-33090-2_7.

4 Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Subtrajectory Clustering: Finding Set Covers for Set Systems of Subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, 2023. doi:10.57717/cgt.v2i1.7.

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 James Biagioni and Jakob Eriksson. Map inference in the face of noise and disparity. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 79–88, 2012. doi:10.1145/2424321.2424333.

7 Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. A survey on trajectory clustering analysis. *arXiv preprint arXiv:1802.06971*, 2018. arXiv:1802.06971.

8 Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.28.

9 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.

10 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.

11 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Rodrigo I Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Roadster: Improved algorithms for subtrajectory clustering and map construction. *Computers & Geosciences*, 196:105845, 2025. doi:10.1016/j.cageo.2024.105845.

12 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.

13 Kevin Buchin, Maike Buchin, Marc Van Kreveld, Maarten Löffler, Rodrigo I Silveira, Carola Wenk, and Lionov Wiratma. Median trajectories. *Algorithmica*, 66:595–614, 2013. doi:10.1007/s00453-012-9654-2.

14 Lili Cao and John Krumm. From gps traces to a routable road map. *ACM international conference on advances in geographic information systems (SIGSPATIAL)*, pages 3–12, 2009. doi:10.1145/1653771.1653776.

15 Chen Chen, Hao Su, Qixing Huang, Lin Zhang, and Leonidas Guibas. Pathlet learning for compressing and planning trajectories. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 392–395, 2013. doi:10.1145/2525314.2525443.

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 Jonathan J Davies, Alastair R Beresford, and Andy Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006. doi:10.1109/MPRV.2006.83.

18 university of Technology Eindhoven. MoveTK: the movement toolkit. http://https://github.com/movetk/movetk, 2020.

19 Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. *ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*, pages 63–72, 1999. doi:10.1145/312129.312197.

20 Joachim Gudmundsson, Andreas Thom, and Jan Vahrenhold. Of motifs and goals: mining trajectory data. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 129–138, 2012. doi:10.1145/2424321.2424339.

21 Joachim Gudmundsson and Nacho Valladares. A GPU approach to subtrajectory clustering using the fréchet distance. *ACM nternational Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 259–268, 2012. doi:10.1145/2424321.2424355.

22 Joachim Gudmundsson and Sampson Wong. Cubic upper and lower bounds for subtrajectory clustering under the continuous fréchet distance. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 173–189, 2022. doi:10.1137/1.9781611977073.9.

23 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.

24 Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *The VLDB Journal*, 24:169–192, 2015. doi:10.1007/s00778-011-0262-6.

25 Sophia Karagiorgou and Dieter Pfoser. On vehicle tracking data-based road network generation. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 89–98, 2012. doi:10.1145/2424321.2424334.

26 Sophia Karagiorgou, Dieter Pfoser, and Dimitrios Skoutas. Segmentation-based road network construction. *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 460–463, 2013. doi:10.1145/2525314.2525460.

27 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. *ACM International Conference on Management of Data (SIGMOD)*, pages 593–604, 2007. doi:10.1145/1247480.1247546.

28 Vanessa Lago Machado, Ronaldo dos Santos Mello, Vânia Bogorny, and Geomar André Schreiner. A survey on the computation of representative trajectories. *GeoInformatica*, pages 1–26, 2024. doi:10.1007/s10707-024-00514-y.

29 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.

30 Cynthia Sung, Dan Feldman, and Daniela Rus. Trajectory clustering for motion prediction. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1547–1552, 2012. doi:10.1109/IROS.2012.6386017.

31 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.

Computing Non-Obtuse Triangulations with Few Steiner Points

Mikkel Abrahamsen
University of Copenhagen, Denmark

Florestan Brunck
University of Copenhagen, Denmark

Jacobus Conradi
University of Bonn, Germany

Benedikt Kolbe
Hausdorff Center for Mathematics, Lamarr Institute for Machine Learning and Artificial Intelligence, University of Bonn, Germany

André Nusser
Université Côte d'Azur, CNRS, Inria, Sophia Antipolis, France

Abstract

We present the winning implementation of the Seventh Computational Geometry Challenge (CG:SHOP 2025). The task in this challenge was to find non-obtuse triangulations for given planar regions, respecting a given set of constraints consisting of extra vertices and edges that must be part of the triangulation. The goal was to minimize the number of introduced Steiner points. Our approach is to maintain a constrained Delaunay triangulation, for which we repeatedly remove, relocate, or add Steiner points. We use local search to choose the action that improves the triangulation the most, until the resulting triangulation is non-obtuse.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases non-obtuse triangulation, local search, competition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.79

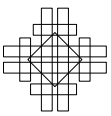
Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/JacobusTheSecond/nonObtuseTri` [1], archived at `swh:1:dir:b6301970ab626c084f309f9b5e6b91e9acf24949`

Funding *Mikkel Abrahamsen*: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.
Florestan Brunck: Supported by a DNRF Chair grant from the Danish National Research Foundation and partially by the BARC grant of the Villum Foundation.
Jacobus Conradi: Funded by the iBehave Network: Sponsored by the Ministry of Culture and Science of the State of North Rhine-Westphalia.
Benedikt Kolbe: This research has partly been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence.
André Nusser: Supported by the France 2030 investment plan managed by the ANR as part of the Initiative of Excellence of Université Côte d'Azur with reference number ANR-15-IDEX-01.

Acknowledgements We thank the Challenge organizers and other competitors for their time, feedback, and making this whole event possible. We thank Max Kanold for invaluable input enabling high quality visualizations which made later discussions on how to improve our algorithm all the more fruitful. We gratefully acknowledge the access to the Marvin cluster of the University of Bonn.

© Mikkel Abrahamsen, Florestan Brunck, Jacobus Conradi, Benedikt Kolbe, and André Nusser;
licensed under Creative Commons License CC-BY 4.0



1 Introduction

The Computational Geometry Challenge (CG:SHOP) is an annual competition on geometric optimization. The 2025 edition involved finding non-obtuse triangulations of a given planar straight-line graph (PSLG), a planar embedding of a graph where edges are represented as line segments; see Figure 1. The input consists of a PSLG $G = (V, E)$ and a simple polygon P with vertices in V. The goal is to output a non-obtuse triangulation $\mathcal{T}$ of P with vertices V' so that $V \subseteq V'$ and the edges of $\mathcal{T}$ cover E. The vertices $V' \setminus V$ are called *Steiner points* and the objective is to minimize them. We refer to the survey [10] for further details.

The problem of finding non-obtuse triangulations has a long history, dating back to the 1960s [6, 4, 3, 11, 13, 5]. In the 2-dimensional setting, the latest development is Bishop's algorithm – the only one to provably output a non-obtuse triangulation of any PSLG with a number of Steiner points polynomial in the vertices of the input. In general, the computational complexity of minimizing the number of Steiner points or triangles remains unknown, and the problem may be intractable already for simple polygons.

In this paper, we present our winning implementation. The instance set of the callenge consisted of 150 instances with $|V|$ and $|E|$ as large as 250. Our computed solutions were the best for 116 instances, while the runner-up team, "Gwamegi" [2], was best in 4. For the remaining 30 instances, we matched Gwamegi in number of Steiner points.

2 Algorithm

Let us first outline the idea behind our algorithm. Since a non-obtuse triangulation is a Delaunay triangulation (the local Delaunay condition is respected by the inscribed angle theorem), our algorithm maintains a (constrained) Delaunay triangulation (CDT) that gets modified successively. Whenever a obtuse triangle is present, the algorithm makes a modification by removing, relocating, or adding a Steiner point and updating the triangulation accordingly. More precisely, the algorithm operates in rounds, and in each round we consider a set of candidates for local changes and choose the most promising one. In each round, the input is the original instance and the current set of Steiner points, which have been added in the previous rounds. We compute a set of candidate actions, of which there are three types: (i) the insertion of a new Steiner point, (ii) the relocation of an existing Steiner point, and (iii) the replacement of two adjacent Steiner points with a single Steiner point. For

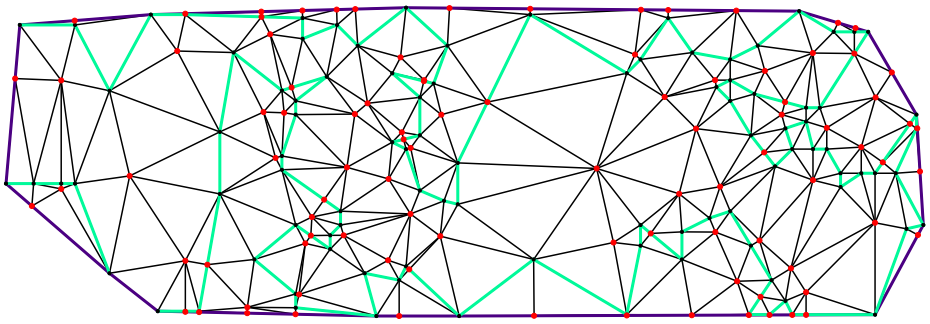


Figure 1 A non-obtuse triangulation produced by our implementation. Here, as in the rest of the paper, the region to be triangulated is enclosed by the fat curve, the given vertices are black, and the constraint edges are green. The Steiner points are red.

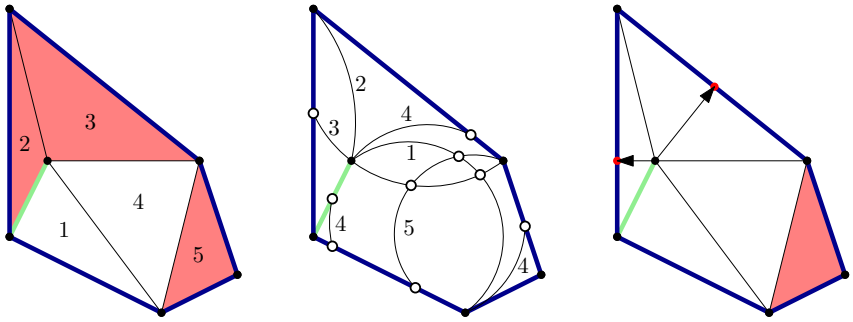


Figure 2 *left:* CDT with obtuse triangles marked in red; *middle:* its `circleArrangement`; *right:* the two `altitudeDrop` points of the two upper obtuse triangles and its resulting CDT.

each action, we compute the resulting triangulation after applying the action. Each of the resulting triangulations is evaluated using a cost function (given in Section 2.3, Equation (1)), which involves the number of Steiner points and the number of obtuse triangles of different types. The output produced by the round is the triangulation that minimizes the cost. The process terminates when the CDT has no obtuse triangles left. In particular, our algorithm produces a non-obtuse triangulation whenever it terminates.

The way in which we insert a new Steiner point is inspired by the work of Erten and Ungör [9, 8, 7]. They described a general approach to finding triangulations avoiding small or large angles, where new Steiner points are chosen as optimal points along the Voronoi diagram (see [7, Section 2.1]). We extend this idea to a visibility-constrained Voronoi diagram [14] which takes into account the constraints of the PSLG.

2.1 Primitives

We first describe the geometric primitives that we are basing our algorithm on:

- `altitudeDrop`: Given an obtuse triangle, compute the unique point on the longest side that splits the obtuse triangle into two right-angled triangles (Figure 2, right).
- `polygonCenter`: Given a simple polygon, compute a center point such that connecting each vertex of the polygon with the center point results in a non-obtuse triangulation of the polygon, or decide no such point exists. Such a point can be computed as follows: Observe that for any two points p and q, a point x forms a non-obtuse triangle pqx if and only if x lies inside the slab defined by the two hyperplanes rooted at p and q with normal $q - p$ and outside the disk centered at $\frac{p+q}{2}$ with radius $\frac{\|q-p\|}{2}$. Pick any point in the intersection of these sets defined by all consecutive pairs of vertices p and q on the boundary of the polygon (Figure 3).
- `visibilityVoronoi`: Given a CDT, compute the visibility-constrained Voronoi diagram (sometimes called bounded Voronoi diagram) [14]. (Figure 4, top right).
- `clippedCircumcircle`: Let T be a triangle in a CDT and C its circumcircle. The disk bounded by C may be partitioned into multiple pieces by the constraints of the CDT. Compute the piece of the partition containing T (Figure 4, bottom left).
- `circleArrangement`: Given a CDT, compute first the `clippedCircumcircle` for each triangle, then the arrangement of these clipped circumcircles (Figure 2, middle).

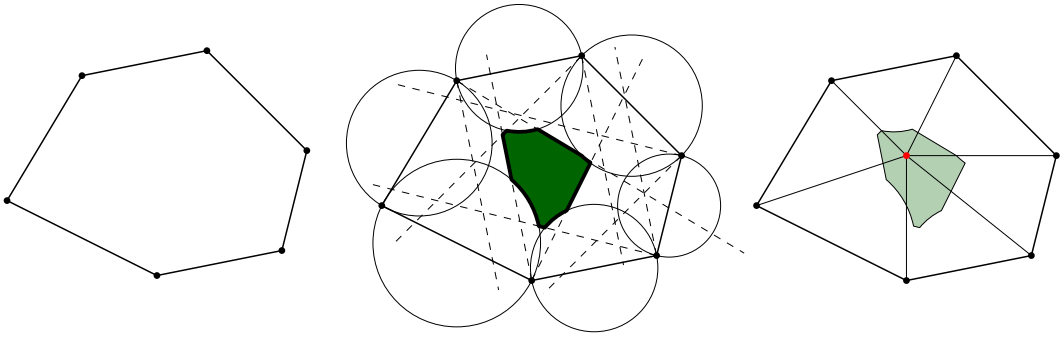


Figure 3 Illustration of the computation of the `polygonCenter` primitive.

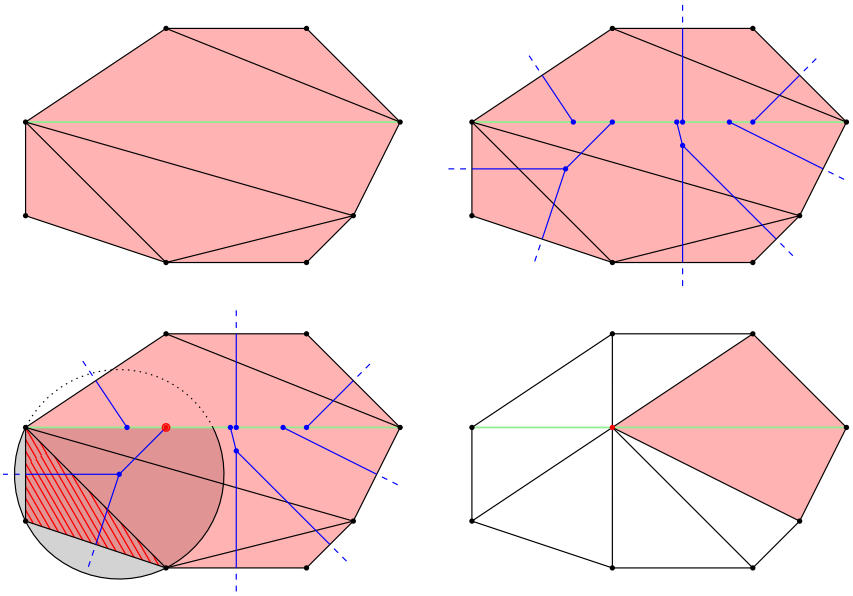


Figure 4 Center identification in the style of Erten and Ungör [9, 8, 7], from the visibility-constrained Voronoi diagram [14].

2.2 Action Generation

In this section we build on the geometric primitives we just introduced, and describe in more detail the three types of candidate actions that the algorithm evaluates for a provided CDT $\mathcal{D}$, which together form the action set $\mathcal{A}(\mathcal{D})$ of $\mathcal{D}$. Every Steiner point, triangle, and edge of $\mathcal{D}$ and every cell in the `circleArrangement` of $\mathcal{D}$ has at most one action associated to it. Every action includes two steps: the insertion, relocation or deletion of Steiner points and ensuring that the maintained triangulation is a CDT. We now describe the types of actions.

Insertion. Let T be a triangle in $\mathcal{D}$. If T is non-obtuse already, then it has no action associated to it. Hence, let T be obtuse instead. If the longest side of T (opposite the unique obtuse angle in T) is a constrained edge of $\mathcal{D}$, the candidate action associated to T adds the unique point described by the `altitudeDrop` primitive. If instead the longest side of T is not constrained, the candidate action associated to T adds the point v in the `clippedCircumcircle` of T that lies on an edge e of the `visibilityVoronoi` of $\mathcal{D}$ and maximizes the distance to the points of $\mathcal{D}$ defining e (see Figure 4). This point v

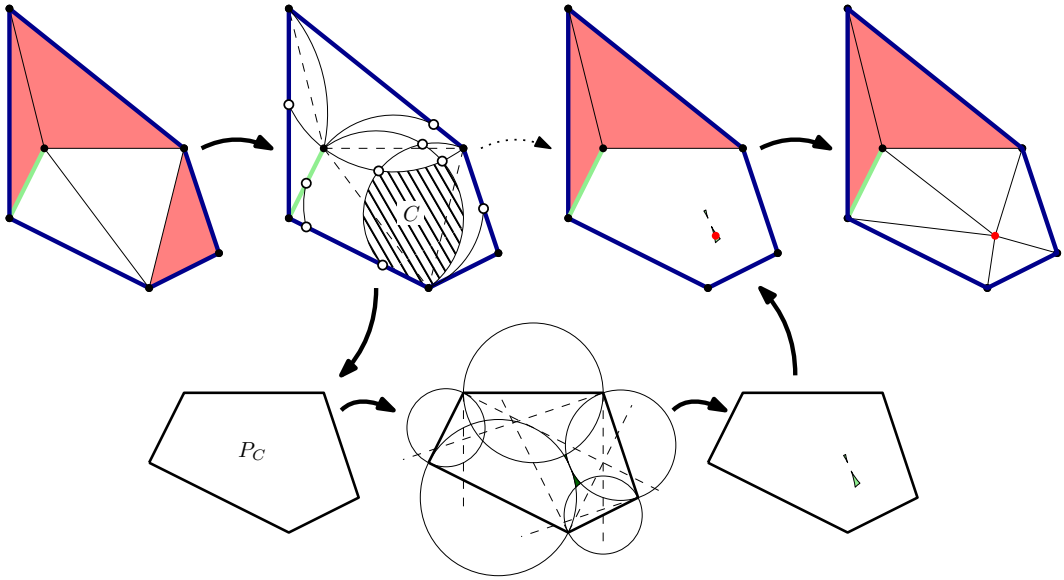


Figure 5 Center identification by choosing a cell C from the `circleArrangement` and computing the `polygonCenter` of its corresponding polygon P_C.

can efficiently be identified via a local search in the `visibilityVoronoi` diagram. We observe that v maximizes the distance to all other points in the CDT while still lying in the `clippedCircumcircle` of T. This guarantees that its insertion into the CDT results in the removal of T from the CDT, accompanied by an intuitively increased likelihood that only few angles of triangles incident to v are obtuse (c.f. [9, 8, 7]).

