

42nd International Symposium on Theoretical Aspects of Computer Science

STACS 2025, March 4–7, 2025, Jena, Germany

Edited by

Olaf Beyersdorff

Michał Pilipczuk

Elaine Pimentel

Nguyễn Kim Thắng



Editors

Olaf Beyersdorff

Friedrich Schiller University Jena, Germany
olaf.beyersdorff@uni-jena.de

Michał Pilipczuk

University of Warsaw, Poland
michal.pilipczuk@mimuw.edu.pl

Elaine Pimentel

University College London, UK
e.pimentel@ucl.ac.uk

Nguyễn Kim Thăng

Grenoble INP, Université Grenoble-Alpes, France
kim-thang.nguyen@univ-grenoble-alpes.fr

ACM Classification 2012

Mathematics of computing → Combinatorics; Mathematics of computing → Graph theory; Theory of computation → Formal languages and automata theory; Theory of computation → Logic; Theory of computation → Design and analysis of algorithms; Theory of computation → Computational complexity and cryptography; Theory of computation → Models of computation

ISBN 978-3-95977-365-2

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/978-3-95977-365-2>.

Publication date

March, 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.STACS.2025.0

ISBN 978-3-95977-365-2

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

LIPICs – Leibniz International Proceedings in Informatics

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

Editorial Board

- Luca Aceto (Reykjavik University, IS and Gran Sasso Science Institute, IT)
- 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)
- Daniel Král' (Masaryk University, Brno, CZ)
- Meena Mahajan (*Chair*, Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Pierre Senellart (ENS, Université PSL, Paris, FR)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

■ Contents

Preface	
<i>Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng</i>	0:xi–0:xii
Conference Organization	
.....	0:xiii–0:xvii
List of Authors	
.....	0:xix–0:xxv

Invited Talks

Proof Complexity and Its Relations to SAT Solving	
<i>Albert Atserias</i>	1:1–1:1
A Strongly Polynomial Algorithm for Linear Programs with at Most Two Non-Zero Entries per Row or Column	
<i>Daniel Dadush, Zhuang Khye Koh, Bento Natura, Neil Olver, and László A. Végh</i> .	2:1–2:1
Algebras for Automata: Reasoning with Regularity	
<i>Anupam Das</i>	3:1–3:1
Some Recent Advancements in Monotone Circuit Complexity	
<i>Susanna F. de Rezende</i>	4:1–4:2

Regular Papers

Parameterized Saga of First-Fit and Last-Fit Coloring	
<i>Akanksha Agrawal, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Shaily Verma</i>	5:1–5:21
Twin-Width One	
<i>Jungho Ahn, Hugo Jacob, Noleen Köhler, Christophe Paul, Amadeus Reinald, and Sebastian Wiederrecht</i>	6:1–6:19
Faster Edge Coloring by Partition Sieving	
<i>Shyan Akmal and Tomohiro Koana</i>	7:1–7:18
Tropical Proof Systems: Between R(CP) and Resolution	
<i>Yaroslav Alekseev, Dima Grigoriev, and Edward A. Hirsch</i>	8:1–8:20
Improved Approximation Algorithms for (1,2)-TSP and Max-TSP Using Path Covers in the Semi-Streaming Model	
<i>Sharareh Alipour, Ermiya Farokhnejad, and Tobias Mömke</i>	9:1–9:17
Monotone Weak Distributive Laws over the Lifted Powerset Monad in Categories of Algebras	
<i>Quentin Aristote</i>	10:1–10:20
Generalized Inner Product Estimation with Limited Quantum Communication	
<i>Srinivasan Arunachalam and Louis Schatzki</i>	11:1–11:17



Results on H -Freeness Testing in Graphs of Bounded r -Admissibility <i>Christine Awofeso, Patrick Greaves, Oded Lachish, and Felix Reidl</i>	12:1–12:16
Hyperbolic Random Graphs: Clique Number and Degeneracy with Implications for Colouring <i>Samuel Baguley, Yannic Maus, Janosch Ruff, and George Skretas</i>	13:1–13:20
Multivariate Exploration of Metric Dilation <i>Aritra Banik, Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Satyabrata Jana, and Saket Saurabh</i>	14:1–14:17
Structure-Guided Automated Reasoning <i>Max Bannach and Markus Hecher</i>	15:1–15:18
Listing Spanning Trees of Outerplanar Graphs by Pivot-Exchanges <i>Nastaran Behrooznia and Torsten Mütze</i>	16:1–16:18
Tight Approximation and Kernelization Bounds for Vertex-Disjoint Shortest Paths <i>Matthias Bentert, Fedor V. Fomin, and Petr A. Golovach</i>	17:1–17:17
Online Disjoint Set Covers: Randomization Is Not Necessary <i>Marcin Bienkowski, Jaroslaw Byrka, and Łukasz Jeż</i>	18:1–18:16
The Complexity of Learning LTL, CTL and ATL Formulas <i>Benjamin Bordais, Daniel Neider, and Rajarshi Roy</i>	19:1–19:20
On Cascades of Reset Automata <i>Roberto Borelli, Luca Geatti, Marco Montali, and Angelo Montanari</i>	20:1–20:22
Computability of Extender Sets in Multidimensional Subshifts <i>Antonin Callard, Léo Paviet Salomon, and Pascal Vanier</i>	21:1–21:19
CMSO-Transducing Tree-Like Graph Decompositions <i>Rutger Campbell, Bruno Guillon, Mamadou Moustapha Kanté, Eun Jung Kim, and Noleen Köhler</i>	22:1–22:18
How to Play the Accordion: Uniformity and the (Non-)Conservativity of the Linear Approximation of the λ -Calculus <i>Rémy Cerda and Lionel Vaux Auclair</i>	23:1–23:21
A Deterministic Approach to Shortest Path Restoration in Edge Faulty Graphs <i>Keerti Choudhary and Rishabh Dhiman</i>	24:1–24:10
Local Density and Its Distributed Approximation <i>Aleksander Bjørn Christiansen, Ivor van der Hoog, and Eva Rotenberg</i>	25:1–25:20
Toward Better Depth Lower Bounds: Strong Composition of XOR and a Random Function <i>Nikolai Chukhin, Alexander S. Kulikov, and Ivan Mihajlin</i>	26:1–26:15
Local Equivalence of Stabilizer States: A Graphical Characterisation <i>Nathan Claudet and Simon Perdrix</i>	27:1–27:18
Can You Link Up With Treewidth? <i>Radu Curticapean, Simon Döring, Daniel Neuen, and Jiaheng Wang</i>	28:1–28:24

Noisy (Binary) Searching: Simple, Fast and Correct <i>Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański</i>	29:1–29:18
Being Efficient in Time, Space, and Workload: a Self-Stabilizing Unison and Its Consequences <i>Stéphane Devismes, David Ilcinkas, Colette Johnen, and Frédéric Mazoit</i>	30:1–30:18
Efficient Approximation Schemes for Scheduling on a Stochastic Number of Machines <i>Leah Epstein and Asaf Levin</i>	31:1–31:18
A Faster Algorithm for Constrained Correlation Clustering <i>Nick Fischer, Evangelos Kipouridis, Jonas Klausen, and Mikkel Thorup</i>	32:1–32:18
Metric Dimension and Geodetic Set Parameterized by Vertex Cover <i>Florent Foucaud, Esther Galby, Liana Khazaliya, Shaohua Li, Fionn Mc Inerney, Roohani Sharma, and Prafullkumar Tale</i>	33:1–33:20
Agreement Tasks in Fault-Prone Synchronous Networks of Arbitrary Structure <i>Pierre Fraigniaud, Minh Hang Nguyen, and Ami Paz</i>	34:1–34:21
Dimension-Free Parameterized Approximation Schemes for Hybrid Clustering <i>Ameet Gadekar and Tanmay Inamdar</i>	35:1–35:20
MaxMin Separation Problems: FPT Algorithms for <i>st</i> -Separator and Odd Cycle Transversal <i>Ajinkya Gaikwad, Hitendra Kumar, Soumen Maity, Saket Saurabh, and Roohani Sharma</i>	36:1–36:21
On the Existential Theory of the Reals Enriched with Integer Powers of a Computable Number <i>Jorge Gallego-Hernández and Alessio Mansutti</i>	37:1–37:18
Two-Dimensional Longest Common Extension Queries in Compact Space <i>Arnab Ganguly, Daniel Gibney, Rahul Shah, and Sharma V. Thankachan</i>	38:1–38:17
A Quasi-Polynomial Time Algorithm for Multi-Arrival on Tree-Like Multigraphs <i>Ebrahim Ghorbani, Jonah Leander Hoff, and Matthias Mnich</i>	39:1–39:19
Identity-Preserving Lax Extensions and Where to Find Them <i>Sergey Goncharov, Dirk Hofmann, Pedro Nora, Lutz Schröder, and Paul Wild</i>	40:1–40:20
Residue Domination in Bounded-Treewidth Graphs <i>Jakob Greilhuber, Philipp Schepper, and Philip Wellnitz</i>	41:1–41:20
Local Enumeration: The Not-All-Equal Case <i>Mohit Gurumukhani, Ramamohan Paturi, Michael Saks, and Navid Talebanfard</i> .	42:1–42:19
Approximating Densest Subgraph in Geometric Intersection Graphs <i>Sariel Har-Peled and Saladi Rahul</i>	43:1–43:17
Independence and Domination on Bounded-Treewidth Graphs: Integer, Rational, and Irrational Distances <i>Tim A. Hartmann and Dániel Marx</i>	44:1–44:19
Forbidden Patterns in Mixed Linear Layouts <i>Deborah Haun, Laura Merker, and Sergey Pupyrev</i>	45:1–45:21

Sampling Unlabeled Chordal Graphs in Expected Polynomial Time <i>Úrsula Hébert-Johnson and Daniel Lokshantov</i>	46:1–46:20
Minimizing the Number of Tardy Jobs with Uniform Processing Times on Parallel Machines <i>Klaus Heeger and Hendrik Molter</i>	47:1–47:17
Subshifts Defined by Nondeterministic and Alternating Plane-Walking Automata <i>Benjamin Hellouin de Menibus and Pacôme Perrotin</i>	48:1–48:15
Cycle Counting Under Local Differential Privacy for Degeneracy-Bounded Graphs <i>Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya</i>	49:1–49:22
Designing Exploration Contracts <i>Martin Hoefer, Conrad Schecker, and Kevin Schewior</i>	50:1–50:19
Protecting the Connectivity of a Graph Under Non-Uniform Edge Failures <i>Felix Hommelsheim, Zhenwei Liu, Nicole Megow, and Guochuan Zhang</i>	51:1–51:21
Polynomial Kernel and Incompressibility for Prison-Free Edge Deletion and Completion <i>Sébane Bel Houari-Durand, Eduard Eiben, and Magnus Wahlström</i>	52:1–52:17
On Read- k Projections of the Determinant <i>Pavel Hrubeš and Pushkar S. Joglekar</i>	53:1–53:7
Multidimensional Quantum Walks, Recursion, and Quantum Divide & Conquer <i>Stacey Jeffery and Galina Pass</i>	54:1–54:16
Modal Separation of Fixpoint Formulae <i>Jean Christoph Jung and Jędrzej Kołodziejcki</i>	55:1–55:20
Transforming Stacks into Queues: Mixed and Separated Layouts of Graphs <i>Julia Katheder, Michael Kaufmann, Sergey Pupyrev, and Torsten Ueckerdt</i>	56:1–56:18
Approximate Minimum Tree Cover in All Symmetric Monotone Norms Simultaneously <i>Matthias Kaul, Kelin Luo, Matthias Mnich, and Heiko Röglin</i>	57:1–57:18
Violating Constant Degree Hypothesis Requires Breaking Symmetry <i>Piotr Kawalek and Armin Weiß</i>	58:1–58:21
Online Matching with Delays and Size-Based Costs <i>Yasushi Kawase and Tomohiro Nakayoshi</i>	59:1–59:18
Modular Counting CSP: Reductions and Algorithms <i>Amirhossein Kazemini and Andrei A. Bulatov</i>	60:1–60:18
Efficiently Computing the Minimum Rank of a Matrix in a Monoid of Zero-One Matrices <i>Stefan Kiefer and Andrew Ryzhikov</i>	61:1–61:22
Faster Algorithms on Linear Delta-Matroids <i>Tomohiro Koana and Magnus Wahlström</i>	62:1–62:19
Approximation of Spanning Tree Congestion Using Hereditary Bisection <i>Petr Kolman</i>	63:1–63:6

Cluster Editing on Cographs and Related Classes <i>Manuel Lafond, Alitzel López Sánchez, and Weidong Luo</i>	64:1–64:21
On Average Baby PIH and Its Applications <i>Yuwei Liu, Yijia Chen, Shuangli Li, Bingkai Lin, and Xin Zheng</i>	65:1–65:19
The Hardness of Decision Tree Complexity <i>Bruno Loff and Alexey Milovanov</i>	66:1–66:13
Commutative N-Rational Series of Polynomial Growth <i>Aliaume Lopez</i>	67:1–67:16
Slightly Non-Linear Higher-Order Tree Transducers <i>Lê Thành Dũng (Tito) Nguyễn and Gabriele Vanoni</i>	68:1–68:20
A Dichotomy Theorem for Ordinal Ranks in MSO <i>Damian Niwiński, Paweł Parys, and Michał Skrzypczak</i>	69:1–69:18
Colorful Vertex Recoloring of Bipartite Graphs <i>Boaz Patt-Shamir, Adi Rosén, and Seeun William Umboh</i>	70:1–70:19
Unfairly Splitting Separable Necklaces <i>Patrick Schneider, Linus Stalder, and Simon Weber</i>	71:1–71:19
Card-Based Protocols Imply PSM Protocols <i>Kazumasa Shinagawa and Koji Nuida</i>	72:1–72:18
Dominating Set, Independent Set, Discrete k -Center, Dispersion, and Related Problems for Planar Points in Convex Position <i>Anastasiia Tkachenko and Haitao Wang</i>	73:1–73:20
Nearly-Optimal Algorithm for Non-Clairvoyant Service with Delay <i>Noam Touitou</i>	74:1–74:21
Canonical Labeling of Sparse Random Graphs <i>Oleg Verbitsky and Maksim Zhukovskii</i>	75:1–75:20
Dynamic Unit-Disk Range Reporting <i>Haitao Wang and Yiming Zhao</i>	76:1–76:19

■ Preface

The International Symposium on Theoretical Aspects of Computer Science (STACS) conference series is an internationally leading forum for original research on theoretical aspects of computer science.

STACS 2025 consists of two tracks, A and B. Track A is dedicated to algorithms and data structures, complexity and games. Track B covers automata, logic, semantics and theory of programming.

STACS is held alternately in France and in Germany. This year's conference, taking place in Jena (Germany) from March 4 to March 7, is the 42nd in the series. Previous meetings took place in Paris (1984), Saarbrücken (1985), Orsay (1986), Passau (1987), Bordeaux (1988), Paderborn (1989), Rouen (1990), Hamburg (1991), Cachan (1992), Würzburg (1993), Caen (1994), München (1995), Grenoble (1996), Lübeck (1997), Paris (1998), Trier (1999), Lille (2000), Dresden (2001), Antibes (2002), Berlin (2003), Montpellier (2004), Stuttgart (2005), Marseille (2006), Aachen (2007), Bordeaux (2008), Freiburg (2009), Nancy (2010), Dortmund (2011), Paris (2012), Kiel (2013), Lyon (2014), München (2015), Orléans (2016), Hannover (2017), Caen (2018), Berlin (2019), Montpellier (2020), Saarbrücken (2021, taking place virtually), Marseille (2022, taking place virtually), Hamburg (2023) and Clermont-Ferrand (2024).

The STACS 2025 call for papers led to 259 submissions (202 for Track A and 57 for Track B). Each paper was assigned to three program committee members who, at their discretion, asked external reviewers for reports. STACS 2025 employed a lightweight double-blind reviewing process and incorporated an author rebuttal period in the reviewing process.

The committee selected 72 papers for presentation at the conference (55 for Track A and 17 for Track B), implying an acceptance rate of approximately 28%. We are thankful to all individuals, institutions, and organizations who contributed to making STACS 2025 a success. We thank all authors for submitting their work to STACS 2025. Our deepest thanks go to all Program Committee members and external expert reviewers for carefully reading the submissions, providing constructive comments, and for participating in extensive discussions that helped in selecting the strongest papers for the technical program of the conference. The very high quality of the submissions made the selection an extremely difficult task. We also thank the Steering Committee members of STACS for providing overall guidance.

We would like to express our gratitude to the three invited speakers: Daniel Dadush (CWI Amsterdam), Anupam Das (University of Birmingham), and Susanna F. de Rezende (Lund University) and to Albert Atserias (UPC Barcelona) as the tutorial speaker.

STACS 2025 was preceded on 3 and 4 March 2025 by a workshop on algorithms, complexity and logic (Theorietag), a workshop under the auspices of the GI interest groups on Algorithms, Complexity and Logic. Invited talks at the workshop were given by Heribert Vollmer (Hanover) and Sebastian Wild (Marburg). We thank Christian Komusiewicz (University of Jena) for co-organising the pre-conference workshop. As in 2024, STACS 2025 was accompanied by an extended stay support program, allowing participants to combine their conference visit with a research trip to a nearby university.

Full versions of selected outstanding papers from STACS 2025 are invited for submission to the journal *TheoretCS*. Further selected papers from Track A are invited to the *ACM Transactions on Computation Theory* and selected papers from Track B to *Logical Methods in Computer Science*.



We thank the LIPIcs team for assisting us in the publication process and the final production of the proceedings. These proceedings contain extended abstracts of the accepted contributions and abstracts of the tutorial and invited talks. The authors retain their rights and make their work available under a Creative Commons license. The proceedings are published electronically by Schloss Dagstuhl – Leibniz-Center for Informatics within their LIPIcs series.

Finally, we would like to thank Friedrich Schiller University of Jena, the Carl Zeiss Foundation, the Interactive Inference project, Inverso, the University Clermont Auvergne and LIMOS for their support. Our special thanks go to Silvia Blaser, Benjamin Böhm, Marlene Gründel, Tim Hoffmann, Kaspar Kasche, Agnes Schleitzer and Luc Spachmann – the local organising team at the University of Jena – for all their help with the organisation, including the webpages, the registration and the social events.

March 2025

Olaf Beyersdorff
Michał Pilipczuk
Elaine Pimentel
Nguyễn Kim Thắng

■ Conference Organisation

Program Committee – Track A

Antonios Antoniadis	University of Twente
Christian Coester	University of Oxford
Johanne Cohen	CNRS, Université Paris-Saclay
Arnaud de Mesmay	CNRS, Université Gustave Eiffel
Holger Dell	Goethe University Frankfurt and IT University of Copenhagen
Omar Fawzi	Inria, ENS Lyon
Paweł Gawrychowski	University of Wrocław
Carla Groenland	TU Delft
Zhiyi Huang	University of Hong Kong
Bart Jansen	TU Eindhoven
Tuukka Korhonen	University of Copenhagen
Jakub Łącki	Google Research, New York
Hung Le	University of Massachusetts at Amherst
Nguyễn Kim Thăng	Université Grenoble-Alpes, co-chair
Michał Pilipczuk	University of Warsaw, co-chair
Lars Rohwedder	Maastricht University
Rahul Santhanam	University of Oxford
Shay Solomon	Tel-Aviv University
Tatiana Starikovskaya	ENS Paris
Jukka Suomela	Aalto University
Jakub Tarnawski	Microsoft Research
Torsten Ueckerdt	Karlsruhe Institute of Technology
Jan van den Brand	Georgia Tech
Karol Węgrzycki	Saarland University and MPI
Andreas Wiese	Technical University of Munich

Program Committee – Track B

Christoph Berkholz	TU Ilmenau
Olaf Beyersdorff	Friedrich Schiller University Jena, co-chair
Benedikt Bollig	CNRS, ENS Paris-Saclay
Flavien Breuvert	LIPN, Université Sorbonne Paris Nord
Michaël Cadilhac	DePaul University, Chicago, IL
Nofar Carmeli	Inria, LIRMM, Montpellier
Moses Ganardi	MPI-SWS Kaiserslautern
Christoph Haase	University of Oxford
Sandra Kiefer	University of Oxford
Alexander Kurz	Chapman
Dietrich Kuske	TU Ilmenau
Karoliina Lehtinen	CNRS, Aix-Marseille University, LIS
Stefan Mengel	CNRS, CRIL Lens
Cláudia Nalon	University of Brasilia
Elaine Pimentel	University College London, co-chair
Jurriaan Rot	Radboud University, Nijmegen

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Steering Committee

Dietmar Berwanger	LMF, CNRS, Université Paris-Saclay
Marthe Bonamy	LaBRI, CNRS, Université de Bordeaux
Cyril Nicaud	LIGM, Université Paris-Est
Sylvain Schmitz	IRIF, Université de Paris
Luc Segoufin	DI ENS, INRIA, ENS Ulm
Ioan Todinca	LIFO, Université d'Orléans, co-chair
Petra Berenbrink	Hamburg University
Olaf Beyersdorff	Friedrich Schiller University of Jena
Florin Manea	University of Göttingen
Arne Meier	University of Hannover
Heiko Röglin	University of Bonn
Thomas Schwentick	University of Dortmund, co-chair

Local Organising Committee (Friedrich Schiller University Jena)

Olaf Beyersdorff
Silvia Blaser
Benjamin Böhm
Marlene Gründel, chair
Tim Hoffmann
Kaspar Kasche
Agnes Schleitzer
Luc Spachmann

Subreviewers

Abheek Ghosh	Barbara Keller	Eduard Eiben
Adam Karczmarz	Baris Can Esmer	Elena Kirshanova
Adam Polak	Barnaby Martin	Elvira Mayordomo
Aditya Prakash	Bartło Miej Bosek	Emile Anand
Ahmad Biniaz	Bartłomiej Dudek	Emmanouil Vasileios Vlatakis
Akanksha Agrawal	Bartłomiej Dudek	Gkaragkounis
Aleksander Łukasiewicz	Ben Cameron	Eniko Kevi
Aleksandrs Belovs	Ben Lee Volk	Enze Sun
Alessandro Ronca	Benedikt Kolbe	Erik Jan van Leeuwen
Alex Crane	Benjamin Monmege	Erik Paul
Alexander Lindermayr	Benjamin Rossman	Esther Galby
Alexander Rabinovich	Bertrand Simon	Euiwoong Lee
Alexander Skopalik	Blaise Genest	Evangelia Gergatsouli
Alexandra Lassota	Bo Li	Evangelos Kipouridis
Alexandra Wesolek	Bruno Lopes	Faith Ellen
Alexandros Hollender	C. S. Bhargav	Fateme Abbasi
Alexis de Colnet	Ce Jin	Fatiha Bendali
Ali Vakilian	Cédric Bentz	Felix Reidl
Alireza Bagheri	Chenglin Fan	Feng Shi
Alkida Balliu	Chetan Gupta	Florent Foucaud
Aloïs Dufour	Chien-Chung Huang	Florin Manea
Alon Efrat	Chris Köcher	Francesco Dagnino
Ama Koranteng	Christian Ikenmeyer	Franziska Eberle
Amin Shiraz Gilani	Christian Scheideler	Frédéric Magniez
Anastasiia Alokina	Christof Löding	Gabriel Bathie
Anay Mehrotra	Christoph Dürr	Gabriel Istrate
André Nichterlein	Christoph Lenzen	Gaétan Berthe
André Nusser	Christophe Tollu	George Christodoulou
Andreas Björklund	Chung-Shou Liao	George Kenison
Andreas Emil Feldmann	Clemens Thielen	George Manoussakis
Andreas Göbel	Clovis Eberhart	George Mertzios
Andreas Maggiori	Colin Geniet	George Osipov
Andreas Maletti	Corentin Barloy	Giannos Stamoulis
Andrei Krokhin	Corto Mascle	Giorgio Lucarelli
Andrew Ryzhikov	Csaba Toth	Giulia Bernardini
Andris Ambainis	Da Wei Zheng	Graham Leigh
Anish Mukherjee	Damien Busatto-Gaston	Greg Bodwin
Anna Gilbert	Dana Ron	Guillaume Ducoffe
Antonio Casares	Daniel Cordeiro	Guillaume Malod
Argyrios Deligkas	Daniel Neuen	Guillermo Perez
Arindam Khan	Daniel Turetsky	Haitao Wang
Aritra Banik	Davi Silva	Hang Zhou
Arka Ghosh	David Auger	Hans Bodlaender
Arne Meier	David Harris	Harry Vinall-Smeeth
Artem Tsikiridis	David Lidell	Hermann Haeusler
Artur Riazanov	David Mix Barrington	Hermann Wilhelm
Arturo Merino	David R. Wood	Hoai-An Nguyen
Aryan Agarwala	Dedy Septono Catur Putranto	Hsi-Ming Ho
Ashkan Norouzi Fard	Dominik Scheder	Hsin-Hao Su
Athanasios Konstantinidis	Dorit Hochbaum	Hugo Akitaya
Augusto Modanese	Dušan Knop	Ian Pratt-Hartmann
Balagopal Komarath	Édouard Bonnet	Ignaz Rutter

Ioan Todinca	Lior Gishboliner	Michael Poss
Ioannis Psarros	Loes Kruger	Michaela Borzechowski
Isabella Ziccardi	Loïc Dubois	Michal Opler
Jacob Focke	Loukas Georgiadis	Mickael Randour
Jacob Imola	Luc Pellissier	Mikaël Rabie
Jaegun Lee	Luca Pascal Staus	Mikko Koivisto
James Brotherston	Lucas De Meyer	Miriam Münch
Jan Boeckmann	Lucia Draque Penso	Mirza Redzic
Jan Bok	Lukas Plätz	Mohammad Roghani
Janik Huth	Lukasz Kowalik	Morgan Rogers
Jannik Olbrich	Lvzhou Li	Moritz Buchem
Jannis Blauth	Maël Dumas	Moritz Lichter
Jaroslav Byrka	Magnus Berg	Moritz Muehlenthaler
Jędrzej Hodor	Manfred Kufleitner	Naonori Kakimura
Jesper Nederlof	Manoj Gupta	Narek Bojikian
Jianqiang Li	Manolis Vasilakis	Nathan Klein
Jinge Bao	Marc Schroder	Neha Rino
Jingyang Zhao	Marcel Roeloffzen	Nicolas Bonichon
Joachim Gudmundsson	Marcella Anselmo	Nicolas Bousquet
Jochen Koenemann	Marcin Bienkowski	Nicolas Heurtel
Joe Sawada	Marcin Pilipczuk	Niels Kornerup
Joel Rybicki	Marck van der Vegt	Niels van der Weide
Johannes Lengler	Marco Bressan	Nikhil Balaji
Jonas Ellert	Marek Sokółowski	Nikhil Bansal
Jonas Sénizergues	Marin Bougeret	Nikhil Mande
Joseph Landsberg	Mark Bun	Niko Hastrich
Joshua Brakensiek	Markus Anders	Nima Anari
Julien Baste	Markus Hecher	Ninad Rajgopal
Kaave Hosseini	Markus Lohrey	Nir Piterman
Karine Altisen	Martin Böhm	Noam Touitou
Karol Pokorski	Martin Koutecký	Nobutaka Shimizu
Karthik C. S.	Martin Lange	Norbert Zeh
Kazuyuki Amano	Martin S. Krejca	Nutan Limaye
Kevin Schewior	Masayuki Miyamoto	Olivier Idir
Khaled Elbassioni	Massimo Equi	Omrit Filtser
Kiril Bangachev	Massimo Lauria	Pål Grønås Drange
Kirill Simonov	Mateusz Skomra	Pamela Fleischmann
Konstantinos Tsakalidis	Mateusz Wasykiewicz	Pan Peng
Krzysztof Fleszar	Mathieu Mari	Panagiotis Charalampopoulos
Kunihiro Wasa	Matias Korman	Panos Giannopoulos
Kyungjin Cho	Matt Kovacs-Deak	Pasin Manurangsi
Lars Jaffke	Matthew Gray	Paul Duetting
Lasse Wulf	Matthew Kwan	Paulin Jacobé De Naurois
Laurent Bienvenu	Matthias Bentert	Paweł Rzażewski
Laurent Feuilloley	Matthias Kaul	Pei Wu
Laurent Gourves	Matthias Mnich	Peter Kling
Lazar Milenkovic	Max Deppert	Peter Manohar
Lê Thành Dũng Nguyen	Max Dupré La Tour	Petr Golovach
Leo Wennmann	Maximilian Merz	Peyman Afshani
León Bohn	Maximilian Weininger	Philipp Hieronymi
Leonid Barenboim	Meike Neuwohner	Philipp Schepper
Leqi Zhu	Michael Bekos	Pierre Bergé
Leroy Chew	Michael Blondin	Pierre Coucheney
Liam Roditty	Michael Kompatscher	Pierre Ohlmann
Lina Vandré	Michael Lampis	Prafullkumar Tale


Prahlad Narasimhan Kasthurirangan	Sándor Kisfaludi-Bak	Tianyi Zhang
Prajakta Nimbhorkar	Sarah Maria Morell	Tim Oosterwijk
Pramod Ganapathi	Sariel Har-Peled	Tim Randolph
Pranjal Dutta	Sathiya Venkatesan Ramesh	Timothy Gomez
Prantar Ghosh	Sathwik Karnik	Tobias Winkler
Prashanth Amireddy	Sebastian Berndt	Toghrul Karimov
Qi Ye	Sebastian Haslebacher	Tom van der Zanden
Qipeng Liu	Sebastian Meyer	Tomohiro Koana
Quentin Bramas	Sebastian Pfau	Toshiki Saitoh
R. Govind	Sebastian Schubert	Tristan Kraft
R.B. Sandeep	Sebastian Siebertz	Ullrich Hustadt
Radosław Piórkowski	Sebastian Zur	Vaishali Surianarayanan
Radu Curticapean	Seeun William Umboh	Vida Dujmović
Rahul Vaze	Sergey Kitaev	Viet Cuong Than
Rainer Gemulla	Sergey Pupyrev	Vignesh Viswanathan
Rajendra Kumar	Sergio Cabello	Viktoriia Korchemna
Rajesh Chitnis	Sergio Rajsbaum	Vincent Chau
Ramanujan M. Sridharan	Shaofeng H.-C. Jiang	Vincent Cohen-Addad
Raul Lopes	Shaull Almagor	Vishwas Bhargava
Reiko Heckel	Shibashis Guha	Vladislav Ryzhikov
Reilly Browne	Shinwoo An	Warut Suksompong
Rémy Belmonte	Shreyas Srinivas	Wei Zhan
Renzo Gomez	Simon D. Fink	Weitian Tong
Reuben Rowe	Simon Döring	Wenjie Fang
Reut Levi	Simon Weber	Will Perkins
Rhea Jain	Sorrachai Yingchareonthawornchai	William Kretschmer
Riccardo Michielan	Stanislav Živný	Wojciech Janczewski
Richard Mayr	Stefan Kiefer	Wojciech Nadara
Rini Wisnu Wardhani	Stefan Weltge	Wolfgang Mulzer
Rob van Stee	Susanna F. de Rezende	Xiao Peng
Robert Mercas	Susanne Albers	Xiaojun Dong
Robin Vacus	Suthee Ruangwises	Ya-Chun Liang
Rodrigo Raya	Sven Jäger	Yanlin Chen
Rohit Gurjar	Sylvain Schmitz	Yann Strozecki
Ronnie Pavlov	Taehoon Ahn	Yasamin Nazari
Roohani Sharma	Taha El Ghazi El Houssaini	Yassine Hamoudi
Ruben F.A. Verhaegh	Talya Eden	Yinzhan Xu
Ruben Hoeksma	Tamio-Vesa Nakajima	Yogesh Dahiya
Sablik Mathieu	Tanmay Inamdar	Yongjie Yang
Saket Saurabh	Thekla Hamm	Yota Otachi
Saladi Rahul	Themistoklis Melissourgos	Youssef Oualhadj
Samah Ghazawi	Thi Quynh Trang Vo	Yu Chen
Samarth Tiwari	Thomas Colcombet	Yun Kuen Cheung
Sami Davies	Thomas Erlebach	Yunchao Liu
Sampson Wong	Thomas Lavastida	Yurong Chen
Sander Gribling	Thomas Seiller	Yusuke Kobayashi
Sándor Fekete	Thorben Tröbst	Zixuan Zhu


■ List of Authors

- Akanksha Agrawal  (5)
Indian Institute of Technology Madras, India
- Jungho Ahn  (6)
Korea Institute for Advanced Study (KIAS),
Seoul, South Korea
- Shyan Akmal  (7)
INSAIT, Sofia University "St. Kliment
Ohridski", Bulgaria
- Yaroslav Alekseev  (8)
Technion - Israel Institute of Technology, Haifa,
Israel
- Sharareh Alipour  (9)
Department of Computer Science, Tehran
Institute for Advanced Studies (TeIAS), Khatam
University, Tehran, Iran
- Quentin Aristote  (10)
Université Paris Cité, CNRS, Inria, IRIF,
F-75013, Paris, France
- Srinivasan Arunachalam  (11)
IBM Quantum, Almaden, CA, USA
- Albert Atserias  (1)
Universitat Politècnica de Catalunya, Barcelona,
Spain; Centre de Recerca Matemàtica,
Bellaterra, Spain
- Christine Awofeso  (12)
Birkbeck, University of London, UK
- Samuel Baguley  (13)
Hasso Plattner Institute, University of Potsdam,
Germany
- Aritra Banik  (14)
National Institute of Science, Education and
Research, An OCC of Homi Bhabha National
Institute, Bhubaneswar, India
- Max Bannach  (15)
European Space Agency, Advanced Concepts
Team, Noordwijk, The Netherlands
- Nastaran Behrooznia (16)
Department of Computer Science, University of
Warwick, Coventry, UK
- Matthias Bentert (17)
University of Bergen, Norway
- Marcin Bienkowski  (18)
University of Wrocław, Poland
- Benjamin Bordaïs  (19)
TU Dortmund University, Center for
Trustworthy Data Science and Security,
University Alliance Ruhr, Dortmund, Germany
- Roberto Borelli  (20)
University of Udine, Italy
- Andrei A. Bulatov (60)
Simon Fraser University, Burnaby, Canada
- Jarosław Byrka  (18)
University of Wrocław, Poland
- Antonin Callard  (21)
Normandie Univ, UNICAEN, ENSICAEN,
CNRS, GREYC, 14000, Caen, France
- Rutger Campbell (22)
Discrete Mathematics Group, Institute for Basic
Science, Daejeon, South Korea
- Rémy Cerda  (23)
Aix-Marseille Université, CNRS, I2M, France;
Université Paris Cité, CNRS, IRIF, F-75013,
Paris, France
- Yijia Chen  (65)
Shanghai Jiao Tong University, China
- Keerti Choudhary  (24)
Department of Computer Science and
Engineering, IIT Delhi, India
- Aleksander Bjørn Christiansen  (25)
Technical University of Denmark, Lyngby,
Denmark
- Nikolai Chukhin (26)
Neapolis University Pafos, Cyprus; JetBrains
Research, Paphos, Cyprus
- Nathan Claudet  (27)
Inria Mocqua, LORIA, CNRS, Université de
Lorraine, F-54000 Nancy, France
- Radu Curticapean  (28)
University of Regensburg, Germany; IT
University of Copenhagen, Denmark
- Daniel Dadush  (2)
Centrum Wiskunde & Informatica, Amsterdam,
The Netherlands
- Anupam Das  (3)
University of Birmingham, UK





Susanna F. de Rezende  (4)
Lund University, Sweden


Dariusz Dereniowski  (29)
Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Poland

Stéphane Devismes  (30)
Laboratoire MIS, Université de Picardie, 33 rue Saint Leu - 80039 Amiens cedex 1, France

Rishabh Dhiman (24)
Department of Computer Science and Engineering, IIT Delhi, India

Simon Döring  (28)
Max Planck Institute for Informatics, Saarbrücken, Germany; Saarland University (SIC), Saarbrücken, Germany


Eduard Eiben  (52)
Department of Computer Science, Royal Holloway University of London, UK


Leah Epstein  (31)
Department of Mathematics, University of Haifa, Israel


Ermiya Farokhnejad  (9)
Department of Computer Science, University of Warwick, Coventry, UK


Nick Fischer  (32)
INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria


Fedor V. Fomin  (14, 17)
University of Bergen, Norway


Florent Foucaud  (33)
Université Clermont Auvergne, CNRS, Mines Saint-Étienne, Clermont Auvergne INP, LIMOS, 63000 Clermont-Ferrand, France

Pierre Fraigniaud  (34)
Institut de Recherche en Informatique Fondamentale (IRIF), CNRS, Université Paris Cité, France

Ameet Gadekar  (35)
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany


Ajinkya Gaikwad  (36)
Indian Institute of Science Education and Research, Pune, India


Esther Galby  (33)
Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Sweden

Jorge Gallego-Hernández  (37)
IMDEA Software Institute, Madrid, Spain; Universidad Politécnica de Madrid, Spain


Arnab Ganguly  (38)
University of Wisconsin, Whitewater, WI, USA

Luca Geatti  (20)
University of Udine, Italy


Ebrahim Ghorbani  (39)
Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Daniel Gibney  (38)
University of Texas at Dallas, TX, USA

Petr A. Golovach  (14, 17)
University of Bergen, Norway


Sergey Goncharov  (40)
University of Birmingham, UK


Patrick Greaves  (12)
Birkbeck, University of London, UK


Jakob Greilhuber  (41)
TU Wien, Austria; CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Dima Grigoriev (8)
CNRS, Mathématique, Université de Lille, Villeneuve d'Ascq, 59655, France

Bruno Guillon (22)
Université Clermont Auvergne, Clermont Auvergne INP, LIMOS, CNRS, Clermont-Ferrand, France