Next, let C be a cell in the `circleArrangement`. Let $\{T_1, \ldots\}$ be the set of all triangles whose `clippedCircumcircle` contains C. By classical Delaunay arguments, the union of triangles in $\{T_1, \ldots\}$ defines a polygon P_C. We note that the CDT resulting from adding a point $v \in C$ is obtained by replacing the edges of $\mathcal{D}$ in P_C with the segments vp for every vertex p of P_C. In particular, if at least one triangle in $\{T_1, \ldots\}$ is obtuse and the `polygonCenter` v of P_C lies in C, then adding v to the CDT ensures that the number of obtuse triangles in the CDT strictly decreases. Thus, if at least one T_i is obtuse and the `polygonCenter` of P_C exists, the candidate action associated to C adds this `polygonCenter` (see Figure 5), otherwise no action is associated to C.

Relocation. Let v be a Steiner point of $\mathcal{D}$. If v is not part of an obtuse triangle, v has no action associated to it. Hence, let v be part of some obtuse triangle instead. Let P be the polygon resulting from the union of all triangles that v is a part of. Assume the `polygonCenter` of P exists, otherwise v has no action associated to it. The action associated to v moves v to the `polygonCenter` of P.

Deletion. Let e be an edge in $\mathcal{D}$ defined by two Steiner points v and w. The union of all triangles that v or w are a part of forms a polygon P. Assume the `polygonCenter` of P exists, otherwise e has no action associated to it. The action associated to e removes the two points v and w from $\mathcal{D}$ and adds the `polygonCenter` of P.

2.3 Evaluation

The evaluation function of a CDT $\mathcal{D}$ consists of two parts: A naive evaluation function and an algorithm that improves the accuracy of the naive evaluation.

Naïve evaluation. We say an obtuse triangle T in $\mathcal{D}$ *supersedes* another obtuse triangle T' in $\mathcal{D}$ if the longest side of T is either of the shorter two sides of T' (Figure 6 shows superseded-superseding triangle pairs). We expect superseded triangles to be influenced by whatever action is associated to its superseding triangle. An example of this phenomenon can be seen in Figure 4. Hence, we argue that handling non-superseded obtuse triangles should be prioritized and therefore the cost of non-superseded obtuse triangles should be higher than that of superseded (obtuse) triangles. The evaluation function we use is

$$\text{eval}(\mathcal{D}) := \#\text{Steiner points} + 1.1 \cdot \#\text{superseded } \triangle\text{'s} + 3.1 \cdot \#\text{non-superseded } \triangle\text{'s}. \tag{1}$$

Figure 6 shows a run of the algorithm, including the actions that were applied and the value of the evaluation function. Note that we did not pursue optimizing the constants in (1).

Probabilistic state-graph evaluation. To improve the rather naïve evaluation function, we endow the evaluation function with a depth parameter $k \geq 0$, akin to the notion of minimax searching [12] over a game-state graph of a single player game, and define

$$\text{eval}_k(\mathcal{D}) := \begin{cases} \min_{A \in \mathcal{A}(\mathcal{D})} \text{eval}_{k-1}(\mathcal{D}_A) & \text{if } k > 0 \\ \text{eval}(\mathcal{D}) & \text{if } k = 0 \end{cases} \tag{2}$$

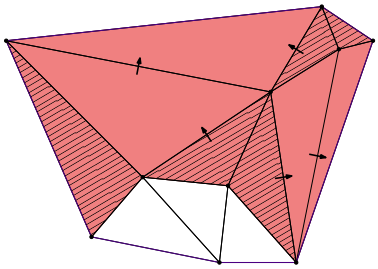
where $\mathcal{D}_A$ is the CDT resulting from $\mathcal{D}$ when applying the action A from the action set $\mathcal{A}(\mathcal{D})$. Even after extensive optimization efforts, we found that even with k as small as 3 the execution time was quite slow. Hence, instead of evaluating $\text{eval}_{k-1}(\mathcal{D}_A)$ for every A in the action set of $\mathcal{D}$, we randomly draw a sample S of actions from $\mathcal{A}(\mathcal{D})$ (with a bias towards more promising moves determined by $\text{eval}_0(\mathcal{D}_A)$) computing $\min_{A \in S} \text{eval}_{k-1}(\mathcal{D}_A)$.

2.4 Solution Merging

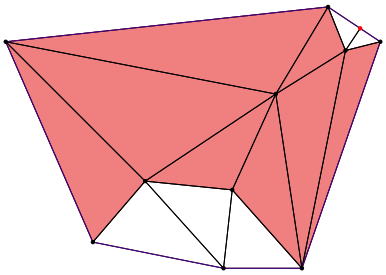
We observed that, due to the probabilistic nature of the evaluation function, different solutions to the same instance often had a similar number of Steiner points but looked vastly different locally. This motivates merging different solutions by sampling a circle and replacing every Steiner point inside this circle from one solution with the Steiner points of the other solution. Note that merging two solutions might result in a CDT that has obtuse triangles. Thus, we apply our algorithm to these CDTs obtaining a new solution with potentially less Steiner points. Repeated application of this merging step reduced the number of Steiner points by as much as 30% for some instances (see Figure 7).

3 Implementation, Experiments, and Examples

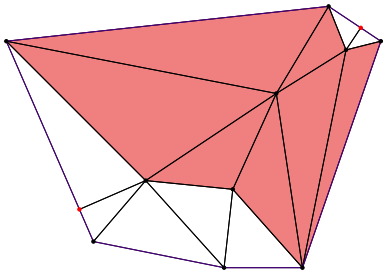
Code and Hardware. Our implementation is in python. In the later stages of the competition we ran our solver on ten nodes of the Marvin cluster of the University of Bonn, each with 1024GB of RAM and 96 threads at 2.10GHz using around 300 000 CPU hours. We did not find any improved solution in the last two weeks of the competition.



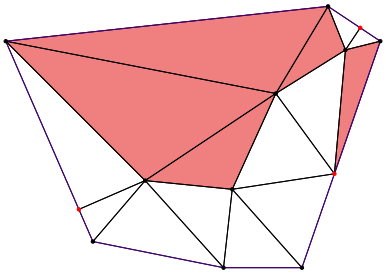
(a) Initial CDT with hatched non-superseded triangles and how triangles are superseded. $\mathrm{eval}(\mathcal{D}) = 19.9$, $\mathrm{eval}_3(\mathcal{D}) = 14.5$.



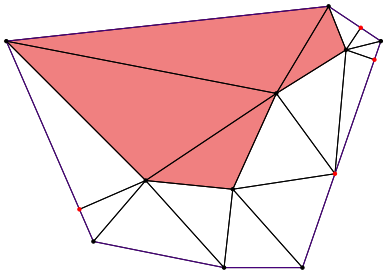
(b) '`altitudeDrop`' insert action. $\mathrm{eval}(\mathcal{D}) = 17.8$, $\mathrm{eval}_3(\mathcal{D}) = 12.4$.



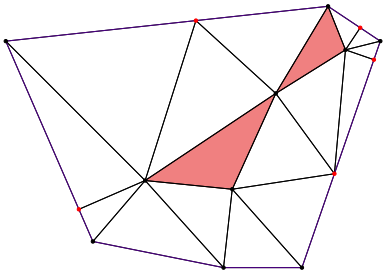
(c) '`altitudeDrop`' insert action. $\mathrm{eval}(\mathcal{D}) = 15.7$, $\mathrm{eval}_3(\mathcal{D}) = 11.2$.



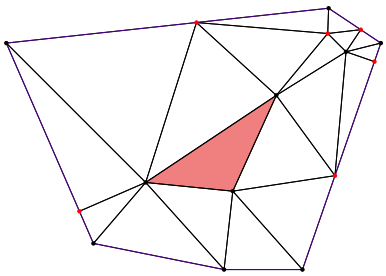
(d) '`visibilityVoronoi`' insert action. $\mathrm{eval}(\mathcal{D}) = 14.5$, $\mathrm{eval}_3(\mathcal{D}) = 9.1$.



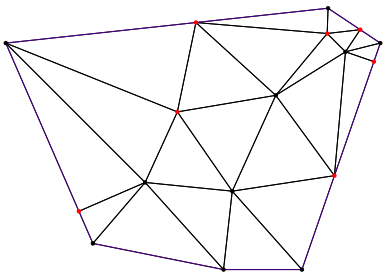
(e) '`altitudeDrop`' insert action. $\mathrm{eval}(\mathcal{D}) = 12.4$, $\mathrm{eval}_3(\mathcal{D}) = 7$.



(f) '`polygonCenter`' insert action. $\mathrm{eval}(\mathcal{D}) = 11.2$, $\mathrm{eval}_3(\mathcal{D}) = 7$.

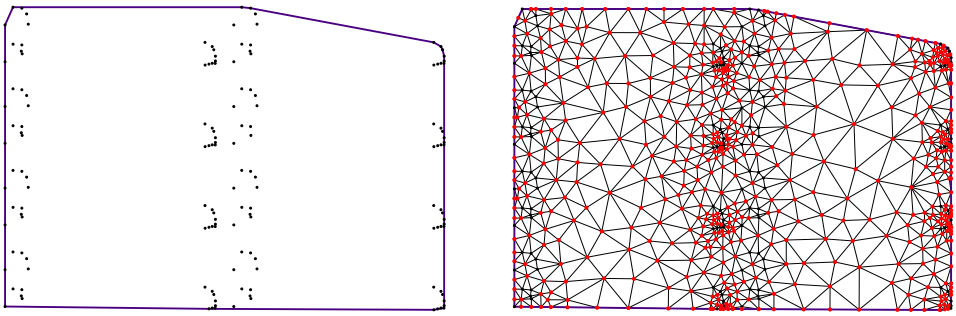


(g) '`polygonCenter`' insert action. $\mathrm{eval}(\mathcal{D}) = 9.1$, $\mathrm{eval}_3(\mathcal{D}) = 7$.



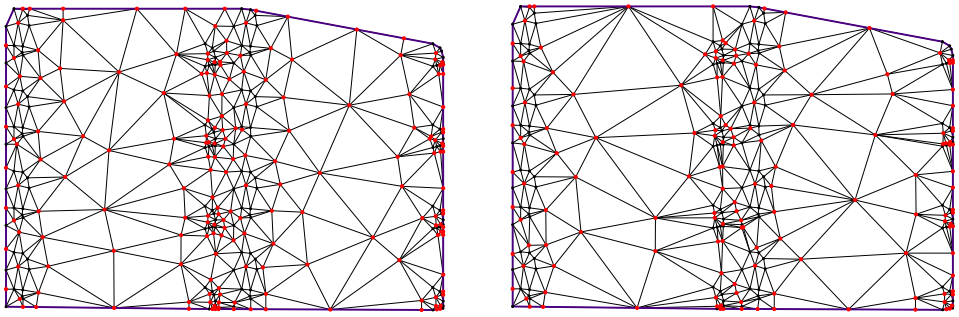
(h) Terminal '`polygonCenter`' insert action. $\mathrm{eval}(\mathcal{D}) = 7$, $\mathrm{eval}_3(\mathcal{D}) = 7$.

Figure 6 Maintained CDT of instance '`point-set_10_c04b0024`' (consisting of 10 points, 6 polygon edges and 0 inner edges) throughout the execution of the local search with depth 0, with all obtuse triangles marked in red. Execution time: 10 seconds. Final number of Steiner points: 7.



(a) Instance '`point-set_150_982c9ab3`' with 150 points, 13 polygon- and 0 inner edges.

(b) Baseline solution via [8, 7, 6] and rounding of coordinates with 573 Steiner points.



(c) Best found solution via probabilistic local search and depth 3 with 190 Steiner points.

(d) Solution via merging and re-solving of existing solutions with 151 Steiner points.

Figure 7 Solution comparison of instance '`point-set_150_982c9ab3`'.

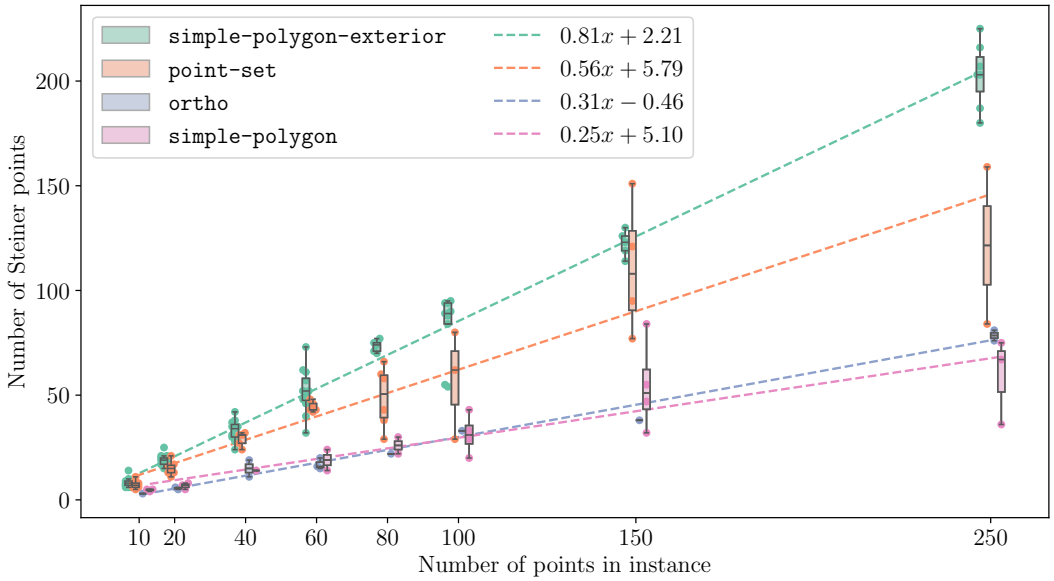


Figure 8 Relationship between solution size and input complexity over different instance types.

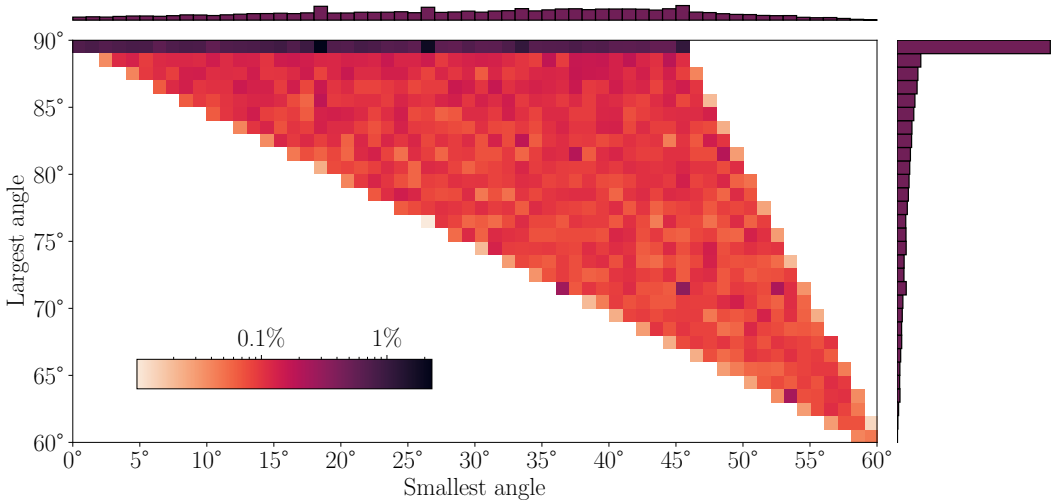


Figure 9 Average distribution of smallest and largest angles, of all triangles in all solutions.

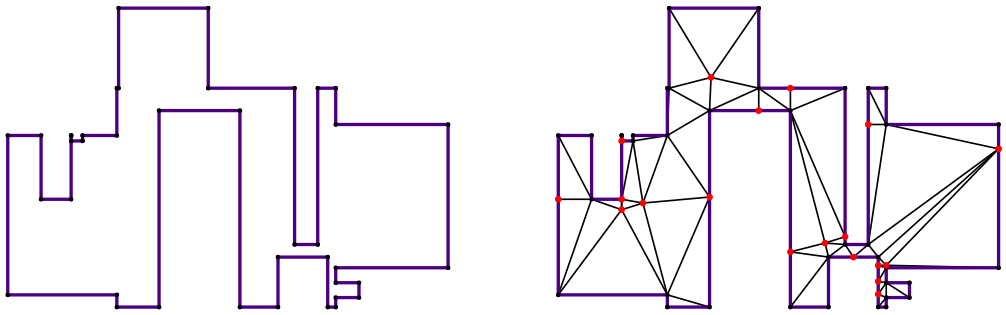


Figure 10 Instance '`ortho_40_56a6f463`' and a solution with 19 Steiner points.