Mohit Gurumukhani  (42)
Cornell University, Ithaca, NY, USA

Sariel Har-Peled  (43)
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Tim A. Hartmann  (44)
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany


Deborah Haun  (45)
Karlsruhe Institute of Technology, Germany

- Markus Hecher  (15)
Univ. Artois, CNRS UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), 6230, France; Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA, USA
- Klaus Heeger  (47)
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel
- Benjamin Hellouin de Menibus  (48)
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France
- Quentin Hillebrand  (49)
The University of Tokyo, Japan
- Edward A. Hirsch  (8)
Department of Computer Science, Ariel University, Israel
- Martin Hoefler  (50)
Department of Computer Science, RWTH Aachen University, Germany
- Dirk Hofmann  (40)
CIDMA, University of Aveiro, Portugal
- Felix Hommelsheim  (51)
University of Bremen, Germany
- Séhane Bel Houari-Durand  (52)
ENS Lyon, France
- Pavel Hrubeš  (53)
Institute of Mathematics of ASCR, Czech Republic
- Úrsula Hébert-Johnson  (46)
University of California, Santa Barbara, CA, USA
- David Ilcinkas  (30)
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France
- Tanmay Inamdar  (14, 35)
Indian Institute of Technology Jodhpur, India
- Hugo Jacob  (6)
LIRMM, Université de Montpellier, CNRS, Montpellier, France
- Satyabrata Jana  (14)
University of Warwick, UK
- Stacey Jeffery  (54)
QuSoft, CWI, Amsterdam, The Netherlands; University of Amsterdam, The Netherlands
- Łukasz Jeż  (18)
University of Wrocław, Poland
- Pushkar S. Joglekar  (53)
Vishwakarma Institute of Technology, Pune, India
- Colette Johnen  (30)
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France
- Jean Christoph Jung  (55)
TU Dortmund University, Germany
- Mamadou Moustapha Kanté  (22)
Université Clermont Auvergne, Clermont Auvergne INP, LIMOS, CNRS, Clermont-Ferrand, France
- Julia Katheder  (56)
Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
- Michael Kaufmann  (56)
Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
- Matthias Kaul  (57)
Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany; University of Bonn, Germany
- Yasushi Kawase  (59)
The University of Tokyo, Japan
- Piotr Kawalek  (58)
TU Wien, Austria; Jagiellonian University in Kraków, Poland
- Amirhossein Kazemini  (60)
Simon Fraser University, Burnaby, Canada
- Liana Khazaliya  (33)
Technische Universität Wien, Austria
- Stefan Kiefer  (61)
Department of Computer Science, University of Oxford, UK
- Eun Jung Kim  (22)
KAIST, Daejeon, South Korea; CNRS, France
- Evangelos Kipouridis  (32)
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
- Jonas Klausen  (32)
BARC, University of Copenhagen, Denmark

- Tomohiro Koana  (7, 62)
Utrecht University, The Netherlands; Research
Institute for Mathematical Sciences, Kyoto
University, Japan
- Zhuan Khye Koh (2)
Centrum Wiskunde & Informatica, Amsterdam,
The Netherlands
- Petr Kolman  (63)
Department of Applied Mathematics, Faculty of
Mathematics and Physics, Charles University,
Prague, Czech Republic
- Jerdrzej Kołodziejcki  (55)
TU Dortmund University, Germany
- Alexander S. Kulikov  (26)
JetBrains Research, Paphos, Cyprus
- Hitendra Kumar (36)
Indian Institute of Science Education and
Research, Pune, India
- Noleen Köhler  (6, 22)
University of Leeds, UK
- Oded Lachish  (12)
Birkbeck, University of London, UK
- Manuel Lafond  (64)
Department of Computer Science, Université de
Sherbrooke, Canada
- Jonah Leander Hoff (39)
Hamburg University of Technology, Institute for
Algorithms and Complexity, Hamburg, Germany
- Asaf Levin  (31)
Faculty of Data and Decisions Science, The
Technion, Haifa, Israel
- Shaohua Li  (33)
School of Computer Science and Engineering,
Central South University, Changsha, China
- Shuangle Li  (65)
Nanjing University, China
- Bingkai Lin  (65)
Nanjing University, China
- Yuwei Liu  (65)
Shanghai Jiao Tong University, China
- Zhenwei Liu  (51)
Zhejiang University, Hangzhou, China;
University of Bremen, Germany
- Bruno Loff  (66)
LASIGE, Faculdade de Ciências, Universidade
de Lisboa, Portugal
- Daniel Lokshtanov  (5, 46)
University of California, Santa Barbara, CA,
USA
- Aliaume Lopez  (67)
University of Warsaw, Poland
- Kelin Luo  (57)
University of Bonn, Germany; University at
Buffalo, NY, USA
- Weidong Luo  (64)
Department of Computer Science, Université de
Sherbrooke, Canada
- Alitzel López Sánchez  (64)
Department of Computer Science, Université de
Sherbrooke, Canada
- Soumen Maity (36)
Indian Institute of Science Education and
Research, Pune, India
- Alessio Mansutti  (37)
IMDEA Software Institute, Madrid, Spain
- Dániel Marx  (44)
CISPA Helmholtz Center for Information
Security, Saarbrücken, Germany
- Yannic Maus  (13)
TU Graz, Austria
- Frédéric Mazoit  (30)
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI,
UMR 5800, F-33400 Talence, France
- Fionn Mc Inerney  (33)
Telefónica Scientific Research, Barcelona, Spain
- Nicole Megow  (51)
University of Bremen, Germany
- Laura Merker  (45)
Karlsruhe Institute of Technology, Germany
- Ivan Mihajlin (26)
JetBrains Research, Paphos, Cyprus
- Alexey Milovanov  (66)
LASIGE, Faculdade de Ciências, Universidade
de Lisboa, Portugal
- Matthias Mnich  (39, 57)
Hamburg University of Technology, Institute for
Algorithms and Complexity, Hamburg, Germany


- Hendrik Molter  (47)
Department of Computer Science, Ben-Gurion
University of the Negev, Beer-Sheva, Israel
- Marco Montali  (20)
Free University of Bozen-Bolzano, Italy
- Angelo Montanari  (20)
University of Udine, Italy
- Tobias Mömke  (9)
Department of Computer Science, University of
Augsburg, Germany
- Torsten Mütze  (16)
Institut für Mathematik, Universität Kassel,
Germany; Department of Theoretical Computer
Science and Mathematical Logic, Charles
University, Prague, Czech Republic
- Tomohiro Nakayoshi  (59)
The University of Tokyo, Japan
- Bento Natura (2)
Columbia University, New York, NY, USA
- Daniel Neider  (19)
TU Dortmund University, Center for
Trustworthy Data Science and Security,
University Alliance Ruhr, Dortmund, Germany
- Daniel Neuen  (28)
University of Regensburg, Germany; Max Planck
Institute for Informatics, Saarbrücken, Germany
- Minh Hang Nguyen  (34)
Institut de Recherche en Informatique
Fondamentale (IRIF), CNRS, Université Paris
Cité, France
- Lê Thành Dũng (Tito) Nguyễn  (68)
CNRS & Aix-Marseille University, France
- Damian Niwiński  (69)
Institute of Informatics, University of Warsaw,
Poland
- Pedro Nora  (40)
Radboud Universiteit, Nijmegen, The
Netherlands
- Koji Nuida  (72)
Institute of Mathematics for Industry (IMI),
Kyushu University, Fukuoka, Japan; National
Institute of Advanced Industrial Science and
Technology (AIST), Tokyo, Japan
- Neil Olver (2)
London School of Economics and Political, UK
- Fahad Panolan  (5)
School of Computer Science, University of Leeds,
UK
- Paweł Parys  (69)
Institute of Informatics, University of Warsaw,
Poland
- Galina Pass (54)
QuSoft, Amsterdam, The Netherlands;
University of Amsterdam, The Netherlands
- Boaz Patt-Shamir  (70)
School of Electrical Engineering, Tel Aviv
University, Israel
- Ramamohan Paturi (42)
Department of Computer Science and
Engineering, University of California San Diego,
La Jolla, CA, USA
- Christophe Paul  (6)
LIRMM, Université de Montpellier, CNRS,
Montpellier, France
- Léo Paviet Salomon  (21)
Université de Lorraine, CNRS, Inria, LORIA,
54000, Nancy, France
- Ami Paz  (34)
Laboratoire Interdisciplinaire des Sciences du
Numérique (LISN), CNRS, Université
Paris-Saclay, France
- Simon Perdrix  (27)
Inria Mocqua, LORIA, CNRS, Université de
Lorraine, F-54000 Nancy, France
- Pacôme Perrotin  (48)
Université Paris-Saclay, CNRS, Laboratoire
Interdisciplinaire des Sciences du Numérique,
91400, Orsay, France
- Sergey Pupyrev  (45, 56)
Menlo Park, CA, USA
- Saladi Rahul  (43)
Indian Institute of Science (IISc), Bangalore,
India
- Felix Reidl  (12)
Birkbeck, University of London, UK
- Amadeus Reinald  (6)
LIRMM, Université de Montpellier, CNRS,
Montpellier, France
- Adi Rosén (70)
CNRS and Université Paris Cité, France


- Eva Rotenberg  (25)
Technical University of Denmark, Lyngby,
Denmark
- Rajarshi Roy  (19)
Department of Computer Science, University of
Oxford, UK
- Janosch Ruff  (13)
Hasso Plattner Institute, University of Potsdam,
Germany
- Andrew Ryzhikov  (61)
Department of Computer Science, University of
Oxford, UK
- Heiko Röglin  (57)
Universität Bonn, Germany
- Michael Saks (42)
Department of Mathematics, Rutgers University,
Piscataway, NJ, USA
- Saket Saurabh  (5, 14, 36)
The Institute of Mathematical Sciences, HBNI,
Chennai, India; Department of Informatics,
University of Bergen, Norway
- Louis Schatzki  (11)
Electrical and Computer Engineering, University
of Illinois Urbana-Champaign, IL, USA
- Conrad Schecker  (50)
Institute for Computer Science, Goethe
University Frankfurt, Germany
- Philipp Schepper  (41)
CISPA Helmholtz Center for Information
Security, Saarbrücken, Germany
- Kevin Schewior  (50)
Department of Mathematics and Computer
Science, University of Cologne, Germany;
University of Southern Denmark, Odense,
Denmark
- Patrick Schnider  (71)
Department of Computer Science, ETH Zürich,
Switzerland
- Lutz Schröder  (40)
Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany
- Rahul Shah  (38)
Louisiana State University, Baton Rouge, LA,
USA
- Roohani Sharma  (33, 36)
University of Bergen, Norway
- Tetsuo Shibuya  (49)
The University of Tokyo, Japan
- Kazumasa Shinagawa  (72)
Ibaraki University, Ibaraki, Japan; National
Institute of Advanced Industrial Science and
Technology (AIST), Tokyo, Japan
- George Skretas  (13)
Hasso Plattner Institute, University of Potsdam,
Germany
- Michał Skrzypczak  (69)
Institute of Informatics, University of Warsaw,
Poland
- Linus Stalder (71)
Department of Computer Science, ETH Zürich,
Switzerland
- Vorapong Suppakitpaisarn  (49)
The University of Tokyo, Japan
- Prafullkumar Tale  (33)
Indian Institute of Science Education and
Research Pune, India
- Navid Talebanfard  (42)
University of Sheffield, UK
- Sharma V. Thankachan  (38)
North Carolina State University, Raleigh, NC,
USA
- Mikkel Thorup  (32)
BARC, University of Copenhagen, Denmark
- Anastasiia Tkachenko  (73)
Kahlert School of Computing, University of
Utah, Salt Lake City, UT, USA
- Noam Touitou  (74)
Unaffiliated, Tel Aviv, Israel
- Torsten Ueckerdt  (56)
Institute of Theoretical Informatics, Karlsruhe
Institute of Technology, Germany
- Seun William Umboh  (70)
School of Computing and Information Systems,
The University of Melbourne, Australia; ARC
Training Centre in Optimisation Technologies,
Integrated Methodologies, and Applications
(OPTIMA), Melbourne, Australia
- Przemysław Uznański  (29)
Institute of Computer Science, University of
Wrocław, Poland
- Ivor van der Hoog  (25)
Technical University of Denmark, Lyngby,
Denmark

Pascal Vanier  (21)
Normandie Univ, UNICAEN, ENSICAEN,
CNRS, GREYC, 14000, Caen, France

Gabriele Vanoni  (68)
IRIF, Université Paris Cité, France


Lionel Vaux Auclair  (23)
Aix-Marseille Université, CNRS, I2M, France

Oleg Verbitsky  (75)
Institut für Informatik, Humboldt-Universität zu
Berlin, Germany


Shaily Verma  (5)
Algorithm Engineering Group, Hasso Plattner
Institute, Potsdam, Germany

László A. Végh (2)
University of Bonn, Germany


Magnus Wahlström  (52, 62)
Department of Computer Science, Royal
Holloway University of London, UK

Haitao Wang  (73, 76)
Kahlert School of Computing, University of
Utah, Salt Lake City, UT, USA


Jiaheng Wang  (28)
University of Regensburg, Germany


Simon Weber  (71)
Department of Computer Science, ETH Zürich,
Switzerland


Armin Weiß  (58)
University of Stuttgart, Germany

Philip Wellnitz  (41)
National Institute of Informatics, Tokyo, Japan;
The Graduate University for Advanced Studies,
SOKENDAI, Tokyo, Japan

Sebastian Wiederrecht  (6)
School of Computing, KAIST, Daejeon, South
Korea


Paul Wild  (40)
Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany

Guochuan Zhang  (51)
Zhejiang University, Hangzhou, China

Yiming Zhao  (76)
Department of Computer Sciences, Metropolitan
State University of Denver, CO, USA

Xin Zheng  (65)
Nanjing University, China

Maksim Zhukovskii  (75)
School of Computer Science, University of
Sheffield, UK

Aleksander Łukasiewicz  (29)
Institute of Computer Science, University of
Wrocław, Poland; Computer Science Institute of
Charles University, Prague, Czech Republic

Proof Complexity and Its Relations to SAT Solving

Albert Atserias 

Universitat Politècnica de Catalunya, Barcelona, Spain
Centre de Recerca Matemàtica, Bellaterra, Spain

Abstract

Propositional proof complexity is the study of algorithms that recognize the set of tautologies in propositional logic. Initially conceived as an approach to address the “P versus NP” problem in computational complexity, the field has gradually expanded its focus to include new objectives. Among these is the goal of providing a theoretical foundation for comparing the effectiveness of heuristics for algorithms that exhaustively explore the solution spaces of combinatorial problems. Dually, and complementarily, the methods of proof complexity can also be used to assess how to certify that a given exploration path of such an algorithm ultimately leads to a dead end. A notable challenge faced by this methodology lies in the fact that, despite the theoretically proved modelling power of propositional logic, as established by the theory of NP-completeness, propositional logic is not always the best specification language for all application domains. Addressing this challenge involves studying the expressive power of various languages and their associated proof systems through the lens of computational complexity.

The first part of this talk will be a survey of how the emergence of these new objectives for propositional proof complexity came to be, and what the theory’s methods offer in pursuing them. The second part will review the current state of the art on the computational complexity of automating the proof search problem for various proof systems for propositional logic and other languages. While it is now known and well understood that fully automating propositional Resolution as a proof system for propositional logic is NP-hard, it remains an open question whether it is possible to distinguish satisfiable formulas from unsatisfiable ones with short Resolution proofs of unsatisfiability in polynomial time. As of the time of writing, there is no consensus among experts on whether this problem should be considered computationally intractable.

2012 ACM Subject Classification Theory of computation → Proof complexity; Theory of computation → Automated reasoning

Keywords and phrases Propositional logic, proof systems, Resolution, cutting planes, integer linear programming, automatability, NP-hardness, satisfiability, heuristics, search

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.1

Category Invited Talk



© Albert Atserias;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 1; pp. 1:1–1:1



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



A Strongly Polynomial Algorithm for Linear Programs with at Most Two Non-Zero Entries per Row or Column

Daniel Dadush ✉ 

Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Zhuan Khye Koh ✉

Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Bento Natura ✉

Columbia University, New York, NY, USA

Neil Olver ✉

London School of Economics and Political, UK

László A. Végh ✉

University of Bonn, Germany

Abstract

We give a strongly polynomial algorithm for minimum cost generalized flow, and hence for optimizing any linear program with at most two non-zero entries per row, or at most two non-zero entries per column. Primal and dual feasibility were shown by Végh (MOR '17) and Megiddo (SICOMP '83) respectively. Our result can be viewed as progress towards understanding whether all linear programs can be solved in strongly polynomial time, also referred to as Smale's 9th problem.

Our approach is based on the recent primal-dual interior point method (IPM) due to Allamigeon, Dadush, Loho, Natura and Végh (FOCS '22). The number of iterations needed by the IPM is bounded, up to a polynomial factor in the number of inequalities, by the *straight line complexity* of the central path. Roughly speaking, this is the minimum number of pieces of any piecewise linear curve that multiplicatively approximates the central path.

As our main contribution, we show that the straight line complexity of any minimum cost generalized flow instance is polynomial in the number of arcs and vertices. By applying a reduction of Hochbaum (ORL '04), the same bound applies to any linear program with at most two non-zeros per column or per row.

To be able to run the IPM, one requires a suitable initial point. For this purpose, we develop a novel multistage approach, where each stage can be solved in strongly polynomial time given the result of the previous stage. Beyond this, substantial work is needed to ensure that the bit complexity of each iterate remains bounded during the execution of the algorithm. For this purpose, we show that one can maintain a representation of the iterates as a low complexity convex combination of vertices and extreme rays. Our approach is black-box and can be applied to any log-barrier path following method.

2012 ACM Subject Classification Theory of computation → Linear programming

Keywords and phrases Linear Programming, Strongly Polynomial Algorithms, Interior Point Methods

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.2

Category Invited Talk



© Daniel Dadush, Zhuan Khye Koh, Bento Natura, Neil Olver, and László A. Végh; licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 2; pp. 2:1–2:1



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Algebras for Automata: Reasoning with Regularity

Anupam Das   

University of Birmingham, UK

Abstract

In the second half of the 20th century various theories of regular expressions were proposed, eventually leading to the notion of a *Kleene Algebra* (KA). Kozen and Krohn independently proved the completeness of KA for the model of regular languages, a now celebrated result that has been refined and generalised over the years. In recent years proof theoretic approaches to regular languages have been studied, providing alternative routes to metalogical results like completeness and decidability.

In this talk I will present a new approach from a different starting point: finite state automata. A notation for non-deterministic finite automata is readily obtained via expressions with least fixed points, leading to a theory of *right-linear algebras* (RLA). RLA is strictly more general than KA, e.g. admitting ω -regular languages as a model too, and enjoys a simpler proof theory than KA. This allows us to recover (more general) metalogical results in a robust way, combining techniques from automata, games, and cyclic proofs. Finally I will discuss extensions of RLA by *greatest* fixed points, comprising a notation for parity automata, to reason about ω -regular languages too.

This talk is based on joint works with Abhishek De [2, 1].

2012 ACM Subject Classification Theory of computation \rightarrow Regular languages; Theory of computation \rightarrow Proof theory

Keywords and phrases Regular languages, Linear grammars, Proof theory, Cyclic proofs, Automata theory, Fixed points, Games

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.3

Category Invited Talk

Funding *Anupam Das*: The author is supported by a UKRI Future Leader's Fellowship, *Structure vs Invariants in Proofs*, project reference MR/Y011716/1.

References

- 1 Anupam Das and Abhishek De. A proof theory of (ω -)context-free languages, via non-wellfounded proofs. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*, volume 14740 of *Lecture Notes in Computer Science*, pages 237–256. Springer, 2024. doi:10.1007/978-3-031-63501-4_13.
- 2 Anupam Das and Abhishek De. A proof theory of right-linear (ω -)grammars via cyclic proofs. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 30:1–30:14. ACM, 2024. doi:10.1145/3661814.3662138.



© Anupam Das;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 3; pp. 3:1–3:1



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Some Recent Advancements in Monotone Circuit Complexity

Susanna F. de Rezende   

Lund University, Sweden

Abstract

In 1985, Razborov [16] proved the first superpolynomial size lower bound for monotone Boolean circuits for the perfect matching the clique functions, and, independently, Andreev [2] obtained exponential size lower bounds. These breakthroughs were soon followed by further advancements in monotone complexity, including better lower bounds for clique [1, 19], superlogarithmic depth lower bounds for connectivity by Karchmer and Wigderson [12], and the separations $\text{mon-NC} \neq \text{mon-P}$ and that $\text{mon-NC}^i \neq \text{mon-NC}^{i+1}$ by Raz and McKenzie [15]. Karchmer and Wigderson [12] proved their result by establishing a relation between communication complexity and (monotone) circuit depth, and Raz and McKenzie [15] introduced a new technique, now called *lifting theorems*, for obtaining communication lower bounds from query complexity lower bounds,

In this talk, we will survey recent advancements in monotone complexity driven by query-to-communication lifting theorems. A decade ago, Göös, Pitassi, and Watson [10] brought to light the generality of the result of Raz and McKenzie [15] and reignited this line of work. A notable extension is the lifting theorem [8] for a model of DAG-like communication [17, 18] that corresponds to circuit size. These powerful theorems, in their different flavours, have been instrumental in addressing many open questions in monotone circuit complexity, including: optimal $2^{\Omega(n)}$ lower bounds on the size of monotone Boolean formulas computing an explicit function in NP [14]; a complete picture of the relation between the mon-AC and mon-NC hierarchies [6]; a near optimal separation between monotone circuit and monotone formula size [5]; exponential separation between NC^2 and mon-P [8, 11]; and better lower bounds for clique [7, 13], improving on [3]. Very recently, lifting theorems were also used to prove supercritical trade-offs for monotone circuits showing that there are functions computable by small circuits for which any small circuit must have superlinear or even superpolynomial depth [4, 9]. We will explore these results and their implications, and conclude by discussing some open problems.

2012 ACM Subject Classification Theory of computation \rightarrow Circuit complexity; Theory of computation \rightarrow Communication complexity; Theory of computation \rightarrow Proof complexity

Keywords and phrases monotone circuit complexity, query complexity, lifting theorems


Digital Object Identifier 10.4230/LIPIcs.STACS.2025.4

Category Invited Talk

Funding ELLIIT, Knut and Alice Wallenberg grants KAW 2021.0307 and 2023.0116, and the Swedish Research Council grant 2021-05104.

References

- 1 Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, March 1987. doi:10.1007/bf02579196.
- 2 Alexander E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Mathematics Doklady*, 31(3):530–534, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.
- 3 Bruno Pasqualotto Cavalari, Mrinal Kumar, and Benjamin Rossman. Monotone circuit lower bounds from robust sunflowers. In *Proceedings of the 14th Latin American Symposium on Theoretical Informatics (LATIN '20)*, volume 12118 of *Lecture Notes in Computer Science*, pages 311–322. Springer, January 2021. doi:10.1007/978-3-030-61792-9_25.

 © Susanna F. de Rezende; licensed under Creative Commons License CC-BY 4.0
42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).
Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;
Article No. 4; pp. 4:1–4:2



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



- 4 Susanna F. de Rezende, Noah Fleming, Duri Andrea Janett, Jakob Nordström, and Shuo Pang. Truly supercritical trade-offs for resolution, cutting planes, monotone circuits, and weisfeiler-leman. Technical Report 2411.14267, arXiv.org, November 2024. doi:10.48550/arXiv.2411.14267.
- 5 Susanna F. de Rezende, Or Meir, Jakob Nordstrom, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS '20)*, November 2020. doi:10.1109/focs46700.2020.00011.
- 6 Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS '16)*, October 2016. doi:10.1109/focs.2016.40.
- 7 Susanna F. de Rezende and Marc Vinyals. Lifting with colourful sunflowers. Manuscript, 2025.
- 8 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory of Computing*, 16(13):1–30, 2020. Preliminary version in *STOC '18*. doi:10.4086/toc.2020.v016a013.
- 9 Mika Göös, Gilbert Maystre, Kilian Risse, and Dmitry Sokolov. Supercritical tradeoffs for monotone circuits. Technical Report 2411.14268, arXiv.org, November 2024. doi:10.48550/arXiv.2411.14268.
- 10 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, January 2018. Preliminary version in *FOCS '15*. doi:10.1137/16m1059369.
- 11 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS '19)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:19, January 2019. doi:10.4230/LIPIcs.ITCS.2019.38.
- 12 Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990. Preliminary version in *STOC '88*. doi:10.1137/0403021.
- 13 Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with Sunflowers. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS '22)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 104:1–104:24, January 2022. doi:10.4230/LIPIcs.ITCS.2022.104.
- 14 Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC '17)*, pages 1246–1255, June 2017. doi:10.1145/3055399.3055478.
- 15 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999. Preliminary version in *FOCS '97*. doi:10.1007/s004930050062.
- 16 Alexander A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady*, 31(2):354–357, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.
- 17 Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: Mathematics*, pages 205–227, February 1995. doi:10.1070/im1995v059n01abeh000009.
- 18 Dmitry Sokolov. Dag-like communication and its applications. In *Proceedings of the 12th International Computer Science Symposium in Russia (CSR '17)*, volume 10304 of *Lecture Notes in Computer Science*, pages 294–307. Springer, June 2017. doi:10.1007/978-3-319-58747-9_26.
- 19 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL: <http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/>.

Parameterized Saga of First-Fit and Last-Fit Coloring

Akanksha Agrawal ✉ 

Indian Institute of Technology Madras, India

Daniel Lokshtanov ✉ 

University of California, Santa Barbara, CA, USA

Fahad Panolan ✉ 

School of Computer Science, University of Leeds, UK

Saket Saurabh ✉ 

The Institute of Mathematical Sciences, HBNI, Chennai, India

Department of Informatics, University of Bergen, Norway

Shaily Verma ✉ 

Algorithm Engineering Group, Hasso Plattner Institute, Potsdam, Germany

Abstract

The classic greedy coloring algorithm considers the vertices of an input graph G in a given order and assigns the first available color to each vertex v in G . In the GRUNDY COLORING problem, the task is to find an ordering of the vertices that will force the greedy algorithm to use as many colors as possible. In the PARTIAL GRUNDY COLORING, the task is also to color the graph using as many colors as possible. This time, however, we may select both the ordering in which the vertices are considered and which color to assign the vertex. The only constraint is that the color assigned to a vertex v is a color previously used for another vertex if such a color is available.

Whether GRUNDY COLORING and PARTIAL GRUNDY COLORING admit fixed-parameter tractable (FPT) algorithms, algorithms with running time $f(k)n^{\mathcal{O}(1)}$, where k is the number of colors, was posed as an open problem by Zaker and by Effantin et al., respectively.

Recently, Aboulker et al. (STACS 2020 and Algorithmica 2022) resolved the question for GRUNDY COLORING in the negative by showing that the problem is $W[1]$ -hard. For PARTIAL GRUNDY COLORING, they obtain an FPT algorithm on graphs that do not contain $K_{i,j}$ as a subgraph (a.k.a. $K_{i,j}$ -free graphs). Aboulker et al. re-iterate the question of whether there exists an FPT algorithm for PARTIAL GRUNDY COLORING on general graphs and also asks whether GRUNDY COLORING admits an FPT algorithm on $K_{i,j}$ -free graphs. We give FPT algorithms for PARTIAL GRUNDY COLORING on general graphs and for GRUNDY COLORING on $K_{i,j}$ -free graphs, resolving both the questions in the affirmative. We believe that our new structural theorems for partial Grundy coloring and “representative-family” like sets for $K_{i,j}$ -free graphs that we use in obtaining our results may have wider algorithmic applications.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Grundy Coloring, Partial Grundy Coloring, FPT Algorithm, $K_{i,j}$ -free graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.5

Related Version *Full Version*: <https://arxiv.org/pdf/2410.20629>

Funding *Akanksha Agrawal*: Supported by the New Faculty Seed Grant, IIT Madras (No. NFSC008972).

Saket Saurabh: Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416).



© Akanksha Agrawal, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Shaily Verma; licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).
Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;
Article No. 5; pp. 5:1–5:21



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



1 Introduction

A *proper coloring* of a graph G is an assignment of colors to its vertices such that none of the edges have endpoints of the same color. In k -COLORING, we are given a graph G , and the objective is to test whether G admits a proper coloring using at most k colors. The k -COLORING problem is one of the classical NP-hard problems, and it is NP-complete for every fixed $k \geq 3$. The problem is notoriously hard even to approximate. Indeed, approximating k -COLORING within $\mathcal{O}(n^{1-\epsilon})$, for any $\epsilon > 0$, is hard [18]. Also, under a variant of Unique Games Conjecture, there is no constant factor approximation for 3-COLORING [14].

The k -COLORING problem has varied applications ranging from scheduling, register allocations, pattern matching, and beyond [8, 9, 29]. Owing to this, several heuristics-based algorithms have been developed for the problem. One of the most natural greedy strategies considers the vertices of an input graph G in an arbitrary order and assigns to each vertex the first available color in the palette (the color palette for us is \mathbb{N}). In literature, this is called the *first-fit* rule. Notice that there is nothing special about using the “first” available color; one may instead opt for any of the previously used colors, if available, before using a new color; let us call this greedy rule the *any-available* rule. It leads to yet another greedy strategy to properly color a graph, and one can easily prove that this greedy strategy is equivalent to the “last-(available) fit” rule.

For any greedy strategy, one may wonder: *How bad can the strategy perform for the given instance?* The above leads us to the well-studied fundamental combinatorial problems, GRUNDY COLORING and PARTIAL GRUNDY COLORING, that arise from the aforementioned greedy strategies for proper coloring. In the GRUNDY COLORING problem, we are given a graph G on n vertices and an integer k , and the goal is to check if there is an ordering of the vertices on which the first-fit greedy algorithm for proper coloring uses at least k colors. Similarly, we can define the PARTIAL GRUNDY COLORING problem, where the objective is to check if, for the given graph G on n vertices and integer k , there is an ordering of the vertices on which the any-available greedy algorithm uses at least k colors. In this paper, we consider these two problems in the realm of parameterized complexity.

The GRUNDY COLORING problem has a rich history dating back to 1939 [21]. Goyal and Vishwanathan [20] proved that GRUNDY COLORING is NP-hard. Since then, there has been a flurry of results on the computational and combinatorial aspects of the problem both on general graphs and on restricted graph classes, see, for instance [3, 6, 7, 10, 16, 23, 24, 26, 27, 33, 35–39] (this list is only illustrative, not comprehensive). The problem PARTIAL GRUNDY COLORING was introduced by Erdős et al. [16] and was first shown to be NP-hard by Shi et al. [34]. The problem has gained quite some attention thereafter; see, for instance [1, 4, 5, 12, 15, 25, 32, 36].

These problems have also been extensively studied from the parameterized complexity perspective. Unlike k -COLORING, both these problems admit XP algorithms [15, 38], i.e., an algorithm running in time bounded by $|V(G)|^{f(k)}$. The above naturally raises the question of whether they are fixed-parameter tractable (FPT), i.e., admit an algorithm running in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$. In fact, these problems have also been explicitly stated as open problems [1, 7, 23, 33].

Havet and Sampaio [23] studied GRUNDY COLORING with an alternate parameter and showed that the problem of testing whether there is a Grundy coloring with at least $|V(G)| - q$ colors is FPT parameterized by q . Bonnet et al. [7] initiated a systematic study of designing parameterized and exact exponential time algorithms for GRUNDY COLORING and obtained FPT algorithms for the problem for several structured graph classes. They gave an exact

algorithm for GRUNDY COLORING running in time $2.443^n \cdot n^{\mathcal{O}(1)}$ and also showed that the problem is FPT on chordal graphs, claw-free graphs and graphs excluding a fixed minor. In the same paper, they stated the tractability status of GRUNDY COLORING on general graphs parameterized by the treewidth or the number of colors as central open questions. Belmonte et al. [6] resolved the first question by proving that GRUNDY COLORING is W[1]-hard parameterized by treewidth, but surprisingly, it becomes FPT parameterized by pathwidth. Later, Aboulker et al. [1] proved that GRUNDY COLORING does not admit an FPT algorithm (parameterized by the number of colors) and obtained an FPT algorithm for PARTIAL GRUNDY COLORING on $K_{t,t}$ -free graphs (which includes graphs of bounded degeneracy, graphs excluding some fixed graph as minor/topological minors, graphs of bounded expansion and nowhere dense graphs). A graph is $K_{i,j}$ -free if it does not have a *subgraph* isomorphic to the complete bipartite graph with i and j vertices, respectively, on the two sides. They concluded their work with the following natural open questions:

Question 1: Does PARTIAL GRUNDY COLORING admit an FPT algorithm?

Question 2: Does GRUNDY COLORING admit an FPT algorithm on $K_{i,j}$ -free graphs?

In this paper, we resolve the questions 1 and 2 in the affirmative by a new structural result and a new notion of representative families for $K_{i,j}$ -free graphs, respectively. In the next section, we give an intuitive overview of both results, highlighting our difficulties and our approaches to overcome them.

1.1 Our Results, Methods and Overview

Our first result is the following.

► **Theorem 1.** *PARTIAL GRUNDY COLORING is solvable in time $2^{\mathcal{O}(k^5)} \cdot n^{\mathcal{O}(1)}$.*

Our algorithm starts with the known “witness reformulation” of PARTIAL GRUNDY COLORING. It is known that (G, k) is a yes-instance of PARTIAL GRUNDY COLORING if and only if there is a vertex subset W of size at most k^2 such that, $(G[W], k)$ is a yes-instance of the problem. In the above, the set W is known as a *small witness set*. Our algorithm is about finding such a set W of size at most k^2 . Observe that this witness reformulation immediately implies that PARTIAL GRUNDY COLORING admits an algorithm with running time $n^{\mathcal{O}(k^2)}$ time. To build our intuition, we first give a simple algorithm for the problem on graphs of bounded degeneracy (or even, nowhere dense graphs). This algorithm has two main steps: (a) classical color-coding of Alon-Yuster-Zwick [2], and (b) independence covering lemma of Lokshtanov et al. [28].

Let (G, k) be a yes instance of PARTIAL GRUNDY COLORING, where G is a d -degenerate graph on n vertices, and W be a small witness set of size at most k^2 . As $(G[W], k)$ must be a yes-instance of the problem, there exists an ordering of the vertices such that when we apply *any-available* greedy rule, it uses at least k colors. Let \hat{c} be this proper coloring of $G[W]$. The tuple $(W_i := \{v \in W \mid \hat{c}(v) = i\})_{i \in [k]}$ is called a *k-partial Grundy witness* for G . Now we apply the color-coding step of the algorithm. That is, we color the vertices of G using k colors independently and uniformly at random, and let Z_1, \dots, Z_k be the color classes of this coloring. The probability that for each $i \in [k]$, $W_i \subseteq Z_i$, is k^{-k^2} . Notice that since G is a d -degenerate graph, we have that $G_i = G[Z_i]$, for each $i \in [k]$, is d -degenerate. Now we exploit this fact and apply the independence covering lemma of Lokshtanov et al. [28]. That is given as input (G_i, k^2) , in time $2^{\mathcal{O}(dk^2)} n^{\mathcal{O}(1)}$ it produces a family \mathcal{F}_i of independent sets of G_i , of size $2^{\mathcal{O}(dk^2)} \cdot \log n$. Furthermore, given any independent set I of G_i of size at most k^2 ,

there exists an independent set $F \in \mathcal{F}_i$, such that $I \subseteq F$ (F covers I). In particular, we know that there is a set $F_i \in \mathcal{F}_i$ that covers W_i . So, the algorithm just enumerates each tuple (F_1, \dots, F_k) of $\mathcal{F}_1 \times \dots \times \mathcal{F}_k$ and checks whether (F_1, \dots, F_k) is a k -partial Grundy coloring of $G[F_1 \cup \dots \cup F_k]$. If (G, k) is a yes instance, our algorithm is successful with probability k^{-k^2} . Moreover, we can convert the described randomized algorithm to a deterministic one by using the standard derandomization technique of “universal sets” [17, 31]. Some remarks are in order; it can be shown that each of $|W_i| \leq k$, and hence we can call the independence covering lemma on (G_i, k) , resulting in an improved running time of $2^{\mathcal{O}(dk^2 \log k)} \cdot n^{\mathcal{O}(1)}$. Aboulker et al. [1] proved that PARTIAL GRUNDY COLORING on $K_{t,t}$ -free graphs is FPT, which includes t -degenerate graphs. Aboulker et al. did not explicitly mention the running time, but their running time is at least $2^{k^t} n$. Our new algorithm improves over this. For general instances, we do not have a bound on the degeneracy of the input graph. However, we can achieve this by using our new structural result.

► **Theorem 2 (Structural result).** *There is a polynomial-time algorithm that given a graph G and a positive integer k , does one of the following:*

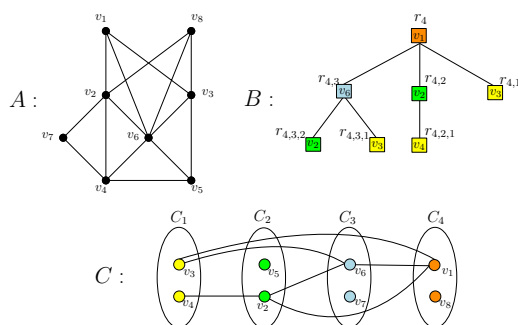
- (i) *Correctly concludes that (G, k) is a yes-instance of PARTIAL GRUNDY COLORING, or*
- (ii) *Outputs at most $2k^3$ induced bicliques A_1, \dots, A_ℓ in G such that the following holds. For any $v \in V(G)$, the degree of v in $G - F$ is at most k^3 , where F is the union of the edges in the above bicliques.*

The structural result (Theorem 2) is one of our main technical contributions. Next, we show how to design an algorithm for PARTIAL GRUNDY COLORING using Theorem 2. We follow the same steps as for the one described for the degenerate case. That is, we have color classes Z_i s and they contain the respective W_i s (the part of the small witness set W). Now, to design a family of independent sets in $G_i = G[Z_i]$, we do as follows. Let (L_j, R_j) be a bipartition of A_j , for each $j \in [\ell]$. Observe that any independent set I (in particular of G_i) intersects L_j or R_j , but *not both*, for any $j \in [\ell]$. Thus, we first guess whether W_i intersects L_j , R_j or none. Let this be given by a function $f_i : [\ell] \rightarrow \{L, R, N\}$, that is, if $f_i(j) = L$, then $W_i \cap L_j = \emptyset$, if $f_i(j) = R$, then $W_i \cap R_j = \emptyset$, else $W_i \cap (L_j \cup R_j) = \emptyset$. Taking advantage of this property, for each guess of which of L_j or R_j is not contained in W_i , we delete the corresponding set (which is one of L_j or R_j , for each $j \in [\ell]$) from G_i . We call the resulting graph $G_i^{f_i}$. This implies that for any f_i , in $G_i^{f_i}$ we delete all edges of F (where F is the union of edges in the bicliques). Hence, the maximum degree of $G_i^{f_i}$ is at most k^3 , and therefore it has degeneracy at most k^3 . Now using the independence covering step of the algorithm for degenerate graphs, we can finish the algorithm. The proof of Theorem 2 is obtained by carefully analyzing the reason for the failure of a greedy algorithm.

Our next result is an affirmative answer to Question 2.

► **Theorem 3.** *For any fixed $i, j \in \mathbb{N}$, there is an FPT algorithm that given a graph G and a positive integer k , decides if there is Grundy coloring of G using at least k colors.*

For our algorithm, we use a reinterpretation of the problem which is based on the existence of a *small witness*. Gyarfas et al. [22], and Zaker [38] independently showed that a given instance (G, k) of GRUNDY COLORING is a yes-instance if and only if there is a vertex subset W of size at most 2^{k-1} , such that $(G[W], k)$ is also a yes-instance of the problem. The existence of this small induced subgraph directly yield an XP algorithm for the problem [38]. Using characterizations of [22, 38] and basic Grundy coloring properties, we can reduce GRUNDY COLORING to finding a homomorphic image, satisfying some independence constraints, of some *specific labeled trees* (see Fig. 1, where different parts of it will be



■ **Figure 1** Illustration of a labelled homomorphism $\omega : V(T_4) \rightarrow V(G)$, where the graph is shown in part (i), T_4 with $V(T_4) = \{r_4, r_{4,3}, r_{4,2}, r_{4,1}, r_{4,3,2}, r_{4,3,1}, r_{4,2,1}\}$ is shown in part (ii), and a relation to Grundy coloring is illustrated in part (iii). Here, $\omega(r_4) = v_1, \omega(r_{4,3}) = v_6, \omega(r_{4,2}) = v_2, \omega(r_{4,1}) = v_3, \omega(r_{4,3,2}) = v_2, \omega(r_{4,3,1}) = v_3, \text{ and } \omega(r_{4,2,1}) = v_4$.

discussed, shortly). Let the pair (T, ℓ) denote a rooted tree T together with a labeling function $\ell : V(T) \rightarrow [k]$. Given (T, ℓ) and a graph G , a function $\omega : V(T) \rightarrow V(G)$ is a *labelled homomorphism* if: i) for each $\{u, v\} \in E(T)$, we have $\{\omega(u), \omega(v)\} \in E(G)$, and ii) for $u, v \in V(T)$, if $\ell(u) \neq \ell(v)$, then $\omega(u) \neq \omega(v)$. In particular, we will reduce the problem to the following.

CONSTRAINED LABEL TREE HOMOMORPHISM (CLTH) **Parameter:** $|V(T)|$

Input: A host graph G and $(T, \ell : V(T) \rightarrow [k])$, where T is a tree.

Question: Does there exist a labeled homomorphism $\omega : V(T) \rightarrow V(G)$ such that for any $z \in [k]$, $W_i = \{\omega(t) \mid t \in V(T) \text{ and } \ell(t) = i\}$ is an independent set in G ?

Now our goal is to identify ω (and thus the witness set W). The first step of our algorithm will be to use the color-coding technique of Alon-Yuster-Zwick [2] to ensure that the labeling requirement of the graph homomorphism ω is satisfied. To this end, we randomly color the vertices of G using k colors, where we would like the random coloring to ensure that for each $z \in [k]$, all the vertices in W_z are assigned the color z . Let X_1, X_2, \dots, X_k be the color classes in a coloring that achieves the above property. Our objective will be to find ω such that vertices of T that are labeled $z \in [k]$ are assigned to vertices in X_z .

Our next challenge is to find a homomorphism that additionally satisfies the independence condition. That is, vertices of the same label in T are assigned to an independent set in G . Note that the number of potential ω s that satisfy our requirements can be very huge; however, we will be able to carefully exploit $K_{i,j}$ -freeness to design a dynamic programming-based algorithm to identify one such ω (and thus the set W). Our approach here is inspired by dynamic programming in the design of FPT algorithms based on computations of “representative sets” [19,30]. However, this inspiration ends here, as to apply known methods we need to have an underlying family of sets that form a matroid. Unfortunately, we do not have any matroid structure to apply the known technique. Here, we exploit the fact that we have a specific labeled tree (T, ℓ) and a $K_{i,j}$ -free graph. Next we define the specific trees that we will be interested in (see Fig. 1, (ii)).

► **Definition 4.** For each $k \in \mathbb{N} \setminus \{0\}$, we (recursively) define a pair $(T_k, \ell_k : V(T_k) \rightarrow [k])$, called a k -Grundy tree, where T_k is a tree and ℓ_k is a *labelling* of $V(T_k)$, as follows :

1. $T_1 = (\{r_1\}, \emptyset)$ is a tree with exactly one vertex r_1 (which is also its root), and $\ell_1(r_1) = 1$.
2. Consider any $k \geq 2$, we (recursively) obtain T_k as follows. For each $z \in [k-1]$, let (T_z, ℓ_z) be the z -Grundy tree with root r_z . We assume that for distinct $z, z' \in [k-1]$, T_z and

$T_{z'}$ have no vertex in common, which we can ensure by renaming the vertices.¹ We set $V(T_k) = (\cup_{z \in [k-1]} V(T_z)) \cup \{r_k\}$ and $A(T_k) = (\cup_{z \in [k-1]} A(T_z)) \cup \{(r_k, r_z) \mid z \in [k-1]\}$. We set $\ell_k(r_k) = k$, and for each $z \in [k-1]$ and $t \in V(T_z)$, we set $\ell_k(t) = \ell_z(t)$. For $v \in V(T_k)$, $\ell_k(v)$ is the *label* of v in (T_k, ℓ_k) , and the elements in $[k]$ are *labels* of T_k .

Observe that the label of a vertex $t \in V(T)$ is the depth of the subtree rooted at t (the depth of a leaf is 1). In particular, the leaves are assigned the label 1, and when they are deleted, we get vertices with the label 2 as leaves, and so on. This allows us to do a bottom-up dynamic programming over T_k . Roughly speaking, for each $z \in [k]$, we solve the special labelled tree homomorphism from $\omega_z : V(T_z) \rightarrow X_1 \cup X_2 \dots \cup X_z$, where the root of T_z is mapped to a fixed vertex $v \in X_z$ as follows: instead of having all potential choices for ω_z (or $W_z = \{\omega_z(t) \mid t \in V(T_z)\}$), we find enough representatives, that will allow us to replace W_z by something that we have stored. It is a priori not clear that such representative sets of small size exist and furthermore, even if they exist, how to find them. The existence and computation of small representative sets in this setting is our main technical contribution for GRUNDY COLORING.

We heavily exploit the $K_{i,j}$ -freeness in our “representative set” computation. Very roughly stating, while we have computed required representatives for W_z , and wish to build such a representatives for W_{z+1} , by exploiting $K_{i,j}$ -freeness, we either find a small hitting set or a large sub-family of pair-wise disjoint sets. In the former case, we can split the family and focus on the subfamily containing a particular vertex from the hitting set and obtain a “representative” for it (and then take the union over such families). In the latter case, we show that we are very close to satisfying the required property, except for the sets containing vertices from an appropriately constructed small set S of vertices. The construction of this *small* set S is crucially based on the $K_{i,j}$ -freeness of the input graph. Once we have the set S , we can focus on sets containing a vertex from it and compute “representatives” for them.

Again, using standard hash functions, we can obtain a deterministic FPT algorithm for the problem by derandomizing the color coding based step [2, 31].

2 Preliminaries

Generic Notations. We denote the set of natural numbers by \mathbb{N} . For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \dots, n\}$. For a function $f : X \rightarrow Y$ and $y \in Y$, $f^{-1}(y) := \{x \in X \mid f(x) = y\}$.

For standard graph notations not explicitly stated here, we refer to the textbook of Diestel [13]. For a graph G , we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively. Also, if the context is clear, we will use n and m to denote the numbers $|V(G)|$ and $|E(G)|$, respectively. The neighborhood of a vertex v in a graph G is the set of vertices that are adjacent to v in G , and we denote it by $N_G(v)$. The degree of a vertex v is the size of its neighborhood in G , and we denote it by $d_G(v)$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = (\cup_{v \in S} N(v)) \setminus S$. When the graph is clear from the context, we drop the subscript G from the above notations. For $X \subseteq V(G)$, the induced subgraph of G on X , denoted by $G[X]$, is the graph with vertex set X and edge set $\{\{u, v\} \mid u, v \in X \ \& \ \{u, v\} \in E(G)\}$. Also, $G[V(G) \setminus X]$ is denoted by $G - X$. For $v \in V(G)$, we use $G - v$ to denote $G - \{v\}$ for ease of notation. For an edge subset $F \subseteq E(G)$, $G - F$ is the graph with vertex set $V(G)$ and edge set $E(G) \setminus F$. A bipartite graph $G = (A \uplus B, E)$ is called a *biclique* if every vertex in A

¹ For the sake of notational simplicity we will not explicitly write the renaming of vertices used to ensure pairwise vertex disjointness of the trees. This convention will be followed in the relevant section.

is adjacent to every vertex in B . We assume that A and B are both non-empty sets. For $d \in \mathbb{N}$, a graph is d -degenerate if each of its subgraphs has a vertex of degree at most d . For terminologies related to parameterized complexity, see the textbook of Cygan et al. [11].

3 FPT Algorithm for Partial Grundy Coloring

Consider a graph G and an integer $k \in \mathbb{N} \setminus \{0\}$. For a (not necessarily proper) coloring $\chi : V(G) \rightarrow [k]$, for simplicity, we sometimes write χ as the ordered tuple $(\chi^{-1}(1), \chi^{-1}(2), \dots, \chi^{-1}(k))$. Recall that a proper coloring of a graph is a coloring of its vertices so that for none of its edges, the two endpoints of it are of the same color. Also, a k -partial Grundy coloring of G is a proper coloring $c : V(G) \rightarrow [k]$, such that for each $i \in [k]$, there is a vertex $v \in V(G)$ with: (i) $c(v) = i$ and (ii) for every $j \in [i - 1]$, there is $u \in N_G(v)$ with $c(u) = j$.

We will begin with some definitions and results that will be useful in obtaining our main structural result (Theorem 2) and our FPT-algorithm (Theorem 1).

Observe that given a k -partial Grundy coloring of an induced subgraph \widehat{G} of a graph G , we can extend this coloring to a partial Grundy coloring of the whole graph G using at least k colors by greedily coloring the uncolored vertices of $G - V(\widehat{G})$. The following observation will be particularly useful when we work with a small “witness”.

► **Observation 5.** *Given a graph G , an induced subgraph \widehat{G} of G , and a partial Grundy coloring of \widehat{G} using k colors, we can find a partial Grundy coloring of G using at least k colors in linear time.*

Proof. Let $\widehat{c} : V(\widehat{G}) \rightarrow [k]$ be a partial Grundy coloring of \widehat{G} with exactly k colors. We construct a partial Grundy coloring $c : V(G) \rightarrow \mathbb{N}$ of G using at least k colors as follows. For each vertex $v \in V(\widehat{G})$, set $c(v) := \widehat{c}(v)$. Let $v_1, v_2, \dots, v_{n'}$ be an arbitrarily fixed order of vertices in $V(G) \setminus V(\widehat{G})$. Let $G_0 = \widehat{G}$, and for each $p \in [n']$, $G_p = G[V(\widehat{G}) \cup \{v_1, v_2, \dots, v_p\}]$. We iteratively create a partial Grundy coloring c_p of G_p using at least k colors (in increasing values of p) as follows. Note that $c_0 = \widehat{c}$ is already a partial Grundy coloring of G_0 that uses at least k colors. Consider $p \in [n'] \setminus \{0\}$, and assume that we have already computed a partial Grundy coloring $c_{p-1} : V(G_{p-1}) \rightarrow \mathbb{N}$ of G_{p-1} that uses $k' \geq k$ colors. For each $z \in [k']$, let $V_z = c_{p-1}^{-1}(z)$. For each $v \in V(G_{p-1})$, we set $c_p(v) := c_{p-1}(v)$. If the vertex v_p has a neighbor in each of the sets $V_1, V_2, \dots, V_{k'}$, i.e., if for each $z \in [k']$, $N_G(v_p) \cap V_z \neq \emptyset$, then set $c_p(v_p) := k' + 1$. Otherwise, let $z^* \in [k']$ be the smallest number such that $N_G(v_p) \cap V_{z^*} = \emptyset$, and set $c_p(v_p) := z^*$. Notice that by construction, c_p is a partial Grundy coloring of G_p using at least k colors. From the above discussions, $c_{n'}$ is a partial Grundy coloring of $G = G_{n'}$ using at least k colors. ◀

► **Definition 6.** Consider a graph G and an integer $k \in \mathbb{N} \setminus \{0\}$. A sequence of pairwise disjoint independent sets (Q_1, Q_2, \dots, Q_k) of G is a k -partial Grundy witness if the following holds. For any $i \in [k]$, there is $v \in Q_i$ such that for all $j \in [i - 1]$, $Q_j \cap N_G(v) \neq \emptyset$. The vertex v is called a *dominator* in Q_i .

► **Observation 7.** *Given a graph G and an integer k , let (Q_1, Q_2, \dots, Q_k) be a k -partial Grundy witness. Suppose Y_1, Y_2, \dots, Y_k are pairwise disjoint independent sets in G such that $Q_i \subseteq Y_i$, for all $i \in [k]$. Then (Y_1, Y_2, \dots, Y_k) is also a k -partial Grundy witness of G .*

A k -partial Grundy witness (X_1, X_2, \dots, X_k) is *small* if for each $i \in [k]$, $|X_i| \leq k - i + 1$. Next, we prove the existence of a small k -partial Grundy witness. This result is the same as the one obtained by Effantin et al. [15]; however, it is stated slightly differently for convenience.

► **Observation 8** (♠).² Let G be a graph and k be an integer, where G has a k -partial Grundy witness (Q_1, \dots, Q_k) . Then, there exists a k -partial Grundy witness (X_1, X_2, \dots, X_k) such that for each $i \in [k]$, $X_i \subseteq Q_i$ and $|X_i| \leq k - i + 1$.

The remainder of this section is organized as follows. In Section 3.1 we prove our key structural result (Theorem 2), and then obtain our algorithm in Section 3.2. (Readers who may want to read the algorithm directly, may skip Section 3.1.)

3.1 Degree Reduction: Proof of Theorem 2

The objective of this section is to prove Theorem 2. The proof of this theorem is based on the following lemma for bipartite graphs.

► **Lemma 9.** *There is a polynomial-time algorithm that, given a bipartite graph $G = (A \uplus B, E)$ and a positive integer k , does one of the following.*

- (i) *Correctly concludes that the partial Grundy coloring of G is at least k .*
- (ii) *Outputs at most $4k - 4$ bicliques A_1, \dots, A_ℓ in G such that for any $v \in V(G)$, degree of v in $G - F$ is at most k^2 , where F is the union of the edges in the above bicliques.*

We first give a proof of Theorem 2 based on the above lemma.

Proof of Theorem 2. Consider a graph G and a positive integer k . First, we run the first-fit greedy algorithm for proper coloring of the graph G for an arbitrarily fixed ordering (v_1, v_2, \dots, v_n) of $V(G)$. For each j , let C_j be the vertices colored j and k' be the largest integer such that $C_{k'} \neq \emptyset$. Note that $(C_1, \dots, C_{k'})$ is a proper coloring of G . Also, for any $j \in [k']$ and any vertex v in C_j , v has a neighbor in $C_{j'}$ for all $j' \in [j - 1]$. If $k' \geq k$, then $(C_1, \dots, C_{k'})$ is a partial Grundy coloring of G using at least k colors, and thus we can correctly report it.

Next, we assume that $k' < k$. Note that all the edges in G are between the color classes $C_1, \dots, C_{k'}$. Now for every distinct $i, j \in [k']$, where $i < j$, we apply Lemma 9 on $(H_{i,j} = (C_i \uplus C_j, E(C_i, C_j)), k)$, where $E(C_i, C_j)$ is the set of edges in G between the color classes C_i and C_j . Our algorithm will declare that G has a partial Grundy coloring using at least k colors if we get the output given in statement (i) in any of the $\binom{k'}{2}$ applications of Lemma 9. Otherwise, for every distinct $i, j \in [k']$, where $i < j$, let $A_{i,j,1}, \dots, A_{i,j,\ell_{i,j}}$ be the bicliques, we get as output by the algorithm in Lemma 9 on $(H_{i,j}, k)$. Note that $\ell_{i,j} \leq 4k - 4$. Now our algorithm will output the bicliques $\{A_{i,j,r} : 1 \leq i < j \leq k', r \in [\ell_{i,j}]\}$. As $k' < k$ for all $1 \leq i < j \leq k'$, the number of bicliques we output is at most $(4k - 4) \binom{k'}{2}$, that is, at most $2k^3$. Since any vertex in a color class C_r has neighbors in other color classes, for $r \in [k']$ and we applied Lemma 9 for every pair of color classes, the degree of v in $G - F$ is at most k^3 for any $v \in V(G)$, where F is the union of the edges in the above bicliques. This completes the proof of the theorem. ◀

We now focus on the proof of Lemma 9. Toward this, we give a polynomial time procedure that, given a bipartite graph $G = (L \uplus R, E)$ and a positive integer k , either concludes that the input graph has partial Grundy coloring using at least k colors or it outputs at most $2k - 2$ bicliques A_1, \dots, A_ℓ in G such that for any $v \in L$, $d_{G-F}(v) \leq k^2$, where F is the union of the edges in the above bicliques. That is, removal of the edges of these bicliques bounds the degree of each vertex in L by k^2 . We get the proof of Lemma 9 by applying this algorithm once for L and then for R .

² The proofs of the result marked with ♠ can be found in the full version of the paper on arXiv.

Overview of our algorithm. Let $\sigma = v_1, v_2, \dots, v_n$ be an ordering of the vertices in L in non-increasing order of their degree in G . The algorithm constructs *specific* color classes Q_1, Q_2, \dots, Q_r in this order so that $(C_1 = Q_r, C_2 = Q_{r-1}, \dots, C_r = Q_1)$ is an r -partial Grundy witness, where $|Q_j| \leq j$. Furthermore, we will construct sets $B_i, i \in [r]$, which will be used to construct the bicliques. Notice that if we obtain $r \geq k$, then we will be able to conclude that G has a partial Grundy coloring using at least k colors. Let $Q_1 = \{v_1\}$ and in our construction v_1 will be the dominator in Q_1 (and our construction needs to ensure this property; see Definition 6). Consider the construction of Q_2 . Let i be the smallest index in $\{2, 3, \dots, n\}$ for which there is a vertex $w \in N_G(v_1)$ such that v_i is not adjacent to w . Then, we set $Q_2 = \{v_i, w\}$, and designate v_i as the dominator in color class $C_{r-1} = Q_2$. Notice that all the vertices in $B_1 = \{v_2, \dots, v_{i-1}\}$ are adjacent to all the vertices in $N_G(v_1)$, and hence they together form a biclique (with bipartition B_1 and $N_G(v_1)$). This property will be extended in building each Q_j s and the required bicliques.

For the construction of Q_j , we consider *unprocessed vertices* (i.e., the vertices that do not belong to the previously constructed sets, i.e., to $Q_1 \cup B_1 \cup \dots \cup Q_{j-1} \cup B_{j-1}$) as follows. We would now like to choose an unprocessed vertex $v_{i'}$, so that we can make $v_{i'}$ the dominator of Q_j , and additionally, for each $j' \in [j-1]$, we can include a neighbor of the dominator from $Q_{j'}$ to the set Q_j . Note for us to do the above, we need to ensure that the vertices that we add to Q_j is an independent set in G , and all the vertices that we want to include in the set Q_j are outside $Q_1 \cup \dots \cup Q_{j-1}$. That is, among the unprocessed vertices, we choose the first vertex $v_{i'}$ with the following property: for each $j' \in [j-1]$, we have a neighbor $w_{j'}$ of the previously constructed dominator in $Q_{j'}$ such that $w_{j'} \notin Q_1 \cup \dots \cup Q_{j-1}$ and $(v_{i'}, w_{j'}) \notin E(G)$; we set $Q_j = \{v_{i'}, w_1, \dots, w_{j-1}\}$. We would like to mention that all the dominators we construct are from the bipartition L and hence $\{w_1, \dots, w_{j-1}\} \subseteq R$. This will imply that Q_j is an independent set.

Moreover, by the choice of $v_{i'}$ as the smallest vertex with the desired property, it follows that for any vertex v_r that appears before $v_{i'}$ in the order σ and $v_r \notin P = Q_1 \cup \dots \cup Q_{j-1}$, the vertex v_r is adjacent to all the vertices in $N(x_{j'}) \setminus P$, where $x_{j'}$ is the dominator in $Q_{j'}$, for some $j' \in [j-1]$. Then we add v_r to $B_{j'}$. Note that in the above process, we still maintain the biclique property, by explicitly ensuring that $B_{j'}$ and $N(x_{j'}) \setminus (Q_1 \cup \dots \cup Q_{j-1} \cup Q_j)$ forms a biclique.

Description of the algorithm. We give a pseudocode of our algorithm in Algorithm 1. First, the algorithm initializes the sets B_i and Q_i to be the empty set, for all $i \in [k]$ (see Algorithm 1). Let $\sigma = v_1, v_2, \dots, v_n$ be an ordering of the vertex set L in the non-increasing order of their degrees. Now, we want to construct the color classes Q_1, Q_2, \dots, Q_k , iteratively, such that $(C_1, C_2, \dots, C_k) = (Q_k, Q_{k-1}, \dots, Q_1)$ is a k -partial Grundy witness. At line 3, we initialize with $Q_1 := \{v_1\}$, $x_1 := v_1$, and fix the index $q = 2$. Here, Q_1 will be the color class C_k with dominator vertex x_1 . Now consider an iteration of the **while** loop. The algorithm checks if the set $L \setminus (\bigcup_{j \in [q-1]} (Q_j \cup B_j))$ is non-empty and executes the **while** loop. At this point we have constructed sets Q_1, \dots, Q_{q-1} such that $(Q_{q-1}, Q_{q-2}, \dots, Q_1)$ is a $(q-1)$ -partial Grundy witness such that each x_i is a dominator vertex in Q_i . Let v_r be the first unprocessed vertex in L and $P = \bigcup_{j \in [q-1]} Q_j$ by Lines 5 and 6. Now, we check if we can construct the current color class Q_q with vertex v_r as a dominator vertex, and for that, we need to add a neighbor w_j (which is not added to any $Q_{i'}$ before) for each already discovered dominator x_j such that w_j is non-adjacent to v_r . Now, if there exists some $j \in [q-1]$ such that each neighbor of x_j is a neighbor of v_r , then we will not be able to construct Q_q with vertex v_r in it. In that case, we choose such a value j and add v_r to B_j (See

5:10 Parameterized Saga of First-Fit and Last-Fit Coloring

■ **Algorithm 1** Algorithm for one-sided bipartite structural result.

```

1 Initialize  $B_i := \emptyset$  and  $Q_i := \emptyset$ , for each  $i \in [k]$ .
2 Let  $\sigma = v_1, \dots, v_n$  be an order of the vertices in  $L$  in the non-increasing order of
  their degrees.
3 Let  $x_1 := v_1$ ,  $Q_1 := \{x_1\}$  and  $q := 2$ .
4 while  $L \not\subseteq \bigcup_{j \in [q-1]} (Q_j \cup B_j)$  do
5   | Let  $v_r$  be the first vertex in  $\sigma$  from  $L \setminus (\bigcup_{j \in [q-1]} (Q_j \cup B_j))$ .
6   | Let  $P = \bigcup_{j \in [q-1]} Q_j$ .
7   | if there exists  $j \in [q-1]$  such that  $N(x_j) \subseteq P \cup N(v_r)$  then
8   |   | choose an arbitrary  $j$  with this property and set  $B_j := B_j \cup \{v_r\}$ .
9   | else
10  |   | For each  $j \in [q-1]$ ,  $N(x_j) \setminus (P \cup N(v_r)) \neq \emptyset$ . Then, for each  $j \in [q-1]$ 
11  |   |   | arbitrarily pick a vertex  $w_j$  from  $N(x_j) \setminus (P \cup N(v_r))$ .
12  |   |   | //(Notice that in the above,  $w_j$  may be equal to  $w_{j'}$  for distinct  $j, j' \in [q-1]$ ).
13  |   |   | Set  $x_q := v_r$ .
14  |   |   | Set  $Q_q := \{x_q\} \cup \{w_1, \dots, w_{q-1}\}$ .
15  |   |   | Set  $q := q + 1$ .
16  |   | end
17 end
18 if  $q \geq k + 1$  then
19 |   | Declare that partial Grundy coloring of  $G$  is at least  $k$ .
20 else
21 |   | For all  $j \in [q-1]$ , let  $A_j$  be the bipartite graph induced on  $B_j$  union  $N(x_j) \setminus P$ 
22 |   |   | and  $S_j$  be the graph induced on  $N[x_j]$ , where  $P = \bigcup_{j \in [q-1]} Q_j$ .
23 |   | Output  $A_1, \dots, A_{q-1}$  and  $S_1, \dots, S_{q-1}$ 
24 end

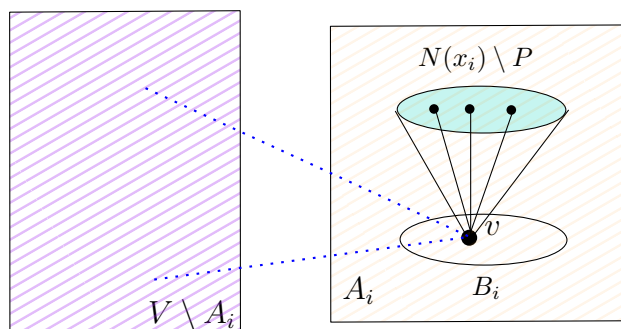
```

Line 8). Here, notice that v_r is adjacent to all the vertices $N(x_j) \setminus P$. We will maintain this property for all the vertices added to B_j , i.e., B_j union $N(x_j) \setminus \bigcup_i Q_i$ forms a biclique. Now consider the case that the condition in the **if** statement in Line 7 is false. Then, we choose a vertex $w_j \in N(x_j) \setminus (P \cup N(v_r))$ for each $j \in [q-1]$, by line 11 and set the vertex v_r as the dominator for Q_q , that is, $Q_q := \{x_q\} \cup \{w_1, \dots, w_{q-1}\}$, at Line 13. Notice that Q_q is an independent set because there is no edge between x_1 and a vertex in $\{w_1, \dots, w_{q-1}\}$, and $\{w_1, \dots, w_{q-1}\}$ is a subset of R , the right part of the bipartition of G . We repeat the iteration until one of the **while** loop conditions at line 4 fails. Next, if $q \geq k + 1$, we conclude that G has a partial Grundy coloring using k colors by line 18, because (Q_{q-1}, \dots, Q_1) is a $(q-1)$ -partial Grundy witness, where $q-1 \geq k$. Otherwise, by line 20, let A_j be the graph induced on $B_j \cup (N(x_j) \setminus P)$ and S_j be the graph induced on $N[x_j]$, for each $j \in [q-1]$. Recall that A_j is a biclique. It is easy to see that S_j is a biclique, because G is a bipartite graph. At line 21, the algorithm outputs the set of graphs A_1, \dots, A_{q-1} and S_1, \dots, S_{q-1} .

The number of iterations of the **while** loop is at most n and each step in the algorithm takes polynomial time, the total running time of the algorithm is polynomial in the input size. Next, we prove the correctness of the algorithm.

► **Lemma 10.** *Algorithm 1 is correct.*

Proof. Let q^* be the value of q at the end of the algorithm. To prove the correctness of the algorithm, first, we prove the following claim.



■ **Figure 2** Here the vertex $v \in B_i$ in the biclique A_i (right side). The number of neighbors of v outside A_i (blue edges) cannot be more than $|P|$ as $d_G(v) \leq d_G(x_i) = |N_G(x_i)|$ and v has $|N(x) \setminus P|$ neighbours in the biclique A_i .

▷ **Claim 11.** The following statements are true.

- (i) For each $i \in [q^* - 1]$, Q_i is an independent set and $Q_i \neq \emptyset$.
- (ii) For each $i \in [q^* - 1]$ and $j \in [i - 1]$, $N(x_j) \cap Q_i \neq \emptyset$.
- (iii) For each $i \in [q^* - 1]$ and $v \in B_i$, v is adjacent to all the vertices in $N(x_i) \setminus (\bigcup_{j \in [q^* - 1]} Q_j)$ and $d_G(v) \leq d_G(x_i)$.

Proof. We prove the statements by induction on i . The base case is when $i = 1$. Clearly, $Q_1 = \{x_1\}$ and hence statement (i) is true. Statement (ii) is vacuously true. Next, we prove statement (iii). Notice that in any iteration of the **while** loop, in Step 8, we may add a vertex v_r to B_1 . If this happens, then we know that $N(x_1) \subseteq P \cup N(v_r)$, where P is a subset of $\bigcup_{j \in [q^* - 1]} Q_j$. That is, all the vertices in $N(x_1) \setminus P$ are adjacent to v_r . This implies that v_r is adjacent to all the vertices in $N(x_1) \setminus (\bigcup_{j \in [q^* - 1]} Q_j)$. Since x_1 is the vertex with the maximum degree, we have that $d_G(v_r) \leq d(x_1)$.

Next, for the induction step, we assume that the induction hypothesis is true for $i - 1$, and we will prove that the hypothesis is true for i . Consider the iteration h^* of the **while** loop when $q = i$ and Steps 11-14 is executed. Let v_r be the vertex mentioned in Step 5 during that iteration. The vertices w_1, \dots, w_{q-1} belongs to R (the right side of the bipartition of G) and hence $\{w_1, \dots, w_{q-1}\}$ is an independent set. Also, notice that each w_j does not belong to $N(v_r)$ (See Step 11). Hence, $Q_q := \{v_r\} \cup \{w_1, \dots, w_{q-1}\}$ is an independent set and $Q_q \neq \emptyset$. Thus, we proved statement (i). Again, notice that $w_j \in N(x_j)$ for all $j \in [q - 1]$ (See Step 11). Thus, statement (ii) follows. Next, we prove statement (iii), which is similar to the proof of it in the base case. Notice that in any iteration of the while loop (after the iteration h^*), in Step 8, we may add a vertex $v_{r'}$ to B_i . If this happens, then we know that $N(x_i) \subseteq P \cup N(v_{r'})$, where P is a subset of $\bigcup_{j \in [q-1]} Q_j$. That is, all the vertices in $N(x_i) \setminus P$ are adjacent to $v_{r'}$. This implies that $v_{r'}$ is adjacent to all the vertices in $N(x_i) \setminus (\bigcup_{j \in [q-1]} Q_j)$. Since $x_i \in Q_i$, considered in iteration h^* , $d_G(v_{r'}) \leq d_G(x_i)$ (See Step 5). This completes the proof of the claim. ◁

Now suppose $q^* \geq k + 1$. Then, by Statements (i) and (ii) in Claim 11, we get that (Q_k, \dots, Q_1) is a partial Grundy coloring of the graph induced on $\bigcup_{j \in [q^* - 1]} Q_j$. Thus, if the algorithm executes Step 18, then it is correct because of Observation 5.

Now suppose $q^* \leq k$. Then the algorithm executes Step 21 and outputs the sets $A_1, \dots, A_{q^* - 1}$ and $S_1, \dots, S_{q^* - 1}$. Statement (iii) in Claim 11 implies that each A_j is a biclique in G , where $j \in [q^* - 1]$. Also, note that S_j is a biclique in G as it is induced on the set $N[x_j]$, for each $j \in [q^* - 1]$. Let F be the union of the edges in the bicliques

5:12 Parameterized Saga of First-Fit and Last-Fit Coloring

A_1, \dots, A_{q^*-1} and S_1, \dots, S_{q^*-1} . Next, we prove that for any $v \in L$, $d_{G-F}(v) \leq k^2$. Let $X = \{x_1, \dots, x_{q^*-1}\}$. It is easy to see that $|Q_j| \leq k$ and $L \cap Q_j = \{x_j\} \subseteq X$, for all $j \in [q^* - 1]$ (See Steps 11-13). Hence, $|\bigcup_{j \in [q^*]} Q_j| \leq k^2$. Let v be an arbitrary vertex in L . Note that if $v \in X$, the degree of v in $G - F$ is zero (by the definition of S_j). Next, suppose that $v \in L \setminus X$. Since $L \subseteq \bigcup_{j \in [q^*-1]} (Q_j \cup B_j)$ (this is the condition for **while** loop to exit) and $L \cap Q_j \subseteq X$ for all $j \in [q^* - 1]$, v belongs to B_i for some $i \in [q^* - 1]$. By statement (iii) in Claim 11, v is adjacent to all the vertices in $N(x_i) \setminus (\bigcup_{j \in [q^*-1]} Q_j)$ and $d_G(v) \leq d_G(x_i)$. Recall that the biclique A_i is the graph with bipartition B_i and $N(x_i) \setminus (\bigcup_{j \in [q^*-1]} Q_j)$. Moreover $v \in B_i$ and $d_G(v) \leq d_G(x_i) = |N_G(x_i)|$. This implies that the number of neighbors of v that does not belong to A_i is at most $|\bigcup_{j \in [q^*-1]} Q_j|$, which is upper bounded by k^2 . Therefore, the degree of v in $G - F$ is at most k^2 . See Figure 2 for an illustration. This concludes the proof. \blacktriangleleft

3.2 FPT Algorithm for Partial Grundy Coloring

In this section, we design an FPT algorithm \mathcal{A} for PARTIAL GRUNDY COLORING when the input has a structure dictated the second item of Lemma 9. Then, we explain how to get an FPT algorithm for PARTIAL GRUNDY COLORING on general graphs using \mathcal{A} and Theorem 2. Moreover, our algorithm \mathcal{A} will provide a faster algorithm for PARTIAL GRUNDY COLORING on d -degenerate graphs, improving the result in [1]. Toward the first task, we define the following problem.

STRUCTURAL PARTIAL GRUNDY COLORING (SPGC)

Input: Positive integers $k, d, \ell \in \mathbb{N}$, a graph G , and ℓ bicliques A_1, \dots, A_ℓ in G such that $G - F$ is d -degenerate, where F is the union of the edges in the above bicliques.

Question: Decide if there is a partial Grundy coloring for G using at least k colors.

First, we design a randomized polynomial time algorithm \mathcal{A}_1 for SPGC with a success probability at least $(k(d+1))^{-2k^2-k} \cdot 2^{-\ell k}$. We increase the probability of success to a constant by running \mathcal{A}_1 multiple times. Finally, we explain the derandomization of our algorithm. Then we prove Theorem 1 using this algorithm and our structural result (Theorem 2). To design the algorithm \mathcal{A}_1 , we use the following result of Lokshtanov et al. [28].

► **Proposition 12** (Lemma 1.1. [28]). *There is a linear-time randomized algorithm that, given a d -degenerate graph H and an integer k , outputs an independent set Y such that for any independent set X in H with $|X| \leq k$, the probability that $X \subseteq Y$ is at least $\left(\binom{k(d+1)}{k} \cdot k(d+1)\right)^{-1}$.*

The algorithm \mathcal{A}_1 has the following steps.

1. Color all vertices in $V(G)$ uniformly and independently at random with colors from the set $[k]$. Let the obtained coloring be $\phi : V(G) \rightarrow [k]$, and $Z_i = \phi^{-1}(i)$, for each $i \in [k]$.
2. For each $i \in [\ell]$, let $A_i = (L_i \uplus R_i, E_i)$. For each $j \in [k]$ and $i \in [\ell]$, uniformly at randomly assign $P_{j,i} := L_i$ or $P_i := R_i$. That is, with probability $\frac{1}{2}$, $P_{j,i} := L_i$ and with probability $\frac{1}{2}$, $P_{j,i} := R_i$. Let $D_j = \bigcup_{i \in [\ell]} P_{j,i}$.
3. Now for each $j \in [k]$, we apply the algorithm in Proposition 12 for $(G[Z_j - D_j], k)$ to obtain an independent set Y_j .
4. If (Y_1, \dots, Y_k) is a k -partial Grundy witness of G , then output **Yes**, else, output **No**.