Solutions. The provided instances fall into various types: (i) `ortho`, consisting of rectilinear simple polygons (Figure 10), (ii) `simple-polygon`, consisting of simple polygons (Figure 11), (iii) `point-set`, consisting of the convex polygons with points in its interior (Figures 13 and 14), and (iv) `simple-polygon-exterior`, consisting of convex polygons with points and additional constraints in its interior (Figures 12 and 15). Our algorithm performs well on all instances types; no manual intervention or distinction is necessary for our approach. The number of Steiner points in our solutions compared to the number of input points in each type suggests that the total number of Steiner points is roughly linear in the input complexity and increases depending on the number of points and constraints in the interior of the PSLG, see Figure 8. We observe that a large fraction of triangles computed by our algorithm are right triangles (see Figure 9, right). All other angles in our solutions (determined by largest and smallest angle) appear to be roughly equally distributed (see Figure 9).

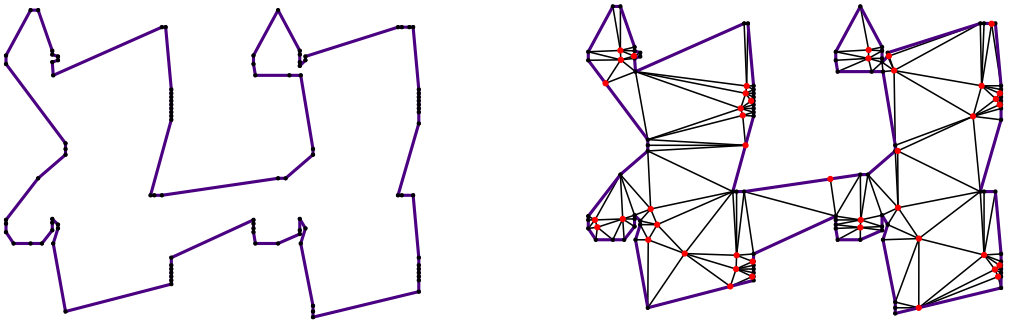


Figure 11 Instance '`simple-polygon_100_cb23308c`' and a solution with 43 Steiner points.

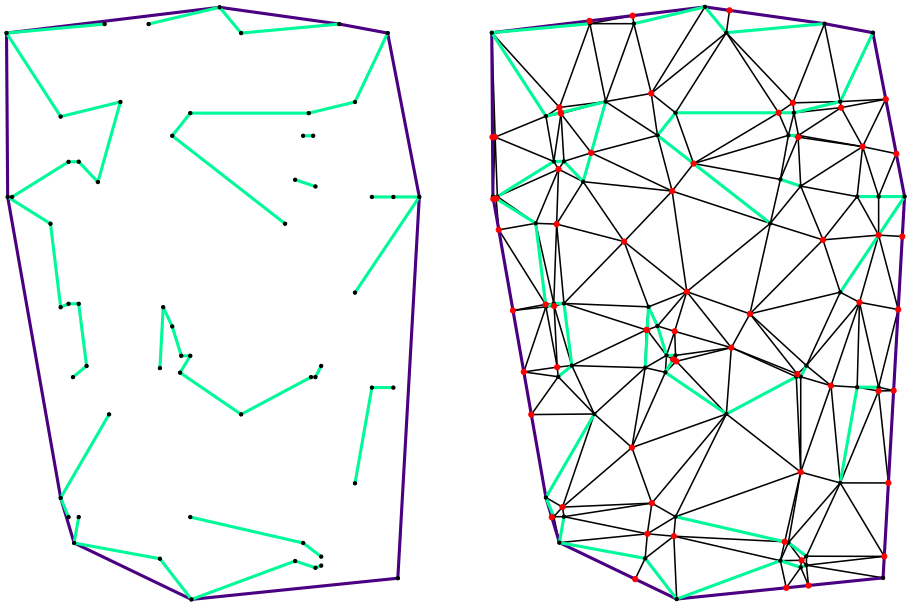


Figure 12 Solution to '`simple-polygon-exterior-20_60_8221c868`' with 62 Steiner points.

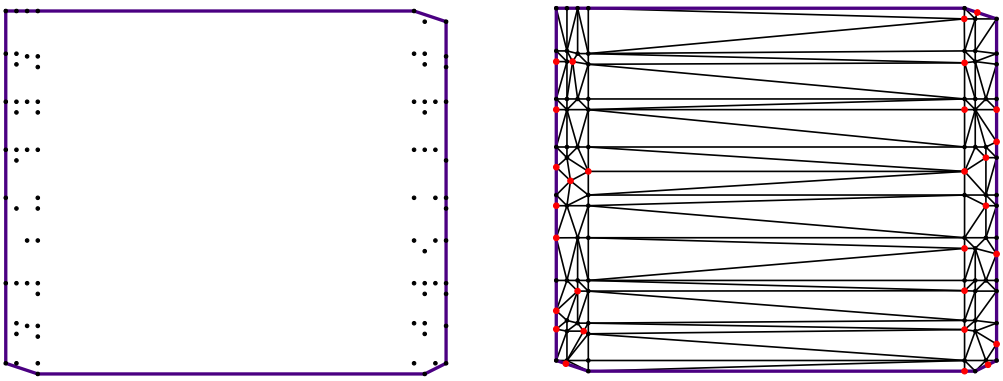


Figure 13 Instance '`point-set_80_9a8373fb`' and a solution with 29 Steiner points.

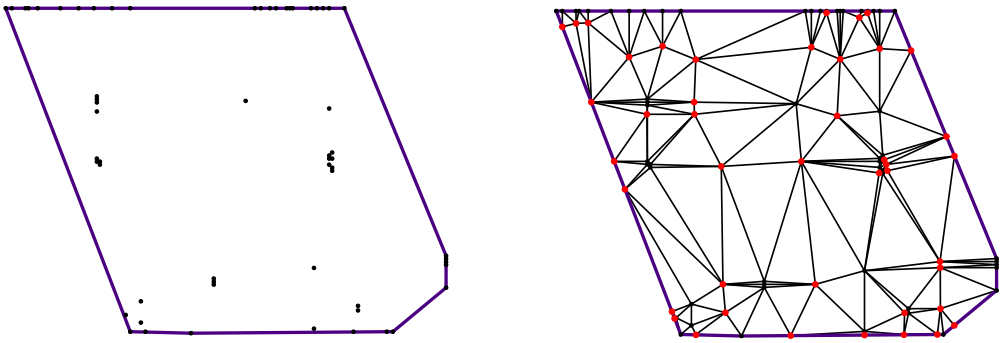


Figure 14 Instance '`point-set_60_27bc003d`' and a solution with 43 Steiner points.

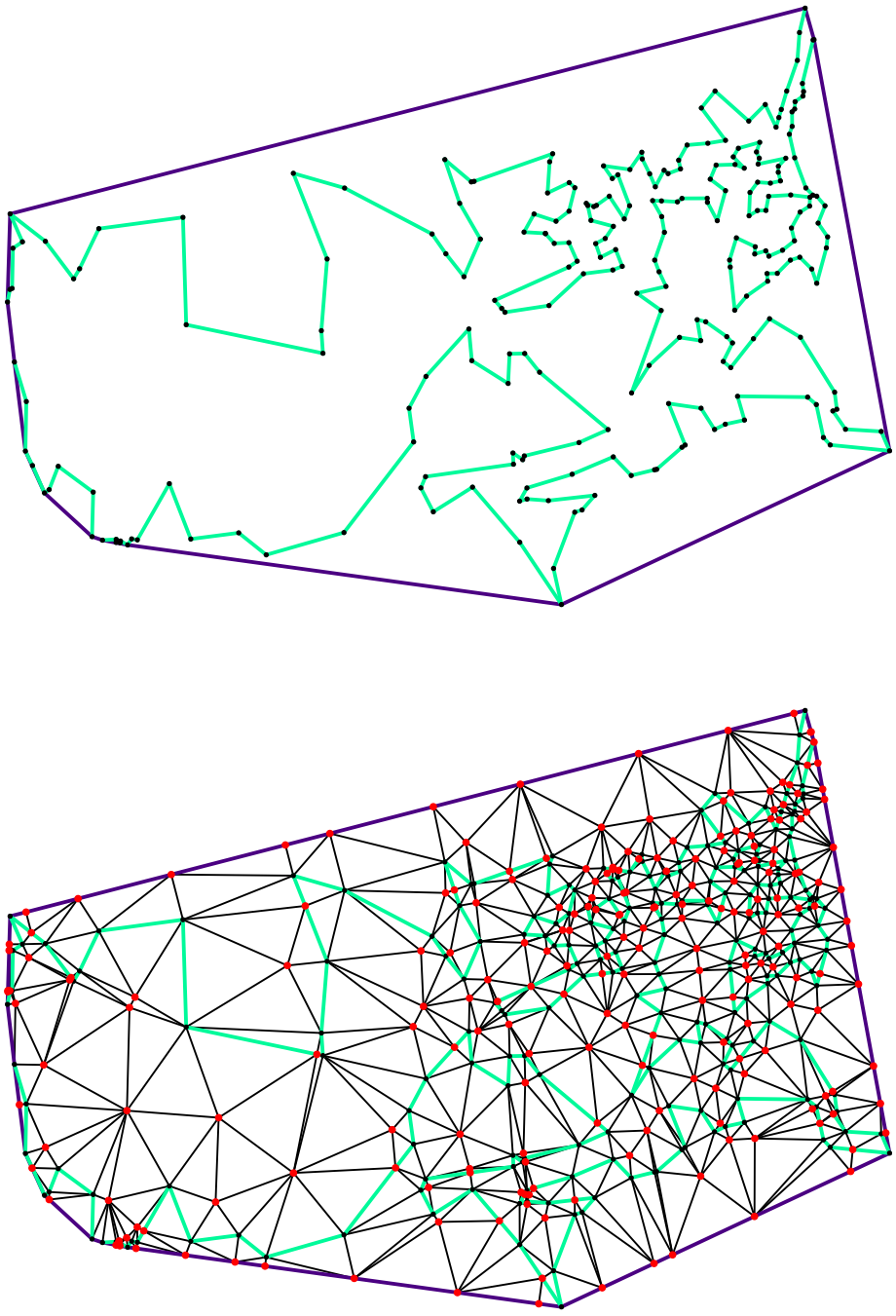


Figure 15 Solution with 225 Steiner points to '`simple-polygon-exterior_250_c0a19392`'.

References

1 Mikkel Abrahamsen, Florestan Brunck, Jacobus Conradi, Benedikt Kolbe, and André Nusser. nonObtuseTri. Software, version v3.0., swhId: `swh:1:dir:b6301970ab626c084f309f9b5e6b91e9acf24949` (visited on 2025-06-02). URL: `https://github.com/JacobusTheSecond/nonObtuseTri`, `doi:10.4230/artifacts.23288`.

2 Taehoon Ahn, Jaegun Lee, Byeonguk Kang, and Hwi Kim. Incremental algorithm and local search for minimum non-obtuse triangulations (CG challenge). In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, 2025. These proceedings. `doi:10.4230/LIPIcs.SoCG.2025.80`.

3 Brenda S. Baker, Eric Grosse, and Conor S. Rafferty. Nonobtuse triangulation of polygons. *Discret. Comput. Geom.*, 3:147–168, 1988. `doi:10.1007/BF02187904`.

4 Marshall W. Bern, Scott A. Mitchell, and Jim Ruppert. Linear-size nonobtuse triangulation of polygons. *Discret. Comput. Geom.*, 14(4):411–428, 1995. `doi:10.1007/BF02570715`.

5 Jan Brandts, Sergey Korotov, Michal Krízek, and Jakub Solc. On nonobtuse simplicial partitions. *SIAM Rev.*, 51(2):317–335, 2009. `doi:10.1137/060669073`.

6 Y. Burago and V. Zalgaller. Polyhedral embedding of a net. *Vestnik Leningrad. Univ*, 15(7):66–80, 1960.

7 Hale Erten and Alper Üngör. Computing acute and non-obtuse triangulations. In Prosenjit Bose, editor, *Proceedings of the 19th Annual Canadian Conference on Computational Geometry, CCCG 2007, August 20-22, 2007, Carleton University, Ottawa, Canada*, pages 205–208. Carleton University, Ottawa, Canada, 2007. URL: `http://cccg.ca/proceedings/2007/09a2.pdf`.

8 Hale Erten and Alper Üngör. Computing triangulations without small and large angles. In Francois Anton, editor, *Sixth International Symposium on Voronoi Diagrams, ISVD 2009, Copenhagen, Denmark, June 23-26, 2009*, pages 192–201. IEEE Computer Society, 2009. `doi:10.1109/ISVD.2009.32`.

9 Hale Erten and Alper Üngör. Quality triangulations with locally optimal steiner points. *SIAM J. Sci. Comput.*, 31(3):2103–2130, 2009. `doi:10.1137/080716748`.

10 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Stefan Schirra. Minimum non-obtuse triangulations: The CG:SHOP Challenge 2025, 2025. URL: `https://arxiv.org/abs/2504.04412`.

11 Hiroshi Maehara. Acute triangulations of polygons. *European Journal of Combinatorics*, 23(1):45–55, 2002. `doi:10.1006/EUJC.2001.0531`.

12 M. Maschler, S. Zamir, and E. Solan. *Game Theory*. Cambridge University Press, 2013. URL: `https://books.google.de/books?id=lqwzqgvhwXsC`.

13 S. Saraf. Acute and nonobtuse triangulations of polyhedral surfaces. *European Journal of Combinatorics*, 30(4):833–840, 2009. `doi:10.1016/J.EJC.2008.08.004`.

14 Jane Tournois, Pierre Alliez, and Olivier Devillers. Interleaving delaunay refinement and optimization for 2d triangle mesh generation. In Michael L. Brewer and David L. Marcum, editors, *Proceedings of the 16th International Meshing Roundtable, October 14-17, 2007, Seattle, Washington, USA, Proceedings*, pages 83–101. Springer, 2007. `doi:10.1007/978-3-540-75103-8_5`.

Incremental Algorithm and Local Search for Minimum Non-Obtuse Triangulations

Taehoon Ahn
National Institute for Mathematical Sciences, Daejeon, South Korea

Jaegun Lee
Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, Korea

Byeonguk Kang
Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, Korea

Hwi Kim
Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, Korea

Abstract

In this year's CG challenge, the task was to compute a non-obtuse triangulation of given planar regions while minimizing the number of Steiner points. Our team (Gwamegi) used two approaches. The first approach incrementally adds Steiner points on the grid defined by the input points in the planar regions, while maintaining a Delaunay triangulation. The second approach is an iterated local search, which runs insertion and deletion steps alternatingly. In the insertion step, we add a new Steiner point inside a maximal convex subpolygon in the current triangulation. In the deletion step, we remove a number of Steiner points packed in a small region. We use both our approaches to obtain non-obtuse triangulations for all 150 instances. We use our second approach to reduce the number of Steiner points from the non-obtuse triangulations. We have successfully computed non-obtuse triangulations using a sufficiently small number of Steiner points for all instances.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Triangulation, Non-obtuse triangle, Steiner point, Incremental algorithm, Local search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.80

Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/slothetic/CG2025`
archived at `swh:1:dir:e8f3167e99fdf8a88291190ad292e15703bb3c03`

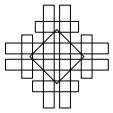
Funding Taehoon Ahn was supported by the National Institute for Mathematical Sciences (NIMS) grant funded by the Korea government (MSIT) (No. NIMS-B25910000). Jaegun Lee, Byeonguk Kang, and Hwi Kim were supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00219980).

Acknowledgements We would like to thank the challenge organizers and other competitors for their effort on the event. We would also like to thank Hee-Kap Ahn for providing the computational resources of the Algorithms laboratory of POSTECH.

1 Introduction

Triangulation is a well-studied topic in computational geometry. It is a subdivision of a planar geometric object into triangles. Such an object is a point set possibly with constraint edges [7, 13], a polygon [10, 11], or a planar straight-line graph (PSLG) [6]. In this year's CG

© Taehoon Ahn, Jaegun Lee, Byeonguk Kang, and Hwi Kim;
licensed under Creative Commons License CC-BY 4.0



challenge, the task was to compute a non-obtuse triangulation of given PSLGs minimizing the number of Steiner points. This paper introduces the approaches of our team (Gwamegi) for the challenge. See also the survey [9] by the challenge organizers and another team's approach [1].

Finding triangulations under various optimality criteria has been well studied, including max-min angle triangulations (called Delaunay triangulations) [4, 12], high aspect ratio triangulations [14], and min-sum side length triangulations (called minimum weight triangulations) [3, 5]. Among them, Delaunay triangulation is the most popular with its various application areas such as mesh generation and path planning [2, 15]. Note that any non-obtuse triangulation (every triangle is acute or right) is Delaunay.

The challenge problem can be formulated as follows.

▶ **Problem 1.** *Given a problem instance $I = (V, R, E)$, where V is a set of points in $\mathbb{R}^2$, R is a simple polygon to be triangulated whose vertex set is a subset of V, and E is the set of constraint edges each connecting two distinct points in V, find a minimum-sized set $S \subset R$ of Steiner points and a non-obtuse triangulation of R with vertex set $V \cup S$ and respecting E.*

There were 150 instances, five types in total. Figure 1 illustrates the instance types.