Since the algorithm in Proposition 12 runs in linear time, the algorithm \mathcal{A}_1 can be implemented to run in linear time because Z_1, \dots, Z_k is a partition of $V(G)$. Clearly, if the algorithm \mathcal{A}_1 outputs **Yes**, then G has a k -Partial Grundy witness and hence G has a partial Grundy coloring using at least k colors. We can prove that if G has a k -Partial Grundy witness the algorithm \mathcal{A}_1 outputs **Yes** with probability $(k(d+1))^{-2k^2-k} \cdot 2^{-\ell k}$.

Also, by running \mathcal{A}_1 , $3 \cdot (k(d+1))^{2k^2+k} 2^{\ell k}$ times and outputting **Yes** if at least one of the runs outputs a **Yes**, and outputs **No**, otherwise, we can boost the success probability to $2/3$, and thus obtain the following result.

► **Theorem 13.** *There is a randomized algorithm for SPGC running in time $\mathcal{O}((k(d+1))^{2k^2+k} \cdot 2^{\ell k} \cdot (m+n))$. In particular, if (G, k) is a no-instance then with probability 1 the algorithm outputs **No**; and if (G, k) is a yes-instance then with probability $2/3$ the algorithm outputs **Yes**.*

Theorem 2 and 13 imply the following theorem.

► **Theorem 14.** *There is a randomized algorithm for PARTIAL GRUNDY COLORING running in time $2^{\mathcal{O}(k^4)} n^{\mathcal{O}(1)}$. In particular, if (G, k) is a no-instance then with probability 1 the algorithm outputs **No**; and if (G, k) is a yes-instance then with probability $2/3$ the algorithm outputs **Yes**.*

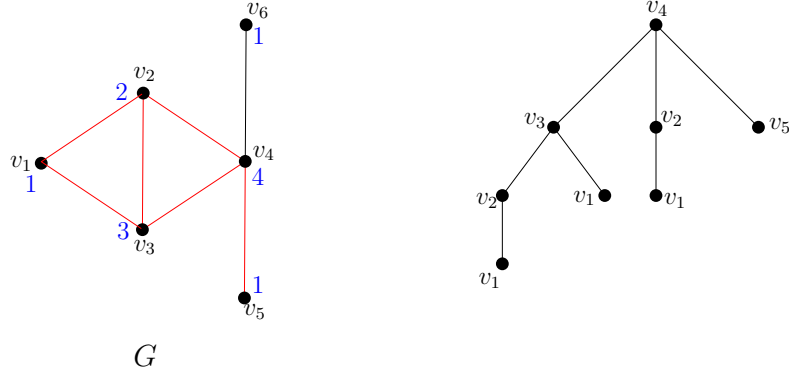
Proof Sketch. First we run the algorithm mentioned in Theorem 2. If it concludes that G has a partial Grundy coloring with at least k colors, then we output **Yes**. Otherwise, we get at most $2k^3$ induced bicliques $A_1 \cdots, A_\ell$ in G such that the following holds. For any $v \in V(G)$, the degree of v in $G - F$ is at most k^3 , where F is the union of the edges in the above bicliques. That is the degeneracy of $G - F$ is at most k^3 . Then, we apply Theorem 13, and outputs accordingly. ◀

The derandomization of our algorithm can be found in the full version.

4 FPT Algorithm for Grundy Coloring on $K_{i,j}$ -free Graphs

This section aims to prove Theorem 3. Consider fixed $i, j \in \mathbb{N} \setminus \{0\}$, where $i \geq j$. Recall that a graph is $K_{i,j}$ -free if it does not contain a subgraph isomorphic to $K_{i,j}$. We call the special case of GRUNDY COLORING where the input graph is $K_{i,j}$ -free, $K_{i,j}$ -FREE GRUNDY COLORING. Let (G, k) be an instance of $K_{i,j}$ -FREE GRUNDY COLORING. We begin by intuitively explaining the flow of our algorithm.

Consider a Grundy coloring $c : V(G) \rightarrow [k']$ of G , where $k' \geq k$, and for each $z \in [k]$, $c^{-1}(z) \neq \emptyset$. Furthermore, for $z \in [k']$, let $C_z = c^{-1}(z)$. Let us focus on the first k color classes, and for $z \in [k]$, arbitrarily fix a vertex $v_z \in C_z$. (Note that v_z has a neighbor in $C_{z'}$, for each $z' \in [z-1]$.) We next intuitively describe construction, for each $z \in [k]$, a set W_z initialized to $\{v_z\}$ as follows. Basically, for each v_z , add an arbitrarily chosen neighbor of it in color class $C_{z'}$, for every $z' < z$. We do the above process exhaustively; whenever we add a vertex to a set W_z , we add an arbitrarily chosen neighbor of it from each color class $C_{z'}$ to $W_{z'}$, where $z' < z$. Then, let $W = \cup_{z \in [k]} W_z$; we will call such a set W a k -Grundy set for G and we will show that such a set of size at most 2^{k-1} exists (for yes instances). For $z \in [k]$, let $W_{\leq z} = \cup_{z' \in [z]} W_{z'}$ and $W_{> z} = \cup_{z' \in [k] \setminus [z]} W_{z'}$. Note that $c|_W$ is a k -Grundy coloring of $G[W]$. Also, we will show that G has a Grundy coloring using at least k colors if and only if some subgraph of G has a Grundy coloring using *exactly* k colors. We remark that the above result and the existence of W are the same as the results of Gyarfas et al. [22] and Zaker [38], although, for the sake of convenience, we state it here slightly differently. If we



■ **Figure 3** An illustration of a graph G that admits a 4-Grundy coloring (on the left) and a 4-Grundy-witness ω (on the right).

can identify all the vertices in W (or some other k -Grundy set), then we will be done. The first step of our algorithm will be to use the technique of color coding to randomly color the vertices of G using k colors so that, for each $z \in [k]$, $v \in W_z$ is colored z ; such a coloring will be a *nice coloring* and it will be denoted by χ .

The next step of our algorithm is inspired by the design of FPT algorithms based on computations of “representative sets” [19,30]. To this end, we will interpret W in a “tree-like” fashion. With this interpretation, in a bottom-up fashion, for each $z \in [k]$ and $v \in X_z$, we will compute a family $\mathcal{F}'_{z,v}$, so that, if $v \in W_z$, then there will be $W' \in \mathcal{F}'_{z,v}$ so that $W' \cup W_{>z}$ is also a k -Grundy set for G . We will now formalize the above steps.

Grundy Tree & Grundy Witness. We recall the Definition 4 from Section 1, and obtain some properties regarding it.

- **Observation 15 (♠).** For $k \in \mathbb{N} \setminus \{0\}$, for the k -Grundy tree (T_k, ℓ_k) , $|V(T_k)| = 2^{k-1}$.
- **Observation 16 (♠).** Consider $k \in \mathbb{N} \setminus \{0\}$ and the k -Grundy tree (T_k, ℓ_k) . We have $|\ell^{-1}(k)| = 1$ and for each $z \in [k-1]$, $|\ell^{-1}(z)| = 2^{k-z-1}$.

Next, we define the notion of k -Grundy witness in a graph G .

► **Definition 17.** Consider $k \in \mathbb{N} \setminus \{0\}$ and a graph G . A k -Grundy witness for G is a function $\omega : V(T_k) \rightarrow V(G)$, where (T_k, ℓ_k) is the k -Grundy tree, such that: 1) for each $z \in [k]$, $\{\omega(t) \mid t \in V(T_k) \text{ and } \ell_k(t) = z\}$ is an independent set in G , 2) for each $t, t' \in V(T_k)$, if $\ell_k(t) \neq \ell_k(t')$ then $\omega(t) \neq \omega(t')$, and 3) for each $(t, t') \in A(T_k)$, we have $\{\omega(t), \omega(t')\} \in E(G)$.

Recall that for $k \in \mathbb{N} \setminus \{0\}$, for the k -Grundy tree (T_k, ℓ_k) , T_k is the tree obtained by adding a root vertex r_k attached to the roots of (pairwise vertex disjoint) trees $T_{k-1}, T_{k-2}, \dots, T_1$, where for each $z \in [k-1]$, (T_z, ℓ_z) is the z -Grundy tree. We have the following observation.

► **Observation 18 (♠).** Consider $k \in \mathbb{N} \setminus \{0\}$, a graph G and a k -Grundy witness $\omega : V(T_k) \rightarrow V(G)$ for G . For each $z \in [k]$, $\omega|_{V(T_z)}$ is a z -Grundy witness for G .

The next observation is a partial Grundy counterpart of Observation 5.

► **Observation 19 (♠).** Consider a graph G , any induced subgraph \widehat{G} of it, and an integer $k \in \mathbb{N}$. If \widehat{G} has a Grundy coloring that uses exactly k colors, then G has a Grundy coloring that uses at least k colors.

In the following two lemmas, we show that the existence of a k -Grundy witness for a graph is equivalent to the graph admitting a Grundy coloring with at least k colors.

► **Lemma 20.** *For any $k \in \mathbb{N} \setminus \{0\}$ and a graph G , if G has a k -Grundy witness, then G has a Grundy coloring with at least k colors.*

Proof. Consider a graph G and any $k \in \mathbb{N} \setminus \{0\}$. For a k -Grundy witness $\omega : V(T_k) \rightarrow V(G)$ of G , let $\widehat{V}_\omega = \{\omega(t) \mid t \in V(T_k)\}$, and for each $z \in [k]$, let $\widehat{V}_{\omega,z} = \{\omega(t) \mid t \in V(T_k) \text{ and } \ell_k(t) = z\}$. Note that from item 2 of Definition 17, $\widehat{V}_{\omega,1}, \widehat{V}_{\omega,2}, \dots, \widehat{V}_{\omega,k}$ is a partition of \widehat{V}_ω , where none of the parts are empty. Let $c_\omega : \widehat{V}_\omega \rightarrow [k]$ be the function such that for each $z \in [k]$ and $v \in \widehat{V}_{\omega,z}$, we have $c_\omega(v) = z$.

For each $k \in \mathbb{N} \setminus \{0\}$ and a k -Grundy witness $\omega : V(T_k) \rightarrow V(G)$ of G , we will prove by induction (on k) that c_ω is Grundy coloring of $G[\widehat{V}_\omega]$ using k colors. The above statement, together with Observation 19, will give us the desired result.

The base case is $k = 1$, where T_1 has exactly one vertex, r_1 . For any 1-Grundy witness, ω of G , note that $c_\omega(\omega(r_1)) = 1$ is a Grundy coloring for $G[\{\omega(r_1)\}]$ using 1 color. Now for the induction hypothesis suppose that for some $\widehat{k} \in \mathbb{N} \setminus \{0, 1\}$, for each $0 < k < \widehat{k}$, the statement is true. Now we will prove the statement for $k = \widehat{k}$, and to this end, we consider a k -Grundy witness $\omega : V(T_k) \rightarrow [k]$, where r_k is the root of T_k . Recall that T_k is the tree obtained by adding a root vertex r_k attached to the roots of (pairwise vertex disjoint) trees $T_{k-1}, T_{k-2}, \dots, T_1$, where for each $z \in [k-1]$, (T_z, ℓ_z) is the z -Grundy tree, and T_z is rooted at r_z . Let $V' = \widehat{V}_\omega \setminus \widehat{V}_{\omega,k}$, and consider a vertex $v \in \widehat{V}_{\omega,z^*}$, where $z^* \in [k-1]$. We will argue that for each $z' \in [z^* - 1]$, $N_G(v) \cap \widehat{V}_{\omega,z'} \neq \emptyset$. Note that there must exist $z \in [k-1]$ and $t \in V(T_z)$ such that $\omega(t) = v$ and $\ell_z(v) = z^*$, and we arbitrarily choose one such z and t . Let $V_z = \{\omega(t) \mid t \in V(T_z)\}$. From Observation 18, $\omega_z = \omega|_{V(T_z)}$ is a z -Grundy witness for G . Thus, from our induction hypothesis, $c_{\omega_z} = c_\omega|_{V_z}$ is a Grundy coloring for $G[V_z]$. From the above we can conclude that for each $q' \in [z^* - 1]$, $N_G(v) \cap \widehat{V}_{\omega,q'} \neq \emptyset$. Now consider the vertex $\omega(r_k) = v_k^*$ and any $z \in [k-1]$. Note that $\ell_k(r_z) = z$ and from item 3 of Definition 17, we can obtain that $\{v_k^*, \omega(r_z)\} \in E(G)$. From the above discussions, we can obtain that c_ω is Grundy coloring of $G[\widehat{V}_\omega]$ using at least z colors. This concludes the proof. ◀

► **Lemma 21.** *For any $k \in \mathbb{N} \setminus \{0\}$ and a graph G , if G has a Grundy coloring with at least k colors, then G has a k -Grundy witness.*

Proof. Consider a Grundy coloring $c : V(G) \rightarrow [k']$ of G with $k' \geq k$ colors, and for each $q \in [k']$, let $C_q = c^{-1}(q)$. We construct a Grundy witness $\omega : V(T_k) \rightarrow V(G)$ by processing labels of T_k starting at k and iteratively proceeding to smaller labels as follows while maintaining the below invariants.

Pre-condition: When we begin processing a label $q \in [k-1]$, for each $t \in V(T_k)$ with $\ell_k(t) \geq q$, we have fixed the vertex $\omega(t)$.

Post-condition: After processing label $q \in [k]$, we have fixed, for each $t \in V(T_k)$ with $\ell_k(t) \geq q$, and $t' \in N_{T_k}[t]$, the vertex $\omega(t')$; and these are the only vertices in T_k for which the vertex in G assigned by ω is determined.

Note that the pre-condition is vacuously satisfied for $q = k$. Recall that T_k is the tree obtained by adding a root vertex r_k attached to the roots $r_{k-1}, r_{k-2}, \dots, r_1$ of (pairwise vertex disjoint) trees $T_{k-1}, T_{k-2}, \dots, T_1$, respectively, where for each $q \in [k-1]$, (T_q, ℓ_q) is a q -Grundy tree. Pick any vertex $v_k \in C_k$, and set $\omega(r_k) := v_k$ and for each $q \in [k-1]$, set $\omega(r_q) := w_q^k$, where w_q^k is an arbitrarily chosen neighbor of v_k from C_q (which exists as c is a Grundy coloring). After the above step, the post-condition is satisfied for $q = k$.

Now we (iteratively, in decreasing order) consider $q \in [k-1] \setminus \{1\}$. From the pre-condition for q , we have fixed, for each $t \in V(T_k)$ with $\ell_k(t) \geq q$, the vertex $\omega(t)$. Consider $t \in V(T_k)$ with $\ell_k(t) = q$ and let $v_t = \omega(t)$. Let \widehat{T}_q be the subtree of T_k rooted at t , and let

$\widehat{\ell}_q = \ell_k|_{V(\widehat{T}_q)}$. Notice that $(\widehat{T}_q, \widehat{\ell}_q)$ is a q -Grundy tree, where \widehat{T}_q is the tree obtained from by adding edge between t and the roots $\widehat{r}_{q-1}, \widehat{r}_{q-2}, \dots, \widehat{r}_1$ of $\widehat{T}_{q-1}, \widehat{T}_{q-2}, \dots, \widehat{T}_1$, respectively, where $(\widehat{T}_{q'}, \ell_k|_{V(\widehat{T}_{q'})})$ is a q' -Grundy tree, for each $q' \in [q-1]$. For each $q' \in [q]$, let $\widehat{w}_{q'}^q$ be an arbitrarily chosen vertex from $N_G(v) \cap C_{q'}$, and we set $\omega(\widehat{r}_{q'}) = \widehat{w}_{q'}^q$. Notice that after the above step, the post-condition is satisfied for q , and the pre-condition is satisfied for $q-1$.

After we are done processing each $q \in [k] \setminus \{1\}$, the post-condition for $q=2$ (and the pre-condition of $q=1$ implies that for each $t \in V(T_k)$, we have determined the vertex $\omega(t)$). Moreover, the construction of ω implies that all the three conditions in Definition 17 are satisfied. This concludes the proof. \blacktriangleleft

We next summarize the result we obtain from the above two lemmas.

► **Lemma 22.** *Consider any $k \in \mathbb{N} \setminus \{0\}$ and a graph G . The graph G has a k -Grundy witness if and only if G has a Grundy coloring with at least k colors.*

Color Coding of G . We will next use the above lemma to simplify our job in the following sense. Let $\omega : V(T_k) \rightarrow V(G)$ be a (fixed) k -Grundy witness of G (if it exists), where (T_k, k) is a k -Grundy tree. Let $\widehat{V}_\omega = \{\omega(t) \mid t \in V(T_k)\}$, and for each $q \in [k]$, let $\widehat{V}_{\omega,q} = \{\omega(t) \mid t \in V(T_k) \text{ and } \ell_k(t) = q\}$. Roughly speaking, our new objective will be to find the vertices in \widehat{V}_ω and say that $G[\widehat{V}_\omega]$ admits a Grundy coloring with at least k colors, using which we can conclude that G admits a Grundy coloring with at least k colors. We will use the technique of color coding introduced by Alon et al. [2], to color the vertices in \widehat{V}_ω “nicely” as follows. Color each vertex in G uniformly at random using a color from $[k]$, and let $\chi : V(G) \rightarrow [k]$ be this coloring. A *nice* coloring is the one where, for each $q \in [k]$, the coloring assigns the color q to all the vertices in $\widehat{V}_{\omega,q}$.

We will work with the assumption that χ is a nice coloring of G , and for each $q \in [k]$, let $X_q = \chi^{-1}(q)$. Our objective will be to look for a k -Grundy witness $\widehat{\omega} : V(T_k) \rightarrow V(G)$, where (T_k, k) is a k -Grundy tree, such that for each $q \in [k]$ and $t \in V(T_k)$ with $\ell_k(t) = q$, we have $\widehat{\omega}(t) \in X_q$. To this end, we will store a “Grundy representative family” for each vertex in a bottom-up fashion, starting from $q=1$. The definition of such a representative is inspired by the q -representative families [19, 30], although here we need a “vectorial” form of representation. To this end, we introduce the following notations and definitions.

Grundy Representative Sets. Recall we have the coloring χ of G with color classes $X_z = \chi^{-1}(z)$, for $z \in [k]$. A vertex subset $A \subseteq V(G)$ is χ -independent if for each $z \in [k]$, $A \cap X_z$ is an independent set in G . For $p \in \mathbb{N}$, a family of vertex subsets \mathcal{F} is a p -family if each set in \mathcal{F} has size at most p and each $A \in \mathcal{F}$ is χ -independent. We will only be working with vectors all of whose entries are from \mathbb{N} without explicitly stating it. For a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, $\text{sum}(\vec{q})$ denotes the number $\sum_{z \in [k]} q_z$. For a vector $\vec{q} = (q_1, q_2, \dots, q_k)$ and $B \subseteq V(G)$, we say that the *size* of B is \vec{q} , written as $|B| = \vec{q}$, if for each $z \in [k]$, $|B \cap X_z| = q_z$. For vertex subsets A and B , A fits B if $A \cup B$ is χ -independent. For two vectors $\vec{q}_1 = (q_1^1, q_2^1, \dots, q_k^1)$ and $\vec{q}_2 = (q_1^2, q_2^2, \dots, q_k^2)$, and $\diamond \in \{\leq, \geq, >, <, =\}$, we write $\vec{q}_1 \diamond \vec{q}_2$ if for each $z \in [k]$, we have $q_z^1 \diamond q_z^2$. We next define the notion of \vec{q} -Grundy representation.

► **Definition 23.** Consider $p \in \mathbb{N}$, a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, and a p -family \mathcal{F} of vertex subsets of G . For a sub-family $\mathcal{F}' \subseteq \mathcal{F}$, we say that \mathcal{F}' \vec{q} -Grundy represents \mathcal{F} , written as $\mathcal{F}' \subseteq_{\text{prep}}^{\vec{q}} \mathcal{F}$, if the following holds. For any set B of size \vec{q} , if there is $A \in \mathcal{F}$ that fits B , then there is $A' \in \mathcal{F}'$ that fits B . In the above, \mathcal{F}' is a \vec{q} -Grundy representative for \mathcal{F} .

Next, we obtain some properties regarding \vec{q} -Grundy representatives.

► **Observation 24** (♠). Consider $p \in \mathbb{N}$, a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, and any two p -families \mathcal{F}_1 and \mathcal{F}_2 . If $\mathcal{F}'_1 \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}_1$ and $\mathcal{F}'_2 \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}_2$, then $\mathcal{F}'_1 \cup \mathcal{F}'_2 \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}_1 \cup \mathcal{F}_2$.

Consider $p \in \mathbb{N}$ and $v \in V(G)$. For a family \mathcal{F} over $V(G)$, $\mathcal{F} + v$ denotes the family $\{A \cup \{v\} \mid A \in \mathcal{F} \text{ and } A \cup \{v\} \text{ is } \chi\text{-independent}\}$. Similarly, $\mathcal{F} - v$ denotes the family $\{A \setminus \{v\} \mid A \in \mathcal{F}\}$. A p -family \mathcal{F} is a (p, v) -family if for each $A \in \mathcal{F}$, we have $v \in A$.

► **Observation 25** (♠). Consider $p \in \mathbb{N}$, a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, a vertex $v \in V(G)$ and a (p, v) -family \mathcal{F} . Let \vec{h} be the vector obtained from \vec{q} by increasing its $\chi(v)$ th coordinate by 1. If $\mathcal{F}' \subseteq_{\vec{h}}^{\text{grep}} \mathcal{F} - v$ and $\mathcal{F}'' \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F} - v$, then $(\mathcal{F}' + v) \cup (\mathcal{F}'' + v) \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}$.

For a p_1 -family \mathcal{F}_1 and a p_2 -family \mathcal{F}_2 , we define a $(p_1 + p_2)$ -family, $\mathcal{F}_1 \star \mathcal{F}_2 = \{A_1 \cup A_2 \mid A_1 \in \mathcal{F}_1, A_2 \in \mathcal{F}_2, \text{ and } A_1 \cup A_2 \text{ is } \chi\text{-independent}\}$. The following lemma will be helpful in obtaining a \vec{q} -representative for $\mathcal{F}_1 \star \mathcal{F}_2$.

► **Lemma 26**. Consider a p_1 -family \mathcal{F}_1 , a p_2 -family \mathcal{F}_2 , and a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, where $\text{sum}(\vec{q}) + p_1 + p_2 \leq 2^{k-1}$. Let $\mathcal{F}'_1 \subseteq \mathcal{F}_1$ be a p_1 -family such that for every vector $\vec{q}'_1 \geq \vec{q}$ with $\text{sum}(\vec{q}'_1) \leq \text{sum}(\vec{q}) + p_2$, $\mathcal{F}'_1 \subseteq_{\vec{q}'_1}^{\text{grep}} \mathcal{F}_1$. Similarly, consider a p_2 -family $\mathcal{F}'_2 \subseteq \mathcal{F}_2$ such that for every vector $\vec{q}'_2 \geq \vec{q}$ with $\text{sum}(\vec{q}'_2) \leq \text{sum}(\vec{q}) + p_1$, $\mathcal{F}'_2 \subseteq_{\vec{q}'_2}^{\text{grep}} \mathcal{F}_2$. Then, $\mathcal{F}'_1 \star \mathcal{F}'_2 \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}_1 \star \mathcal{F}_2$.

Proof. Consider any $B \subseteq V(G)$ of size \vec{q} for which there is $A \in \mathcal{F}_1 \star \mathcal{F}_2$, such that A fits B . As $A \in \mathcal{F}_1 \star \mathcal{F}_2$, there must exist sets $A_1 \in \mathcal{F}_1, A_2 \in \mathcal{F}_2$, such that $A_1 \cup A_2 = A$.

Let $\vec{\delta}'_1 = (\delta'_1 = |(A_2 \cap X_z) \setminus B|)_{z \in [k]}$. Note that $|B \cup A_2| = \vec{q} + \vec{\delta}'_1$, A_1 fits $B \cup A_2$ and $\text{sum}(\vec{q}) + \text{sum}(\vec{\delta}'_1) \leq \text{sum}(\vec{q}) + p_2$. By the premise of the lemma, there exist $A'_1 \in \mathcal{F}'_1$ such that A'_1 fits $B \cup A_2$, as $\mathcal{F}'_1 \subseteq_{\vec{q} + \vec{\delta}'_1}^{\text{grep}} \mathcal{F}_1$. The above implies that A_2 fits $B \cup A'_1$, where $A'_1 \in \mathcal{F}'_1$. Let $\vec{\delta}'_2 = (\delta'_2 = |(A'_1 \cap X_z) \setminus B|)_{z \in [k]}$, and note that $|B \cup A'_1| = \vec{q} + \vec{\delta}'_2$, A_2 fits $B \cup A'_1$ and $\text{sum}(\vec{q}) + \text{sum}(\vec{\delta}'_2) \leq \text{sum}(\vec{q}) + p_1$. Again, as $\mathcal{F}'_2 \subseteq_{\vec{q} + \vec{\delta}'_2}^{\text{grep}} \mathcal{F}_2$, there exists $A'_2 \in \mathcal{F}'_2$ such that A'_2 fits $B \cup A'_1$. The above discussions imply that, $A'_1 \in \mathcal{F}'_1, A'_2 \in \mathcal{F}'_2$, and thus $A'_1 \cup A'_2 \in \mathcal{F}'_1 \star \mathcal{F}'_2$, where $A'_1 \cup A'_2$ fits B . This concludes the proof. ◀

Recall that G is a $K_{i,j}$ -free graph, where $i \geq j$. Consider any computable function $f(k)$. Let $\eta_{f(k)} := i \cdot f(k) \cdot k$; where we skip the subscript $f(k)$ when the context is clear. Also, for $p \in \mathbb{N}$, let $\alpha_p := 3 \cdot k \cdot (p\eta)^{i+1}$; again we skip the subscript p , when the context is clear. We next state the main lemma, which lies at the crux of our algorithm.

► **Lemma 27** (♠). Consider any computable function $f : \mathbb{N} \rightarrow \mathbb{N} \setminus \{0\}$. There is an algorithm that takes as input $k \in \mathbb{N} \setminus \{0\}$, $p \in \mathbb{N}$, a vector $\vec{q} = (q_1, q_2, \dots, q_k)$, and a p -family \mathcal{F} of vertex subsets of a $K_{i,j}$ -free graph G on n vertices with a coloring $\chi : V(G) \rightarrow [k]$, where $p + \text{sum}(\vec{q}) \leq f(k)$. In time bounded by $\mathcal{O}(\alpha^{2p + \text{sum}(\vec{q})} \cdot p \cdot |\mathcal{F}|)$ we can find $\mathcal{F}' \subseteq \mathcal{F}$ with at most $\alpha^{2p + \text{sum}(\vec{q})}$ sets such that $\mathcal{F}' \subseteq_{\vec{q}}^{\text{grep}} \mathcal{F}$.

In the remainder of this section, we prove Theorem 3, assuming the correctness of Lemma 27.

Some Useful Notations. For $z \in \mathbb{N} \setminus \{0\}$ and $z' \in [z]$, let $\gamma_{z,z'}$ be the number of vertices with label z' in the z -Grundy tree (T_z, ℓ_z) , i.e., $\gamma_{z,z'} = |\ell_z^{-1}(z')|$.

Let $\vec{q}^* = \vec{\gamma}_k := (\gamma_{k,1}, \gamma_{k,2}, \dots, \gamma_{k,k})$. We will define a vector $\vec{q}_z^* = (q_{z,1}^*, q_{z,2}^*, \dots, q_{z,k}^*)$, for every $z \in [k]$. Intuitively speaking, the z' th entry of \vec{q}_z^* will denote the number of vertices with label z' appearing in T_k after removing exactly one subtree rooted at a vertex with label

z . Formally, for each $z' \in \{z+1, z+2, \dots, k\}$, we have $q_{z,z'}^* = \gamma_{k,z'}$, and for each $z' \in [z]$, $q_{z,z'}^* = \gamma_{k,z'} - \gamma_{z,z'}$. For $z \in [k]$, we let $\vec{0}_z$ be the vector of dimension k where the z th entry is 1, and all the other entries are 0.

For a tree \widehat{T} rooted at r and $t \in V(\widehat{T})$, we let \widehat{T}^t be the subtree of \widehat{T} rooted at t , i.e., $V(\widehat{T}^t) = \{t' \in V(\widehat{T}) \mid t' = t, \text{ or } t' \text{ is a descendant of } t \text{ in } \widehat{T}\}$ and $\widehat{T}^t = \widehat{T}[V(\widehat{T}^t)]$.

For a set $W \subseteq V(G)$, we say that W is a k -Grundy set if there is a k -Grundy witness $\omega : V(T_k) \rightarrow W$ for G . Moreover, W is *minimal* if no proper subset $W' \subset W$ is a k -Grundy set for G . For a k -Grundy set W and a k -Grundy witness $\omega : V(T_k) \rightarrow W$ for G , for $t \in V(T_k)$, we let $W_{\text{sub},t} = \{\omega(t') \mid t' \in V(T_k^t)\}$ and $W_{\text{exc},t} = \{\omega(t') \mid t' \in V(T_k) \setminus V(T_k^t)\}$.

Recall that we have a graph G and a coloring $\chi : V(G) \rightarrow [k]$, where for $z \in [k]$, we have $X_z = \chi^{-1}(z)$. For $z \in [k]$ and $v \in V(G)$, we define $\mathcal{F}_{z,v} := \{W \subseteq \cup_{z' \in [z]} X_{z'} \mid v \in W, W \text{ is } \chi\text{-independent and } z \leq |W| \leq 2^{z-1}\}$.

Description of the Algorithm. The objective of our algorithm will be to compute, for each $z \in [k]$ and $v \in X_z$, a family $\mathcal{F}'_{z,v} \subseteq \mathcal{F}_{z,v}$; starting from $z = 1$ (and then iteratively, for other values of z in increasing order), satisfying the following constraints:

Size Constraint. $|\mathcal{F}'_{z,v}| \leq \alpha^{2^k+1}$.

Correctness Constraint. For any $z \in [k]$ and $v \in X_z$, the following holds:

1. Each $A \in \mathcal{F}'_{z,v}$ is a z -Grundy set in G .
2. Consider any minimal k -Grundy set W , such that $v \in W$ (if it exists). Furthermore, let $\omega : V(T_k) \rightarrow W$ be a k -Grundy witness for G . For any $t \in V(T_k)$ with $\omega(t) = v$, where $\vec{q}_t = |W_{\text{exc},t}|$, there is $W' \in \mathcal{F}'_{z,v} \subseteq \mathcal{F}_{z,v}$ such that $W_{\text{exc},t} \cup W'$ is a k -Grundy set in G , i.e., $\mathcal{F}'_{z,v} \subseteq_{\text{grep}}^{\vec{q}_t} \mathcal{F}_{z,v}$.

Base Case. We are in our base case when $z = 1$; note that $[2^0] = \{1\}$. For each $v \in X_1$, set $\mathcal{F}'_{1,v} := \mathcal{F}_{1,v} = \{\{v\}\}$. Note that $\mathcal{F}'_{1,v}$ satisfies both the size and the correctness constraints.

Recursive Formula. Consider $z \in [k] \setminus \{1\}$ and $v \in X_z$. We suppose that for each $z' \in [z-1]$ and $v' \in X_{z'}$, we have computed $\mathcal{F}'_{z',v'}$ that satisfies both the size and the correctness constraints.

For each $z' \in [z-1]$, we create a family $\mathcal{F}_{z,v,z'}$, initialized to \emptyset as follows. For each $u \in X_{z'} \cap N_G(v)$ and $W \in \mathcal{F}'_{z',u}$, if $W \cup \{v\}$ is χ -independent and $|W \cup \{v\}| \leq 2^{z-1}$, then add $W \cup \{v\}$ to $\mathcal{F}_{z,v,z'}$. Note that $|\mathcal{F}_{z,v,z'}| \leq n \cdot \alpha_p^{2^k+1}$, where $p = 2^{z'-1}$. Using Lemma 27, for each vector $\vec{q} \leq \vec{q}_z^*$, we compute $\mathcal{F}'_{z,v,z',\vec{q}} \subseteq_{\text{grep}}^{\vec{q}} \mathcal{F}_{z,v,z'}$, where $|\mathcal{F}'_{z,v,z',\vec{q}}| \leq \alpha^{2^k}$, and set $\mathcal{F}'_{z,v,z'} = \cup_{\vec{q} \leq \vec{q}_z} \mathcal{F}'_{z,v,z',\vec{q}}$. Note that $|\mathcal{F}'_{z,v,z'}| \leq 2^{(k-1)k} \cdot \alpha^{2^k} \leq \alpha^{2^k+1}$ and we can compute it in time bounded by $\mathcal{O}(\alpha^{2^k+1} \cdot 2^{k-1} \cdot |\mathcal{F}_{z,v,z'}|)$.

Next we will iteratively “combine and reduce” the families $\mathcal{F}'_{z,v,z'}$, for $z' \in [z]$, to obtain a family $\widehat{\mathcal{F}}_{z,v} \subseteq \mathcal{F}_{z,v}$ as follows. We set $\widehat{\mathcal{F}}_{z,v,1} := \mathcal{F}'_{z,v,1}$. Iteratively, (in increasing order), for each $z' \in [z-1] \setminus \{1\}$, we do the following:

1. Set $\widetilde{\mathcal{F}}_{z,v,z'} := \widehat{\mathcal{F}}_{z,v,z'-1} \star \mathcal{F}'_{z,v,z'}$.
2. Compute $\widehat{\mathcal{F}}_{z,v,z',\vec{q}} \subseteq_{\text{grep}}^{\vec{q}} \widetilde{\mathcal{F}}_{z,v,z'}$, for each $\vec{q} \leq \vec{q}_z^* = \vec{\gamma}_k - (\sum_{z' \in [z']} (\vec{\gamma}_k - \vec{q}_z^*)) - \vec{0}_z$, and set $\widehat{\mathcal{F}}_{z,v,z'} = \cup_{\vec{q} \leq \vec{q}_z^*} \widehat{\mathcal{F}}_{z,v,z',\vec{q}}$. Note that $|\widehat{\mathcal{F}}_{z,v,z'}| \leq \alpha^{2^k+1}$ and it can be computed in time bounded by $\mathcal{O}(\alpha^{2^k+1} \cdot 2^{k-1} \cdot |\widetilde{\mathcal{F}}_{z,v,z'}|)$.

We add each $A \in \widehat{\mathcal{F}}_{z,v,z'-1}$ to $\mathcal{F}'_{z,v}$, which is a z -Grundy set in G (note that since the size of each set is bounded by 2^{k-1} , we can easily do it in the allowed amount of time). In the following lemma, we show that $\mathcal{F}'_{z,v}$ satisfies the correctness constraints.

► **Lemma 28.** $\mathcal{F}'_{z,v}$ satisfies the correctness constraint.

Proof. Consider any $v \in X_z$ and a minimal k -Grundy set W , such that $v \in W$ (if it exists) and let $\omega : V(T_k) \rightarrow W$ be a k -Grundy witness for G . Next consider any $t \in V(T_k)$ with $\omega(t) = v$. We will argue that, there is $W' \in \mathcal{F}'_{z,v}$ such that $W_{\text{exc},t} \cup W'$ is a k -Grundy set in G . For each $z' \in [z-1]$, let $t_{z'}$ be the child of t in T_k with $\ell_k(t_{z'}) = z'$ and $v_{z'} = \omega(t_{z'})$. Note that for each $z' \in [z-1]$, $v_{z'} \in X_{z'}$. For each $z' \in [z-1]$, let $\widehat{A}_{z'} = \{\omega(t') \mid t' \in V(T_k^{t_{z'}})\}$, and note that $|\widehat{A}_{z'}| \leq 2^{z'-1}$. Now we iteratively take the union of the above sets as follows. For each $z' \in [z-1]$, let $A_{z'} = \bigcup_{z \in [z']} \widehat{A}_{z'}$. Now for each $z' \in [z-1]$, we construct a subset, $B_{z'}$ of W that contains $\omega(t')$, for each $t' \in V(T_k)$ that does not belong to the subtrees rooted at any of the vertices $t_1, t_2, \dots, t_{z'}$. Formally, for $z' \in [z-1]$, let $B_{z'} = \{\omega(t') \mid t' \in V(T_k) \setminus (\bigcup_{z'' \in [z']} V(T_k^{t_{z''}}))\}$. Furthermore, let $\vec{s}_{z'}$ be the size of $B_{z'}$. Notice that for each $z' \in [z-1]$, all of the following holds:

1. $\vec{s}_{z'} \leq \vec{q}_{z'}$,
2. $|A_{z'}| \leq \sum_{z \in [z']} 2^{\widehat{z}-1}$,
3. $|\widehat{A}_{z'}| \leq 2^{z'-1}$ and $\widehat{A}_{z'} \in \mathcal{F}'_{z',v_{z'}}$,
4. $A_{z'} \cup B_{z'} = W$, and thus, $A_{z'}$ fits $B_{z'}$.

We will now iteratively define sets $A'_1, A'_2, \dots, A'_{z-1}$ and functions $\omega_1, \omega_2, \dots, \omega_{z-1}$, and we will ensure that, for each $z' \in [z-1]$, we have: i) $\omega_{z'} : V(T_k) \rightarrow A'_{z'} \cup B_{z'}$ is a k -Grundy witness for G , ii) for each $z' \in [z-1]$ and $t' \in V(T_k) \setminus (\bigcup_{z'' \in [z']} V(T_k^{t_{z''}}))$, we have $\omega_{z'}(t') = \omega(t')$, iii) $A'_{z'} \in \widehat{\mathcal{F}}_{z,v,z'}$, and iv) for each $z'' \in [z']$, there is a minimal z'' -Grundy set $A'_{z',z''} \subseteq A'_{z'}$, where the unique vertex in $A'_{z',z''} \cap X_{z''}$ is a neighbor of v .