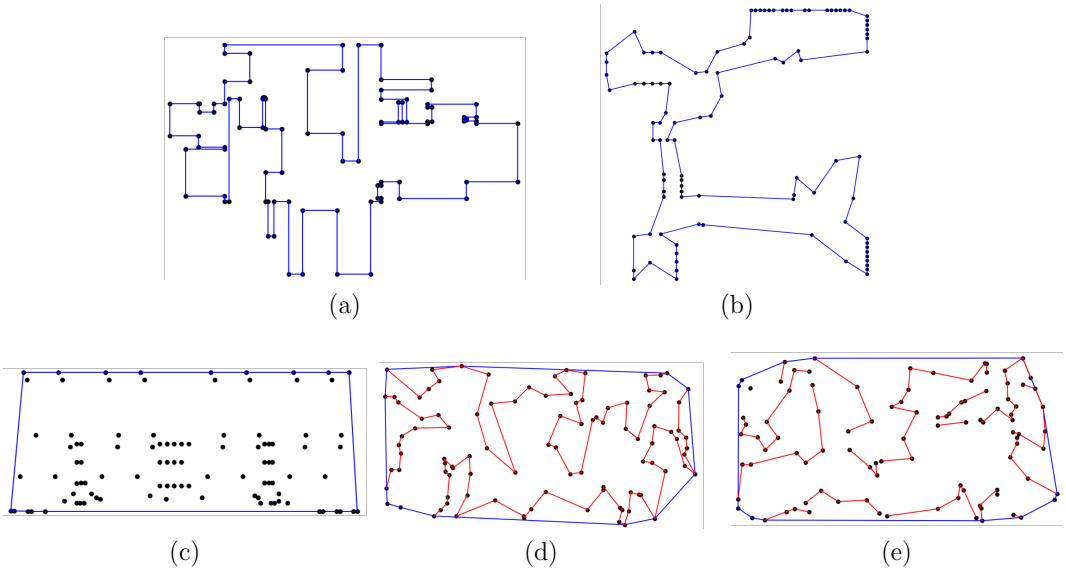


Figure 1 Example instance per type. Blue segments form the boundary of R. Red segments are constraint edges E. Type (a) ortho, (b) simple-polygon, (c) point-set, (d) simple-polygon-exterior, and (e) simple-polygon-exterior-20.

2 Algorithms

We used two approaches to compute non-obtuse triangulations: (1) incremental algorithm on a grid and (2) iterated local search. For all instances of type ortho and some instances of type point-set and type simple-polygon, we used our first approach. For the other instances, we used our second approach. After computing non-obtuse triangulations for all instances, we again used our second approach to reduce the number of Steiner points for each of them.

Let $\mathsf{DT}(V, R, E)$ be a Delaunay triangulation of R with point set V, respecting E. If there is more than one such Delaunay triangulation, we let $\mathsf{DT}(V, R, E)$ to be one with the minimum number of obtuse triangles. In our approaches, we maintain $\mathsf{DT}(V \cup S, R, E)$ while updating the set S of Steiner points. We say that S is a *feasible solution* of $I = (V, R, E)$ if $\mathsf{DT}(V \cup S, R, E)$ is a non-obtuse triangulation.

2.1 Incremental algorithm on a grid

We introduce our incremental algorithm. It was mostly used for instances of type ortho. For a type ortho instance $I = (V, R, E)$, R is an orthogonal polygon (any edge is vertical or horizontal), V is the vertex set of R, and $E = \emptyset$. Define $x(p)$ and $y(p)$ as the x- and y-coordinate of a point $p \in \mathbb{R}^2$, respectively. Let $X = \{x(p) \mid p \in V\}$ and $Y = \{y(p) \mid p \in V\}$, and let $S_G = \{p \in R \mid x(p) \in X, y(p) \in Y\}$ be the *grid points* of I. The set $S_G \setminus V$ is an obvious feasible solution of I. However, its size can be $\Omega(|V|^2)$. In the following, we show how to find a moderately-sized feasible solution $S \subset S_G$.

Starting with $S = \emptyset$, we compute $\mathsf{DT}(V \cup S, R, E)$ in each step. If it is non-obtuse, S is a feasible solution, and we are done. Otherwise, we pick a random obtuse triangle in $\mathsf{DT}(V \cup S, R, E)$. Let p be its obtuse vertex. The angle interval defined by p contains at least one of the four axis-aligned directions from p: $+x$, $+y$, $-x$ and $-y$. Without loss of generality, assume that it contains the $+y$-direction as in Figure 2, and let $\vec{r}$ be the ray emanating from p and going towards the $+y$-direction. Let q be a point moving from p along $\vec{r}$. During the movement of q, if $y(q) = y(v)$ for a vertex v of the triangle containing q and $q \neq p$, it stops, and we add q to S. It is easy to see that q always stops before moving out of R. We repeat the procedure until S becomes feasible.

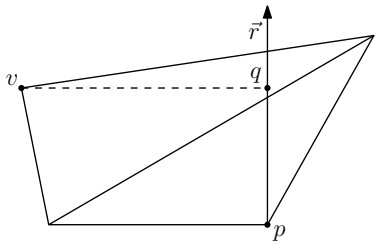


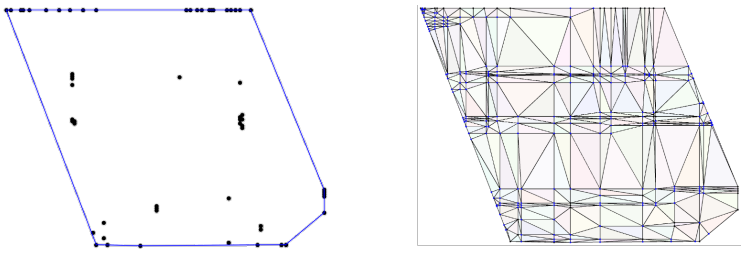
Figure 2 Point q moves from p along $\vec{r}$. It stops when $y(q) = y(v)$ for a vertex v of the triangle containing q.

Table 1 shows that the above algorithm uses no more Steiner points than the size of V, for type ortho instances.

Table 1 $|V|$ is the size of the instances. $|S|$ is the average number of Steiner points added, when we executed the algorithm five times for each instance of type ortho.

$\lvert V\rvert$	10	20	40	60	80	100	150	250
$\lvert S\rvert$	3.0	9.3	23.6	38.4	45.4	74.6	125.2	184.9

We can apply the algorithm to some instances of type point-set and simple-polygon by invoking MakeNonObtuseBoundary procedure whenever we add a new Steiner point. MakeNonObtuseBoundary finds an obtuse triangle from $\mathsf{DT}(V \cup S, R, E)$ having its hypotenuse on the boundary of R, and adds the projection from the opposite vertex onto the hypotenuse to S. It repeats until there is no such obtuse triangle. See Figure 3.



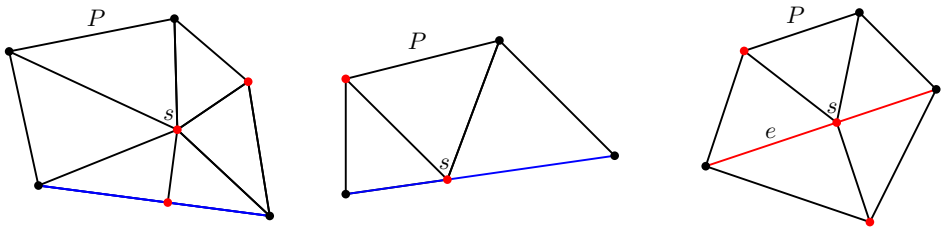
(a) (b)

Figure 3 (a) Instance point-set_60_27bc003d of type point-set and (b) its feasible solution obtained by our incremental algorithm.

2.2 Iterated local search

We introduce our local search algorithm. Starting from $S := \emptyset$, the algorithm runs *insertion steps* and *deletion steps* alternatingly to update S.

Unlike the incremental algorithm in Section 2.1, our local search algorithm may place non-grid Steiner points in the insertion step. We first observe the following. Let $I = (V, R, E)$ be an instance. Consider a feasible solution S of I. Fix a point $s \in S$, and let $T(s)$ be the set of triangles in $\mathsf{DT}(V \cup S, R, E)$ with s as a vertex. Since all triangles in $T(s)$ are non-obtuse, their union is a convex polygon P. The interior of P does not contain any constraint edge if and only if s does not lie on any constraint edge. See Figure 4(a, b). If s lies on some $e \in E$, then $P \cap e$ is an edge connecting two vertices of P, and P does not contain any other constraint edge. See Figure 4(c). Based on the observation, we consider the following.



(a) (b) (c)

Figure 4 Union P of $T(s)$. Black points are the given points in V, and red points are the Steiner points. (a) s lies in the interior of P and s does not lie on any $e \in E$. (b) s lies on the boundary of P. (c) s lies on some $e \in E$.

▶ **Problem 2.** *Given a convex polygon P, find a point $s \in P$ such that $\triangle spq$ is non-obtuse for any edge $\overline{pq}$ of P. Additionally, if a diagonal e of P is given, s must lie on e.*

Fix an edge $\overline{pq}$ of P and consider the triangle $\triangle spq$ for a point $s \in P$. We have $\angle spq \leq \pi/2$ and $\angle sqp \leq \pi/2$ if and only if s lies in the strip L perpendicular to $\overline{pq}$, with each bounding line going through p and q, respectively. We have $\angle psq \leq \pi/2$ if and only if s lies outside the disk D having $\overline{pq}$ as its diameter. See Figure 5(a). Summing up, $\triangle spq$ is non-obtuse if and only if s lies in $\mathcal{R}(\overline{pq}) := L \setminus D$, and $s \in P$ is a solution of Problem 2 if and only if it lies in the intersection of $\mathcal{R}(\overline{pq})$'s for every edge $\overline{pq}$ of P. See Figure 5(b). A similar subprocedure was previously introduced by Erten and Üngör [8], and it was also used in the team Naughty NOTers' approach [1], named as polygonCenter. If a diagonal e is given additionally as in

the problem statement, we check if such intersection of $\mathcal{R}(\overline{pq})$'s intersects e. We solve its generalized version, Problem 3, defined as follows, because there may be no solution for Problem 2. Let V_P be the set of vertices of P, and let $E_P \coloneqq \{e \cap P \mid e \in E\}$.

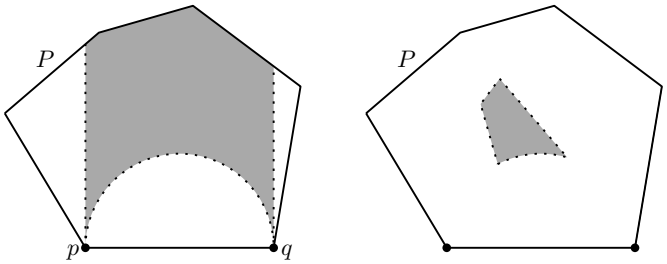


Figure 5 (a) $\mathcal{R}(\overline{pq}) \cap P$. (b) The intersection of $\mathcal{R}(\overline{pq})$'s for every edge $\overline{pq}$ of P.

▶ **Problem 3.** *Given a convex polygon P and a set of constraint edges E_P in P, find a point $s \in P$ minimizing the number of obtuse triangles in $\mathsf{DT}(V_P \cup \{s\}, P, E_P)$.*

To solve Problem 3, we need to define the following sets.

- $\mathcal{D}$: the set of circles, each having an edge of P as its diameter.
- $\mathcal{L}$: the set of lines, each perpendicular to an edge of P and going through one of its endpoints.
- $\mathcal{C}$: the set of circumcircles of the triangles in $\mathsf{DT}(V_P, P, E_P)$.

Consider the arrangement $\mathcal{A}$ of $\mathcal{D} \cup \mathcal{L} \cup \mathcal{C} \cup E_P$ in P. For each cell of $\mathcal{A}$, the number of obtuse triangles is the same. Thus we aim to find a vertex v of $\mathcal{A}$ minimizing the number of obtuse triangles in $\mathsf{DT}(V_P \cup \{v\}, P, E_P)$.

Since we are using exact arithmetic, it consumes too much time and space to exactly compute vertices of $\mathcal{A}$. So instead, we approximate each point with another point at close range with simpler coordinates. For each vertex of $\mathcal{A}$ that does not lie on any constraint edge, we approximate both its x- and y-coordinate using quantized values represented as $i \cdot q$ for some integer i and a fixed parameter $q \in \mathbb{Q}$. For each vertex of $\mathcal{A}$ lying on a constraint edge, we approximate only one of its coordinates with the quantized values, to ensure that the approximated point also lies on the constraint edge. We start with $q = 1$ and make it smaller whenever the algorithm gets stuck. From the approximated points, we choose one minimizing the number of obtuse triangles. If there are multiple such points, we choose a point at random. Let $\mathsf{AddPoint}(P)$ be the corresponding subprocedure.

In the insertion step, we first compute $\mathsf{DT}(V \cup S, R, E)$. If it is non-obtuse, we are done. Otherwise, we pick an obtuse triangle t in $\mathsf{DT}(V \cup S, R, E)$ at random. Then we compute a maximal convex polygon P which is the union of some triangles in $\mathsf{DT}(V \cup S, R, E)$ containing t. This is done by maintaining a convex polygon P, starting from $P = t$ and adding triangles incident to P until there is no triangle t' such that $P \cup t'$ is convex. Finally, we add a point to S obtained from $\mathsf{AddPoint}(P)$. During each insertion step, we repeat the above several times.

In the deletion step, we randomly choose a Steiner point p, and remove every Steiner point (including p itself) within distance $r \cdot d$ where r is a parameter that increases as time goes by, and d is the distance between p and its closest Steiner point.

We store the current solution as the best solution whenever the number of obtuse triangles decreases, or the number of Steiner points decreases while the number of obtuse triangles remains the same. In each iteration, we start from the best solution and apply the insertion and the deletion steps alternatingly. Figure 6(a) illustrates a deletion step, and Figure 6(b) illustrates a non-obtuse solution obtained by running an insertion step afterwards.

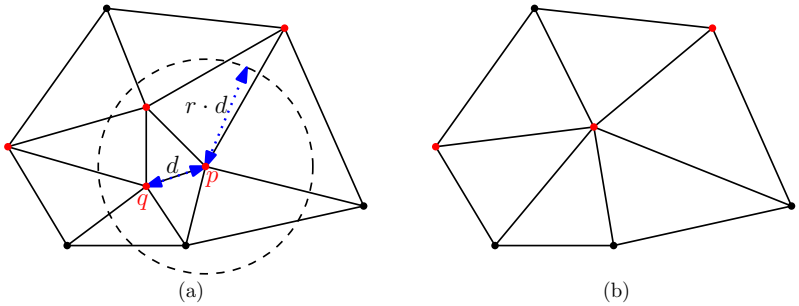


Figure 6 (a) Given a Steiner point p, we remove every Steiner point whose distance from p is at most $r \cdot d$, for some parameter r and the distance d between p and its closest Steiner point q. (b) Applying a deletion step and an insertion step in order reduces the number of Steiner points by two.

3 Results

We implemented our algorithms in Python 3. We used two servers: (i) 4.40GHz Intel Core i5 12400F CPU with 32GB RAM, and (ii) 4.70GHz AMD Ryzen 9 7900 CPU with 32GB RAM.

We now discuss the results. We have found non-obtuse triangulation for all 150 instances. We ranked first on 34 instances including 4 exclusive ones, see Figure 7. We ranked second in almost all the other instances, including 35 instances where we used only one more Steiner point than the best solution. As a result, we ranked second among 19 teams, and reached the top among the junior teams.

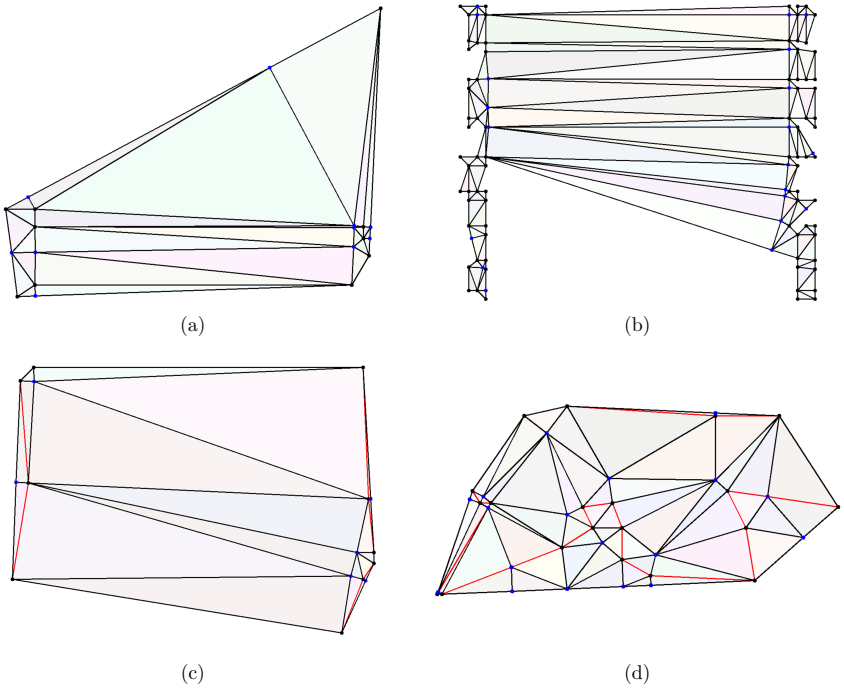


Figure 7 Our exclusive best solutions: (a) point-set_10_f999dc7f, (b) simple-polygon_80_7b8f6c4c, (c) simple-polygon-exterior_10_a5f0f2fc, and (d) simple-polygon-exterior_20_92dcd467.

We analyze our results by the types and the sizes of the instances. We achieved the highest average score for instances of ortho types. See Figure 8(a). We got higher scores for smaller instances than for the larger instances. See Figure 8(b).