Recall that $z \geq 2$ and $\widehat{\mathcal{F}}_{z,v,1} = \mathcal{F}'_{z,v,1} \subseteq \mathcal{F}_{z,v,1}$. Also, we have $\widehat{A}_1 = A_1 = \{u'\}$, for some $u' \in N_G(v) \cap X_1$, and $A_1 \cup B_1 = W$ is a k -Grundy set. Thus, there must exist $A'_1 \in \mathcal{F}'_{z,v,1}$ such that $A'_1 \cup B_1$ is χ -independent. Moreover by the construction of $\mathcal{F}'_{z,v,1}$, $A'_1 = \{u\}$, for some $u \in N_G(v) \cap X_1$. Let $\omega_1 : V(T_k) \rightarrow A'_1 \cup B_1$ be the function such that for each $t' \in V(T_k) \setminus \{t_1\}$, we have $\omega_1(t') = \omega(t')$ and $\omega_1(t_1) = u$. As $A'_1 \cup B_1$ is χ -independent and $\{u, v\} \in E(G)$, we can obtain that ω_1 is a k -Grundy witness for G . Note that if $z = 2$, then by the above arguments, we have constructed the desired sets and functions, which is just the set A'_1 and the function ω_1 .

We now consider the case when $z' \geq 2$. Also, we assume that for some $\widehat{z} \in [z-2]$, for each $z' \in [\widehat{z}]$, we have constructed $A'_{z'}$ and $\omega_{z'}$ satisfying the desired condition. Now we prove the statement for $z' = \widehat{z} + 1$. Note that $A'_{z'-1} \cup B_{z'-1}$ is a k -Grundy set and $\omega_{z'-1} : V(T_k) \rightarrow A'_{z'-1} \cup B_{z'-1}$ is a k -Grundy witness for G , where $A'_{z'-1} \in \widehat{\mathcal{F}}_{z,v,z'}$. Note that $\widehat{A}_{z'} \subseteq B_{z'-1}$. Let $B'_{z'} = \{\omega_{z'-1}(t') \mid t' \in V(T_k) \setminus V(T_k^{t_{z'}})\}$. Note that $|B'_{z'}| \leq \vec{q}_{z'}$ and $\widehat{A}_{z'}$ fits $B'_{z'}$, and recall that $\widehat{A}_{z'} \in \mathcal{F}'_{z',v_{z'}}$. As $\mathcal{F}'_{z',v_{z'}} \subseteq_{\vec{q}} \mathcal{F}_{z',v_{z'}}$, for every $\vec{q} \leq \vec{q}_{z'}$, there must exist $\widetilde{A}_{z'} \in \mathcal{F}'_{z',v_{z'}}$, such that $\widetilde{A}_{z'}$ fits $B'_{z'}$. By the construction of $\mathcal{F}'_{z',v_{z'}}$, we have $v_{z'} \in \widetilde{A}_{z'}$ and $\widetilde{A}_{z'} \cup B'_{z'}$ is χ -independent, and also $v \in B'_{z'}$. From the above discussions we can conclude that $\widetilde{A}_{z'} \cup \{v\} \in \mathcal{F}_{z,v,z'}$. Moreover, as $A'_{z'-1} \in \widehat{\mathcal{F}}_{z,v,z'-1}$, $A'_{z'-1} \subseteq B'_{z'}$ and $\widetilde{A}_{z'} \cup B'_{z'}$ is χ -independent, we can obtain that $A'_{z'-1} \cup \widetilde{A}_{z'} \cup \{v\} \in \widehat{\mathcal{F}}_{z,v,z'-1} \star \mathcal{F}'_{z,v,z'} = \widehat{\mathcal{F}}_{z,v,z'}$. As $\widehat{\mathcal{F}}_{z,v,z'} \subseteq_{\vec{q}} \mathcal{F}_{z,v,z'}$, for every $\vec{q} \leq \vec{q}_{z'}^*$ and $|B_{z'}| \leq \vec{q}_{z'}^*$, there must exist $A'_{z'} \in \widehat{\mathcal{F}}_{z,v,z'}$ such that $A'_{z'} \cup B_{z'}$ is χ -independent.

As $A'_{z'} \in \widehat{\mathcal{F}}_{z,v,z'}$, there must exist $\widehat{C} \in \widehat{\mathcal{F}}_{z,v,z'-1}$ and $C' \cup \{v\} \in \mathcal{F}'_{z,v,z'}$, such that $A'_{z'} = \widehat{C} \cup C' \cup \{v\}$. By the correctness for $z'-1$, C' contains for each $z'' \in [z'-1]$, a minimal z'' -Grundy set $A'_{z'-1,z''} \subseteq A'_{z'-1}$, where the unique vertex in $A'_{z'-1,z''} \cap X_{z''}$ is a neighbor of v . Also by the construction of $\mathcal{F}'_{z,v,z'}$, C' contains a minimal z' -Grundy set C'' in G , where the unique vertex in $C'' \cap X_{z'}$ is a neighbor of v . From the above discussions, we can conclude that $A'_{z'} \cup B_{z'}$ is a k -Grundy set in G . \blacktriangleleft

Using the above algorithm, we can compute for each $z \in [k]$ and $v \in X_z$, a family $\mathcal{F}'_{z,v} \subseteq \mathcal{F}_{z,v}$ that satisfies the correctness and the size constraints, in time bounded by $\alpha^{\mathcal{O}(2^k+1)} \cdot n^{\mathcal{O}(1)}$. Note that G has a Grundy coloring using at least k colors if and only if for some $v \in X_k$, $\mathcal{F}'_{z,v} \neq \emptyset$. This implies a proof of Theorem 3.

References

- 1 Pierre Aboulker, Édouard Bonnet, Eun Jung Kim, and Florian Sikora. Grundy coloring and friends, half-graphs, bicliques. *Algorithmica*, pages 1–28, 2022. doi:10.1007/S00453-022-01001-2.
- 2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
- 3 Júlio César Silva Araújo and Cláudia Linhares Sales. Grundy number on p_4 -classes. *Electron. Notes Discret. Math.*, 35:21–27, 2009. doi:10.1016/J.ENDM.2009.11.005.
- 4 R. Balakrishnan and T Kavaskar. Interpolation theorem for partial Grundy coloring. *Discrete Mathematics*, 313(8):949–950, 2013. doi:10.1016/J.DISC.2013.01.018.
- 5 Rangaswami Balakrishnan and T. Kavaskar. Color chain of a graph. *Graphs Comb.*, 27(4):487–493, 2011. doi:10.1007/S00373-010-0989-7.
- 6 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In *28th Annual European Symposium on Algorithms (ESA 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 7 Édouard Bonnet, Florent Foucaud, Eun Jung Kim, and Florian Sikora. Complexity of Grundy coloring and its variants. *Discret. Appl. Math.*, 243:99–114, 2018. doi:10.1016/J.DAM.2017.12.022.
- 8 Gregory J. Chaitin, Marc A. Auslander, Ashok K. Chandra, John Cocke, Martin E. Hopkins, and Peter W. Markstein. Register allocation via coloring. *Comput. Lang.*, 6(1):47–57, 1981. doi:10.1016/0096-0551(81)90048-5.
- 9 CWK Chen and David YY Yun. Unifying graph-matching problem with a practical solution. In *Proceedings of International Conference on Systems, Signals, Control, Computers*, volume 55, 1998.
- 10 C. A. Christen and S. M. Selkow. Some perfect coloring properties of graphs. *Journal of Combinatorial Theory, Series B*, 27(1):49–59, 1979. doi:10.1016/0095-8956(79)90067-4.
- 11 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 12 Lyes Dekar, Brice Effantin, and Hamamache Kheddouci. An incremental distributed algorithm for a partial Grundy coloring of graphs. In Ivan Stojmenovic, Ruppia K. Thulasiram, Laurence Tianruo Yang, Weijia Jia, Minyi Guo, and Rodrigo Fernandes de Mello, editors, *Parallel and Distributed Processing and Applications, 5th International Symposium, ISPA 2007, Niagara Falls, Canada, August 29-31, 2007, Proceedings*, volume 4742 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2007. doi:10.1007/978-3-540-74742-0_18.
- 13 Reinhard Diestel. *Graph theory*, volume 173. Springer-Verlag Heidelberg, 2017.
- 14 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM J. Comput.*, 39(3):843–873, 2009. doi:10.1137/07068062X.
- 15 B. Effantin, N. Gastineau, and O. Togni. A characterization of b -chromatic and partial Grundy numbers by induced subgraphs. *Discrete Mathematics*, 339(8):2157–2167, 2016. doi:10.1016/J.DISC.2016.03.011.
- 16 P. Erdős, S. T. Hedetniemi, R. C. Laskar, and G. C. E. Prins. On the equality of the partial Grundy and upper chromatic numbers of graphs. *Discrete Mathematics*, 272(1):53–64, 2003. doi:10.1016/S0012-365X(03)00184-5.
- 17 P Fahad. *Dynamic Programming using Representative Families*. PhD thesis, Homi Bhabha National Institute, 2015.

- 18 Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57(2):187–199, 1998. doi:10.1006/JCSS.1998.1587.
- 19 Fedor V. Fomin, Daniel Lokshтанov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.
- 20 N. Goyal and S. Vishwanathan. Np-completeness of undirected grundy numbering and related problems. *Unpublished manuscript*, 1997.
- 21 P. M. Grundy. Mathematics and games. *Eureka*, 2:6–9, 1939.
- 22 András Gyárfás, Zoltán Király, and Jenő Lehel. On-line 3-chromatic graphs i. triangle-free graphs. *SIAM J. Discret. Math.*, 12(3):385–411, 1999. doi:10.1137/S089548019631030X.
- 23 F. Havet and L. Sampaio. On the grundy and b-chromatic numbers of a graph. *Algorithmica*, 65(4):885–899, 2013. doi:10.1007/S00453-011-9604-4.
- 24 S. M. Hedetniemi, S. T. Hedetniemi, and T. Beyer. A linear algorithm for the grundy (coloring) number of a tree. *Congressus Numerantium*, 36:351–363, 1982.
- 25 Allen Ibiapina and Ana Silva. b-continuity and partial grundy coloring of graphs with large girth. *Discrete Mathematics*, 343(8):111920, 2020. doi:10.1016/J.DISC.2020.111920.
- 26 Hal A. Kierstead and Karin Rebecca Saoub. First-fit coloring of bounded tolerance graphs. *Discret. Appl. Math.*, 159(7):605–611, 2011. doi:10.1016/J.DAM.2010.05.002.
- 27 Guy Kortsarz. A lower bound for approximating the grundy number. *Discrete Mathematics & Theoretical Computer Science*, 9, 2007.
- 28 Daniel Lokshтанov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. *ACM Transactions on Algorithms (TALG)*, 16(3):1–31, 2020. doi:10.1145/3379698.
- 29 Dániel Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering (Archives)*, 48(1-2):11–16, 2004.
- 30 Dániel Marx. A parameterized view on matroid optimization problems. *Theor. Comput. Sci.*, 410(44):4471–4479, 2009. doi:10.1016/J.TCS.2009.07.027.
- 31 Moni Naor, Leonard J Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191. IEEE, 1995. doi:10.1109/SFCS.1995.492475.
- 32 B. S. Panda and Shaily Verma. On partial grundy coloring of bipartite graphs and chordal graphs. *Discret. Appl. Math.*, 271:171–183, 2019. doi:10.1016/J.DAM.2019.08.005.
- 33 L. Sampaio. *Algorithmic aspects of graph colourings heuristics*. PhD thesis, University Nice Sophia Antipolis, 2012.
- 34 Z. Shi, W. Goddard, S. T. Hedetniemi, K. Kennedy, R. Laskar, and A. McRae. An algorithm for partial grundy number on trees. *Discrete Mathematics*, 304(1-3):108–116, 2005. doi:10.1016/J.DISC.2005.09.008.
- 35 Zixing Tang, Baoyindureng Wu, Lin Hu, and Manouchehr Zaker. More bounds for the grundy number of graphs. *J. Comb. Optim.*, 33(2):580–589, 2017. doi:10.1007/S10878-015-9981-8.
- 36 Shaily Verma and B. S. Panda. Grundy coloring in some subclasses of bipartite graphs and their complements. *Information Processing Letters*, 163:105999, 2020. doi:10.1016/J.IPL.2020.105999.
- 37 M. Zaker. Grundy chromatic number of the complement of bipartite graphs. *Australasian Journal of Combinatorics*, 31:325–330, 2005. URL: http://ajc.maths.uq.edu.au/pdf/31/ajc_v31_p325.pdf.
- 38 M. Zaker. Results on the grundy chromatic number of graphs. *Discrete mathematics*, 306(23):3166–3173, 2006. doi:10.1016/J.DISC.2005.06.044.
- 39 Manouchehr Zaker. Inequalities for the grundy chromatic number of graphs. *Discret. Appl. Math.*, 155(18):2567–2572, 2007. doi:10.1016/J.DAM.2007.07.002.

Twin-Width One

Jungho Ahn ✉ 🏠 

Korea Institute for Advanced Study (KIAS), Seoul, South Korea

Hugo Jacob ✉ 

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Noleen Köhler ✉ 

University of Leeds, UK

Christophe Paul ✉ 

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Amadeus Reinald ✉ 

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Sebastian Wiederrecht ✉ 

School of Computing, KAIST, Daejeon, South Korea

Abstract

We investigate the structure of graphs of twin-width at most 1, and obtain the following results:

- Graphs of twin-width at most 1 are permutation graphs. In particular they have an intersection model and a linear structure.
- There is always a 1-contraction sequence closely following a given permutation diagram.
- Based on a recursive decomposition theorem, we obtain a simple algorithm running in linear time that produces a 1-contraction sequence of a graph, or guarantees that it has twin-width more than 1.
- We characterise distance-hereditary graphs based on their twin-width and deduce a linear time algorithm to compute optimal sequences on this class of graphs.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Twin-width, Hereditary graph classes, Intersection model

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.6

Related Version *Full Version:* [arxiv:2501.00991](https://arxiv.org/abs/2501.00991)

Funding *Jungho Ahn:* Supported by the KIAS Individual Grant (CG095301) at Korea Institute for Advanced Study.

Hugo Jacob: Supported by the ANR project GODASse ANR-24-CE48-4377.

Christophe Paul: Supported by the ANR project GODASse ANR-24-CE48-4377 and the ANR-DFG project UTMA ANR-20-CE92-0027.

Amadeus Reinald: Supported by the ANR project DIGRAPHS ANR-19-CE48-0013.

Acknowledgements The authors wish to thank the organisers of the 1st Workshop on Twin-width, which was partially financed by the grant ANR ESIGMA (ANR-17-CE23-0010) of the French National Research Agency. The second author wishes to thank Paul Bastide and Carla Groenland for interesting initial discussions on the topic of this paper.

1 Introduction

Twin-width is a graph invariant introduced by Bonnet, Kim, Thomassé, and Watrigant [11] as a generalisation of a parameter on permutations introduced by Guillemot and Marx [21]. The main result of the seminal paper [11] is an FPT algorithm for FO model-checking on graphs of bounded twin-width, when given a *contraction sequence* certifying their width. Graphs of



© Jungho Ahn, Hugo Jacob, Noleen Köhler, Christophe Paul, Amadeus Reinald, and Sebastian Wiederrecht;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 6; pp. 6:1–6:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



bounded twin-width capture a wide variety of classes such as bounded rank-width and proper minor-closed classes, unifying many results on tractable FO model-checking. Furthermore, it gives an exact dichotomy between tractability and intractability of FO model-checking for hereditary classes of ordered structures [8] and of tournaments [19].

While these results are sufficient to consider twin-width an important graph parameter, we are still lacking an FPT algorithm for computing “good” contraction sequences, i.e. contraction sequences of width bounded by a function of the twin-width. Such an algorithm is required for FO model-checking to be FPT on bounded twin-width classes. Regarding exact algorithms for twin-width, it is known that computing twin-width is hard even for constant values: distinguishing graphs of twin-width 4 from graphs of twin-width 5 is NP-hard [5]. Still, a polynomial-time algorithm for recognizing graphs of twin-width 1 was given in [10], where the worst-case time complexity is not explicitly given but corresponds to a polynomial of degree 4 or 5 depending on implementation. Meanwhile, the hardness of recognizing twin-width 2 and 3 graphs remains open. Nevertheless, it is known that sparse graphs of twin-width 2 have bounded treewidth [7]. As for approximation, there is an algorithm to compute contraction sequences of approximate width for ordered structures [8]. It is still wide open whether there is an XP, let alone FPT, approximation algorithm in the general case. Regarding parameterized algorithms, some results are known with parameters that are still quite far from twin-width [3, 4]. Interestingly, the more general parameter flip-width introduced by Torunczyk [25] has an XP approximation algorithm but the status of FO model-checking for this parameter is unknown.

In this paper, we are interested in the structure of graphs of twin-width 1, and in the complexity of their recognition. Let us briefly recall the definition of k -contraction sequences, which are witnesses for twin-width at most k . Given a graph G , a contraction sequence of G starts with G , and consists in a sequence of identifications of pairs of vertices (contractions) recording “neighbourhood errors” as *red edges*, ending with a graph on a single vertex. Specifically, at each step, we create red edges from the contracted vertex to vertices that were not homogeneous to the pair. That is, vertices which were not both adjacent or non-adjacent to the pair. In particular, red edges remain red. Then, a k -contraction sequence is one where at each step the vertices have most k incident red edges.

Towards understanding graphs of twin-width one, it will be useful to look at the structure and behaviour of classes of graphs which are related. There is a rich literature on hereditary classes of graphs related to perfect graphs. It follows from the following observation and the strong perfect graph theorem [13] that twin-width 1 graphs are perfect graphs. It is then natural to wonder how they compare to other subclasses of perfect graphs.

► **Observation 1** (See [1, Lemma 2.3]). *Every cycle of length at least 5 has twin-width 2. This also implies that complements of cycles of length at least 5 have twin-width 2.*

A natural way of understanding classes of twin-width at most k is to interpret them as a generalization of cographs. Indeed, cographs are exactly the graphs which admit a sequence of twin identifications ending in a single vertex, that is, graphs of twin-width 0. Then, graphs of twin-width k correspond to relaxing the twin condition to allow for at most k “twin errors”.

We now compare twin-width 1 graphs with other generalisations of cographs. Cographs are the graphs of clique-width at most 2. In this direction, cographs are generalised by the class of distance-hereditary graphs which are themselves closely related to graphs of clique-width at most 3 [14]. Distance-hereditary (DH) graphs are not closed by complementation, and have a tree-like structure. They also have a characterisation by a sequential elimination of twins and pendant vertices. Cographs are permutation graphs, which is exactly the complementation-closed class of graphs whose edges can be transitively oriented [17]. Permutation graphs are

asteroidal triple-free (AT-free), as such, they are dominated by a path [15]. A caterpillar is an AT-free tree, the following result suggests a relation between AT-free graphs and twin-width 1.

► **Lemma 2** (Ahn, Hendrey, Kim, and Oum [2, Lemma 6.6]). *For a tree T , $\text{tw}(T) \leq 1$ if and only if T is a caterpillar.*

Cographs, distance-hereditary graphs, and permutation graphs can all be recognised in linear time. It is only natural to expect that the closely related class of twin-width 1 graphs can also be recognised efficiently.

Previous results on graphs of twin-width at most 1

The recognition algorithm for twin-width one given in [10] makes use of the relation between modules and twin-width, by observing that the twin-width of a graph can be determined by considering independently the subgraphs induced by modules (see Lemma 5). It is straightforward to deduce that the twin-width of a graph is the maximal twin-width over all prime nodes of its modular decomposition (defined in Section 2). Since this decomposition may be computed in linear time [24], one then only needs to recognise prime graphs of twin-width one. In their paper [10], the recognition is done by branching on valid sequences that have at most one red edge at any intermediate step of the sequence, after the observation that such sequences always exist. Furthermore, when the trigraph has a red edge, the only possible contractions in such a sequence involve at least one vertex of the red edge. Essentially, the complexity of their algorithm stems from the need to branch over all possible first contractions, to greedily simulate a contraction sequence for each possible first contraction, and the lack of an efficient way to detect eventual twins to be contracted at each step. In particular, a more explicit understanding of the structure of graphs of twin-width at most 1 seems to be required to avoid the enumeration of all pairs of possible first contractions.

Our results

We investigate the structure of graphs of twin-width at most 1, and uncover that it defines a relatively well-behaved hereditary class of graphs, which fits surprisingly well in the landscape of classical hereditary classes. Graphs of twin-width at most 1 were already known to have some form of linear structure, due to a bound on their linear rankwidth if they are prime (see Lemma 7 and [9]). We show that they are in fact contained in the class of permutation graphs, thus they admit an intersection model called “permutation diagram”, making it easier to reason on their structure. Furthermore, we show that there is a close relationship between permutation diagrams and 1-contraction sequences. We use these structural results to obtain a recursive decomposition theorem for prime graphs of twin-width at most 1. We then use it to devise a simple linear time algorithm to compute a 1-contraction sequence, or conclude that the graph has twin-width more than 1. The algorithm starts by computing a permutation diagram and a modular decomposition which can be done in time $O(n + m)$, where n is the number of vertices and m is the number of edges, and then it checks that the diagram respects the decomposition theorem in time $O(n)$.

Regarding the challenge of avoiding the enumeration of possible first contractions, we can significantly reduce the number of candidates by using the permutation diagram and related properties of contraction sequences. We consider a slightly different scheme to avoid this challenge. Instead of guessing the first contraction, we guess the vertex that will be contracted last. It turns out that, in well-behaved contraction sequences, such a vertex

holds a special position in permutation diagrams, reducing the number of candidates to 4. However, we do not exclude the possibility that the scheme of the previous algorithm can be implemented in linear time with some further arguments.

Interestingly, our structural analysis via the permutation diagrams allows us to avoid a characterisation by forbidden induced subgraphs. Moreover, we prove most structural tools inductively and in a unified way instead of proving the observations required for our algorithm separately.

In the full version, we also present some intermediate results on distance-hereditary graphs and split decompositions that turned out to be unnecessary for the recognition algorithm thanks to the use of permutation diagrams. We show that twin-width 1 trigraphs with a pendant red edge are distance-hereditary. We also show that distance-hereditary graphs have twin-width at most 2. Combining these results, we can compute an optimal contraction sequence for distance-hereditary graphs in linear time. We also obtain a simple inductive proof that all distance-hereditary AT-free graphs are permutation graphs, and a characterisation of the twin-width of a distance-hereditary graph by the structure of its split decomposition. This is a generalisation of the characterisation of twin-width 1 trees [2] mentioned above.

Distance-hereditary graphs are also related to the following infinite family of permutation graphs of twin-width 2: linking two obstructions to distance-hereditary graphs (domino, house, or gem) by a path of arbitrary length. In particular, this class of examples shows that a linear time algorithm for the recognition of twin-width 1 graphs cannot be deduced from the linear algorithm to find patterns in a permutation of Guillemot and Marx [21].

We also observe that bipartite graphs of twin-width 1 constitute a well-behaved subclass. Indeed, in their 1-contraction sequences, all trigraphs are bipartite. This is because there are no induced odd cycles of length more than 3, and triangles have at most one red edge from which we can deduce an original edge and complete it in a triangle of the starting graph (this is a particular case of Observation 14).

Perspectives

One may hope that understanding the structure of graphs of twin-width 1 well enough can guide a subsequent study of graphs of twin-width 2. We expect that this hereditary class can be characterised by a tree-like decomposition with a very constrained local structure. Our results also point toward the fact that split decompositions could be a convenient tool for the analysis of the structure of twin-width 2 graphs.

In another direction, one may wonder if it is possible to compute other values of the twin-width efficiently for permutation graphs. An FPT approximation algorithm is already known in this setting [11], although one may try to find tighter approximation bounds.

Organisation

We organise this paper as follows. In Section 2, we present some terminology and useful known results on twin-width, and on permutation graphs. In Section 3, we show that every graph of twin-width at most 1 is a permutation graph. In Section 4, we present a linear-time algorithm that recognises graphs of twin-width at most 1.

2 Preliminaries

In this paper, all graphs are finite and simple. For an integer i , we denote by $[i]$ the set of positive integers at most i . Note that if $i \leq 0$, then $[i]$ is an empty set. A subset $I \subseteq [i]$ is an *interval* of $[i]$ if there are integers $i_1, i_2 \in [i]$ such that $I = \{j : i_1 \leq j \leq i_2\}$.

Let G be a graph. We denote the complement graph by \overline{G} . For a vertex v of G , we denote by $N_G(v)$ the set of neighbours of v , and let $N_G[v] := N_G(v) \cup \{v\}$. Then, $\overline{N}_G(v) = V(G) \setminus N[v]$. For a set $X \subseteq V(G)$, let $N_G[X] := \bigcup_{v \in X} N_G[v]$ and let $N_G(X) := N_G[X] \setminus X$. Distinct vertices v and w of G are *twins in G* if $N_G(v) \setminus \{w\} = N_G(w) \setminus \{v\}$. For a set $X \subseteq V(G)$, we denote by $G[X]$ the subgraph of G induced by X . A *dominating set* of G is a set $D \subseteq V(G)$ such that $N_G[D] = V(G)$. A *dominating path* of G is a path whose vertex set is a dominating set of G . Three vertices form an *asteroidal triple* if each pair of vertices is connected by a path that does not dominate the third vertex. A graph is *asteroidal triple-free* (AT-free) if it contains no asteroidal triple. An *independent set* of a graph G is a set $I \subseteq V(G)$ such that G has no edge between two vertices in I .

A *module* of a graph G is a set $M \subseteq V(G)$ such that for every vertex $v \in V(G) \setminus M$, v is either adjacent to every vertex in M , or nonadjacent to every vertex in M . If M is not a module, we call *splitter* of M a vertex $v \in V(G) \setminus M$, that has both a neighbour and a non-neighbour in M . Note that M is a module exactly if it has no splitter. A module is *trivial* if it either has size at most 1, or is equal to $V(G)$. A graph is *prime with respect to modules*, or simply *prime*, if all of its modules are trivial. A *modular partition* of a graph G is a partition of $V(G)$ where each part is a module of G . For a modular partition $\mathcal{M} = (M_1, \dots, M_\ell)$ of a graph G , we denote by G/\mathcal{M} , the subgraph of G induced by a set containing exactly one vertex from each M_i . A module M is *strong* if it does not overlap with any other module, i.e. for every module M' , we have $M \subseteq M'$, or $M' \subseteq M$, or $M \cap M' = \emptyset$.

The *modular decomposition* of a graph G is a tree T corresponding to the recursive modular partition by the maximal strong modules. The leaves of the tree are vertices of G , and each internal node n has an associated quotient graph $Q(n)$ which encodes the adjacency relation between vertices of G introduced in subtrees below n . In particular, quotient graphs are induced subgraphs of G , and the modular decomposition of an induced subgraph of G can easily be deduced from the modular decomposition of G . There are three types of internal nodes: *series*, *parallel*, and *prime*, corresponding to cliques, independent sets and prime graphs (with respect to modules). Prime graphs are graphs whose modules are all trivial i.e. singletons and the complete vertex set. Series and parallel nodes are called *degenerate*, and two degenerate nodes of the same type may not be adjacent in T . The reason they are called degenerate is because any subset of vertices of a clique or an independent set is a module. Cographs are exactly the graphs whose modular decompositions have no prime nodes, their modular decomposition is usually called a cotree.

2.1 Permutation graphs

A graph G on n vertices is a *permutation graph* if there are two linear orderings $\sigma : V(G) \rightarrow [n]$ and $\tau : V(G) \rightarrow [n]$ such that two vertices u and v are adjacent in G if and only if $\{u, v\}$ is an inversion of $\sigma^{-1} \circ \tau$. We remark that $\sigma^{-1} \circ \tau$ and its inverse permutation $\tau^{-1} \circ \sigma$ have the same set of inversions. A *permutation diagram* of G with respect to σ and τ is a drawing of n line segments between two parallel lines ℓ_1 and ℓ_2 such that each of ℓ_1 and ℓ_2 has n distinct points, and for each $v \in V(G)$, there is a line segment between $\sigma(v)$ -th point of ℓ_1 and $\tau(v)$ -th point of ℓ_2 . Note that two vertices are adjacent in G if and only if their corresponding line segments intersect each other in the permutation diagram.

We denote by $G[\sigma, \tau]$ the graph on V where uv is an edge if and only if $\{u, v\}$ is an inversion of $\sigma^{-1} \circ \tau$. As pointed earlier, we have $G[\sigma, \tau] = G[\tau, \sigma]$. If $G = G[\sigma, \tau]$, we call (σ, τ) a *realiser* of G (as a permutation graph).

For σ an ordering of V , we define intervals of V as follows: for $\{i, i+1, \dots, j-1, j\}$ an interval of $[n]$, we have an interval of V for σ $\{\sigma^{-1}(i) = u, \dots, \sigma^{-1}(j) = v\}$ that we denote by $[u, v]_\sigma$. We extend the notation to open intervals similarly.

We say that $I \subseteq V$ is an *interval* of realiser (σ, τ) if it is an interval for σ or for τ . We also say that $u, v \in V(G_i)$ are *consecutive* for (σ, τ) if $\{u, v\}$ is an interval of (σ, τ) . Similarly, two intervals are *consecutive* if their union is an interval. We say that I is a *common interval* of realiser (σ, τ) if it is an interval for σ and for τ . A vertex v of G is *extremal* for realiser (σ, τ) (or equivalently the corresponding permutation diagram) if it is the first or last vertex of σ or τ .

It is known that for a prime permutation graph, its permutation diagram (or equivalently its realiser) is unique up to symmetry [18, 20]. More generally, the modular decomposition encodes all possible realisers. Furthermore, we have the following known observation which will be very convenient for our analysis.

► **Observation 3.** *If M is a common interval of a realiser of G , then M is a module of G . Conversely, if M is a strong module of G , then it is a common interval of all realisers.*

It will also be useful to note that extremal vertices of a prime permutation graph do not depend on the realiser. See [6, 12, 16, 22] for further details on this.

Moreover, for a permutation diagram of G with respect to linear orderings σ and τ of $V(G)$, if we reverse one of σ and τ , then we obtain a permutation diagram of \overline{G} . In other words, \overline{G} has a permutation diagram with respect to σ and $n+1-\tau$, or to $n+1-\sigma$ and τ .

Thus, a graph G is a permutation graph if and only if \overline{G} is a permutation graph.

2.2 Trigraphs and twin-width

A *trigraph* is a triple $H = (V(H), B(H), R(H))$ where $B(H)$ and $R(H)$ are disjoint sets of unordered pairs of $V(H)$. We denote by $E(H)$ the union of $B(H)$ and $R(H)$. The *edges* of H are the elements in $E(H)$. The *black edges* of H are the elements of $B(H)$, and the *red edges* of H are the elements of $R(H)$. We identify a graph $G = (V, E)$ with a trigraph (V, E, \emptyset) .

The *underlying graph* of a trigraph H is the graph $(V(H), E(H))$. We identify H with its underlying graph when we use standard graph-theoretic terms and notations. A *black neighbour* of v is a vertex u with $uv \in B(H)$ and a *red neighbour* of v is a vertex w with $vw \in R(H)$. The *red degree* of v is the number of red neighbours of v . For an integer d , a *d-trigraph* is a trigraph with maximum red degree at most d . For a set $X \subseteq V(H)$, we denote by $H - X$ the trigraph obtained from H by removing all vertices in X . If $X = \{v\}$, then we write $H - v$ for $H - X$.

For a trigraph H and distinct vertices u and w of H , we denote by $H/\{u, v\}$ the trigraph H' obtained from $H - \{u, v\}$ by adding a new vertex x such that for every vertex $y \in V(H) \setminus \{u, v\}$, the following hold:

- if y is a common black neighbour of u and v , then $xy \in B(H')$,
- if y is adjacent to none of u and v , then $xy \notin E(H')$, and
- otherwise, $xy \in R(H')$.

We say that H' is obtained by *contracting u and v* .

A *contraction sequence* of G is a sequence of trigraphs G_n, \dots, G_1 where $G_n = G$, G_1 as a single vertex, and G_{i-1} is obtained from G_i by contracting a pair of vertices. For an integer d , a contraction sequence G_n, \dots, G_1 is a *d -sequence* if for every $i \in [t]$, the maximum red-degree of G_i is at most d . The *twin-width* of G , denoted by $\text{tw}(G)$, is the minimum integer d such that there is a d -sequence of G .

The vertices of the trigraphs in the contraction sequence can be interpreted as a partition of $V(G)$, this leads to a very natural characterisation of red edges in a trigraph as pairs of part such that the relation between vertices in the two parts is not homogeneous (i.e. there is an edge and a non-edge).

We have the following observations from [11].

► **Observation 4** (Bonnet, Kim, Thomassé, and Watrigant [11]). *For a graph G , the twin-width of G is equal to that of \overline{G} , and every induced subgraph of G has twin-width at most $\text{tw}(G)$.*

► **Lemma 5** (Bonnet, Kim, Reinald, Thomassé, and Watrigant [10, Lemma 9]). *Let G be a graph and let $\mathcal{M} = (M_1, \dots, M_\ell)$ be a modular partition. Then*

$$\text{tw}(G) = \max \left\{ \text{tw}(G/\mathcal{M}), \max_{i \in [\ell]} \text{tw}(G[M_i]) \right\}.$$

We sketch the proof as this will be of importance throughout the paper.

Proof. A contraction in a module M does not create red edges incident to $V(G - M)$. Therefore, we can always first contract the induced subgraphs $G[M_i]$ in any order, and then contract G/\mathcal{M} once all subgraphs $G[M_i]$ have been contracted to single vertices. ◀

► **Corollary 6.** *The twin-width of a graph G is equal to the maximum twin-width of the quotient graphs of nodes of its modular decomposition.*

Proof. We may always contract leaf nodes of the modular decomposition via their optimal contraction sequence until the whole graph is reduced to a single vertex. This corresponds exactly to a recursive application of the above lemma with maximal strong modules. ◀

The above statements show that computing the twin-width of a graph reduces to the case of a prime graph. Thus, throughout this paper, we focus on prime graphs of twin-width at most 1. The following lemma on the structure of twin-width 1 graphs will be useful.

► **Lemma 7** (Bonnet, Kim, Reinald, Thomassé, and Watrigant [10, Lemma 31]). *Let G be a prime graph of twin-width 1 and let $G_n (= G), \dots, G_1$ be a 1-sequence of G . Then for every $i \in [n - 1] \setminus \{1\}$, the trigraph G_i has exactly one red edge.*

3 Permutation diagrams

In this section, we show that every graph of twin-width at most 1 is a permutation graph.

► **Theorem 8.** *Every graph of twin-width at most 1 is a permutation graph.*

It would be sufficient to prove that graphs of twin-width at most 1 are comparability graphs. Indeed, since twin-width is stable by complementation, this implies that graphs of twin-width at most 1 are also co-comparability graphs. We could then conclude because permutation graphs are exactly the graphs that are both comparability and co-comparability graphs [17]. To do this, one could make use of the following characterisation of comparability graphs via forbidden induced subgraphs given by Gallai [18] (see also [23]).

► **Theorem 9.** *A graph G is a comparability graph if and only if G does not contain any graph of Figure 3 as induced subgraph, and \overline{G} does not contain any graph of Figure 4 as induced subgraph.*

We instead present a constructive proof because it gives combinatorial insights on the relationship between contraction sequences and permutation diagrams that will be useful to find an efficient recognition algorithm. The trick is to decompose the graph by considering the contraction sequence backwards and to realise that vertices that are contracted together at some point in the contraction sequence should be consecutive or almost consecutive in a realiser.

The following recursive decomposition of twin-width 1 graphs is meant as a simple introduction to the technique. This decomposition will later be strengthened in Lemma 11 and Corollary 15. We abusively extend the terminology of universal vertex and isolated vertex to modules M such that replacing M by a single vertex makes it universal or isolated.

► **Lemma 10.** *The class \mathcal{G} of graphs of twin-width at most 1 satisfies the following recursive characterisation:*

$$\mathcal{G} = \left\{ G \mid \exists M \subset V(G), \begin{array}{l} G - M \in \mathcal{G}, G[M] \in \mathcal{G}, M \text{ is a module of } G, \\ M \text{ is universal or } M \text{ is isolated or} \\ G - M \text{ admits a 1-contraction sequence ending} \\ \text{with the (possibly red) edge } \{N(M), \overline{N}(M)\} \end{array} \right\}.$$

Proof. Consider a graph G such that there exists a module $M \subset V(G)$ such that $G - M \in \mathcal{G}$ and $G[M] \in \mathcal{G}$. If M is universal or isolated, $G - M$ is a module of G and, by Lemma 5, G is also a graph of \mathcal{G} . If $G - M$ admits a partial 1-contraction sequence π ending with edge $\{N(M), \overline{N}(M)\}$, then π is also a partial 1-contraction sequence of G , and can be extended to be 1-contraction sequence of G . Indeed, no red edge incident to M appears when applying π to G since pairs of contracted vertices are always either in $N(M)$ or $\overline{N}(M)$, we can then apply the 1-contraction sequence of M , and finally contract arbitrarily the resulting 3-vertex trigraph.

Conversely, consider a graph of \mathcal{G} . It admits a 1-contraction sequence with a 3-vertex trigraph G_3 . G_3 has at most one red edge ab (pick a, b arbitrarily in $V(G_3)$ if there is no red edge). We consider $v \in V(G_3) - \{a, b\}$. The set M of vertices of G that were contracted to form v is a module since there are no incident red edges in G_3 . $G - M \in \mathcal{G}$ and $G[M] \in \mathcal{G}$ since they are induced subgraphs of G . Let A and B be the sets of vertices that were contracted to form a and b , respectively. Since v has no red neighbour, it must be that $A \subseteq N(M)$ or $A \subseteq \overline{N}(M)$, similarly, $B \subseteq N(M)$ or $B \subseteq \overline{N}(M)$. We conclude that M is universal, isolated or $G - M$ admits a contraction sequence ending with the edge $\{N(M), \overline{N}(M)\}$. ◀

Given a permutation graph $G = (V, E)$ and $U \subseteq V$, we denote by $\sigma[U]$ and $\tau[U]$ the restrictions of σ and τ (re-numbered by $[[U]]$). They form a realiser of $G[U]$ as a permutation graph. We also use the notation $\sigma \cdot \tau$ to denote the concatenation of two orderings on disjoint domains (all elements of σ are before elements of τ , relative orders in σ and τ are preserved).

The following lemma implies Theorem 8.

► **Lemma 11.** *Let G be a graph of twin-width at most 1, and G_n, \dots, G_2, G_1 be a fixed 1-contraction sequence of G .*

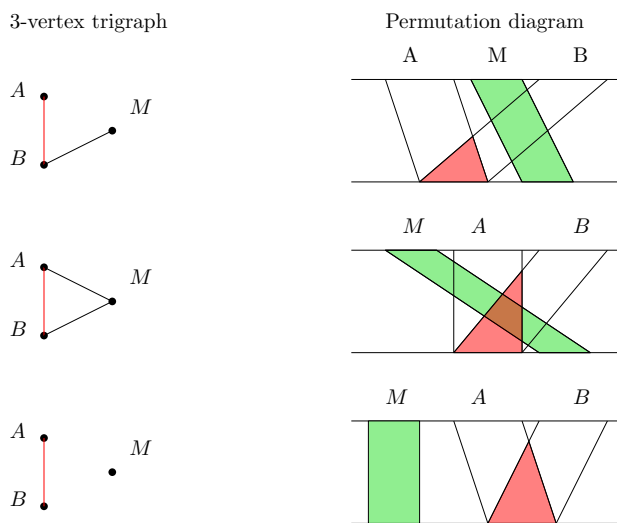
Then G is a permutation graph admitting a realiser (σ, τ) satisfying the following.

- *For every $x \in V(G_i)$, $\{v \in V(G) \mid v \in x\}$ is an interval of (σ, τ) .*

- For every $x, y \in V(G_i)$, if $xy \in R(G_i)$, then $\{v \in V(G) \mid v \in (x \cup y)\}$ is an interval on one of the orderings $\pi \in \{\sigma, \tau\}$, and both $\{v \in V(G) \mid v \in x\}$ and $\{v \in V(G) \mid v \in y\}$ are intervals on the other ordering.

Proof. We proceed by induction on the number of vertices of the graph by leveraging the recursive characterization given in Lemma 10.

The statement is trivial for graphs on at most 3 vertices. We now consider a graph $G \in \mathcal{G}$ on more than 3 vertices and its contraction sequence $G_n, \dots, G_3, G_2, G_1$. Let M be the set of vertices of G that were contracted into a vertex of G_3 not incident to a red edge in G_3 , M is a module of G . Restricting the sequence G_n, \dots, G_1 to $G - M$ and $G[M]$, respectively, yields a 1-contraction sequence for each of them. By induction hypothesis applied with these sequences, we obtain realisers σ_1, τ_1 and σ_2, τ_2 of $G - M$ and $G[M]$, respectively. The following cases are illustrated in Figure 1. If M is universal (resp. isolated), $\sigma_1 \cdot \sigma_2, \tau_2 \cdot \tau_1$ (resp. $\sigma_1 \cdot \sigma_2, \tau_1 \cdot \tau_2$) is a realiser of G . Otherwise, the contraction sequence $G_n - M, \dots, G_3 - M$ is such that $V(G_3 - M) = \{N(M), \overline{N}(M)\}$. In particular, we know from the induction hypothesis that $B = N(M)$ and $A = \overline{N}(M)$ are intervals of either σ_1 or τ_1 , as they correspond to the sets of vertices contracted into the last two vertices on the sequence. Without loss of generality, we assume they are intervals of σ_1 . We can now produce the realiser $\sigma_1[B] \cdot \sigma_2 \cdot \sigma_1[A], \tau_2 \cdot \tau_1$. In all cases, the realiser satisfies the desired properties: it suffices to observe that by construction we do not contract vertices from $G - M$ with vertices from $G[M]$ until the last two contractions, so we must check only the last two contractions which are on trigraphs of order at most 3 hence the properties are satisfied. ◀



■ **Figure 1** The different cases of the induction constructing the realiser.

We now move on to show related properties that will be useful in further understanding permutation diagrams and their relation to contraction sequences.

In the particular case of a prime graph, we deduce the following two corollaries of Lemma 11.

- **Corollary 12.** In a prime graph G of twin-width 1, the first contraction must involve two vertices that are consecutive for one ordering, and with exactly one vertex between them for the other ordering.

Proof. In G_{n-1} , let x', z be the vertices incident to the red edge (there is a red edge because the graph is prime, see Lemma 7) with x' resulting of the contraction of vertices x and y of G . $X' = \{x, y\}$ is an interval for one ordering σ of the realiser. z must be a splitter of X' because it is incident to the red edge, this implies that z is between x and y in the ordering τ . ◀

Unfortunately, there can be linearly many such pairs. On the other hand, there are only 4 extremal vertices in a prime permutation graph. Hence, the following corollary is useful for obtaining a linear time recognition algorithm.

► **Corollary 13.** *In a prime graph of twin-width 1, for any 1-contraction sequence, the last vertex to become incident to the red edge is an extremal vertex of the realiser.*

Proof. Due to the graph being prime, the last vertex of G to become incident to the red edge is also a vertex of G_3 and is adjacent to exactly one other vertex of G_3 . The other two vertices of G_3 are connected because prime graphs are connected, hence G_3 is isomorphic to P_3 whose vertices are all extremal. In particular, the last vertex incident to the red edge is extremal. ◀

► **Observation 14.** *If G_n, \dots, G_1 are the trigraphs of a 1-contraction sequence of G , their underlying graphs form a chain for the induced subgraph relation, i.e. G_{i-1} is an induced subgraph of G_i for $i \in [2, n]$.*

Proof. We describe how to construct the induced subgraph of G by picking a representative in G for each vertex in G_i .

Consider a vertex v of G_i , either it has only black incident edges and we may pick any vertex of G that was contracted into v , or it has exactly one incident red edge vw , in which case we can pick the endpoints of some edge of G that is between the vertices contracted into v and the vertices contracted into w .

This construction picks only one vertex of G per vertex of G_i because the red degree is at most one meaning we do not pick more than once. ◀

This property only holds for 1-contraction sequences (e.g. contracting non-adjacent vertices of a matching creates a path on 3 vertices which is not an induced subgraph). It allows to view the contraction sequence as a sequence of vertex deletions. In particular, underlying graphs of the trigraphs of a 1-contraction sequence are also permutation graphs and one of their permutation diagrams is the induced permutation diagram obtained by seeing G_i as an induced subgraph of G . Observe that this diagram preserves the relative order of vertices that are not deleted. This will be important for our inductive reasoning since we can keep the same diagram while decomposing the graph. We denote by $\sigma[G_i]$ the ordering induced by G_i seen as an induced subgraph of G .

► **Corollary 15.** *Let G be a graph and G_n, \dots, G_1 a 1-contraction sequence of G . The realiser constructed from G_n, \dots, G_1 using Lemma 11 has the property that, for every $i > 1$, the vertices being contracted from G_i to G_{i-1} are consecutive for $(\sigma[G_i], \tau[G_i])$, and endpoints of red edges are consecutive for $(\sigma[G_i], \tau[G_i])$.*

Conversely, for any realiser (σ', τ') , there exists a 1-contraction sequence G_n, \dots, G_1 such that for every $i > 1$ the contracted vertices of G_i are consecutive in $(\sigma'[G_i], \tau'[G_i])$ and for any $xy \in R(G_i)$, x and y are consecutive in $(\sigma'[G_i], \tau'[G_i])$.

Proof. Suppose we contract $a, b \in V(G_i)$ into vertex $c \in V(G_{i-1})$. Let C be the set of vertices of G contracted into c , and A, B those contracted into a, b . By Lemma 11, A, B and C are all intervals of (σ, τ) . Since $C = A \cup B$, we may assume without loss of generality they

are all intervals of σ . Hence, a and b are consecutive for $\sigma[G_i]$. Similarly, the set of vertices of G contracted into a fixed red edge is an interval of (σ, τ) . Since the sets of vertices of G incident to the red edge are also intervals, this means the endpoints of the red edge in G_i are consecutive.

Now, consider any realiser (σ', τ') for graph G , and let us show the second part of the lemma by induction on the modular decomposition of G . The result holds trivially for leaves of the decomposition. Let us then consider an internal node q of the decomposition of G and consider the subgraphs H_1, \dots, H_k of G induced by the children of q . Now, since each H_i is a strong module of G , they each form a common interval of (σ', τ') by Observation 3. They also have twin-width 1, so our induction hypothesis along with Lemma 5 yields that we can contract each child fully by respecting the consecutivity properties on $(\sigma'[H_i], \tau'[H_i])$, and thus on (σ', τ') . Then, if q is a degenerate node, we can always find a pair of consecutive twins and contract them. Otherwise, q is prime, and as such it admits a unique realiser (up to symmetry), which is given by Lemma 11, for which the contraction sequence satisfies the desired properties by the first part of the lemma. ◀

4 Linear-time recognition algorithm

A realiser (σ, τ) , if it exists, can be found in linear time [24]. From the Corollaries 6, 12 and 15, we deduce that the algorithm of [10] can be implemented in time $O(n^2 + m)$. Indeed, for every prime node of the modular decomposition, it suffices to compute σ and τ for its quotient graph. At this point, we have restricted the set of possible first contractions to the set all pairs of consecutive vertices, and we may then simulate contraction sequences efficiently using the fact that new vertices to be contracted can be found in constant time, since they are consecutive. In order to obtain a linear time algorithm, we will instead guess the end of the sequence, and try to extend the sequence from its end, see Figure 2.

► **Lemma 16.** *Let G be a prime permutation graph of twin-width 1, which admits a 1-contraction sequence where an extremal vertex s is last to become incident to a red edge. Let also $G^1 = G[N(s)]$ and $G^2 = G[\overline{N}(s)]$.*

Then, there is a module M of G^i for some i that has a unique splitter $s' \in V(G^{3-i})$, such that $G[M \cup \{s'\}]$ admits a 1-contraction sequence where s' is the last vertex to be incident to the red edge, where:

- (i) *either M is a pair of consecutive twins in G^i and there are no prime nodes in the modular decompositions of G^1 and G^2 ,*
- (ii) *or M corresponds to the subtree rooted at a prime node p in the modular decomposition, and p is the unique prime node in the modular decompositions of G^1 and G^2 .*

Moreover, let $k = |V(G) - (M \cup \{s'\})|$, there is an ordering $\pi : [k] \rightarrow V(G) - (M \cup \{s'\})$ such that for all $i \in [k]$, $M \cup \{s'\} \cup \pi([i])$ forms an interval of σ and two intervals of τ for a realiser (σ, τ) where σ is the linear ordering for which s is extremal.

Proof. Let (σ, τ) be a realiser of G and s be an extremal vertex for σ which is the last vertex incident to the red edge in some 1-contraction sequence of G . Such a vertex exists due to Corollary 13. Let $G^1 = G[N(s)]$ and $G^2 = G[\overline{N}(s)]$ and observe that since G is prime, G^1 and G^2 are not empty. As s is extremal for σ , adjacent to $V(G^1)$ and non-adjacent to $V(G^2)$, $V(G^1)$ and $V(G^2)$ must be disjoint intervals of τ .

▷ **Claim 17.** In any contraction sequence as considered, we only contract pairs of vertices of G^1 or pairs of vertices of G^2 until we reach the 3-vertex trigraph.

Proof. Due to the assumption on s being the last vertex incident to a red edge, we cannot contract a vertex of G^1 with a vertex of G^2 , unless G^1 and G^2 both have been contracted to a single vertex, because they are split by s . \triangleleft

▷ **Claim 18.** For any non-trivial module M of G^i there is a splitter $s' \in V(G^{3-i})$. Furthermore, for any splitter $s' \in V(G^{3-i})$ of M there are a, b in M such that $\sigma(a) < \sigma(s') < \sigma(b)$.

Proof. Indeed, M cannot be a module of G , as G is a prime graph, so it must have a splitter s' in $G - V(G^i)$. Also note that s is not a splitter of M by definition of G^i . Furthermore, since s' is a splitter it must be adjacent to some $a \in M$ and non-adjacent to some $b \in M$. Since $V(G^1)$ and $V(G^2)$ are intervals of τ , we get that $\sigma(a) < \sigma(s') < \sigma(b)$ or $\sigma(a) < \sigma(s') < \sigma(b)$ by definition of the permutation diagram. \triangleleft

The following observation will allow to simplify the analysis.

► **Observation 19.** *If X induces a prime subgraph of H , and $t \in V(H) \setminus X$ is a splitter of X , then we can both extract a pair of vertices of X that are adjacent and split by t , and a pair of vertices of X that are non-adjacent and split by t .*

Proof. Consider the cut $C = (X \cap N(t), X - N(t))$ in $H[X]$, since $H[X]$ is prime both $H[X]$ and $\overline{H[X]}$ are connected, yielding both an edge and a non-edge across C . \blacktriangleleft

▷ **Claim 20.** There cannot be disjoint non-trivial strong modules in G^i .

Proof. We proceed by contradiction and consider M, M' two maximal disjoint non-trivial strong modules of G^i , without loss of generality $i = 1$. We first show that, in G , M and M' must have distinct splitters from $V(G^2)$, and then conclude that this contradicts the assumption that G has a 1-contraction sequence respecting Claim 17.

Using the fact that $V(G^1)$ and $V(G^2)$ are intervals of τ and the common interval characterisation of strong modules of G^1 given by Observation 3, we deduce that a splitter t , which necessarily belongs to $V(G^2)$, cannot split both M and M' . Indeed, we have $\sigma(M) < \sigma(M')$ or $\sigma(M') < \sigma(M)$, so if t is a splitter of M there exist $a, b \in M$ such that $\sigma(a) < \sigma(t) < \sigma(b)$ as seen in the previous claim. In particular, t is not a splitter of M' .

Now because M and M' are non-trivial modules of G^1 , they each have a splitter in G^2 . Let t and t' be such splitters. By maximality, M and M' correspond to subtrees rooted at siblings of the same node n in the modular decomposition of G^1 . Let us first consider the case when n is a prime node. In this case, Lemma 7 guarantees a red edge for $Q(n)$, and the two splitters t, t' forbid the existence of a contraction sequence with consecutivity properties as guaranteed by Lemma 11 and Corollary 15.

We may now assume that n is a degenerate node, and exhibit an induced subgraph H of G that does not admit a 1-contraction sequence respecting our assumptions. This will contradict the existence of such a sequence for the whole G , as its restriction to H would also be valid (as in Observation 4). The graph H consists of splitters t, t' , as well as two pairs of twins $a, b \in M$ and $a', b' \in M'$ distinguished by t, t' respectively. Then, vertices a, b, a', b' induce a cograph whose corresponding cotree is a complete binary tree of depth 2. In other words, (a, b) and (a', b') are present if and only if $\{a, b\}$ is adjacent to $\{a', b'\}$.

We first reduce M and M' so as to induce exactly the quotient graph of their corresponding node in the modular decomposition (recall that this is well defined because they are strong) by picking one vertex in each subtree below this node. We may ensure that t and t' still split this subset of vertices (e.g. by picking extremal vertices of the interval of σ corresponding to M and M' , recalling Observation 3).

Now we observe that if M (resp. M') has a degenerate root node, it is of the opposite type of the node n , and any pair of vertices of M (resp. M') that is split by t (resp. t') can be taken for our subgraph H . Otherwise, M (resp. M') has a prime root node, and we apply Observation 19 to obtain a pair of vertices that is split and has the required adjacency relation.

Let us now show that H does not admit a 1-contraction sequence satisfying our assumptions. Indeed, t and t' may only be contracted together, because they are the only vertices in $V(G^2)$ of our subgraph (Claim 17), but they are split by at least two vertices in $V(G^1)$. Now, any order of contractions on a, b, a', b' will create two red edges. More precisely, contracting a vertex from each nontrivial module creates two red edges, unless one module was already reduced to a single vertex, in which case it has a red edge to its splitter from $V(G^2)$. This will inevitably contradict Lemma 7. \triangleleft

\triangleright Claim 21. If G^i has a prime node p in its modular decomposition, the first red edge must appear in the module M corresponding to the subtree rooted at p . In this case, M has exactly one splitter s' from G^{3-i} and s' should not be incident to a red edge until M is contracted to a single vertex.

Proof. Let p be a prime node of the modular decomposition of G^i and M the module corresponding to the subtree rooted at p . Without loss of generality assume $i = 1$. We first make the observation that, since there cannot be two disjoint nontrivial strong modules in G^1 by Claim 20, and since prime graphs have at least 4 vertices, there are vertices introduced by node p .

We first consider the case when a red edge incident to a vertex of G^2 is created before we fully contract M . This red edge has to remain incident to a vertex of G^2 due to Claim 17. Contracting a vertex introduced in prime node p will create another red edge with both endpoints in G^1 , which is thus different and contradicts Lemma 7.

We now deal with the case where no red edge incident to a vertex of G^2 appears before we fully contract M . Assume the first red edge does not have both endpoints in M . Observe that the first contraction cannot be between vertices that are both not in M but have a different adjacency to it because M is a module on at least 4 vertices, and this would create a red edge to all of them. Observe also that if the first contraction had involved a vertex of M , there would also be a red edge with both endpoints in M . Indeed, on the one hand, if both contracted vertices were in M , red edges with both endpoints in G^i would have both endpoints in M because it is a module. On the other hand, M is also a strong module with a prime node at its root, so all of its vertices have a mixed adjacency to the rest of M , contrary to vertices outside M . We conclude that the first contraction is between two vertices outside of M , so the corresponding red edge has both endpoints outside of M . This has the following implications: both contracted vertices have an homogeneous adjacency to M , but they also have a unique splitter in G^1 , this implies we cannot contract the red edge before contracting the subgraph corresponding to p . Indeed, the two vertices incident to the red edge must have opposite adjacency relations with respect to M . Since contracting M will force the creation of a second red edge, this contradicts Lemma 7.

M is a non-trivial strong module of G^1 so it has at least one splitter in G^2 . From Lemma 7 and Claim 17, it follows that a red edge incident to this splitter cannot appear before M is contracted to a single vertex. Finally, only one red edge may appear when M is contracted to a single vertex meaning the splitter must be unique. \triangleleft

In particular, this shows that only one of G^1 and G^2 can have a prime node in their modular decomposition. We can moreover deduce that if there is a prime node in G^i , it must be unique. Indeed, a red edge has to appear in some M corresponding to the subtree of the

prime node that is deepest in the modular decomposition of G^i . At any step of the sequence, M contains a red edge, until it is a single vertex with a red edge towards its splitter. Then, vertices introduced in any higher prime node cannot be contracted on such red edges without producing an additional red edge, which contradicts Lemma 7.

We are now ready to define M . Consider the first red edge appearing between $X_1 \subseteq V(G^1)$ and $X_2 \subseteq V(G^2)$ in a 1-contraction sequence as considered, and assume without loss of generality that it stems from a contraction in G^1 . Then, X_2 can only consist in a single vertex s' due to Claim 17, Lemma 7, and the fact that this is the first red edge with an endpoint in both G^1 and G^2 . Then, X_1 is included in a module of G^1 which admits as unique splitter s' . In the case when there is no prime node, X_1 induces a cograph, so we may then take M to be any pair of consecutive twins $a, b \in X_1$, which are also consecutive twins in G^1 by Corollary 15, distinguished by s' (because G is prime and s' is the only splitter of X_1).

Otherwise, we can take M and s' as guaranteed by Claim 21 on G^1 and G^2 . The restriction of our contraction sequence to $G[M \cup s']$ yields a 1-contraction sequence in which s' becomes incident to a red edge last, as desired.

Having defined M , we are ready to show the following.

▷ **Claim 22.** There exists a 1-contraction sequence of G that starts by contracting only vertices of M until it is reduced to a single vertex. In particular, this creates a red edge towards s' at the last contraction.

Proof. We can extend the contraction sequence that contracts M first into a sequence contracting the smallest interval I of $\sigma[V(G^1)]$ containing X_1 because X_1 is not split by any splitter from G^2 other than s' , and it does not have disjoint non trivial strong modules. After contraction of M to a single vertex, the remaining vertices of I induce a cograph that is not split by any vertex of G^2 other than s' . We can then proceed with the rest of the contraction of G simply by contracting remaining vertices according to our initial contraction sequence. ◁

At this point, in both cases, we know M forms a common interval for $(\sigma[V(G^1)], \tau[V(G^1)])$, and that there is some 1-sequence for G that first contracts M and in which s is the last vertex to be incident to a red edge. We deduce that $M \cup \{s'\}$ satisfies the setting of Corollary 15, and any further contractions must be consecutive to M due to Corollary 15, ensuring the existence of π . ◀

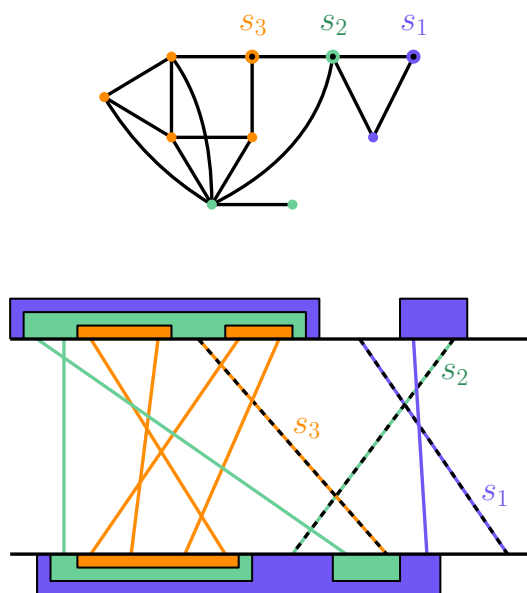
The crucial observation now is that π can easily be computed greedily using the realiser.

► **Lemma 23.** *Given a permutation diagram of a prime permutation graph G and an extremal vertex s of G , we can produce in time linear in the number of vertices of G a 1-contraction sequence such that s is the last vertex of G to be incident to the red edge, or conclude that no such sequence exists.*

Proof. This is essentially an implementation of the result of Lemma 16, we reuse similar notations and provide an illustration in Figure 2. The algorithm works as follows.

We initialise indices at the endpoints of the intervals A and B of τ that are split by s , and at the endpoints of the interval $C = V(G) - \{s\}$ of σ . We add s to the global list Π and mark it as a splitter.

While there is one endpoint v of C that is also an endpoint of A or an endpoint of B , we add v to the global list Π and update indices describing A, B , and C in accordance to the removal of v . We call such a vertex v *doubly extremal*.



■ **Figure 2** An example of a twin-width 1 graph and its permutation diagram. Shown here is the decomposition of Lemma 23 starting from extremal vertex s_1 . The segments of splitters s_1, s_2, s_3 are represented in dashed blue, green and orange respectively. Then, intervals of the corresponding colour are the intervals A, B, C produced during the iteration in which the splitter is considered. A and B induce subgraphs G^i from Lemma 16. Then, segments of the same colour as a splitter correspond to doubly extremal vertices eliminated in the corresponding step.

If the intervals have become empty, return Π , otherwise, if either A or B contains a single vertex, we set s' to be this vertex and M to be the other interval, and apply the algorithm recursively to $G[M \cup \{s'\}]$ with s' as the last vertex incident to a red edge. Finally, if both A and B contain more than one vertex, we reject.

We can use marked vertices to deduce a contraction sequence: we iterate through Π , starting from the last vertices added. When reaching a marked vertex, we contract the red edge. Otherwise, we contract the current vertex to the vertex of the red edge with the same adjacency to the next marked vertex.

If the algorithm does not reject, let us show by induction on the recursive calls that the list Π , along with its marked vertices, describes a valid order of contractions for G . First, we clarify the starting red edge that we consider in the part of the contraction sequence built at this step of the recursion. In the case when we emptied A and B (so M and s' are not defined), take the last vertex in A and the last vertex in B as a starting red edge. We move to the case where M and s' are defined. Here, by induction hypothesis, the algorithm finds a 1-contraction sequence with s' as the last vertex incident to a red edge for $G[M \cup \{s'\}]$. This contraction sequence for $G[M \cup \{s'\}]$ gives a partial 1-contraction sequence for G (see Observation 4) yielding a red edge between M and s' , with all other vertices of G not contracted yet.

We now check that each vertex added to Π at this step can be contracted in the order described by Π . This is the case because the vertex does not split the vertices of A contracted before it, nor the vertices of B contracted before it. By contracting it to the vertex of the red edge with the same adjacency to s , we create no additional red edge. In particular, s is still not incident to a red edge. Once we contracted all vertices of A and B , the red edge can be safely contracted. Indeed, s is the unique splitter of $A \cup B$.

If the algorithm rejects, it contradicts the existence of π guaranteed by Lemma 16. Indeed, if for all i , the $M \cup \{s'\} \cup \pi([i])$ are intervals as claimed, it must be that for each i , $\pi(i)$ is extremal on $\sigma[M \cup \{s'\} \cup \pi([i])]$ and on $\tau[(M \cup \{s'\} \cup \pi([i])) \cap V(G^j)]$ for some $j \in \{1, 2\}$. In particular, all vertices that are not in $M \cup \{s'\}$ will become doubly extremal if π exists. Note that this is independent of the order in which they are contracted. Indeed, a vertex remains (doubly) extremal while we delete other vertices.

We conclude that G has twin-width more than 1. ◀

► **Theorem 24.** *We can decide in linear time if a given graph G has twin-width at most 1.*

Proof. We first compute a modular decomposition of G and realisers for the quotient graphs of prime nodes [24]¹, or conclude that G is not a permutation graph and therefore not a graph of twin-width 1. It is well-known that the total size of the quotient graphs of prime nodes is linear in the size of the original graph.

We then check all prime nodes with corresponding quotient graph H as follows:

Guess an extremal vertex s (out of at most 4) that can be the last vertex incident to the red edge in a 1-contraction sequence of H , and then apply the recursive algorithm of Lemma 23.

If all prime nodes have twin-width 1, we obtained a contraction sequence for all of them and they can be combined to form a contraction sequence of G using Corollary 6.

All of the above procedures can be implemented to run in linear time. ◀

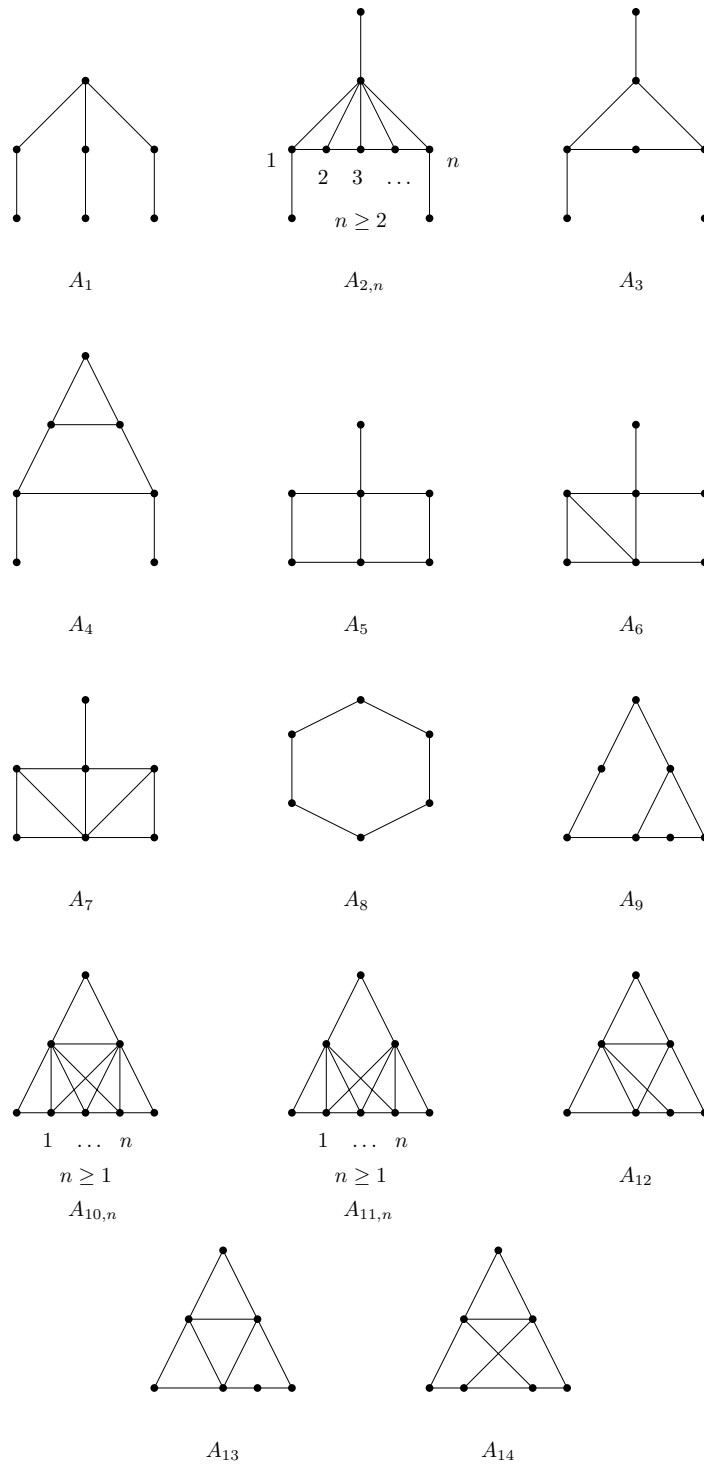
References

- 1 Jungho Ahn, Debsoumya Chakraborti, Kevin Hendrey, and Sang-il Oum. Twin-width of subdivisions of multigraphs, 2024. doi:10.48550/arXiv.2306.05334.
- 2 Jungho Ahn, Kevin Hendrey, Donggyu Kim, and Sang-il Oum. Bounds for the twin-width of graphs. *SIAM J. Discret. Math.*, 36(3):2352–2366, 2022. doi:10.1137/21m1452834.
- 3 Jakub Balabán, Robert Ganian, and Mathis Rocton. Computing twin-width parameterized by the feedback edge number. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshantov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPICs*, pages 7:1–7:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.STACS.2024.7.
- 4 Jakub Balabán, Robert Ganian, and Mathis Rocton. Twin-width meets feedback edges and vertex integrity. *CoRR*, abs/2407.15514, 2024. doi:10.48550/arXiv.2407.15514.
- 5 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is np-complete. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.18.
- 6 Anne Bergeron, Cédric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. *SIAM J. Discret. Math.*, 22(3):1022–1039, 2008. doi:10.1137/060651331.
- 7 Benjamin Bergougnoux, Jakub Gajarský, Grzegorz Guspiel, Petr Hlinený, Filip Pokrývka, and Marek Sokolowski. Sparse graphs of twin-width 2 have bounded tree-width. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPICs*, pages 11:1–11:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ISAAC.2023.11.

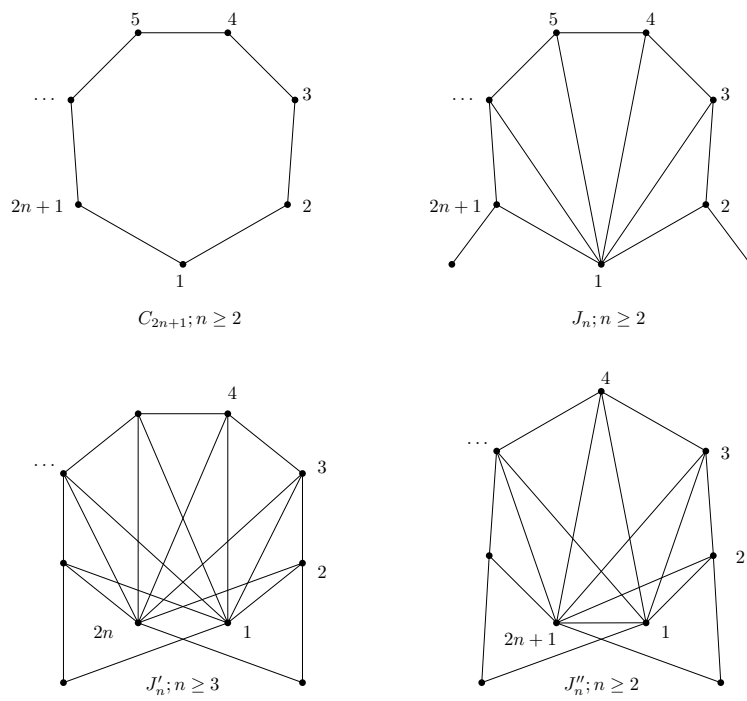
¹ In fact, it would be more reasonable to compute a permutation diagram and deduce a modular decomposition via common intervals for a practical implementation.

- 8 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. *J. ACM*, 71(3):21, 2024. doi:10.1145/3651151.
- 9 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. Twin-width VI: the lens of contraction sequences. 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 1036–1056. SIAM, 2022. doi:10.1137/1.9781611977073.45.
- 10 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. *Algorithmica*, 84(11):3300–3337, 2022. doi:10.1007/s00453-022-00965-5.
- 11 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: Tractable FO model checking. *J. ACM*, 69(1):Art. 3, 46, 2022. doi:10.1145/3486655.
- 12 Christian Capelle, Michel Habib, and Fabien de Montgolfier. Graph decompositions and-factorizing permutations. *Discret. Math. Theor. Comput. Sci.*, 5(1):55–70, 2002. doi:10.46298/DMTCS.298.
- 13 Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, July 2006. doi:10.4007/annals.2006.164.51.
- 14 Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce A. Reed, and Udi Rotics. Polynomial-time recognition of clique-width ≤ 3 graphs. *Discret. Appl. Math.*, 160(6):834–865, 2012. doi:10.1016/J.DAM.2011.03.020.
- 15 Derek G Corneil, Stephan Olariu, and Lorna Stewart. Asteroidal triple-free graphs. *SIAM Journal on Discrete Mathematics*, 10(3):399–430, 1997. doi:10.1137/S0895480193250125.
- 16 Fabien de Montgolfier. *Décomposition modulaire des graphes. Théorie, extension et algorithmes. (Graph Modular Decomposition. Theory, extension and algorithms)*. PhD thesis, Montpellier 2 University, France, 2003. URL: <https://tel.archives-ouvertes.fr/tel-03711558>.
- 17 Ben Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600–610, July 1941.
- 18 Tibor Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18:25–66, 1967. German.
- 19 Colin Geniet and Stéphan Thomassé. First order logic and twin-width in tournaments. 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 53:1–53:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ESA.2023.53.
- 20 M.C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.
- 21 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 82–101. SIAM, 2014. doi:10.1137/1.9781611973402.7.
- 22 Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Comput. Sci. Rev.*, 4(1):41–59, 2010. doi:10.1016/J.COSREV.2010.01.001.
- 23 Ekkehard Köhler. *Graphs Without Asteroidal Triples*. PhD thesis, Technischen Universität Berlin, 1999.
- 24 Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discret. Math.*, 201(1-3):189–241, 1999. doi:10.1016/S0012-365X(98)00319-7.
- 25 Szymon Torunczyk. Flip-width: Cops and robber on dense graphs. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 663–700. IEEE, 2023. doi:10.1109/FOCS57990.2023.00045.

A Appendix



■ **Figure 3** Minimal graphs containing an asteroid triple.



■ **Figure 4** Complements of minimal graphs containing a $(2k + 1)$ -asteroid for $k > 1$.

Faster Edge Coloring by Partition Sieving

Shyan Akmal   

INSAIT, Sofia University “St. Kliment Ohridski”, Bulgaria

Tomohiro Koana  

Utrecht University, The Netherlands

Research Institute for Mathematical Sciences, Kyoto University, Japan

Abstract

In the EDGE COLORING problem, we are given an undirected graph G with n vertices and m edges, and are tasked with finding the smallest positive integer k so that the edges of G can be assigned k colors in such a way that no two edges incident to the same vertex are assigned the same color. EDGE COLORING is a classic NP-hard problem, and so significant research has gone into designing fast *exponential-time* algorithms for solving EDGE COLORING and its variants exactly. Prior work showed that EDGE COLORING can be solved in $2^m \text{poly}(n)$ time and polynomial space, and in graphs with average degree d in $2^{(1-\varepsilon_d)m} \text{poly}(n)$ time and exponential space, where $\varepsilon_d = (1/d)^{\Theta(d^3)}$.

We present an algorithm that solves EDGE COLORING in $2^{m-3m/5} \text{poly}(n)$ time and polynomial space. Our result is the first algorithm for this problem which simultaneously runs in faster than $2^m \text{poly}(m)$ time and uses only polynomial space. In graphs of average degree d , our algorithm runs in $2^{(1-6/(5d))m} \text{poly}(n)$ time, which has far better dependence in d than previous results. We also consider a generalization of EDGE COLORING called LIST EDGE COLORING, where each edge e in the input graph comes with a list $L_e \subseteq \{1, \dots, k\}$ of colors, and we must determine whether we can assign each edge a color from its list so that no two edges incident to the same vertex receive the same color. We show that this problem can be solved in $2^{(1-6/(5k))m} \text{poly}(n)$ time and polynomial space. The previous best algorithm for LIST EDGE COLORING took $2^m \text{poly}(n)$ time and space.