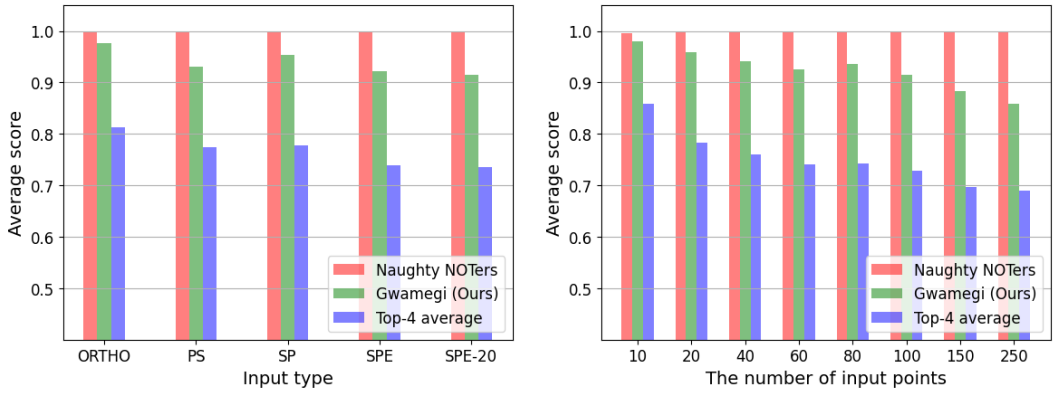


Figure 8 Average score compared with Naughty NOTers [1] and with top four teams (Naughty NOTers, Gwamegi, KITriangle, and Obtuse Terminators). We write ORTHO for ortho type instances, PS for point-set type instances, SP for simple-polygon type instances, SPE for simple-polygon-exterior type instances, and SPE-20 for simple-polygon-exterior-20 type instances. (a) Average score per input type. (b) Average score per number of input points.

References

1 M. Abrahamsen, F. Brunck, J. Conradi, B. Kolbe, and A. Nusser. Computing non-obtuse triangulations with few Steiner points. In *41st International Symposium on Computational Geometry (SoCG 2025)*, SoCG 2025, 2025. doi:10.4230/LIPIcs.SoCG.2025.79.

2 S. J. Anderson, S. B. Karumanchi, and K. Iagnemma. Constraint-based planning and control for safe, semi-autonomous operation of vehicles. In *IEEE Intelligent Vehicles Symposium*, pages 383–388, 2012. doi:10.1109/IVS.2012.6232153.

3 F. Aurenhammer and Y.-F. Xu. Optimal triangulations. In *Encyclopedia of Optimization, Second Edition*. Springer, 2009. doi:10.1007/978-0-387-74759-0_475.

4 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

5 M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry. Second Edition*. World Scientific, 1995. doi:10.1142/9789812831699_0003.

6 C. J. Bishop. Nonobtuse triangulations of PSLGs. *Discrete & Computational Geometry*, 56:43–92, 2016. doi:10.1007/s00454-016-9772-8.

7 L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989. doi:10.1007/BF01553881.

8 Hale Erten and Alper Üngör. Triangulations with locally optimal steiner points. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP 2007, pages 143–152. Eurographics Association, 2007. doi:10.2312/SGP/SGP07/143-152.

9 S. P. Fekete, P. Keldenich, D. Krupke, and S. Schirra. Minimum non-obtuse triangulations: The CG:SHOP challenge 2025, 2025. doi:10.48550/arXiv.2504.04412.

10 P. D. Gilbert. New results on planar triangulations. *Coordinated Science Laboratory Report*, 1979.

11 G. T. Klincsek. Minimal triangulations of polygonal domains. In *Annals of Discrete Mathematics*, volume 9, pages 121–123. Elsevier, 1980. doi:10.1016/S0167-5060(08)70044-X.

12 D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980. doi:10.1007/BF00977785.

13 W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):1–29, 2008. doi:10.1145/1346330.1346336.

14 M.-A. K. Posenau. Approaches to high aspect ratio triangulations. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, CCCG 1993, pages 30–35, 1993.

15 J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22(1):21–74, 2002. doi:10.1016/S0925-7721(01)00047-5.

AmoebotSim 2.0: A Visual Simulation Environment for the Amoebot Model with Reconfigurable Circuits and Joint Movements

Matthias Artmann
Paderborn University, Germany

Tobias Maurer[1]
Paderborn University, Germany

Andreas Padalkin
Paderborn University, Germany

Daniel Warner
Paderborn University, Germany

Christian Scheideler
Paderborn University, Germany

Abstract

We present AmoebotSim 2.0, a simulation environment for the geometric amoebot model of programmable matter that supports the reconfigurable circuit and joint movement extensions of the model. In the geometric amoebot model, we consider systems of simple computational entities called *amoebots* in a regular triangular grid environment. We are interested in distributed algorithms that solve coordination and shape formation problems. The *reconfigurable circuit* and *joint movement* extensions of the model allow the amoebots to communicate over greater distances and perform more complex movements, overcoming some limitations of the original model. AmoebotSim 2.0 is an open-source simulation environment that supports these extensions and provides a rich graphical interface, flexible simulation features, an extensive API, and comprehensive documentation.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Distributed computing models; Theory of computation → Self-organization

Keywords and phrases Programmable matter, amoebot model, reconfigurable circuits, joint movements, simulator

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.81

Category Media Exposition

Supplementary Material
Software (Source Code): `https://github.com/martmannupb/AmoebotSim-2.0` [1]
archived at `swh:1:dir:41a83f85a3e7c6c1d5a2fec79533bef04c7ff770`
Other (Website): `https://cs.uni-paderborn.de/en/ti/forschung/open-source-projekte/amoebotsim-2`
InteractiveResource (Web App): `https://groups.uni-paderborn.de/fg-ti/AmoebotSim2.0/`

Funding This work was supported by the DFG Project SCHE 1592/10-1.

[1] In memoriam, † 2023

© Matthias Artmann, Tobias Maurer, Andreas Padalkin, Daniel Warner, and Christian Scheideler;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Programmable matter envisions materials that can change their physical properties like shape, elasticity, conductivity, and color in a programmable fashion and react autonomously to external influences [10]. Potential application areas include minimally invasive medical treatment, maintenance, exploration, and manufacturing. We typically view such materials as systems of many identical nano-scale computational entities with minimal individual movement and communication abilities.

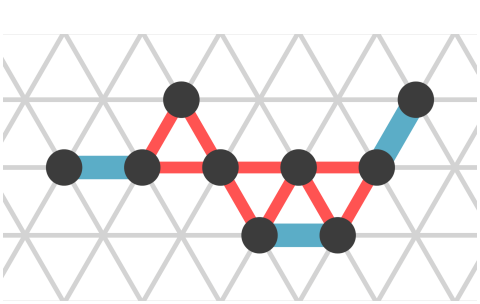
We consider the *geometric amoebot model* [5, 4], where a set of n entities, called *amoebots*, is placed on the nodes of the infinite equilateral triangular grid graph (see Fig. 1a). The set of occupied nodes must be connected and every node can be occupied by at most one amoebot. By performing an *expansion*, an amoebot occupying one node can expand its body onto an adjacent unoccupied node. An expanded amoebot can *contract* into either of the two occupied nodes. By repeated expansion and contraction, the amoebots can move along the edges of the grid. Two amoebots occupying adjacent nodes can sense each other and exchange simple messages. The amoebots are controlled by anonymous, identical finite state machines with a constant number of states that is fixed by the algorithm. We consider a *fully synchronous* execution model where time proceeds in discrete rounds and in each round, all amoebots are activated simultaneously.

The *reconfigurable circuit* extension [6] allows amoebots to construct communication networks by placing *pins* on the amoebots' borders (see Fig. 1b). The pins of adjacent amoebots connect to form communication channels. Each amoebot can internally connect arbitrary disjoint subsets of its pins into *partition sets*. Using partition sets as nodes and connections between pins of adjacent amoebots as edges, we define a graph. We call the connected components of this graph *circuits*. Amoebots can use circuits as communication channels on which they can broadcast primitive signals called *beeps*. A beep carries no information about its origin and is always received by all amoebots on a circuit. Using circuits, many coordination problems can be solved in polylogarithmic time [6, 9, 8, 2].

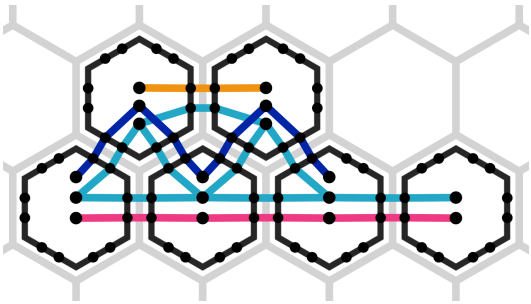
The *joint movement* extension [7] enables amoebots to perform more complex movements by pushing and pulling each other during expansions and contractions (see Fig. 1c). By combining their local movements, amoebots can rapidly reconfigure and move their structure.

In this paper, we present AmoebotSim 2.0, a visual simulator for algorithms in the geometric amoebot model that supports both model extensions. The source code is available on GitHub[2], and there is a website[3] showcasing the main features, some example simulations, and a browser-based demo. The simulator was developed for the Unity[4] game engine, which provides a platform for a highly interactive user interface and efficient rendering of complex simulations. The project can only be opened and executed within Unity's editor. Our simulator was inspired by AmoebotSim [3], a simulator for the original amoebot model. The main difference between the two simulators is that AmoebotSim uses a sequential execution model, while the extensions supported by our simulator require a synchronous model.

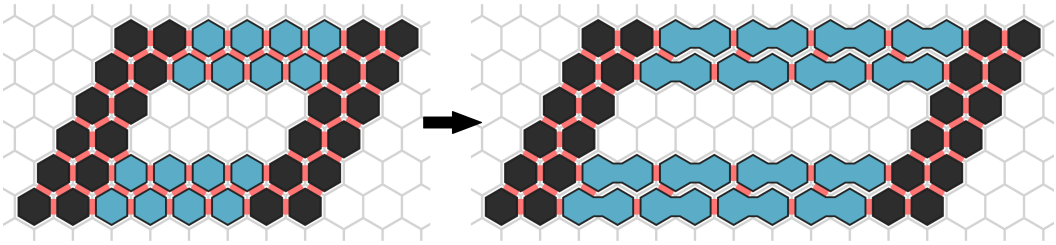
[2] `https://github.com/martmannupb/AmoebotSim-2.0`
[3] `https://cs.uni-paderborn.de/en/ti/forschung/open-source-projekte/amoebotsim-2`
[4] `https://unity.com/`



(a) Contracted and expanded amoebots.



(b) Reconfigurable circuit extension.



(c) Joint movement extension.

Figure 1 (a) Contracted and expanded amoebots (blue) in the triangular grid. Adjacent amoebots are connected by red edges. (b) The reconfigurable circuit extension. Amoebots are drawn as hexagons, pins are black circles on their borders and partition sets are drawn as black circles inside the hexagons. The partition sets are connected to the pins they contain. Partition sets in the same circuit have lines of the same color. (c) Amoebots before and after a joint movement.

2 Features

User Interface. The simulator's user interface (see Fig. 2) consists of a central viewport showing the amoebot structure, a top toolbar with visualization options and general controls, a bottom toolbar with simulation controls, a collapsible log, and two collapsible side panels. The left panel displays the state variables of a selected amoebot, which can be edited while the simulation is paused, and the right panel either displays advanced visualization settings or algorithm parameters, depending on the current mode. When launched, the simulator starts in *initialization mode*, where the algorithm can be selected and the amoebot structure is initialized. Starting the simulation switches to *simulation mode*, where the simulation controls are enabled and the selected algorithm is executed.

Visualization Options. Visualizing amoebot states and complex circuit configurations is the main task of AmoebotSim 2.0. Therefore, each amoebot can control its color, the colors of its circuits, and the placement of its partition sets. Using the top toolbar, the amoebot visualization can be switched between three different modes, and circuits and joint movement bonds can be toggled off. To visualize the dynamics of a simulation, all movements are animated and beeps are highlighted with a flashing effect. The animation speed can be adjusted and the animations can be turned off if desired. Additional UI elements like lines and arrows can be drawn on top of the amoebots to display structures like spanning trees.

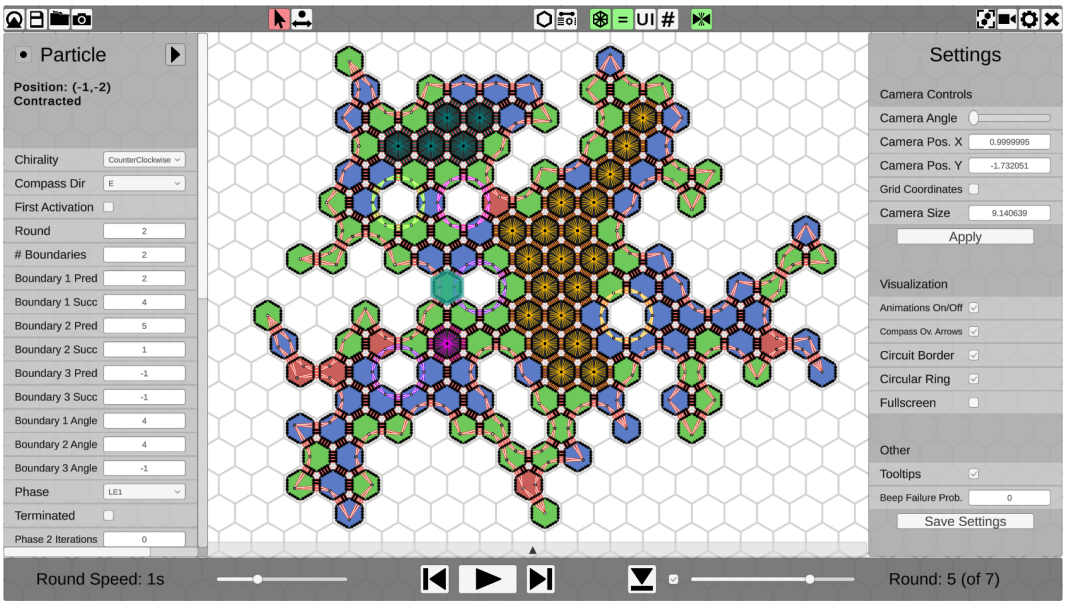


Figure 2 The AmoebotSim 2.0 user interface.

Algorithm Development. Algorithms are developed as C# classes that inherit from a base class and implement or override certain methods and fields. The simulation environment provides an algorithm creation tool that generates a functional template file with one click. Inheriting from the base class allows the simulator to detect the algorithm class by reflection, so no further registration in the core system is necessary. The base class and additional utilities provide a variety of methods and constants for all possible amoebot actions and information retrieval, e. g., discovering neighbors, manipulating direction values, setting up pin configurations, performing movements, etc. An algorithm can also be developed as a *subroutine*, allowing it to be reused by other algorithms and subroutines. A library of subroutines for some basic algorithms like leader election and boundary identification is provided. In addition to the documentation comments annotating all API source code, a comprehensive documentation with installation and usage instructions is hosted on GitHub.

Initialization. Every algorithm can implement an initialization procedure with custom parameters to place and initialize the amoebots. Using a global view of the structure, developers can establish any initial conditions, e. g., elect a leader amoebot, ensure hole-freeness, and provide input data. Afterwards, the amoebot structure can be edited manually by using the top toolbar and left side panel to add, remove, move, and edit amoebots.

Simulation History. The simulator records all rounds of a simulation. Using the bottom toolbar, any part of the recorded history can be reviewed and replayed. The history can also be cut off after the selected round so that the next rounds can be simulated again, possibly leading to a different outcome if the algorithm is randomized or some amoebot states were edited. Simulations can be stored in JSON files and loaded later.

References

1 Matthias Artmann, Tobias Maurer, Andreas Padalkin, Daniel Warner, and Christian Scheideler. AmoebotSim 2.0. Software, version 1.8.4., DFG-SCHE 1592/10-1, swhId: swh:1:dir:41a83f85a3e7c6c1d5a2fec79533bef04c7ff770 (visited on 2025-06-02). URL: https://github.com/martmannupb/AmoebotSim-2.0, doi:10.4230/artifacts.23289.

2 Matthias Artmann, Andreas Padalkin, and Christian Scheideler. On the shape containment problem within the amoebot model with reconfigurable circuits. *CoRR*, abs/2501.16892, 2025. doi:10.48550/arXiv.2501.16892.

3 Joshua J. Daymude, Robert Gmyr, and Kristian Hinnenthal. AmoebotSim: A Visual Simulator for the Amoebot Model of Programmable Matter. Available online at https://github.com/SOPSLab/AmoebotSim, 2021.

4 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The canonical amoebot model: Algorithms and concurrency control. *Distributed Computing*, 36(2):159–192, 2023. doi:10.1007/s00446-023-00443-3.

5 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief Announcement: Amoebot – A New Model for Programmable Matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 220–222, Prague, Czech Republic, 2014. ACM. doi:10.1145/2612669.2612712.

6 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating Amoebots via Reconfigurable Circuits. *Journal of Computational Biology*, 29(4):317–343, 2022. doi:10.1089/cmb.2021.0363.

7 Andreas Padalkin, Manish Kumar, and Christian Scheideler. Reconfiguration and Locomotion with Joint Movements in the Amoebot Model. In Arnaud Casteigts and Fabian Kuhn, editors, *3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2024)*, volume 292 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SAND.2024.18.

8 Andreas Padalkin and Christian Scheideler. Polylogarithmic Time Algorithms for Shortest Path Forests in Programmable Matter. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *43rd ACM Symposium on Principles of Distributed Computing*, PODC '24, pages 65–75, New York, NY, USA, 2024. ACM. doi:10.1145/3662158.3662776.

9 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The Structural Power of Reconfigurable Circuits in the Amoebot Model. In Thomas E. Ouldridge and Shelley F. J. Wickham, editors, *28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.28.8.

10 Tommaso Toffoli and Norman Margolus. Programmable Matter: Concepts and Realization. *International Journal of High Speed Computing*, 05(02):155–170, 1993. doi:10.1142/S0129053393000086.