Our algorithms are algebraic, and work by constructing a special polynomial P based off the input graph that contains a multilinear monomial (i.e., a monomial where every variable has degree at most one) if and only if the answer to the LIST EDGE COLORING problem on the input graph is YES. We then solve the problem by detecting multilinear monomials in P . Previous work also employed such monomial detection techniques to solve EDGE COLORING. We obtain faster algorithms both by carefully constructing our polynomial P , and by improving the runtimes for certain structured monomial detection problems using a technique we call partition sieving.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Coloring, Edge coloring, Chromatic index, Matroid, Pfaffian, Algebraic algorithm

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.7

Related Version *Full Version:* <https://arxiv.org/abs/2501.05570>

Funding *Shyan Akmal:* Partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

Tomohiro Koana: Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (project CRACKNP under grant agreement No. 853234). Also supported by JSPS KAKENHI Grant Numbers JP20H05967.

Acknowledgements We thank the anonymous reviewers for helpful feedback on this work.

1 Introduction

Coloring graphs is a rich area of research in graph theory and computer science. Given a graph G , a *proper vertex coloring* of G is an assignment of colors to its nodes such that no two adjacent vertices receive the same color. In the VERTEX COLORING problem, we are given an undirected graph G on n vertices, and are tasked with computing the smallest positive



© Shyan Akmal and Tomohiro Koana;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 7; pp. 7:1–7:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



integer k such that G admits a proper vertex coloring using k colors. VERTEX COLORING is a classic combinatorial problem, NP-hard to solve even approximately [27]. An influential line of research has worked on designing faster and faster exponential-time algorithms for this problem [24, 17, 13, 8], culminating in an algorithm solving VERTEX COLORING in $O^*(2^n)$ time and space [11], where throughout we write $O^*(f(n))$ as shorthand for $f(n) \text{ poly}(n)$.

In this paper, we study exact algorithms for the closely related EDGE COLORING problem. Given a graph G , a *proper edge coloring* of G is an assignment of colors to its edges such that no two edges incident to a common vertex receive the same color. The smallest positive integer k such that G admits a proper edge coloring using k colors is the *chromatic index* of G , denoted by $\chi'(G)$. In the EDGE COLORING problem, we are given an undirected graph G on n vertices and m edges, and are tasked with computing $\chi'(G)$.

Let Δ denote the maximum degree of the input graph G . For each vertex v in G , a proper edge coloring of G must assign distinct colors to the edges incident to v . Thus $\chi'(G) \geq \Delta$. A classic result in graph theory known as Vizing's theorem proves that in fact $\chi'(G) \leq \Delta + 1$, so that the simple Δ lower bound is close to the truth. Moreover, a proper edge coloring of G using $(\Delta + 1)$ colors can be found in near-linear time [2]. Consequently, solving EDGE COLORING amounts to distinguishing between the cases $\chi'(G) = \Delta$ and $\chi'(G) = \Delta + 1$. Despite this powerful structural result, the EDGE COLORING problem is NP-hard just like VERTEX COLORING, even for graphs where all vertices have degree exactly $\Delta = 3$ [21].

In recent years, researchers have made major strides in our knowledge of algorithms for *approximate* variants of EDGE COLORING in the distributed [19, 5, 3] and dynamic settings [15, 14, 6]. In comparison, much less is known about the exponential-time complexity for solving the EDGE COLORING problem *exactly*.

A simple way to solve EDGE COLORING is by reduction to VERTEX COLORING. Given the input graph G , we can construct in polynomial time the *line graph* $L(G)$ whose nodes are edges of G , and whose nodes are adjacent if the corresponding edges in G are incident to a common vertex. By construction, the solution to VERTEX COLORING on $L(G)$ gives the solution to EDGE COLORING on G . If G has m edges and maximum degree Δ , then $L(G)$ has m vertices and maximum degree $(2\Delta - 1)$. Using the aforementioned $O^*(2^n)$ time and space algorithm for VERTEX COLORING, this immediately implies a $O^*(2^m)$ time and space algorithm for EDGE COLORING. In graphs with maximum degree Δ , VERTEX COLORING can be solved in $O^*(2^{(1-\varepsilon)n})$ time and space, where $\varepsilon = (1/2)^{\Theta(\Delta)}$ [9]. By applying the above reduction, EDGE COLORING can similarly be solved in $O^*(2^{(1-\varepsilon)m})$ time and space, for $\varepsilon = (1/2)^{\Theta(\Delta)}$.

Line graphs are much more structured objects than generic undirected graphs (for example, there are small patterns that line graphs cannot contain as induced subgraphs [4, Theorem 3]), so one might hope to solve EDGE COLORING faster without relying on a black-box reduction to VERTEX COLORING. Nonetheless, there is only one result from previous work which solves EDGE COLORING on general undirected graphs without using this reduction. Specifically, [10, Section 5.6] shows that EDGE COLORING can be solved in $O^*(2^m)$ time and polynomial space. In comparison, it remains open whether VERTEX COLORING can be solved in $O^*(2^n)$ time using only polynomial space.

In summary, algorithms from previous work can solve EDGE COLORING in

- $O^*(2^m)$ time using polynomial space, or in
- faster than $O^*(2^m)$ time in bounded degree graphs, using exponential space.

Given this state of affairs, it is natural to ask:

Can EDGE COLORING be solved in faster than $O^(2^m)$ time, using only polynomial space?*

We answer this question affirmatively by proving the following result.

► **Theorem 1.1.** *There is a randomized algorithm which solves EDGE COLORING with high probability and one-sided error in $O^*(2^{m-3n/5})$ time and polynomial space.*

If the input graph has average degree d , then $m = dn/2$, so Theorem 1.1 equivalently states that EDGE COLORING can be solved in $O^*(2^{(1-\varepsilon)m})$ time for $\varepsilon = \frac{6}{5d}$. In comparison, the current fastest algorithm for VERTEX COLORING on graphs with average degree d takes $O^*(2^{(1-\delta)n})$ time, where $\delta = (1/d)^{\Theta(d^3)}$ [18, Lemma 4.4 and Theorem 5.4]. Prior to our work, the fastest algorithm for EDGE COLORING in the special case of regular graphs (i.e., graphs where all vertices have the same degree) took $O^*(2^{m-n/2})$ time [10, Theorem 6]. The regularity assumption simplifies the EDGE COLORING problem significantly, as in this case the problem reduces to finding a collection of $(\Delta - 1)$ mutually disjoint perfect matchings, which together consist of only $(m - n/2)$ edges. Theorem 1.1 improves upon this runtime with an algorithm that succeeds on *all* undirected graphs, not just regular graphs.

In the case of regular graphs, we obtain improvements beyond Theorem 1.1 when the graphs have high degree. We say a graph is d -regular if all its nodes have degree d . Given a positive integer k , let

$$H_k = \sum_{j=1}^k \frac{1}{j} \tag{1}$$

denote the k^{th} Harmonic number. We prove the following result.

► **Theorem 1.2.** *For each integer $d \geq 6$, there is a randomized algorithm that solves EDGE COLORING with high probability and one-sided error on d -regular graphs in $O^*(2^{m-\alpha_d n})$ time and polynomial space, for $\alpha_d = 1 - H_{d+1}/(d+1)$.*

For example for $d = 6$, Theorem 1.2 shows that we can solve EDGE COLORING in 6-regular graphs in $O^*(2^{m-0.62n})$ time, slightly faster than the runtime presented in Theorem 1.1. The savings in the exponent are better for larger d , approaching a runtime of $O^*(2^{m-n})$ as the degree d grows since we have $\alpha_d = 1 - \Theta((\log d)/d)$.

We also consider a generalization of EDGE COLORING called LIST EDGE COLORING. In this problem, we are given an undirected graph G with m edges as before, together with a positive integer k and a list of possible colors $L_e \subseteq \{1, \dots, k\}$ for each edge e . We are tasked with determining whether G admits a proper edge coloring, which assigns each edge e a color from the list L_e . If we set $k = \Delta$ to be the maximum degree of G and set $L_e = \{1, \dots, k\}$ for each edge e , we recover the standard EDGE COLORING problem. Using more sophisticated arguments, we generalize Theorem 1.1 to the LIST EDGE COLORING problem.

► **Theorem 1.3.** *There is a randomized algorithm that solves LIST EDGE COLORING with high probability and one-sided error in $O^*(2^{m-3n/5})$ time and polynomial space.*

Prior to our work, no algorithm was known for LIST EDGE COLORING which simultaneously ran in faster than $O^*(2^m)$ time and used polynomial space.

1.1 Our Techniques

Our algorithms for EDGE COLORING and LIST EDGE COLORING are algebraic, and involve reducing these problems to certain *monomial detection* tasks, by now a common paradigm in graph algorithms. To achieve the $O^*(2^{m-3n/5})$ runtime and polynomial space bounds for these problems, we combine and improve techniques in polynomial sieving, matroid constructions, and enumeration using matrices.

7:4 Faster Edge Coloring by Partition Sieving

The main technical ingredient in our work builds off the *determinantal sieving* technique introduced in [16]. Given a homogeneous multivariate polynomial $P(X)$ of degree k and a linear matroid \mathcal{M} on X of rank k , this technique allows one to test in $O^*(2^k)$ time whether $P(X)$ contains a multilinear monomial (i.e., a monomial where each variable has degree at most one) that forms a basis in \mathcal{M} . We recall the formal definition of matroids in Section 2. For the purpose of this overview, a matroid \mathcal{M} is a family of subsets of variables of P satisfying certain properties, and a basis in \mathcal{M} is a set in this family whose size equals the rank k . In our applications, we construct P to enumerate certain structures in the input graph. Finding a multilinear monomial in P corresponds to identifying a particularly nice structure in the graph (e.g., a solution to the EDGE COLORING problem). Determinantal sieving is useful because one can pick \mathcal{M} in such a way that identifying a monomial in P that forms a basis in \mathcal{M} recovers extra information about the relevant structures in the graph.

We build off determinantal sieving and introduce the *partition sieving* method. This technique involves a *partition matroid* \mathcal{D} , which is a simple type of matroid defined using an underlying partition of the variable set of P . Given a partition matroid \mathcal{D} whose partition has p parts, and a polynomial P *compatible* with \mathcal{D} (“compatible” is a technical condition defined in Section 3 – intuitively, it requires that monomials in P have degrees respecting the partition defining \mathcal{D}), we can test whether P contains a multilinear monomial forming a basis in \mathcal{D} in $O^*(2^{k-p})$ time and polynomial space. This offers an exponential speed-up over determinantal sieving. To apply partition sieving, we need to carefully design both the polynomial P and the partition matroid \mathcal{D} to be compatible in our technical framework.

We design the polynomial P using the notion of the Pfaffian of a matrix. If A is a symbolic skew-symmetric adjacency matrix of G , then its Pfaffian $\text{Pf } A$ is a polynomial that enumerates the perfect matchings in G . The EDGE COLORING problem may be rephrased as asking whether the edge set of G can be partitioned into a collection of k matchings. One can design a polynomial that enumerates k matchings by computing a product of k Pfaffians of adjacency matrices. However, this simple construction does not meet the technical conditions we need to employ partition sieving. Instead we utilize the Ishikawa-Wakayama formula [22], a convolution identity for Pfaffians and determinants, to design a polynomial that enumerates matchings which satisfy certain degree constraints. Specifically, the formula lets us construct a polynomial P whose monomials correspond to k -tuples of matchings in G , where each vertex in G appears as an endpoint in exactly $\deg_G(v)$ of the matchings in the tuple.

We design the partition matroid \mathcal{D} in terms of a *dominating set* of the input graph. A subset of vertices D is a dominating set in G if every vertex in $V \setminus D$ is adjacent to a node in D . Given a dominating set D in G , we show that it is always possible to construct a partition matroid \mathcal{D} with $p = |V \setminus D|$ parts, compatible with our enumerating polynomial P (in our proofs, \mathcal{D} is actually only compatible with a polynomial obtained from P by the inclusion-exclusion principle, but we ignore that distinction in this overview). We achieve compatibility using the degree condition imposed on P , mentioned at the end of the previous paragraph. Partition sieving then lets us detect a multilinear monomial in P , and thus solve EDGE COLORING, in $O^*(2^{m-|V \setminus D|})$ time.

► **Lemma 1.4.** *There is a randomized algorithm that, given a graph G on n vertices and m edges and a dominating set D of G , solves EDGE COLORING on G with high probability and one-sided error in $O^*(2^{m-n+|D|})$ time and polynomial space.*

In other words, by finding smaller dominating sets in G , we can solve EDGE COLORING faster. In polynomial time, it is easy to construct a dominating set of size at most $n/2$ in any connected graph G with n vertices. Consequently, the partition sieving framework lets us solve EDGE COLORING in $O^*(2^{m-n/2})$ time. We obtain faster algorithms using the fact

that if a graph with $n \geq 8$ nodes has minimum degree two, then it contains a dominating set of size at most $2n/5$ [29]. We make this result effective, showing how to *find* such a dominating set in $O^*(2^{m-3n/5})$ time and polynomial space. When solving EDGE COLORING, it turns out one can assume without loss of generality that the graph has minimum degree two, and so this framework yields a $O^*(2^{m-3n/5})$ time algorithm for the problem. The situation is far more complex for LIST EDGE COLORING, where unit-degree vertices cannot be removed freely. For this problem, we construct a more complicated polynomial P , by enforcing additional matroid conditions in the Ishikawa-Wakayama formula. Intuitively, we identify a subgraph G_{new} of minimum degree two in G , and use P to determine whether G_{new} admits a list edge coloring which can be *extended* to all of G .

Organization

In Section 2 we review notation, basic assumptions, and useful facts about polynomials, matrices, and matroids. In Section 3 we introduce our partition sieving method for monomial detection. In Section 4 we present a generic algorithmic template for solving coloring problems using polynomials. In Section 5 we apply this framework to design our algorithms for EDGE COLORING and prove Theorems 1.1 and 1.2. We conclude in Section 6 with a summary of our work and some open problems. Our LIST EDGE COLORING algorithm and the proof of Theorem 1.3 is deferred to the full version of this paper.

2 Preliminaries

General Notation

Given a positive integer a , we let $[a] = \{1, \dots, a\}$ denote the set of the first a consecutive positive integers. Given a set S and positive integer k , we let $\binom{S}{k}$ denote the family of subsets of S of size k . Given a positive integer k , we let H_k denote the k^{th} Harmonic number, defined in Equation (1).

Graph Notation and Assumptions

Throughout, we consider graphs which are undirected. We let n and m denote the number of vertices and edges in the input graphs for the problems we study. We assume that the input graphs are connected, so that $m \geq n - 1$. This is without loss of generality, since when solving EDGE COLORING and LIST EDGE COLORING on disconnected graphs, it suffices to solve the problems separately on each connected component. Given a graph G and vertex v , we let $\deg_G(v)$ denote the number of neighbors of v (i.e., the degree of v in G). For the LIST EDGE COLORING problem where each edge is assigned a list of colors from a subset of $[k]$, we assume that $\deg_G(v) \leq k$ for all vertices v in G , since if some vertex v has degree greater than k , its incident edges must all be assigned distinct colors in a proper edge coloring, and the answer to the LIST EDGE COLORING problem is trivially no.

We let $\Pi(G)$ denote the set of perfect matchings of G – partitions of the vertex set of G into parts of size two, such that each part consists of two adjacent vertices.

Dominating Sets

A dominating set D in a graph G is a subset of vertices of G such that every vertex in G is either in D or adjacent to a node in D . We collect some useful results relating to finding small dominating sets in graphs.

7:6 Faster Edge Coloring by Partition Sieving

► **Lemma 2.1** (Simple Dominating Set). *There is a polynomial time algorithm which takes in a connected graph on n nodes and outputs a dominating set for the graph of size at most $n/2$.*

Lemma 2.1 is due to Ore [32], and we include a proof in the full version of this paper.

► **Lemma 2.2** (Dominating Sets in Dense Graphs [12, 29]). *If G has $n \geq 8$ nodes, and every vertex in G has degree at least two, then G has a dominating set of size at most $2n/5$.*

► **Lemma 2.3** (Dominating Sets in Regular Graphs [1, 33]). *Any d -regular graph on n vertices has a dominating set of size at most $(H_{d+1}/(d+1))n$.*

► **Lemma 2.4** (Dominating Set Algorithm). *Let G' be a graph obtained by starting with G , and repeatedly deleting vertices of degree one until no vertices of degree one remain. Suppose at most $n/5$ unit-degree vertices were deleted from G to produce G' . Then a minimum-size dominating set of G' can be found in $O^*(2^{m-3n/5})$ time and polynomial space.*

Lemma 2.4 is proved in the full version of this paper.

Finite Field Arithmetic

Throughout, we work with polynomials and matrices over a field $\mathbb{F} = \mathbb{F}_{2^\ell}$ of characteristic two, where $\ell = \text{poly}(n)$ is sufficiently large. Arithmetic operations over \mathbb{F} take $\text{poly}(n)$ time. All elements $a \in \mathbb{F}$ satisfy $a^{2^\ell} = a$. Consequently, given any element $a \in \mathbb{F}$, we can compute its unique square root as $a^{2^{\ell-1}}$ by repeated squaring in $\text{poly}(n)$ time.

Polynomials

Let $P(X)$ be a polynomial over a set of variables $X = \{x_1, \dots, x_n\}$. A monomial \mathbf{m} is a product $\mathbf{m} = x_1^{m_1} \cdots x_n^{m_n}$ for some nonnegative integers m_1, \dots, m_n . A monomial \mathbf{m} is *multilinear* if $m_i \leq 1$ for each $i \in [n]$. We say \mathbf{m} appears in the polynomial P if the coefficient of \mathbf{m} in P is nonzero. We define the *support* of \mathbf{m} to be the set of indices

$$\text{supp}(\mathbf{m}) = \{i \in [n] \mid m_i > 0\}$$

of variables which appear with positive degree in \mathbf{m} .

Similarly, we define the *odd support* of \mathbf{m} to be the set

$$\text{osupp}(\mathbf{m}) = \{i \in [n] \mid m_i \equiv 1 \pmod{2}\}$$

of indices of variables which appear with odd degree in \mathbf{m} . By definition, $\text{osupp}(\mathbf{m}) \subseteq \text{supp}(\mathbf{m})$ for all monomials \mathbf{m} .

The *degree* of a monomial $\mathbf{m} = x_1^{m_1} \cdots x_n^{m_n}$ is defined to be $\deg(\mathbf{m}) = m_1 + \cdots + m_n$. Given $S \subseteq [n]$, we define the degree of \mathbf{m} restricted to the variables indexed by S to be

$$\sum_{i \in S} m_i.$$

The *total degree* (or just *degree*) of P is defined to be

$$\deg P = \max_{\mathbf{m}} \deg(\mathbf{m})$$

where the maximum is taken over all monomials \mathbf{m} with nonzero coefficient in P . We say that a polynomial P is *homogeneous* if $\deg(\mathbf{m}) = \deg P$ for all monomials \mathbf{m} in P .

We recall the Lagrange Interpolation formula, which allows one to recover coefficients of a low degree univariate polynomial from a small number of evaluations of that polynomial.

► **Proposition 2.5** (Lagrange Interpolation). *Let $P(z)$ be a univariate polynomial of degree less than n . Suppose that $P(z_i) = p_i$ for distinct $z_1, \dots, z_n \in \mathbb{F}$. Then*

$$P(z) = \sum_{i \in [n]} p_i \prod_{j \in [n] \setminus \{i\}} \frac{z - z_j}{z_i - z_j}.$$

So given n evaluations of P at distinct points, we can compute all the coefficients of $P(z)$ in polynomial time.

We also record the following observation, proven for example in [30, Theorem 7.2], which shows that to test whether a low degree polynomial P is nonzero, it suffices to check whether a random evaluation of P over a sufficiently large field is nonzero.

► **Proposition 2.6** (Schwartz-Zippel Lemma). *Let P be a nonzero polynomial over a finite field \mathbb{F} of degree at most d . If each variable of P is assigned an independent, uniform random value from \mathbb{F} , then the corresponding evaluation of P is nonzero with probability $1 - d/|\mathbb{F}|$.*

Matrices

Given a matrix A and sets of rows I and columns J , we let $A[I, J]$ denote the submatrix of A restricted to rows in I and columns in J . We let $A[I, \cdot]$ denote the submatrix of A restricted to rows in I but using all columns, and $A[\cdot, J]$ denote the submatrix of A restricted to columns in J but using all rows. If the rows and column sets of A are identical, we write $A[S]$ as shorthand for $A[S, S]$. We let O denote the all-zeros matrix, whose dimensions will always be clear from context. We let A^\top denote the transpose of A . A matrix A is *symmetric* if $A = A^\top$, and *skew-symmetric* if it is symmetric and has zeros along its main diagonal (we employ this definition because we work over characteristic two).

If A is a square matrix, we let $\det A$ denote the determinant of A .

Given a set V , a perfect matching on V is a partition of V into sets of two elements each. We let $\Pi(V)$ denote the set of perfect matchings on V . Given a skew-symmetric matrix A with rows and columns indexed by a set V , we define the *Pfaffian* of A to be

$$\text{Pf } A = \sum_{M \in \Pi(V)} \prod_{\{u, v\} \in M} A[u, v]. \quad (2)$$

The standard definition for Pfaffians involves a sign term (see e.g., [31, Section 7.3.2]), which we ignore here because we work over a field of characteristic two. It is well known that for all skew-symmetric A , we have $\det A = (\text{Pf } A)^2$ (see e.g., [31, Proposition 7.3.3] for a proof). Consequently, the Pfaffian of matrix A over the field \mathbb{F}_{2^ℓ} can be computed in $\text{poly}(n, \ell) \leq \text{poly}(n)$ time by using the equation $\text{Pf } A = \sqrt{\det A} = (\det A)^{2^{\ell-1}}$.

We record the following useful facts for computation with Pfaffians and determinants.

► **Proposition 2.7** (Direct Sums). *For any skew-symmetric matrices A_1 and A_2 , we have*

$$\text{Pf} \begin{pmatrix} A_1 & O \\ O & A_2 \end{pmatrix} = (\text{Pf } A_1) (\text{Pf } A_2)$$

over a field of characteristic two.

Proposition 2.7 is proved in the full version of this paper.

► **Proposition 2.8** (Ishikawa-Wakayama Formula). *For a skew-symmetric $n \times n$ -matrix A and a $k \times n$ -matrix B with $k \leq n$, we have*

$$\text{Pf } BAB^\top = \sum_{S \in \binom{[n]}{k}} \det B[\cdot, S] \text{Pf } A[S].$$

7:8 Faster Edge Coloring by Partition Sieving

Proposition 2.8 is due to [22] (see [23, Section 4.3] for a recent alternate proof).

Matroids

A *matroid* is a pair $\mathcal{M} = (X, \mathcal{I})$, consisting of a *ground set* X and a collection \mathcal{I} of subsets of X called *independent sets*, satisfying the following axioms:

1. $\emptyset \in \mathcal{I}$.
2. If $B \in \mathcal{I}$ and $A \subset B$, then $A \in \mathcal{I}$.
3. For any $A, B \in \mathcal{I}$ with $|A| < |B|$, there exists an element $b \in B \setminus A$ such that $A \cup \{b\} \in \mathcal{I}$.

A maximal independent set in a matroid \mathcal{M} is called a *basis*. It is well known that all bases of \mathcal{M} have the same size r , which is referred to as the *rank* of \mathcal{M} . We say $S \subseteq V$ *spans* \mathcal{M} if S is a superset of a basis of \mathcal{M} . If there exists a matrix A with column set X such that for every $S \subseteq X$, the set S is independent in \mathcal{M} if and only if $A[:, S]$ has full rank, then we say that \mathcal{M} is a *linear matroid represented* by A . Provided we work over a large enough field of size $\text{poly}(|X|)$, it is known that if \mathcal{M} is linear, then it can be represented by a matrix whose number of rows equals the rank of \mathcal{M} (see e.g., [26] or [28, Section 3.7]).

Given matroids $\mathcal{M}_1, \dots, \mathcal{M}_k$ over disjoint ground sets X_1, \dots, X_k respectively, we define the direct sum

$$\mathcal{M} = \bigoplus_{i=1}^k \mathcal{M}_i$$

of these matroids to be the ground set $X = X_1 \sqcup \dots \sqcup X_k$ together with the family \mathcal{I} of subsets S of X with the property that $S \cap X_i$ is independent in \mathcal{M}_i for all $i \in [k]$. It is well known that \mathcal{M} is a matroid as well, whose rank is equal to the sum of the ranks of the \mathcal{M}_i .

The following result lets us represent direct sums of linear matroids [28, Section 3.4].

► **Proposition 2.9 (Matroid Sum).** *Given linear matroids $\mathcal{M}_1, \dots, \mathcal{M}_k$ represented by matrices A_1, \dots, A_k respectively, their direct sum is represented by the block-diagonal matrix*

$$A = \begin{pmatrix} A_1 & O & \dots & O \\ O & A_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & A_k \end{pmatrix}.$$

In our EDGE COLORING algorithm, we make use of two simple types of matroids: uniform and partition matroids. Given a ground set X and a nonnegative integer r , the *uniform matroid* over X of rank r has as its independent sets all subsets of X of size at most r .

► **Proposition 2.10 (Uniform Matroid Representation [28, Section 3.5]).** *Given a set X , a nonnegative integer r , and a field \mathbb{F} with $|\mathbb{F}| > |X|$, we can construct an $r \times |X|$ matrix representing the uniform matroid over X of rank r in $\text{poly}(|X|)$ time.*

Given a set X partitioned $X = X_1 \sqcup \dots \sqcup X_k$ into parts X_i , and a list of nonnegative integer capacities c_1, \dots, c_k , the *partition matroid* \mathcal{M} over the ground set X for this partition and list of capacities has as its independent sets all subsets $S \subseteq X$ satisfying $|S \cap X_i| \leq c_i$ for all $i \in [k]$. Equivalently, \mathcal{M} is the direct sum of uniform matroids of rank c_i over X_i for all $i \in [k]$. By definition, \mathcal{M} has rank $r = (c_1 + \dots + c_k)$. By Propositions 2.9 and 2.10 we see that we can efficiently construct representations of partition matroids [28, Proposition 3.5].

► **Proposition 2.11 (Partition Matroid Representation).** *Given a partition matroid \mathcal{M} over X of rank r , and a field \mathbb{F} with $|\mathbb{F}| > |X|$, we can construct an $r \times |X|$ matrix representing \mathcal{M} in $\text{poly}(|X|)$ time.*

3 Polynomial Sieving

In this section, we develop faster algorithms for a certain monomial detection problem. This is our main technical result, which allows us to obtain faster algorithms for EDGE COLORING and LIST EDGE COLORING. We recall the following well-known sieving technique:

► **Proposition 3.1** (Inclusion-Exclusion). *Let $P(X)$ be a polynomial over a field of characteristic two, with variable set $X = \{x_1, \dots, x_n\}$. Fix $T \subseteq X$. Let $Q(X)$ be the polynomial consisting of all monomials (with corresponding coefficients) in P that are divisible by $\prod_{x \in T} x$. Then*

$$Q = \sum_{S \subseteq T} P_S$$

where P_S is the polynomial obtained from P by setting $x_i = 0$ for all $i \in S$.

See [35, Lemma 2] for a proof of Proposition 3.1.

Combining Propositions 2.5 and 3.1, we obtain the following result.

► **Lemma 3.2** (Coefficient Extraction). *Let $P(X)$ be a polynomial over a field of characteristic two, with variable set $X = \{x_1, \dots, x_n\}$. For $T \subseteq X$, let $Q(X \setminus T)$ be the coefficient of $\prod_{x \in T} x$ in $P(X)$, viewed as a polynomial over the variable set $X \setminus T$. Then we can evaluate $Q(X \setminus T)$ at a point as a linear combination of $2^{|T|}$ poly(n) evaluations of $P(X)$.*

Proof. Fix a subset $T \subseteq X$. Let F be the polynomial with variable set $X \cup \{z\}$ obtained by substituting each variable $x \in T$ with xz in P . Let $G(X)$ be the coefficient of $z^{|T|}$ in F . By definition of F , G is the polynomial obtained by taking the monomials in P whose degree restricted to T equals $|T|$. By Proposition 2.5, we can compute $G(X)$ at any point as a linear combination of poly(n) evaluations of P .

By definition, $Q(X \setminus T)$ is the polynomial obtained by taking monomials of $G(X)$ that are divisible by $\prod_{x \in T} x$, and then setting all variables in T equal to 1. So by Proposition 3.1 we can evaluate $Q(X \setminus T)$ as the sum of $2^{|T|}$ evaluations of G . Then by the conclusion of the previous paragraph, we get that we can evaluate $Q(X \setminus T)$ at any point as a linear combination of $2^{|T|}$ poly(n) evaluations of P as claimed. ◀

The inclusion-exclusion principle allows us to sieve for the multilinear term in a polynomial that is the product of all its variables. This was extended to the parameterized setting, where the goal is to sieve for multilinear terms of degree k in $O^*(2^k)$ time [7, 10, 36], and then further extended to multilinear monomial detection in the matroid setting [16].

► **Lemma 3.3** (Basis Sieving [16, Theorem 1.1]). *Let $P(X)$ be a homogeneous polynomial of degree k over a field \mathbb{F} of characteristic 2, and let $\mathcal{M} = (X, \mathcal{I})$ be a matroid on X of rank k , with a representation over \mathbb{F} given. Then there is a randomized algorithm running in $O^*(2^k)$ time and polynomial space, which determines with high probability if $P(X)$ contains a multilinear monomial \mathbf{m} such that $\text{supp}(\mathbf{m})$ is a basis of \mathcal{M} .*

Moreover, [16] introduced a variant of sieving that takes the odd support into account. This tool is key to our exponential speed-up for the EDGE COLORING problem.

► **Lemma 3.4** (Odd Sieving [16, Theorem 1.2]). *Let $P(X)$ be a polynomial of degree d over a field \mathbb{F} of characteristic 2, and let $\mathcal{M} = (X, \mathcal{I})$ be a matroid on X of rank k , represented by a $k \times n$ matrix over \mathbb{F} . Then using $O^*(d2^k)$ evaluations of $P(X)$ and polynomial space, we can determine if $P(X)$ contains a monomial \mathbf{m} such that $\text{osupp}(\mathbf{m})$ spans \mathcal{M} .*

7:10 Faster Edge Coloring by Partition Sieving

Let $P(X)$ be a homogeneous polynomial of degree d in variables $X = \{x_1, \dots, x_n\}$. Consider a partition $X = X_1 \sqcup \dots \sqcup X_p$ into parts X_i . Let $c_1, \dots, c_p \geq 0$ be integers with

$$c_1 + \dots + c_p = d.$$

Let \mathcal{M} be the partition matroid defined by the partition of X into the parts X_i and the capacities c_i for $i \in [p]$. We say $P(X)$ is *compatible* with \mathcal{M} if for every $i \in [p]$ and every monomial \mathbf{m} in P , the degree of \mathbf{m} restricted to X_i equals $c_i \geq 1$.

The following result gives an improvement over Lemma 3.3 for polynomials that are compatible with partition matroids.

► **Theorem 3.5 (Partition Sieving).** *Let $P(X)$ be a polynomial of degree d over a field \mathbb{F} of characteristic 2 with $|\mathbb{F}| > d$, and let $\mathcal{M} = (X, \mathcal{I})$ be a partition matroid on X of rank d with p partition classes. If P is compatible with \mathcal{M} , then there is a randomized algorithm which runs in $O^*(2^{d-p})$ time and polynomial space that determines with high probability if $P(X)$ contains a monomial \mathbf{m} such that $\text{supp}(\mathbf{m})$ is a basis of \mathcal{M} .*

Proof. Let $X = X_1 \sqcup \dots \sqcup X_p$ be the partition and c_1, \dots, c_p be the capacities defining the partition matroid \mathcal{M} . By assumption, $c_i \geq 1$ for all $i \in [p]$. Let \mathcal{M}' be the partition matroid over X defined by the same partition as \mathcal{M} , but with capacities $(c_i - 1)$ for $i \in [p]$. From this construction, \mathcal{M}' has rank $(d - p)$.

Our algorithm to determine whether $P(X)$ contains a monomial whose support spans \mathcal{M} works as follows. By Proposition 2.11, a linear representation of \mathcal{M} over \mathbb{F} can be computed in polynomial time. We run the odd sieving algorithm of Lemma 3.4 with P and \mathcal{M}' to determine in $O^*(2^{d-p})$ time whether P contains a monomial whose odd support spans \mathcal{M}' . We return YES if such a monomial exists, and return NO otherwise. We now prove that this algorithm is correct.

First, suppose that $P(X)$ has a monomial \mathbf{m} whose support $\text{supp}(\mathbf{m})$ is a basis of \mathcal{M} . By the compatibility condition, \mathbf{m} is multilinear, which implies that $\text{supp}(\mathbf{m}) = \text{osupp}(\mathbf{m})$. Since $\text{osupp}(\mathbf{m})$ is a basis of \mathcal{M} , and thus spans \mathcal{M} , it follows that $\text{osupp}(\mathbf{m})$ spans \mathcal{M}' .

Conversely, suppose $P(X)$ has a monomial \mathbf{m} such that $\text{osupp}(\mathbf{m})$ spans \mathcal{M}' . We claim that $\text{supp}(\mathbf{m})$ spans \mathcal{M} . Indeed, since $\text{osupp}(\mathbf{m})$ spans \mathcal{M}' , the set $\text{osupp}(\mathbf{m})$ contains at least $(c_i - 1)$ variables of X_i for each $i \in [p]$. Hence its superset $\text{supp}(\mathbf{m})$ contains at least $(c_i - 1)$ variables of each part X_i as well.

▷ **Claim 3.6.** The set $\text{supp}(\mathbf{m})$ contains exactly c_i variables of each part X_i .

Proof. Suppose to the contrary that $\text{supp}(\mathbf{m})$ does not contain c_j variables of X_j for some index $j \in [p]$. Then \mathbf{m} contains exactly $(c_j - 1)$ variables of X_j . Since P is compatible with \mathcal{M} , the monomial \mathbf{m} has degree exactly c_j when restricted to X_j . The only way this is possible is if $c_j \geq 2$, and \mathbf{m} has exactly $(c_j - 2)$ variables in X_j of degree one and a single variable in X_j with degree two. The variable of degree two does not show up in the odd support of \mathbf{m} , which implies that $\text{osupp}(\mathbf{m})$ has at most $(c_j - 2)$ elements, which contradicts the assumption that $\text{osupp}(\mathbf{m})$ spans \mathcal{M}' . ◁

By Claim 3.6, the set $\text{supp}(\mathbf{m})$ has exactly c_i variables from X_i for each $i \in [p]$. Since P is compatible with \mathcal{M} , this implies that \mathbf{m} has degree one in every variable in X . Hence $\text{supp}(\mathbf{m}) = \text{osupp}(\mathbf{m})$ spans \mathcal{M} , as claimed.

This shows that the algorithm is correct, and proves the desired result. ◀

4 Coloring Algorithm Template

Recall that in the LIST EDGE COLORING problem we are given a graph $G = (V, E)$, an integer $k \geq 1$, and lists $L_e \subseteq [k]$ of colors for every edge $e \in E$, and are tasked with determining if G contains an assignment of colors $c(e) \in L_e$ to each edge such that no two edges incident to the same vertex are assigned the same color. We call such an assignment a proper edge coloring of G with respect to the lists L_e , or just a proper list edge coloring if the L_e are clear from context.

For each $i \in [k]$, let $G_i = (V, E_i)$ be the subgraph of G , where we define $E_i \subseteq E$ to be the set of all edges e in the graph which have $i \in L_e$ available as a color. Let \mathcal{F} denote the collection of k -tuples (M_1, \dots, M_k) of matchings such that

1. for all $i \in [k]$, M_i is a matching in G_i , and
2. for all vertices $v \in V$, v appears as the endpoint of exactly $\deg_G(v)$ of the M_i matchings.

Observe that G admits a proper list edge coloring if and only if \mathcal{F} contains a tuple of edge-disjoint matchings. Indeed, suppose G admits a proper list edge coloring. For each $i \in [k]$, the set of edges assigned color i must be a matching $M_i \subseteq E_i$ in G . Moreover, every edge is assigned a color, so each vertex v appears as an endpoint in $\deg_G(v)$ of these matchings, hence the matchings form an edge-disjoint tuple in \mathcal{F} . Conversely, given an edge-disjoint tuple $(M_1, \dots, M_k) \in \mathcal{F}$, assigning the edges in M_i color i gives a color to every edge in G by condition 2 above, and thus yields a proper list edge coloring by condition 1.

The basic idea of our algorithmic template is to construct a polynomial P which enumerates a certain subfamily $\mathcal{C} \subseteq \mathcal{F}$. We then argue that P contains a multilinear monomial satisfying certain matroid constraints if and only if \mathcal{C} contains a tuple of edge-disjoint matchings. Using similar reasoning to the previous paragraph, this then lets us solve LIST EDGE COLORING by running monomial detection algorithms on P .

4.1 Enumerating Matchings

In this section, we design a polynomial P whose monomials enumerate tuples of matchings satisfying special conditions. To construct P , we first define certain polynomial matrices.

For each edge $e \in E$ we introduce a variable x_e , and for each pair $(e, i) \in E \times [k]$ we introduce a variable y_{ei} . Let X be the set of x_e variables and Y be the set of y_{ei} variables.

For each $i \in [k]$, define A_i to be the matrix with rows and columns indexed by V with

$$A_i[u, w] = A_i[w, u] = \begin{cases} x_e y_{ei} & \text{if } e = \{u, w\} \in E_i, \\ 0 & \text{otherwise.} \end{cases}$$

For each $i \in [k]$, define $V_i = \{v_i \mid v \in V\}$ to be a copy of V . Let

$$W = \bigsqcup_{i=1}^k V_i$$

be the union of these copies. Let A be the symmetric matrix with rows and columns indexed by W , defined by taking

$$A = \begin{pmatrix} A_1 & O & \dots & O \\ O & A_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & A_k \end{pmatrix} \tag{3}$$

and identifying V_i as the row and column set indexing A_i .