Software for the Thompson and Funk Polygonal Geometry

Hridhaan Banerjee
Thomas Jefferson High School for Science and Technology, Alexandria, VA, USA

Carmen Isabel Day
California State University Channel Islands, CA, USA

Auguste H. Gezalyan
Department of Computer Science, University of Maryland, College Park, MD, USA

Olga Golovatskaia
Mount Holyoke College, South Hadely, MA, USA

Megan Hunleth
Montgomery Blair High School, Silver Spring, MD, USA

Sarah Hwang
University of Maryland, College Park, MD, USA

Nithin Parepally
Department of Computer Science, University of Maryland, College Park, MD, USA

Lucy Wang
Princeton University, NJ, USA

David M. Mount
Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

Metric spaces defined within convex polygons, such as the Thompson, Funk, reverse Funk, and Hilbert metrics, are subjects of recent exploration and study in computational geometry. This paper contributes an educational piece of software for understanding these unique geometries while also providing a tool to support their research. We provide dynamic software for manipulating the Funk, reverse Funk, and Thompson balls in convex polygonal domains. Additionally, we provide a visualization program for traversing the Hilbert polygonal geometry.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Thompson metric, Hilbert metric, Funk metric, balls

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.82

Category Media Exposition

Related Version *Previous Version*: `https://arxiv.org/abs/2503.01988`

Supplementary Material *Software (Source Code)*: `https://github.com/nithin1527/funk-geo-visualizer` [3], archived at `swh:1:dir:074df2e40eb974083f969981c7b82f83a059488b`
Audiovisual (Video): `https://www.youtube.com/watch?v=IeRIO25iK-o`
Software (Application): `https://funk-geo-visualizer.vercel.app/`

Funding The REU grant CNS-2150382 for REU-CAAR funded part of this work.

Acknowledgements Ryan Parker, for his coding suggestions.

1 Introduction

Thompson, Funk, reverse Funk, and Hilbert are closely related distance measures that apply to points in the interior of a convex body (see Section 2 for definitions). The Hilbert metric has applications in convex approximation [1, 2, 16], real analysis [9], linear algebra [10], clustering [12], and machine learning [15] and has been the subject of recent study for this reason [4–7, 11]. The Funk, reverse Funk, and Thompson metrics are comparatively unexplored but all share similarities with the Hilbert metric [13, 14].

© Hridhaan Banerjee, Carmen Isabel Day, Auguste H. Gezalyan, Olga Golovatskaia, Megan Hunleth, Sarah Hwang, Nithin Parepally, Lucy Wang, and David M. Mount; licensed under Creative Commons License CC-BY 4.0

In this paper, we present new dynamic software for manipulating the Funk, reverse Funk, and Thompson balls in convex polygonal domains. Additionally, we provide visualization software for traversing the Hilbert polygonal geometry. Our code is predominantly written in Javascript and is available at `https://github.com/nithin1527/funk-geo-visualizer`. To use the software, please go to `https://funk-geo-visualizer.vercel.app/`. For more information on the usage, please see the README on the GitHub repository or watch the accompanying video available at `https://www.youtube.com/watch?v=IeRIO25iK-o`.

2 Definitions

Throughout, Ω refers to a convex polygon in $\mathbb{R}^2$ with m sides. Let $\partial\Omega$ denote its boundary and let $\operatorname{int}\Omega$ refer to its interior. When clear, we omit explicit reference to Ω.

▶ **Definition 1** (Funk weak metric). *Given two distinct points $p, q \in \operatorname{int}\Omega$ in $\mathbb{R}^2$, let q' denote the intersection of the ray from p through q with $\partial\Omega$. Define the* Funk weak metric *to be:*

$$F_\Omega(p,q) \;=\; \ln \frac{\|p-q'\|}{\|q-q'\|},$$

where $F_\Omega(p,p) = 0$.

The above definition is also sometimes called the *forward Funk metric*. Note that the Funk weak metric is asymmetric. Its reverse, the *reverse Funk metric*, is defined ${}^rF_\Omega(p,q) = F_\Omega(q,p)$ [13]. Define the *Funk ball* centered at a point p of radius r, denoted $B_F(p,r)$ (note the subscript refers to the distance function), to be the set of points $q \in \operatorname{int}\Omega$ such that $F_\Omega(p,q) \le r$. The reverse Funk ball is defined analogously. These are scaled homotheties of Ω (see Figure 1(b) and (c)).

▶ **Lemma 2.** *The open forward Funk ball of radius r around a point p is the image of Ω under Euclidean homothety around the point p with dilation factor of $(1 - e^{-r})$ [13].*

▶ **Lemma 3.** *The open reverse Funk ball of radius r around a point p is the image of Ω under Euclidean homothety around the point p with dilation factor of $(e^r - 1)$ [13].*

The *Hilbert metric* can be defined as the average of the forward and reverse Funk metrics [13].

▶ **Definition 4** (Hilbert metric). *Given two distinct points p, q in a convex polygonal $\Omega \subset \mathbb{R}^2$ we define the* Hilbert metric *to be:*

$$H_\Omega(p,q) \;=\; \frac{1}{2}(F_\Omega(p,q) + {}^rF_\Omega(p,q)),$$

Note that the Hilbert metric is a proper metric, obeying the identity of indiscernibles, symmetry, and the triangle inequality. Hilbert balls are polygons with at most $2m$ sides (see Figure 1(a)). For a characterization of Hilbert balls see "Balls in the Hilbert Polygonal Geometry" by Nielsen and Shao [11].

The notion of the Hilbert metric gave rise to the definition of another metric by A.C. Thompson in 1963 called the *Thompson metric* [14] on convex cones. The formulation of this metric can be generalized through the Funk metric.

▶ **Definition 5** (Thompson metric). *Given two distinct points p, q in a convex polygonal $\Omega \subset \mathbb{R}^2$ we define the* Thompson metric *to be:*

$$T_\Omega(p,q) \;=\; \max(F_\Omega(p,q), {}^rF_\Omega(p,q)),$$

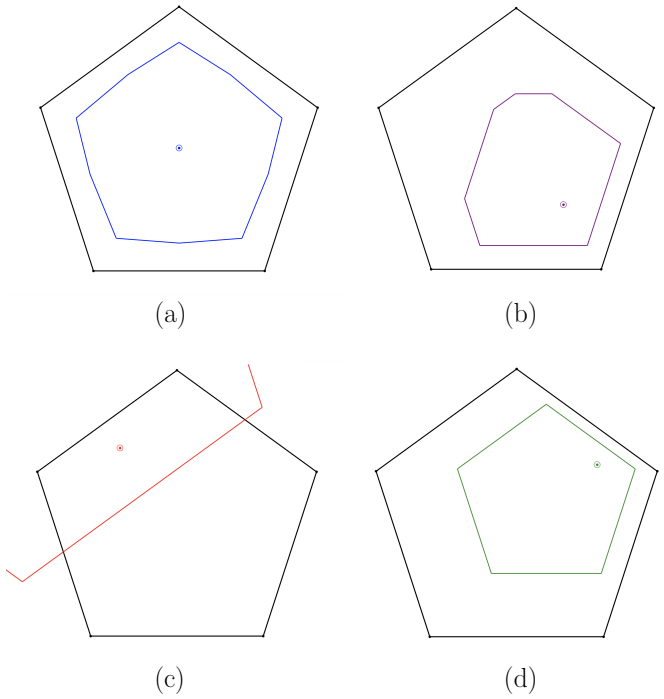


Figure 1 The Hilbert (a), Funk (b), Reverse Funk (c), and Thompson (d) balls around a point.

Note that the identity of indiscernibles and triangle inequality clearly carry over from the Funk metric. The symmetry of the Thompson metric follows from the fact that it is the maximum of the forward and reverse of a weak metric.

The Thompson ball (see Figure 1(d)) is the intersection of the forward and reverse Funk balls. Since both these balls are convex (as they are homotheties of Ω intersected with Ω [13]), computing the Thompson ball reduces to calculating the intersection of two convex polygons. This can be done linearly in the complexity of the polygons. Hence, we get:

▶ **Lemma 6.** *The Thompson ball around a point p with radius r has at most $2m$ sides and is the intersection of the forward and reverse Funk balls around p with radius r.*

Proof. This follows from the fact that each segment of the Funk balls can contribute at most once to the complexity of the Thompson ball by convexity. ◀

Thompson balls have a property that sets them apart from many other metrics.

▶ **Observation 7.** *Thompson balls are not pseudo-disks (see Figure 2(a)).*

We believe the following to be a new result.

▶ **Lemma 8.** *A Thomspon ball of radius r centered at p can be nested between two Hilbert balls of radius $0.5r$ and r centered at p (see Figure 2(b)).*

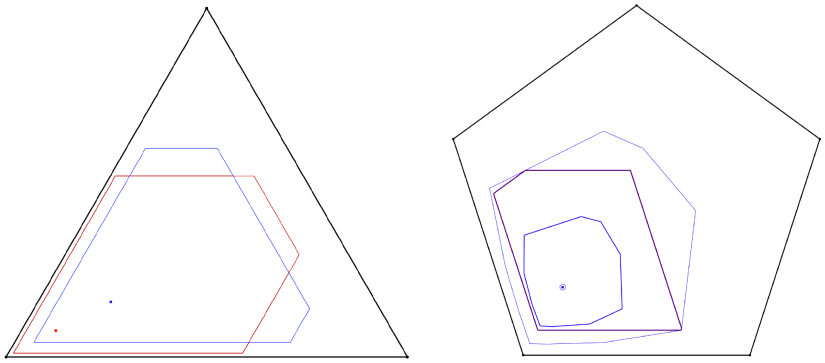
Proof. Given a convex body $\Omega \subset \mathbb{R}^d$, for every point $p \in \Omega$ and radius $0 < r$, we will show:

$$B_H(p, r/2) \subset B_T(p, r) \subset B_H(p, r).$$

To prove that $B_T(p, r) \subset B_H(p, r)$, we must show that the Thompson distance from p to a point on the boundary of the ball is greater than or equal to the Hilbert distance. This follows directly from the definition of the Hilbert and Thompson metric distances:

$$H(a, b) = \frac{1}{2}(F(a, b) + F(b, a)) \leq 2 \cdot \frac{1}{2}(\max(F(a, b), F(b, a)) = T(a, b)).$$

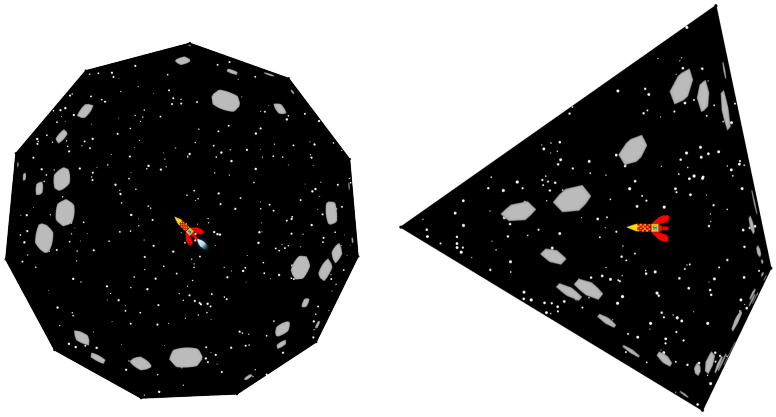
To prove that $B_H(p, r/2) \subset B_T(p, r)$, we must show that the Thompson distance from p to a point on the boundary of the ball is less than or equal to the Hilbert distance. This follows from the fact that if $H(a, b) = \frac{1}{2}r$ then $\frac{1}{2}(F(a, b) + F(b, a)) = \frac{1}{2}r$ so $F(a, b) + F(b, a) = r$ and so $T(a, b) = \max(F(a, b), F(b, a)) \leq r$. ◀



(a) (b)

Figure 2 (a) Thompson balls are not pseudo-disks, (b) a Thompson ball of radius 1 between two Hilbert balls of radii 1/2 and 1.

3 Traversing the Hilbert Geometry



(a) (b)

Figure 3 (a) Travelling in a hendecagon and (b) after travelling some distance in the square.

We provide a visualization software for inserting Hilbert balls into a user-specified Hilbert polygonal geometry, and then traveling around that geometry (see Figure 3). In order to give users the ability to move in the Hilbert geometry, we implement a projective mapping that first affinely shifts Ω so that its centroid is at the origin O. Then, using a map ϕ_v, we map $O + v$ to the centroid $\phi_v(\Omega)$ where v is an accumulated displacement vector. This mapping is given by:

$$\phi_v(p) \;=\; \frac{p}{1+\langle p, v\rangle},$$

where p is an arbitrary point [8].

However, this projective mapping often deforms Ω into very skinny forms. To handle this situation, we first capture the approximate John ellipsoid of the original shape scaled with the Mahalanobis distance. After movement, we calculate the new approximate John ellipsoid with respect to the projection of the vertices of Ω. Then, we map the new John ellipsoid to the unit circle using a Cholesky decomposition. Lastly, we map the unit circle to the original approximate John ellipsoid to center the new geometry. Since all the maps we use are affine maps, they preserve the Hilbert metric. For more information on the mapping used, see the README on the GitHub repository.

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, 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. Software, swhId: swh:1:dir:074df2e40eb974083f969981c7b82f83a059488b (visited on 2025-06-03). URL: https://github.com/nithin1527/funk-geo-visualizer, doi:10.4230/artifacts.23291.
4. 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.
5. 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.
6. 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.
7. 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.
8. Ivan Izmestiev. Matching centroids by a projective transformation. *Geometriae Dedicata*, 217(3):53, 2023.
9. 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.
10. Carlangelo Liverani and Maciej P. Wojtowski. Generalization of the Hilbert metric to the space of positive definite matrices. *Pacific Journal Of Mathematics*, 166(2):339, 1994.

11 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.

12 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.

13 Athanase Papadopoulos and Marc Troyanov. *Handbook of Hilbert geometry*, volume 22 of *IRMA Lectures in Mathematics and Theoretical Physics*. European Mathematical Society Publishing House, 2014. doi:10.4171/147.

14 Anthony C. Thompson. On certain contraction mappings in a partially ordered vector space. *Proceedings of the American Mathematical Society*, 14(3):438–443, 1963. doi:10.2307/2033816.

15 Julian Vanecek, Auguste H Gezalyan, and David M Mount. Support vector machines in the hilbert geometry. *FWGIC24*, 2024. URL: https://www.cs.tufts.edu/research/geometry/FWCG24/papers/FWCG_24_paper_27.pdf.

16 Constantin Vernicos and Cormac Walsh. Flag-approximability of convex bodies and volume growth of Hilbert geometries. In *Annales Scientifiques de l'École Normale Supérieure*, volume 54, pages 1297–1315, 2021.

French Onion Soup, Ipelets for Points and Polygons

Klint Faber
University of Maryland, College Park, MD, USA

Auguste H. Gezalyan
Department of Computer Science, University of Maryland, College Park, MD, USA

Adam Martinson
University of Maryland, College Park, MD, USA

Aniruddh Mutnuru
University of Maryland, College Park, MD, USA

Nithin Parepally
Department of Computer Science, University of Maryland, College Park, MD, USA

Ryan Parker
University of Maryland, College Park, MD, USA

Mihil Sreenilayam
University of Maryland, College Park, MD, USA

Aram Zaprosyan
University of Maryland, College Park, MD, USA

David M. Mount
Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

There are many structures, both classical and modern, involving point-sets and polygons whose deeper understanding can be facilitated through interactive visualizations. The Ipe extensible drawing editor, developed by Otfried Cheong, is a widely used software system for generating geometric figures. One of its features is the capability to extend its functionality through programs called Ipelets. In this media submission, we showcase a collection of new Ipelets that construct a variety of geometric structures based on point sets and polygons. These include quadtrees, trapezoidal maps, beta skeletons, floating bodies of convex polygons, onion graphs, fractals (Sierpiński triangle and carpet), simple polygon triangulations, and random point sets in simple polygons. All our Ipelets are programmed in Lua and are freely available.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Hilbert metric, Macbeath Regions, Polar Bodies, convexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.83

Category Media Exposition

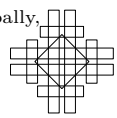
Related Version *Previous Version*: `https://arxiv.org/abs/2503.01979`

Supplementary Material *Software*: `https://github.com/nithin1527/onion-soup` [8]
archived at `swh:1:dir:8d9d49631e2688049de55f2c874e85e56cb50620`

1 Introduction

We present several **Ipelets** for sets of points and polygons in the $\mathbb{R}^2$ plane. These Ipelets include computing quadtrees, trapezoidal maps, onion decompositions, random point sampling in polygons, beta skeletons, floating bodies, the Sierpiński carpet and triangle fractals, and simple polygon triangulations. Some of these are also implemented in CGAL [9]. All the Ipelets are programmed in Lua and are freely available at `https://github.com/nithin1527/onion-soup`. To install an Ipelet, download the file and place it in the `ipelets` subfolder of your Ipe folder.

© Klint Faber, Auguste H. Gezalyan, Adam Martinson, Aniruddh Mutnuru, Nithin Parepally, Ryan Parker, Mihil Sreenilayam, Aram Zaprosyan, and David M. Mount;
licensed under Creative Commons License CC-BY 4.0

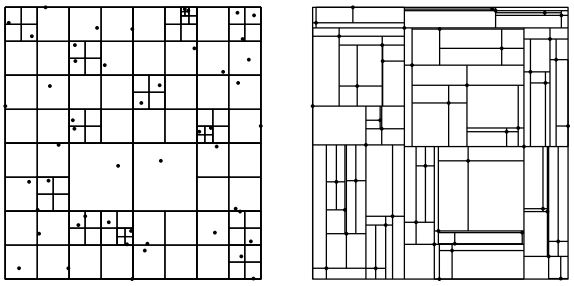


2 Points

In this section, we describe the geometric structures that our Ipelets compute on points. There are a few Ipelets available that already build geometric structures on point sets, such as Voronoi diagrams [5, 14] and Delaunay triangulations [5]. We contribute to this with the additions of the following Ipelets.

2.1 Quadtrees

A quadtree is a data structure used to recursively divide two-dimensional space into four regions/nodes, usually named northwest, northeast, southwest, and southeast [6, Ch. 14]. A point quadtree is a specific type of quadtree in which each node houses a point and its four regions are determined by dividing its region into four sub-regions, with the point itself as the center (see Figure 1(b)). A point-region quadtree is another type of quadtree in which each node represents a region of space with a certain maximum capacity of points, such that points are only housed in leaf nodes (see Figure 1(a)). The importance of quadtrees arises from their enabling of efficient operations on data points in two-dimensional space. Consequently, their applications include areas of image representation/processing, mesh generation, and two-dimensional collision detection [6, 16]. We contribute an Ipelet for generating both these types of quadtrees, and we produce the array format for its contents in a copyable form.

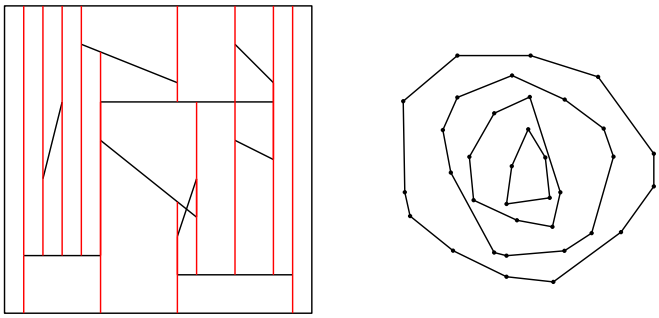


(a) (b)

Figure 1 (a) Point-region quadtree and (b) point quadtree.

2.2 Trapezoidal Maps

Given a collection of line segments, a trapezoidal map is a subdivision of the plane that respects these segments [6, Ch. 6.1]. The trapezoidal map is created by shooting vertical rays up and down from each of the endpoints of the segments until the vertical line intersects another segment or the bounding box (see Figure 2(a)). In practice, trapezoidal maps allow users to perform planar point location, with respect to the trapezoids that make up the map, in $O(\log(n))$ time. This finds applications in many areas, including geographic information systems, computer-aided design, among others [6, 16]. We provide an Ipelet which, given a set of line segments and an optional bounding box, computes a trapezoidal map of these line segments. It also provides a method for copying the details of the map in an array format.



(a) (b)

Figure 2 (a) Trapezoidal map of a set of segments and (b) an onion decomposition of a point set.

2.3 Onion Decompositions

▶ **Definition 1** (Onion Decomposition). *Given a finite set of points $P \subset \mathbb{R}^n$, the onion decomposition of P is the nested sequence of convex sets $\{C_0, C_1, \ldots, C_m\}$, where:*
(1) *C_i is the convex hull of P_i.*
(2) *$P_0 = P$ and $P_i = P \setminus \bigcup_{j=0}^{i-1} C_j$, for $i \geq 1$.*
(3) *P_m is the last P_i containing any points.*

This is the sequence of convex polygons that result by repeatedly taking the convex hull of the set, and then removing these points from the set [15, Ch. 4] [3] (see Figure 2(b)). Onion decompositions of points have found a variety of uses in robust statistics [1, 7] and fractional cascading [4]. We provide an Ipelet that allows users to select a point set and generate its onion decomposition in the plane.

2.4 Beta Skeletons

▶ **Definition 2** (Beta Skeleton). *Given a set of points (vertices) V and a parameter $\beta \in \mathbb{R}^+$, the beta skeleton of the set of points is the graph $G = (V, E)$ where: There exists an edge $\{v_1, v_2\} \in E$ if and only if for all $v_3 \neq v_1, v_2$, the angle $\angle v_1 v_3 v_2$ is smaller than θ, where:*

$$\theta = \begin{cases} \arcsin \frac{1}{\beta} & \text{if } \beta \geq 1, \\ \pi - \arcsin(\beta) & \text{if } \beta \leq 1. \end{cases}$$

Note that the Gabriel graph and the relative neighborhood graph (RNG) of the point set arise as special cases when $\beta = 1$ and $\beta = 2$, respectively. The beta skeleton [18, Ch. 32] has applications ranging from machine learning [19] to astronomy [10].

3 Polygons

In this section, we describe the geometric structures that our Ipelets compute on polygons, for more, see "Ipelets for the Convex Polygonal Geometry" [13].

3.1 Floating Body

Given a convex body $\Omega \subset \mathbb{R}^d$, a *cap* is the intersection of Ω with a halfspace H. Letting h denote the bounding hyperplane of H, the *base* of the cap is $h \cap \Omega$. When $d = 2$, the *area* of a cap is its Lebesgue measure.

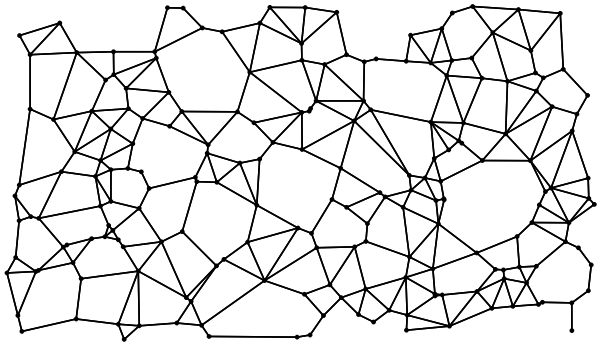
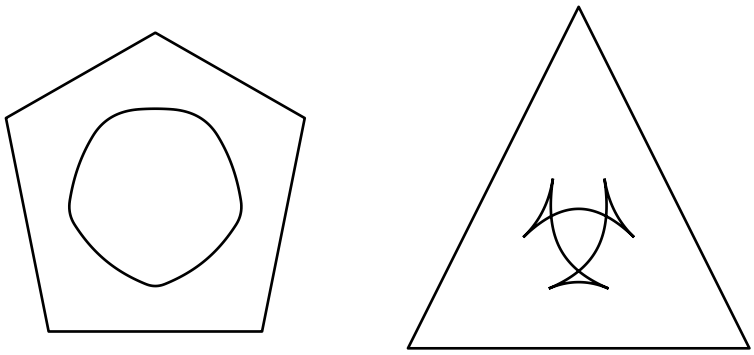


Figure 3 Beta skeleton for a point set for $\beta = 1$, which is equivalent to the Gabriel graph.

▶ **Definition 3** (Dupin Floating Body). *Given a convex body $\Omega \subset \mathbb{R}^n$ and $\delta \in [0, 1]$, the Dupin floating body, F_δ, is the (possibly non-convex) body defined by the set of points that are the centroids of the bases of all caps that cut off a volume of $\delta \cdot \mathrm{vol}(\Omega)$ [2, Ch. 3].*

▶ **Definition 4** (Convex Floating Body). *Given a convex body $\Omega \subset \mathbb{R}^n$ and $\delta \in [0, 1]$, the convex floating body K_δ is the intersection of all halfspaces whose defining hyperplanes cut off a set of volume $\delta \cdot \mathrm{vol}(\Omega)$ from Ω [2, Ch. 3] [17].*

Generally, the Dupin floating body need not be convex (see Figure 4(b)), but when it is convex it equals the convex floating body [2]. We provide an Ipelet for generating the Dupin floating body in arbitrary convex polygons (see Figure 4). For the purposes of the Ipelet we take δ as the percentage of the area of the given polygon in $\mathbb{R}^2$ to cut away with caps. Floating bodies are used for understanding the boundary structure of convex sets through affine surface area [17].

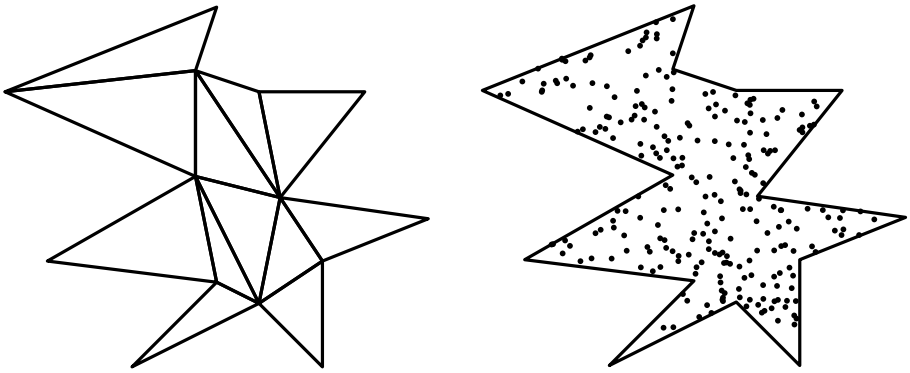


(a) (b)

Figure 4 Two examples of the Dupin floating body, (a) convex and (b) non-convex.

3.2 Polygon Triangulations and Point Sampling

Polygon triangulations are a ubiquitous tool in computer science [6, Ch. 3]. One application of such triangulations is that it allows efficient point sampling inside non-convex polygons. To do this, points are randomly chosen from inside the triangles (weighted by area) using barycentric coordinates. We provide an Ipelet which, given a polygon, decomposes it into a triangulation (based on code by Nils Olovsson [12]) and randomly places a user specified amount of points in it using this method.

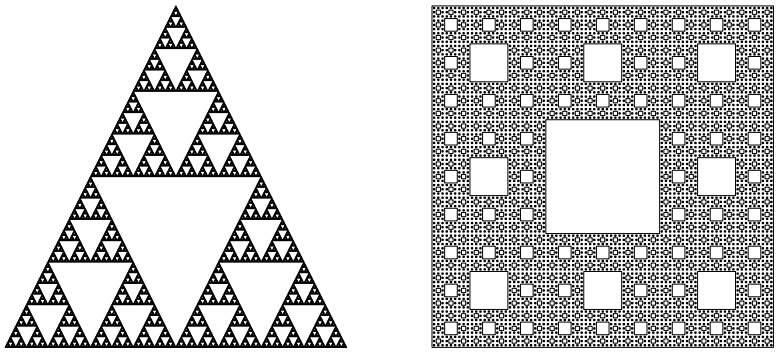


(a) (b)

Figure 5 (a) Triangulating a simple polygon and (b) randomly sampling points in it.

3.3 Sierpiński Triangle and Carpet

The Sierpiński Triangle and Carpet [11, Ch. 1] are two self similar fractals defined on simple polygons. They are known for having a few interesting properties, such as Lebesgue measure 0 and Haussdorf dimensions $\log_2(3)$ and $\log_3(8)$. They contain uncountably many points.



(a) (b)

Figure 6 (a) The Sierpiński Triangle and (b) The Sierpiński Carpet.

References

1 Vic Barnett. The ordering of multivariate data. *Journal of the Royal Statistical Society: Series A (General)*, 139(3):318–344, 1976.

2 Umut Caglar. Floating bodies. Master's thesis, Case Western Reserve University, 2010.

3 Bernard Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985. doi:10.1109/TIT.1985.1057060.

4 Bernard Chazelle, Leo J Guibas, and Der-Tsai Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25(1):76–90, 1985. doi:10.1007/BF01934990.

5 Otfried Cheong. The IPE extensible drawing editor. Version 7.2.28, 2024-03-12. URL: https://ipe.otfried.org/.

6 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2010.

7 William F Eddy. Convex hull peeling. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982: Part I: Proceedings in Computational Statistics*, pages 42–47. Springer, 1982.

8 Klint Faber, Auguste H. Gezalyan, Adam Martinson, Aniruddh Mutnuru, Nithin Parepally, Ryan Parker, Mihil Sreenilayam, Aram Zaprosyan, and David M. Mount. Ipelets for Points and Polygons. Software, swhId: `swh:1:dir:8d9d49631e2688049de55f2c874e85e56cb50620` (visited on 2025-06-05). URL: `https://github.com/nithin1527/onion-soup`, `doi:10.4230/artifacts.23340`.

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 Feng Fang, Jaime Forero-Romero, Graziano Rossi, Xiao-Dong Li, and Long-Long Feng. β-skeleton analysis of the cosmic web. *Monthly Notices of the Royal Astronomical Society*, 485(4):5276–5284, 2019.

11 Gilbert Helmberg. *Getting acquainted with fractals*. Walter de Gruyter, 2008.

12 Nils Olovsson. Ear clipping triangulation, 2021. Accessed: 2025-02-25. URL: `https://nils-olovsson.se/articles/ear_clipping_triangulation/`.

13 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

14 postechDNN. Ipelet. `https://github.com/postechDNN/postechDNN/commits/ipelet/dnn/IPELET`, 2022. (Commit hash: de82a8d).

15 Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

16 Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

17 Carsten Schütt and Elisabeth Werner. The convex floating body. *Mathematica Scandinavica*, pages 275–290, 1990.

18 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

19 Wan Zhang and Irwin King. Locating support vectors via /spl beta/-skeleton technique. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, volume 3, pages 1423–1427. IEEE, 2002.

Incremental and Interactive PQ- and PC-Trees

Simon D. Fink
Algorithms and Complexity Group, TU Wien, Austria

Dominik Peters
CNRS, LAMSADE, Université Paris Dauphine – PSL, France

Abstract

PQ-trees [2, 3] (and their more general variant PC-trees [13, 17]) are a well-known data structure for representing the set of linear (or for PC-trees, circular) orders respecting a given set of consecutivity constraints. Each such constraint is specified by a set of elements and requires that these elements appear consecutively in the linear (or circular) order; thus, they disallow the set to be interleaved with its complement. The main operation supported by these data structures is thus the so-called *update*, which takes as input a set that forms an additional constraint, and in response changes the tree in order to restrict the represented orders to those satisfying the new constraint. Interpreting a given tree is straightforward: leaves represent the underlying elements, while inner nodes either allow the order of their subtrees to be reversed (Q/C-nodes) or to be arbitrarily permuted (P-nodes). However, the way this structure and the set of represented orders change under updates is less intuitive. We present an interactive web app that allows users to specify sets of consecutivity constraints in the form of a 0/1-matrix and then calculates and visualizes the corresponding PQ- or PC-tree. The constraints can then be changed dynamically while observing how this changes the structure of the tree and the set of represented orders. Through this interactive exploration, we hope to make PQ- and PC-trees more accessible to a wider audience.

2012 ACM Subject Classification Human-centered computing → Graphical user interfaces; Mathematics of computing → Permutations and combinations; Applied computing → Interactive learning environments; Theory of computation → Computational geometry

Keywords and phrases PQ trees, PC trees, planarity, consecutive ones problem, interactive exploration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.84

Category Media Exposition

Supplementary Material *InteractiveResource (Website)*: `https://n-coder.github.io/pqtree.js`
Software (Source Code): `https://github.com/N-Coder/pqtree.js`
archived at `swh:1:dir:711cf71799908f0a3e15dadcdd93f0499d88cd1a`

Funding *Simon D. Fink*: Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].

1 Background

PQ- and PC-trees have applications in many graph drawing problems related to planarity [3, 17] and its variants, such as Level Planarity [6] and Cluster Planarity [9], as well as other layout problems such as NodeTrix [15] or the geometric Segment Intersection Representation problem [11]. More broadly, they are also used for solving variants of the Travelling Salesperson Problem [4], in Computational Biology (such as for Genome Ordering [14]) and to solve the Consecutive Ones Problem on 0/1-matrices [3, 13], which has many applications, for example in the analysis of preferences in Computational Social Choice [8].

A *PQ-tree* T is a rooted tree the represents a set of linear orders on some ground set X [2, 3]. The leaves of T are exactly the elements of X, while its inner nodes are labelled as either P-node or Q-node; see Figure 1(b). A *P-node* allows arbitrarily permuting its incident subtrees, while a *Q-node* comes with a fixed order of its incident subtrees that may

© Simon D. Fink and Dominik Peters;
licensed under Creative Commons License CC-BY 4.0

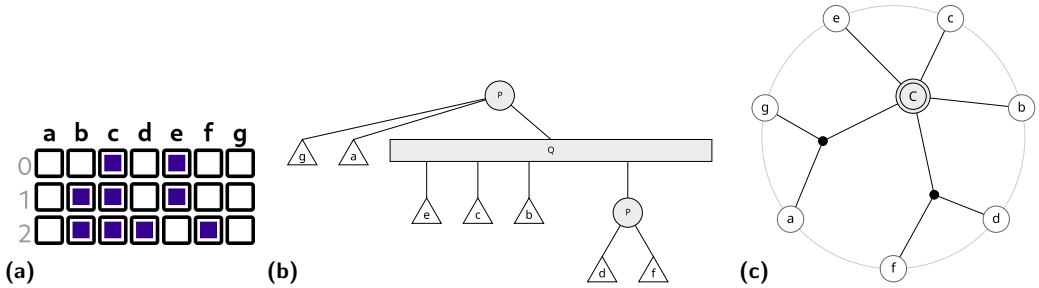


Figure 1 **(a)** A binary matrix representing the elements $X = \{a, b, c, d, e, f, g\}$ as its columns and the three consecutivity restrictions $\{c, e\}, \{b, c, e\}, \{b, c, d, f\}$ through its three rows. **(b)** A PQ-tree representing all linear orders satisfying the restrictions from (a). **(c)** A PC-tree representing all circular orders satisfying the restrictions from (a). Images generated by the portfolio website.

only be reversed. All orders of leaves obtained by making these changes at the inner nodes are the linear orders *represented* by the PQ-tree T. Similarly, a *PC-tree* T is an unrooted tree that represents a set of circular orders on some ground set X [13, 17]. In addition to being unrooted, its only further difference to a PQ-tree is that Q-nodes are called *C-nodes*, but otherwise behave the same in only allowing their fixed order or the reversal thereof; see Figure 1(c). As there is no designated root, the leaf orders obtained from a PC-tree can be circularly shifted, yielding circular instead of linear orders. It has been shown that still, both kinds of trees are actually linear-time equivalent [12].

A *trivial* PQ- or PC-tree consists only of a single inner P-node and represents all permutations of its leaves X. Any tree can be obtained from the trivial one by applying a sequence of *update* operations to it. Each update takes as input a subset $A \subset X$ and changes the tree such that the represented orders are restricted to those in which A is consecutive. If there are no represented orders where A is consecutive, the *null-tree* representing an empty set of orders is returned. The update operation can be implemented to run in $O(|A|)$ time [2, 17], although (as we discuss at the end of this paper) this algorithm is far from trivial to implement or explain [1, 7, 10].

In this way, consecutivity restrictions are represented as a family of subsets of X. This family can be represented as a binary 0/1 matrix, where each column corresponds to an element of X, and each consecutivity restriction corresponds to a row in the matrix, requiring exactly its 1-entries to be consecutive; see Figure 1(a). Note that the problem of finding orders satisfying all consecutivity constraints can now be seen as reordering the columns of the matrix such that all ones are (circularly) consecutive, which is also known as the (Circularly) Consecutive Ones Problem [3, 13].

2 Our Portfolio

We use the matrix representation in our portfolio to allow interactively specifying constraints. After the user changes the matrix, the PQ- or PC-tree displayed next to it is automatically updated to represent the desired orders. The columns of the matrix can be reordered manually through drag and drop, or automatically on clicking a button, such that the matrix has the (Circularly) Consecutive Ones Property. Additionally, the website displays how many admissible orders there are and also provides an explicit enumeration of them. In case of a no-instance (where no order satisfies the consecutivity restrictions), we highlight the first restriction that would yield a null-tree, and display the tree obtained from the

restrictions up to right before that point. This makes it easy to verify which elements of the next restriction cannot be made consecutive. A guided tour further provides a tutorial explaining the PQ- and PC-tree data structures and how the website can be used. Allowing to interactively explore PQ- and PC-trees and their relationship to consecutivity-constrained orders, we hope to make this data structure more accessible to a wider audience. We also provide visually-pleasing automatic visualizations that can, e.g., easily used for scientific work building on these structures; see Figure 1.

For PQ-trees, we use a simple top-down tree layout, while the layout of the circular and unrooted PC-trees is slightly more involved. Here, we use a Tutte drawing [16], fixing the leaves in an order admissible by the tree on a circle and then using the force-based approach to draw the inner nodes within the circle. Chiu, Eppstein and Goodrich suggested to assign weights to edges to further improve the aesthetics of Tutte drawings [5]. They use the formula a/r^i, where a is some scaling constant, r is the weighting parameter, and i is the minimum distance of the edge to a leaf. As the scaling is applied linearly to all weights and thus does not influence the resulting drawing, we set $a = 1$ in our implementation and only allow the user to tune the r value to their liking. Interestingly, while the authors suggested r values of 3 and 5 in their paper [5], we find values of 1 or slightly below to better suit the (rather small) graphs we draw.

Future improvements to the website could also allow interactively changing the displayed order right within the currently visualized tree according to the P- and Q/C-node rules. Furthermore, an interactive tour through the individual steps used to update a PQ- or PC-tree could further help in understanding this algorithm. Note that the update of PQ-trees is through a rather involved recursive matching and application of 9 different "templates" [2], resulting in an algorithm that is famously hard to implement correctly [7, 10]. The update on PC-trees streamlines this to an update in one go, although its correct linear-time implementation is still rather challenging [1, 10]. Thus, some trade-offs for simplicity of explanation will need to be made when interactively executing these algorithms, as is e.g. also done by the static explanation of the update we currently link to [1].

References

1 Rami Benelmir, Christoph Dürr, Erisa Kohansal, and Yanni Lefki. PC Trees, January 2024. URL: https://tryalgo.org/en/data%20structures/2024/01/03/pc-trees/.

2 Kellogg S. Booth. *PQ-tree algorithms*. PhD thesis, University of California, Berkeley, 1975.

3 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

4 Rainer E. Burkard, Vladimir G. Deĭneko, and Gerhard J. Woeginger. The Travelling Salesman and the PQ-Tree. *Mathematics of Operations Research*, 23(3):613–623, 1998. Publisher: INFORMS. doi:10.1287/MOOR.23.3.613.

5 Alvin Chiu, David Eppstein, and Michael T. Goodrich. Manipulating Weights to Improve Stress-Graph Drawings of 3-Connected Planar Graphs. In Michael A. Bekos and Markus Chimani, editors, *Graph Drawing and Network Visualization*, pages 141–149, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-49275-4_10.

6 G. Di Battista and E. Nardelli. Hierarchies and planarity theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):1035–1046, November 1988. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics. doi:10.1109/21.23105.

7 Christoph Dürr. PQ trees, December 2017. URL: https://tryalgo.org/en/data%20structures/2017/12/15/pq-trees/.

8 Edith Elkind, Martin Lackner, and Dominik Peters. Preference Restrictions in Computational Social Choice: A Survey, May 2022. arXiv:2205.09092 [cs]. doi:10.48550/arXiv.2205.09092.

9 Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, and Paul Spirakis, editors, *Algorithms — ESA '95*, volume 979, pages 213–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. Series Title: Lecture Notes in Computer Science. doi:10.1007/3-540-60313-1_145.

10 Simon D. Fink, Matthias Pfretzschner, and Ignaz Rutter. Experimental Comparison of PC-Trees and PQ-Trees. *ACM J. Exp. Algorithmics*, 28:1.10:1–24, 2023. Publisher: Association for Computing Machinery (ACM). doi:10.1145/3611653.

11 Simon D. Fink, Matthias Pfretzschner, and Peter Stumpf. Segment Intersection Representations, Level Planarity and Constrained Ordering Problems, February 2025. arXiv:2502.16621 [cs]. doi:10.48550/arXiv.2502.16621.

12 Bernhard Haeupler and Robert E. Tarjan. Planarity Algorithms via PQ-Trees (Extended Abstract). *Electronic Notes in Discrete Mathematics*, 31:143–149, August 2008. doi:10.1016/j.endm.2008.06.029.

13 Wen-Lian Hsu and Ross M. McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296(1):99–116, March 2003. doi:10.1016/S0304-3975(02)00435-8.

14 Gad M. Landau, Laxmi Parida, and Oren Weimann. Using PQ trees for comparative genomics. *Lecture Notes in Computer Science*, 3537:128–143, 2005. doi:10.1007/11496656_12.

15 Giuseppe Liotta, Ignaz Rutter, and Alessandra Tappini. Simultaneous FPQ-ordering and hybrid planarity testing. *Theoretical Computer Science*, 874:59–79, June 2021. doi:10.1016/j.tcs.2021.05.012.

16 W. T. Tutte. How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 1963. doi:10.1112/plms/s3-13.1.743.

17 Shih Wei-Kuan and Hsu Wen-Lian. A new planarity test. *Theoretical Computer Science*, 223(1):179–191, July 1999. doi:10.1016/S0304-3975(98)00120-0.

Finding Shortest Reconfiguration Sequences for Modular Robots

UML Modular Robotics Group
University of Massachusetts Lowell, MA, USA

Hugo A. Akitaya
University of Massachusetts Lowell, MA, USA

Andrew Clements
University of Massachusetts Lowell, MA, USA

Sam Downey
University of Massachusetts Lowell, MA, USA

Jonathan Eisenbies
University of Massachusetts Lowell, MA, USA

Soham Samanta
Greater Commonwealth Virtual School, Medford, MA, USA

Gabriel Shahrouzi
University of Massachusetts Lowell, MA, USA

Frederick Stock
University of Massachusetts Lowell, MA, USA

Abstract

This paper introduces a set of tools built to help researchers design algorithms for modular robots. These tools can brute force solutions to specific reconfigurations, visualize movements of modular robots, and can be used to design specific configurations of robots. Multiple models of modular robots are supported, and can be added by users.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases modular reconfigurable robots, sliding cube model, reconfiguration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2025.85

Category Media Exposition

Funding Research supported by the NSF award CCF-2348067.

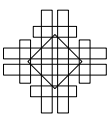
1 Introduction

In *Modular Robotic Systems* (MRS) a "robot" consists of several small independent robotic units (called *modules*) that can move around each other (re-configure) to change the overall shape of the robot. This gives the system flexibility to adapt to different situations and resilience since modules are interchangeable. In many MRS models, a *configuration* of the system is a polyform composed of lattice-aligned modules. A *move* transforms a configuration C into another configuration C', where C and C' differ by the position of a single module m. A sequence of configurations $C^0, C^1, C^2, C^3, \ldots$ where each C^{i+1} is obtained by applying a single move to C^i is called a *reconfiguration sequence*. The *single backbone condition* [6] is often required, to move a module m in a configuration C, $C \setminus \{m\}$ must be connected.

A moving module must not collide with a stationary module. The *free space requirements* define the local arrangement of modules required to prevent collisions. Many models are defined by their free space requirements, such as sliding [5, 7] or pivoting [3, 8] cubes. These two models and their 2D analogs (sliding and pivoting squares) are the focus of this paper. Their free-space requirements are shown in Figure 1.

A model is *universal* if any configuration can be reconfigured into any other. Clearly, MRS models that admit universal reconfiguration are particularly desirable. However, this is not a common property, and small changes in free space requirements can affect whether a MRS model is universal or not. For example, the sliding square and cube models are universal [2, 5], while the pivoting square model is not. Further, it is PSPACE-complete to decide whether two given configurations can even be reconfigured into each other [1].

© UML Modular Robotics Group, Hugo A. Akitaya, Andrew Clements, Sam Downey, Jonathan Eisenbies, Soham Samanta, Gabriel Shahrouzi, and Frederick Stock;
licensed under Creative Commons License CC-BY 4.0



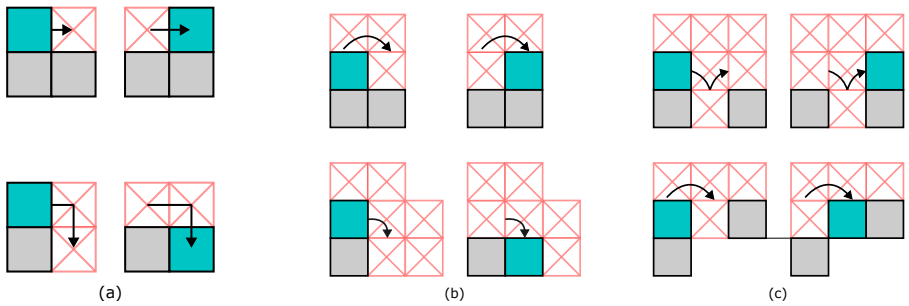


Figure 1 Free space requirements different movement models. Cubes with x's must be empty, the blue cube is the moving module. (a) sliding cube model (b) the restricted pivoting cube (c) "leapfrog" and "monkey" moves from [1].

Contribution. This paper introduces three new tools built to ease this process. A user can define a modular robot model and define starting and ending configurations and then via brute-force search (taking exponential time), find the shortest reconfiguration sequence between them. This can not only find optimal reconfiguration sequences but also be used to verify the correctness of a meta-module.[1] This work would be presented at CG Week via a live demonstration. At the time of writing this is an active research project, hence the software maybe updated, the linked GitHub will have updated documentation.

2 Software

Tiler. Our tool "3D Tiler" is a web-based application designed to create and manipulate 3D polyforms, inspired by the SVG Painter by Erik Demaine[2]. It supports the cube and rhombic dodecahedron lattices. The tool provides a variety of controls for users to interact with their designs. Layer controls allow users to move up or down layers and highlight the current layer, while color controls enable changing the color. The interface is illustrated in Figure 2. Additionally, 3D Tiler offers export options to JSON and OBJ formats, facilitating the sharing and utilization of designs in other applications. Importing designs from JSON files is also supported, making collaboration and modification straightforward.

Pathfinder. We built a tool "Pathfinder" which finds the shortest reconfiguration sequence between two configurations. Move definitions (free space requirements), and initial and final configurations are specified as input JSON files. A variety of cube/square models, as well as rhombic dodechahedra are supported "out of the box" and users can specify their own models.

It natively supports two search algorithms to explore and find shortest paths in the configuration space. We are given initial and final configurations[3] together with a description of the allowed moves[4]. The pathfinder tool uses a shortest path algorithm to incrementally build the configuration graph G. At each step, starting with the initial configuration, we

[1] The source code is available at `https://github.com/Modular-Robotics-Group/modular-robotics/`.
[2] Available at `https://erikdemaine.org/svgpainter/`
[3] `https://github.com/Modular-Robotics-Group/modular-robotics/wiki/Pathfinder#state-definitions`
[4] `https://github.com/Modular-Robotics-Group/modular-robotics/wiki/Move-Definitions`

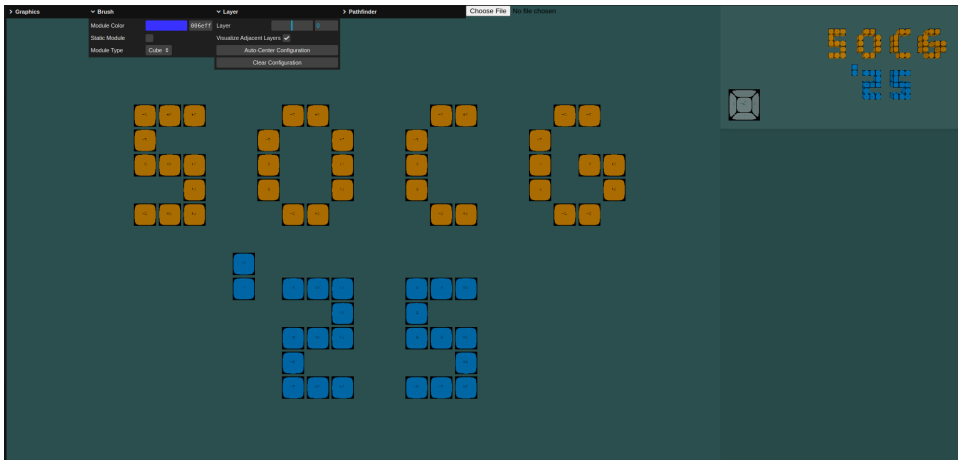


Figure 2 3D Tiler interface.

check whether each non-cut, non-static module has a valid move, each producing a new configuration added to G, which we then recurse into, in an order specified by the selected shortest path algorithm. Once a move generates the desired configuration, a file is produced for use in the visualization tool demonstrating the corresponding reconfiguration sequence.

BFS. New states are added to a queue and processed in a FIFO order. Each new state is kept in a hash table, so states can be checked if they have been previously visited. While a state is being processed, all new states reachable from the current state are generated and added to the queue. When the hash for the current state matches the hash for the final state, the algorithm terminates and returns the path from the start state to the end state.

A* search. A* search algorithm was implemented with various heuristics to improve search times. This search works almost identically to BFS, with the only difference being the use of a priority queue to store adjacent states. States are then processed in an order based on their depth added to a weight assigned to them by a heuristic. The advantages of the A* search are most clear when testing scenarios in 3D space, where the number of possible moves is far greater, and the heuristic tends to avoid irrelevant configurations.

Visualization. "WebVis" is an interactive, browser-based tool to visualize configurations and move sequences, as well as help to conceptualize potential new meta-module configurations. The tool takes a "scenario" file as input, which specifies a reconfiguration sequence and additional data such as module color, size, and shape. There are controls available to step forward and backward through the move sequence, change the animation speed, load in example scenarios, move the camera, and change the camera projection scheme (perspective versus orthographic). Details of input file specification can be found on the project wiki.

3 Results and Discussion

Pivoting to Sliding Meta-module. We analyzed the meta-module shown in Figure 3 made of pivoting square modules which simulate a sliding square module, demonstrating its correctness. The animation of the reconfiguration shown can be found in the repository.

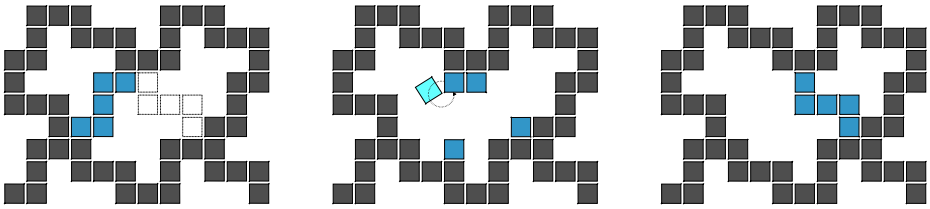


Figure 3 An initial, intermediate, and final configuration in the pivoting square model.

Heuristics. Table 1 shows the differences between search algorithms utilized.

Table 1 Search methods used on the initial and final configurations in Figure 3. All methods found a path of length 14.

Search Method:	BFS	A* (Manhattan Distance)	A* (Symmetric Difference)
Duplicate States Avoided:	148407	40783	58172
Unique States Visited:	24770	13114	14820
Average Time:	1123.67ms	455.33ms	756.67ms

Rhombic Dodecahedron Pivoting Model. In the young researchers' forum at SOCG 2023 Akitaya & Stock [4] showed that Rhombic Dodecahedral Modular Robots admit *super-rigid* configurations (Fig. 4), meaning that there are rigid configurations that remain rigid even with the addition of helper modules. Although the claims in [4] were proven, the proof requires a lot of casework, a technique prone to error. With our tool, we verified the validity of their results.

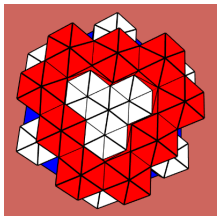


Figure 4 An example of a super rigid configuration from [4].

References

1 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing Universal Reconfigurability of Modular Pivoting Robots. 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 10:1–10:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICS.SOCG.2021.10.

2 Zachary Abel, Hugo A. Akitaya, Scott Duke Kominers, Matias Korman, and Frederick Stock. A universal in-place reconfiguration algorithm for sliding cube-shaped robots in a quadratic number of moves. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 1:1–1:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SoCG.2024.1.

3 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmović, Robin Flatland, Matias Korman, Belen Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: the O(1)-musketeers. *Algorithmica*, 83(5):1316–1351, 2021. doi:10.1007/S00453-020-00784-6.

4 Hugo A. Akitaya and Frederick Stock. Reconfiguration of 3D pivoting modular robots. *arXiv preprint arXiv:2304.09990*, 2023. doi:10.48550/arXiv.2304.09990.

5 Adrian Dumitrescu and János Pach. Pushing Squares Around. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 116–123, 2004. doi:10.1145/997817.997838.

6 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20(3):409–418, 2004. doi:10.1109/TRA.2004.824936.

7 Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2460–2467. IEEE, 2003. doi:10.1109/IROS.2003.1249239.

8 Cynthia Sung, James Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1933–1940. IEEE, 2015. doi:10.1109/ICRA.2015.7139451.