7:12 Faster Edge Coloring by Partition Sieving

Note that A is $(kn) \times (kn)$, and over a field of characteristic two is skew-symmetric.

For each vertex $v \in V$, let $S_v = \{v_i \mid i \in [k]\}$ be the set of k copies of v .

For each vertex v , let \mathcal{M}_v be a linear matroid of rank $\deg_G(v)$ over S_v . These matroids will be specified later on, in our algorithms for EDGE COLORING and LIST EDGE COLORING. Let \mathcal{W} be the direct sum

$$\mathcal{W} = \bigoplus_{v \in V} \mathcal{M}_v \quad (4)$$

of these matroids.

By Proposition 2.7 the matroid \mathcal{W} is linear and has rank equal to the sum of the degrees of vertices in G , which is $2m$. Let B be a $2m \times |W|$ matrix representing \mathcal{W} .

Let \mathcal{C} be the collection of k -tuples of matchings (M_1, \dots, M_k) in G such that

1. all edges in M_i have lists containing the color i ,
2. every vertex v is incident to exactly $\deg_G(v)$ edges across the matchings M_i , and
3. for each vertex v , the set of copies v_i taken over all i such that v appears as an endpoint of an edge in M_i forms an independent set in \mathcal{M}_v .

We define the polynomial

$$P(X, Y) = \text{Pf } BAB^\top. \quad (5)$$

The next result shows that P enumerates tuples of matchings from \mathcal{C} .

► **Lemma 4.1** (Enumerating Matchings). *We have*

$$\text{Pf } BAB^\top = \sum_{(M_1, \dots, M_k) \in \mathcal{C}} c_{M_1, \dots, M_k} \left(\prod_{i \in [k]} \prod_{e \in M_i} y_{ei} \right) \left(\prod_{e \in M_1 \cup \dots \cup M_k} x_e \right),$$

where each $c_{M_1, \dots, M_k} \neq 0$ is a constant depending only on M_1, \dots, M_k .

Proof. Let $U \subseteq W$ be a subset of W of size $2m$. By Proposition 2.8, we have

$$\text{Pf } BAB^\top = \sum_{U \in \binom{W}{2m}} \det B[\cdot, U] \text{Pf } A[U]. \quad (6)$$

By Equation (3) and Proposition 2.7, for every subset $U \subseteq W$ of size $2m$ we have

$$\text{Pf } A[U] = \prod_{i \in [k]} \text{Pf } A_i[U \cap V_i].$$

Substituting the above equation into Equation (6) yields

$$\text{Pf } BAB^\top = \sum_{U \in \binom{W}{2m}} \det B[\cdot, U] \prod_{i \in [k]} \text{Pf } A_i[U \cap V_i].$$

Recall that given a graph H , we let $\Pi(H)$ denote the set of perfect matchings of H . By expanding out the definition of $\text{Pf } A_i[U \cap V_i]$ in the above equation, we see that

$$\text{Pf } BAB^\top = \sum_{U \in \binom{W}{2m}} \det B[\cdot, U] \prod_{i \in [k]} \sum_{M_i \in \Pi(G[U_i])} \prod_{e \in M_i} A_i[e],$$

where $U_i \subseteq V$ is defined as a copy of $U \cap V_i$, i.e., $U_i = \{v \mid v_i \in U \cap V_i\}$. By interchanging the order of the product and the summation, we see that the above is equivalent to summing over all collections (M_1, \dots, M_k) , where M_i is a matching in $G[U_i]$ for each $i \in [k]$:

$$\text{Pf } BAB^\top = \sum_{U \in \binom{W}{2m}} \det B[\cdot, U] \sum_{(M_1, \dots, M_k)} \prod_{i \in [k]} \prod_{e \in M_i} A_i[e].$$

Let $\vec{M} = (M_1, \dots, M_k)$ be a tuple such that the summand corresponding to \vec{M} in the above equation appears with nonzero coefficient in $\text{Pf } BAB^\top$. Since $A_i[e]$ is nonzero if and only if $i \in L_e$, the tuple \vec{M} meets condition 1 in the definition of \mathcal{C} . By the definition of B , we have $\det B[\cdot, U] \neq 0$ if and only if $U \cap S_v$ is independent in \mathcal{M}_v for every vertex $v \in V$. Thus \vec{M} meets condition 3 the definition of \mathcal{C} . Since \mathcal{M}_v has rank $\deg_G(v)$ for each vertex v , U has size $2m$, and the sum of degrees of vertices in G is $2m$, we see that the only way for $U \cap S_v$ to be independent in \mathcal{M}_v for all v is if in fact $|U \cap S_v| = \deg_G(v)$ always. So \vec{M} satisfies condition 2 from the definition of \mathcal{C} .

Thus, the nonzero terms in the above sum correspond precisely to tuples $\vec{M} \in \mathcal{C}$, which proves the desired result. \blacktriangleleft

4.2 Partition Sieving With Dominating Sets

In this section, we describe a generic way of going from a dominating set in G to a certain partition matroid \mathcal{D} . After defining this \mathcal{D} , we will apply Theorem 3.5 to P and \mathcal{D} to achieve a fast algorithm for detecting a multilinear monomial in P .

Recall that a *dominating set* of a graph G is a subset of vertices D such that every vertex in G is either in D or adjacent to a vertex in D .

Fix a dominating set $D \subseteq V$ of the input graph G . Let $V' = V \setminus D$. Let E' be the set of edges in G with one endpoint in D and one endpoint in V' , and $X' = \{x_e \mid e \in E'\}$ be the corresponding set of variables. For each $v \in V'$, let

$$\partial(v) = \{e \in E' \mid e \ni v\}$$

be the set of edges incident to v which connect v to a vertex in D . By definition, we have a partition

$$E' = \bigsqcup_{v \in V'} \partial(v).$$

Let \mathcal{D} be the partition matroid over E' with respect to the above partition, and with capacities $c_v = \deg_{E'}(v)$ for each $v \in V'$, where E' is viewed as a subgraph of G . Since D is a dominating set, every vertex $v \in V'$ is adjacent to some node in D , and thus the capacities $c_v \geq 1$ are all positive.

We now use this partition matroid to sieve over the polynomial P , defined in Equation (5).

► Lemma 4.2 (Accelerated Multilinear Monomial Detection). *There is a randomized algorithm that determines with high probability whether $P(X, Y)$ has a monomial divisible by $\prod_{e \in E} x_e$, running in $O^*(2^{m-|V'|})$ time and polynomial space.*

Proof. Let $Q(X', Y)$ be the coefficient of

$$\prod_{e \in E \setminus E'} x_e$$

in $P(X, Y)$. By definition, $P(X, Y)$ contains a monomial divisible by $\prod_{e \in E} x_e$ if and only if $Q(X', Y)$ contains a monomial divisible by $\prod_{e \in E'} x_e$.

7:14 Faster Edge Coloring by Partition Sieving

Let $R(Y)$ be the coefficient of $\prod_{e \in E'} x_e$ in Q , viewed as a polynomial with variable set Y . Note that the total degree of $R(Y)$ is $\text{poly}(n)$. Take a uniform random assignment over \mathbb{F} to the variables in Y . By Proposition 2.6, if R is nonzero, then it remains nonzero with high probability after this evaluation, provided we take $|\mathbb{F}| = \text{poly}(n)$ to be sufficiently large. For the rest of this proof, we work with polynomials P, Q, R after this random evaluation, so that P is a polynomial in X , Q is a polynomial in X' , and R is a field element.

By the discussion in the first paragraph of this proof, it suffices to check whether $Q(X')$ contains the monomial $\prod_{e \in E'} x_e$ in $O^*(2^{m-|V'|})$ time and polynomial space.

We observe the following helpful property of Q .

▷ **Claim 4.3.** The polynomial $Q(X')$ is compatible with the matroid \mathcal{D} .

Proof. Take an arbitrary monomial \mathbf{m} in Q . By definition, there is a corresponding monomial

$$\mathbf{m}' = \mathbf{m} \cdot \prod_{e \in E \setminus E'} x_e$$

in P . By Lemma 4.1, for every $v \in V$, \mathbf{m}' has degree exactly $\deg_G(v)$ when restricted to the variables x_e corresponding to edges e containing v . Consequently, for all $v \in V'$, \mathbf{m} has degree exactly

$$\deg_G(v) - |\{e \notin E' \mid e \ni v\}| = \deg_{E'}(v)$$

when restricted to variables x_e with $e \in \partial(v)$. Since this holds for arbitrary monomials \mathbf{m} in Q , the polynomial Q is compatible with the partition matroid \mathcal{D} as claimed. ◁

By Lemma 4.1, P , viewed as a polynomial in X , is homogeneous of degree $|E| = m$. Hence Q , viewed as a polynomial in X' , is homogeneous of degree $d = |E'|$. By Claim 4.3, Q is compatible with \mathcal{D} . Since \mathcal{D} is a partition matroid with positive capacities and $p = |V'|$ parts in its underlying partition, by Theorem 3.5 we can determine whether Q contains a monomial \mathbf{m} in the X' variables such that $\text{supp}(\mathbf{m})$ is a basis of \mathcal{D} using

$$O^*(d2^{d-p}) \leq O^*(2^{|E'| - |V'|}) \tag{7}$$

evaluations of Q . Since E' is the unique basis of \mathcal{D} , this means we can test whether $Q(X')$ contains the monomial $\prod_{e \in E'} x_e$ using $O^*(2^{|E'| - |V'|})$ evaluations of $Q(X')$.

By Lemma 3.2, $Q(X')$ can be evaluated at any point by computing a linear combination of $O^*(2^{m-|E'|})$ evaluations of $P(X)$. Combining this observation with Equation (7), we get that we can determine whether $Q(X')$ contains the monomial $\prod_{e \in E'} x_e$ by computing a linear combination of

$$O^*(2^{m-|E'|} \cdot 2^{|E'| - |V'|}) \leq O^*(2^{m-|V'|})$$

evaluations of $P(X, Y)$. Each evaluation of $P(X, Y) = \text{Pf } BAB^\top$ takes polynomial time, so the desired result follows. ◀

5 Edge Coloring Algorithm

In this section, we present our algorithm for **EDGE COLORING** and prove Theorems 1.1 and 1.2. Recall that in this problem, we are given an input graph G with n vertices, m edges, and maximum degree Δ . To solve the problem, it suffices to determine whether G admits a proper edge coloring using at most Δ colors (i.e., whether $\chi'(G) \leq \Delta$).

5.1 Removing Low Degree Vertices

► **Lemma 5.1** (Unit-Degree Deletion). *Let G be a graph with a vertex v of degree one. Let G' be the graph obtained by deleting v from G . Then $\chi'(G') \leq \Delta$ if and only if $\chi'(G) \leq \Delta$.*

Proof. If G admits a proper edge coloring with Δ colors, then this coloring is also a proper edge coloring for G' with Δ colors.

Conversely, suppose G' admits a proper edge coloring with Δ colors. Since $\deg_G(v) = 1$, there is a unique vertex w in G that v is adjacent to. Then

$$\deg_{G'}(w) \leq \deg_G(w) - 1 \leq \Delta - 1$$

because G has maximum degree Δ . So vertex w has edges using at most $(\Delta - 1)$ distinct colors incident to it. Taking the coloring of G' and assigning edge $\{v, w\}$ a color not used by the edges incident to w in G' recovers a proper edge coloring of G with at most Δ colors. ◀

5.2 Instantiating the Template

Recall the definitions from Section 4. We view the EDGE COLORING problem as a special case of LIST EDGE COLORING, where $k = \Delta$ and every edge e is assigned the list $L_e = [\Delta]$. For each vertex v , we set \mathcal{M}_v to be the uniform matroid over S_v of rank $\deg_G(v)$. Then by Equation (4), \mathcal{W} is the partition matroid over W defined by the partition

$$W = \bigsqcup_{v \in V} S_v$$

and the capacities $c_v = \deg_G(v)$ for parts S_v . By Proposition 2.11, we can construct the matrix B representing \mathcal{W} in polynomial time. In this case, condition 3 is identical to condition 2 in the definition of the collection \mathcal{C} .

► **Lemma 5.2** (Characterization by Polynomials). *We have $\chi'(G) \leq \Delta$ if and only if the polynomial $P(X, Y)$ has a monomial divisible by $\prod_{e \in E} x_e$.*

Proof. Suppose $P(X, Y) = \text{Pf } BAB^\top$ contains a monomial divisible by $\prod_{e \in E} x_e$. Since P is homogeneous of degree m in the X variables, by Lemma 4.1 this means there exists a Δ -tuple of *edge-disjoint* matchings $(M_1, \dots, M_\Delta) \in \mathcal{C}$, such that every edge of G appears in some matching M_i . Consequently, assigning the edges in M_i color i yields a proper edge coloring of G using at most Δ colors.

Conversely, suppose G has a proper edge coloring with Δ colors. Without loss of generality, let the set of colors used be $[\Delta]$. For each $i \in [\Delta]$, let M_i be the set of edges assigned color i . Since this is a proper edge coloring, each M_i is a matching in G . Since every edge is assigned a unique color, each vertex v has exactly $\deg_G(v)$ edges across the matchings M_i . Hence $(M_1, \dots, M_\Delta) \in \mathcal{C}$. Then by Lemma 4.1, P contains a monomial divisible by $\prod_{e \in E} x_e$. ◀

We can now prove Lemma 1.4 as a simple application of Lemma 5.2.

► **Lemma 1.4.** *There is a randomized algorithm that, given a graph G on n vertices and m edges and a dominating set D of G , solves EDGE COLORING on G with high probability and one-sided error in $O^*(2^{m-n+|D|})$ time and polynomial space.*

Proof. Let G have vertex set V and edge set E , and let $V' = V \setminus D$. By Lemma 5.2, we have $\chi'(G) \leq \Delta$ if and only if $P(X, Y)$ has a monomial divisible by $\prod_{e \in E} x_e$. By Lemma 4.2, we can determine whether such a monomial exists in

$$O^*(2^{m-|V'|}) \leq O^*(2^{m-n+|D|})$$

time and polynomial space, as desired. ◀

Having established Lemma 1.4, we are ready to present our EDGE COLORING algorithms.

► **Theorem 1.1.** *There is a randomized algorithm which solves EDGE COLORING with high probability and one-sided error in $O^*(2^{m-3n/5})$ time and polynomial space.*

Proof. To solve EDGE COLORING, it suffices to determine whether $\chi'(G) \leq \Delta$.

If G has a vertex of degree one, delete it. Keep performing such deletions, until we are left with a graph containing no vertices of degree one. By repeated application of Lemma 5.1, the resulting graph admits a proper edge coloring using Δ colors if and only if $\chi'(G) \leq \Delta$. If the resulting graph has constant size, we can determine this trivially. Each such deletion decreases the number of edges and vertices in the graph by one, and thus also decreases the value of $(m - 3n/5)$. Consequently, to show the desired result, we may assume without loss of generality that G has minimum degree two, and at least eight vertices.

Since G has minimum degree two and $n \geq 8$ nodes, by Lemma 2.2 the graph has a dominating set of size at most $2n/5$. By Lemma 2.4, we can then find a dominating set D in G with $|D| \leq 2n/5$ in $O^*(2^{m-3n/5})$ time and polynomial space. Hence by Lemma 1.4 we can solve EDGE COLORING on G in $O^*(2^{m-3n/5})$ time and polynomial space, as desired. ◀

► **Theorem 1.2.** *For each integer $d \geq 6$, there is a randomized algorithm that solves EDGE COLORING with high probability and one-sided error on d -regular graphs in $O^*(2^{m-\alpha_d n})$ time and polynomial space, for $\alpha_d = 1 - H_{d+1}/(d+1)$.*

Proof. Let G be the input graph. By exhaustive search over all subsets of vertices, we can find a minimum-size dominating set D of G in $O^*(2^n)$ time and polynomial space. Being d -regular, G has $m = dn/2 \geq 3n$ edges, so we find D in $O^*(2^{m-n}) \leq O^*(2^{m-\alpha_d n})$ time.

Since G is d -regular, by Lemma 2.3 we must have $|D| \leq n(H_{d+1}/(d+1))$. Then by Lemma 1.4 we can solve EDGE COLORING on G in

$$O^*(2^{m-n+|D|}) \leq O^*(2^{m-\alpha_d n})$$

time as claimed. ◀

6 Conclusion

In this paper, we presented the first algorithms for EDGE COLORING and LIST EDGE COLORING which run in faster than $O^*(2^m)$ time while using only polynomial space. Our algorithm is based off a *partition sieving* procedure that works over polynomials of degree d and partition matroids with p parts. We showed how to implement this sieving routine in $O^*(2^{d-p})$ time when the polynomial is in some sense compatible with the matroid, an improvement over the $O^*(2^d)$ runtime that arose in previous techniques.

Overall, we solve EDGE COLORING in faster than $O^*(2^m)$ time by

- Step 1** designing a polynomial P enumerating matchings meeting certain degree constraints,
- Step 2** finding a dominating set D of size at most $2n/5$ in the input graph, and
- Step 3** applying partition sieving with a partition matroid on $p = |D|$ parts.

Implementing **step 2** above was possible for EDGE COLORING because we could assume that the input graph had minimum degree two without loss of generality, and such graphs on $n \geq 8$ nodes always have dominating sets of size at most $2n/5$. For LIST EDGE COLORING we cannot make this assumption, but nonetheless implement a variant of **step 2** by identifying a subgraph with minimum degree two and using extension matroids to handle unit-degree vertices for free. For all sufficiently large n , it is also known that graphs on n nodes with minimum degree three have dominating sets of size at most $3n/8$ [34]. If degree-two vertices could somehow be “freely removed” from the input graph just like unit-degree vertices can,

one could use this bound on the dominating set to solve coloring problems faster. More general connections between the minimum degree of a graph and the minimum size of its dominating set have also been studied [20].

Understanding the precise time complexity of EDGE COLORING and its variants remains an interesting open research direction. Is EDGE COLORING solvable in $O^*(1.9999^m)$ time, or at least in $O^*(2^{m-n})$ time? Establishing any nontrivial lower bounds for EDGE COLORING would also be very interesting. The reductions from [21] imply that solving EDGE COLORING in graphs with maximum degree $\Delta = 3$ requires $2^{\Omega(n)}$ time, assuming the Exponential Time Hypothesis (ETH), a standard hypothesis from fine-grained complexity. Does ETH similarly imply a $2^{\Omega(m)}$ or even a $2^{\Omega(n \log n)}$ time lower bound for EDGE COLORING in dense graphs (this question was previously raised in [25, Open problem 4.25])?

References

- 1 Vladimir I Arnautov. Estimation of the exterior stability number of a graph by means of the minimal degree of the vertices. *Prikl. Mat. i Programirovanie*, 11(3-8):126, 1974.
- 2 Sepehr Assadi, Soheil Behnezhad, Sayan Bhattacharya, Martín Costa, Shay Solomon, and Tianyi Zhang. Vizing’s theorem in near-linear time, 2024. [arXiv:2410.05240](https://arxiv.org/abs/2410.05240).
- 3 Alkida Balliu, Sebastian Brandt, Fabian Kuhn, and Dennis Olivetti. Distributed edge coloring in time polylogarithmic in Δ . In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC ’22, pages 15–25. ACM, July 2022. doi:10.1145/3519270.3538440.
- 4 Lowell W Beineke. Derived graphs and digraphs. *Beiträge zur graphentheorie*, pages 17–33, 1968.
- 5 Anton Bernshteyn. A fast distributed algorithm for $(\Delta + 1)$ -edge-coloring. *Journal of Combinatorial Theory, Series B*, 152:319–352, January 2022. doi:10.1016/j.jctb.2021.10.004.
- 6 Sayan Bhattacharya, Martín Costa, Nadav Panski, and Shay Solomon. Arboricity-Dependent Algorithms for Edge Coloring. In *Proceedings of the 19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2024.12.
- 7 Andreas Björklund. Determinant sums for undirected Hamiltonicity. *SIAM Journal on Computing*, 43(1):280–299, 2014. doi:10.1137/110839229.
- 8 Andreas Björklund and Thore Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica*, 52(2):226–249, December 2007. doi:10.1007/s00453-007-9149-8.
- 9 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed moebius inversion and graphs of bounded degree. *Theory of Computing Systems*, 47(3):637–654, January 2009. doi:10.1007/s00224-009-9185-7.
- 10 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *Journal of Computer and System Sciences*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.
- 11 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009. doi:10.1137/070683933.
- 12 M Blank. An estimate of the external stability number of a graph without suspended vertices. *Prikl. Mat. i Programirovanie*, 10:3–11, 1973.
- 13 Jesper Makhholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, November 2004. doi:10.1016/j.orl.2004.03.002.
- 14 Aleksander B. G. Christiansen, Eva Rotenberg, and Juliette Vlieghe. Sparsity-Parameterised Dynamic Edge Colouring. In *Proceedings of the 19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2024.20.

- 15 Ran Duan, Haoqing He, and Tianyi Zhang. *Dynamic Edge Coloring with Improved Approximation*, pages 1937–1945. Society for Industrial and Applied Mathematics, January 2019. doi:10.1137/1.9781611975482.117.
- 16 Eduard Eiben, Tomohiro Koana, and Magnus Wahlström. Determinantal sieving. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 377–423. SIAM, 2024. doi:10.1137/1.9781611977912.16.
- 17 David Eppstein. *Small Maximal Independent Sets and Faster Exact Graph Coloring*, pages 462–470. Springer Berlin Heidelberg, 2001. doi:10.1007/3-540-44634-6_42.
- 18 Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Families with infants: Speeding up algorithms for NP-hard problems using FFT. *ACM Trans. Algorithms*, 12(3):35:1–35:17, 2016. doi:10.1145/2847419.
- 19 David G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 700–724, 2019. doi:10.1109/FOCS.2019.00048.
- 20 Michael A. Henning. Bounds on domination parameters in graphs: a brief survey. *Discuss. Math. Graph Theory*, 42(3):665–708, 2022. doi:10.7151/DMGT.2454.
- 21 Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981. doi:10.1137/0210055.
- 22 Masao Ishikawa and Masato Wakayama. Minor summation formula of pfaffians. *Linear and Multilinear algebra*, 39(3):285–305, 1995.
- 23 Tomohiro Koana and Magnus Wahlström. Faster algorithms on linear delta-matroids. *arXiv preprint arXiv:2402.11596*, 2024. doi:10.48550/arXiv.2402.11596.
- 24 Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Process. Lett.*, 5(3):66–67, 1976. doi:10.1016/0020-0190(76)90065-X.
- 25 Moshe Lewenstein, Seth Pettie, and Virginia Vassilevska Williams. Structure and hardness in P (dagstuhl seminar 16451). *Dagstuhl Reports*, 6(11):1–34, 2016. doi:10.4230/DAGREP.6.11.1.
- 26 Daniel Lokshantov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Transactions on Algorithms*, 14(2):14:1–14:20, 2018. doi:10.1145/3170444.
- 27 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, September 1994. doi:10.1145/185675.306789.
- 28 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009. doi:10.1016/j.tcs.2009.07.027.
- 29 William McCuaig and Bruce Shepherd. Domination in graphs with minimum degree two. *Journal of Graph Theory*, 13(6):749–762, 1989. doi:10.1002/JGT.3190130610.
- 30 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, August 1995. doi:10.1017/cbo9780511814075.
- 31 Kazuo Murota. *Matrices and matroids for systems analysis*, volume 20. Springer Science & Business Media, 1999.
- 32 Oystein Ore. Theory of graphs. In *Colloquium Publications*. American Mathematical Society, 1962.
- 33 Charles Payan. Sur le nombre d’absorption d’un graphe simple, 1975.
- 34 Bruce A. Reed. Paths, stars and the number three. *Comb. Probab. Comput.*, 5:277–295, 1996. doi:10.1017/S0963548300002042.
- 35 Magnus Wahlström. Abusing the Tutte matrix: An algebraic instance compression for the K-set-cycle problem. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *LIPICs*, pages 341–352. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPICs.STACS.2013.341.
- 36 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letter*, 109(6):315–318, 2009. doi:10.1016/J.IPL.2008.11.004.

Tropical Proof Systems: Between R(CP) and Resolution

Yaroslav Alekseev  

Technion – Israel Institute of Technology, Haifa, Israel

Dima Grigoriev 

CNRS, Mathématique, Université de Lille, Villeneuve d’Ascq, 59655, France

Edward A. Hirsch  

Department of Computer Science, Ariel University, Israel

Abstract

Propositional proof complexity deals with the lengths of polynomial-time verifiable proofs for Boolean tautologies. An abundance of proof systems is known, including algebraic and semialgebraic systems, which work with polynomial equations and inequalities, respectively. The most basic algebraic proof system is based on Hilbert’s Nullstellensatz [7]. Tropical (“min-plus”) arithmetic has many applications in various areas of mathematics. The operations are the real addition (as the tropical multiplication) and the minimum (as the tropical addition). Recently, [8, 17, 21] demonstrated a version of Nullstellensatz in the tropical setting.

In this paper we introduce (semi)algebraic proof systems that use min-plus arithmetic. For the dual-variable encoding of Boolean variables (two tropical variables x and \bar{x} per one Boolean variable x) and $\{0, 1\}$ -encoding of the truth values, we prove that a *static* (Nullstellensatz-based) tropical proof system polynomially simulates *daglike* resolution and also has short proofs for the propositional pigeon-hole principle. Its dynamic version strengthened by an additional derivation rule (a tropical analogue of resolution by linear inequality) is equivalent to the system **Res(LP)** (aka **R(LP)**), which derives nonnegative linear combinations of linear inequalities; this latter system is known to polynomially simulate Krajíček’s **Res(CP)** (aka **R(CP)**) with unary coefficients. Therefore, tropical proof systems give a finer hierarchy of proof systems below **Res(LP)** for which we still do not have exponential lower bounds. While the “driving force” in **Res(LP)** is resolution by linear inequalities, dynamic tropical systems are driven solely by the transitivity of the order, and static tropical proof systems are based on reasoning about differences between the input linear functions. For the truth values encoded by $\{0, \infty\}$, dynamic tropical proofs are equivalent to **Res(∞)**, which is a small-depth Frege system called also DNF resolution.

Finally, we provide a lower bound on the size of derivations of a much simplified tropical version of the BINARY VALUE PRINCIPLE in a static tropical proof system. Also, we establish the non-deducibility of the tropical resolution rule in this system and discuss axioms for Boolean logic that do not use dual variables. In this extended abstract, full proofs are omitted.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Cutting Planes, Nullstellensatz refutations, Res(CP), semi-algebraic proofs, tropical proof systems, tropical semiring

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.8

Related Version *Full Version:* <https://ecc.weizmann.ac.il/report/2024/072/> [3]

Funding *Edward A. Hirsch:* This research was conducted with the support of the State of Israel, the Ministry of Immigrant Absorption, and the Center for the Absorption of Scientists.

Acknowledgements The authors are very grateful to Dmitry Itsykson and Dmitry Sokolov for fruitful discussions, and to Marc Vinyals for his inspiring Proof Complexity Zoo project and for pointing to the recent work of Gläser and Pfetsch.



© Yaroslav Alekseev, Dima Grigoriev, and Edward A. Hirsch;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 8; pp. 8:1–8:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction and Organization of this Extended Abstract

This paper introduces tropical proof systems, that is, proof systems that use min-plus arithmetic. To the best of our knowledge, these are the first tropical proof systems described in the literature though one of them is equivalent to a known proof system **Res**(LP) [19], which is a weakened version of **Res**(CP) (**R**(CP)) [23], these systems are working with disjunctions of inequalities. Thus our proof systems not only introduce a new paradigm, but also give a scale of proof systems between **Res**(CP) and resolution (this scale is visualised in Fig. 1).

In this extended abstract, we briefly recall the standard setup for propositional proof complexity and survey previous results concerning relevant proof systems (Sect. 2), recall tropical arithmetic (Sect. 3) and introduce tropical proof systems (Sect. 4), survey our results (Sect. 6), and discuss further directions (Sect. 7). Preliminary versions of full proofs can be found in the preprint [3].

2 General setup

2.1 Propositional proof complexity

A proof system for language L is¹ a deterministic polynomial-time algorithm V such that for every $x \in L$, there is a *proof* $\pi \in \{0, 1\}^*$ such that $V(x, \pi) = 1$, and for every $x \notin L$ and every candidate proof π , it holds that $V(x, \pi) = 0$. In this paper, we are interested in proofs for the language **UNSAT** of unsatisfiable Boolean formulas in conjunctive normal form (CNF) and, more broadly, in proofs for the language of unsolvable systems of linear equations (and even their disjunctions) with rational coefficients. Frequently (but not always) a proof of the unsatisfiability of a formula derives semantically implied statements from previously derived (or input) statements (in the case of a formula in CNF, the input statements are Boolean clauses). The existence of a proof system that has a polynomial-size proof for every $x \in L$ is equivalent to **NP** = **co-NP**, this equivalence of the classes is unlikely and proving or disproving it is far beyond the reach of the current methods.

Propositional proof complexity is a rapidly developing area where we typically prove four kinds of results:

- A superpolynomial lower bound on the size of the shortest proof for a certain (infinite) set of inputs $x_1, x_2 \dots \in L$ in some specific proof system.
- A polynomial *simulation* between two proof systems, that is, for every $x \in L$, one proof system has a proof of x that has the same or a smaller length (up to a polynomial factor) than the shortest proof of x in the other proof system.
- A polynomial upper bound on the proof size for a certain (infinite) set of inputs $x_1, x_2 \dots \in L$ in a specific proof system.
- A superpolynomial *separation* between proof systems, which is typically obtained by providing a set of inputs for which we can prove a polynomial size upper bound in one proof system and a superpolynomial size lower bound in another proof system.

When we have both a polynomial simulation and a superpolynomial separation between two specific systems, we say that one system is strictly stronger than the other one. Thus proof systems (for the same language) form a lattice² with respect to the partial non-strict order composed of the simulations; if one system is strictly stronger than the other one, then the

¹ This definition deviates from Cook and Reckhow's definition [11], but for our purpose they are equivalent.

² Note that we do not require the simulations to be computable in polynomial time (p-simulations) though in most cases the simulations are indeed efficient.

order is strict for these two systems. *Cook's (or Cook-Reckhow's) program* in the propositional proof complexity is the intention for proving superpolynomial lower bounds for stronger and stronger proof systems, thus developing new methods for proving superpolynomial lower bounds, which are the crux of computational complexity.

2.2 Previous proof complexity results relevant to this paper

The area essentially started with superpolynomial (and then exponential) lower bounds for various versions of the resolution proof system [31, 18, 32], where proofs proceed by deriving the resolvent $C \vee D$ of two already derived (or input) clauses $C \vee x$ and $D \vee \bar{x}$ until we derive the empty clause. We refer the reader to Krajíček's book [25] for a detailed overview of the area.

The previously known proof systems that are the most relevant to us are proof systems that work with linear inequalities.

The Cutting Plane (CP) proof system [12] uses the rounding rule and nonnegative linear combinations of inequalities

$$\frac{\sum_i ca_i x_i - d \geq 0}{\sum_i a_i x_i - \lceil d/c \rceil \geq 0} \quad (c, a_i, d \in \mathbb{Z}), \quad \frac{f_1 \geq 0 \quad f_2 \geq 0}{\alpha_1 f_1 + \alpha_2 f_2 \geq 0} \quad (\alpha_1, \alpha_2 > 0)$$

as its derivation steps (here f_1 and f_2 are the affine forms). The derivation starts from linear inequalities expressing Boolean clauses and from the axioms $x_i \geq 0$, $1 - x_i \geq 0$ for every variable x_i . It finishes with deriving the contradictory inequality $-1 \geq 0$. An exponential lower bound for CP was proved in [29].

Krajíček [23] generalized CP to a new system $\mathbf{R}(\mathbf{CP})$, also called $\mathbf{Res}(\mathbf{CP})$, by allowing to reason about disjunctions of *affine* inequalities (the disjunctions are interpreted as sets, trivially false constant inequalities are dropped out, and a disjunction can be weakened by adding new inequalities to it). The two rules above are generalized to

$$\frac{\sum_i ca_i x_i - d \geq 0 \vee \Gamma}{\sum_i a_i x_i - \lceil d/c \rceil \geq 0 \vee \Gamma} \quad (c, a_i, d \in \mathbb{Z}), \quad \frac{(f_1 \geq 0) \vee \Gamma \quad (f_2 \geq 0) \vee \Gamma}{(\alpha_1 f_1 + \alpha_2 f_2 \geq 0) \vee \Gamma} \quad (\alpha_1, \alpha_2 > 0). \quad (+\mathbf{RES})$$

Also the notion of the negation of an inequality is introduced through the rule

$$\frac{\emptyset}{f - 1 \geq 0 \vee -f \geq 0}.$$

The same idea has been used to define other proof systems (for example, $\mathbf{Res}(k)$ [24], $\mathbf{Res}(\mathbf{Lin})$ [30], $\mathbf{Res}(\oplus)$ [20]). In particular, Hirsch and Kojevnikov stripped $\mathbf{Res}(\mathbf{CP})$ of the negation and the rounding rule and defined the system $\mathbf{Res}(\mathbf{LP})$ that uses the splitting rule

$$\frac{\emptyset}{(x - 1 \geq 0) \vee (-x \geq 0)} \quad (x \text{ is a variable}).$$

No superpolynomial size lower bounds for $\mathbf{Res}(\mathbf{CP})$ or $\mathbf{Res}(\mathbf{LP})$ are known. Derivations can be daglike (as usual) or treelike (where we have to re-derive a statement again every time we use it). Beame et al. defined the Stabbing Planes proof system [6] that is equivalent to **treelike** $\mathbf{Res}(\mathbf{CP})$. While usually the coefficients of linear inequalities are written in *binary*, one can consider weaker proof systems when they are written in *unary*. In the unary coefficients setting, Fleming et al. have shown a quasipolynomial simulation of Stabbing Planes in CP (with binary coefficients) thus obtaining an exponential lower bound for it [14]. Very recently, Gläser and Pfetsch have shown an exponential bound for Stabbing Planes for the case of binary coefficients by providing a quasipolynomial monotone interpolation [15]. The daglike versions of $\mathbf{Res}(\mathbf{CP})$ and $\mathbf{Res}(\mathbf{LP})$ with unary coefficients are polynomially equivalent [19] and no superpolynomial lower bounds are known for them to the date.

3 Tropical arithmetic

Tropical (or min-plus) arithmetic involves operations $\min, +$ in place of $+, \times$ in classical arithmetic; we refer the interested reader to [27] for the introduction and survey of tropical arithmetic and in particular its history and the origin of the name “tropical”. Tropical arithmetic has several sources including algebraic geometry (valuations), mathematical physics, and optimization, and, respectively, numerous applications (some of them can be found in [27], also neural networks are a more recent application).

We consider a tropical semifield based on $\mathbb{Q} \cup \{+\infty\}$. Many of the results of this paper can be also formulated and proved using a similar semifield based on \mathbb{Q} .

Tropical operations. We consider the min-plus (or tropical) semifield defined by the set $\mathbb{Q}_\infty = \mathbb{Q} \cup \{+\infty\}$ endowed with two operations: the tropical addition \oplus and the tropical multiplication \odot defined in the following way:

$$a \oplus b = \min\{a, b\}, \quad a \odot b = a + b,$$

where \min and $+$ are the usual (traditional) arithmetic operations extended to work with the neutral element ∞ : namely, $a \oplus \infty = a$ and $a \odot \infty = \infty$. A tropical *power* n of a is defined as

$$a^{\odot 0} = 0, \quad a^{\odot n} = \underbrace{a \odot \dots \odot a}_{n \text{ copies}},$$

where n is a positive integer. Sometimes we use a bigger \bigoplus to facilitate reading.

Tropical polynomials.

► **Definition 1.** A tropical monomial is a tropical product of tropical powers of variables. For a vector of variables $\vec{x} = (x_1, \dots, x_n)$ and a vector of integers $I = (i_1, \dots, i_n)$ we introduce the notation

$$\vec{x}^I = x_1^{\odot i_1} \odot \dots \odot x_n^{\odot i_n}.$$

Then $\text{degtr}(\vec{x}^I) = i_1 + \dots + i_n$ is called the (total) (tropical) degree of this monomial.

Note that we never use the word “monomial” for a submonomial, that is, a subset of monomial (in other words, the monomial $x \odot y$ does *not* occur in $x \odot y^{\odot 2} \oplus x \odot y \odot z$).

In this paper, a (tropical, or min-plus) *term* $t = c \odot m$ is a tropical product of a tropical monomial m and a constant $c \in \mathbb{Q}_\infty$. One can treat a tropical term classically as a linear function $a + i_1 x_1 + \dots + i_n x_n$. By analogy with the traditional arithmetic (and its zero), a constant term is the only situation where the coefficient $c = \infty$ is meaningful (since $\infty \odot m = \infty$). We assume that if a term is non-constant, it has a finite coefficient. This is important when we say “monomial m occurs” somewhere: we mean that a term based on m occurs. The degree of a term is the degree of its monomial.

Note that when we work with constants, we use traditional operations and treat the constant as a whole, for example, $x \odot (10 - 2 + 1)$ is the same as $x \odot 9$.

► **Definition 2.** Let x_1, \dots, x_n be variables, and let \mathcal{I} be a finite set of their power vectors ($\mathcal{I} \subseteq \mathbb{N}_0^n$). A tropical polynomial is an element of $(\mathbb{Q}_\infty, \oplus, \odot)[x_1, \dots, x_n]$, that is, the tropical sum of a set of tropical terms $t_I(\vec{x}) = c_I \odot \vec{x}^I$ with distinct power vectors $I \in \mathcal{I}$:

$$f(\vec{x}) = \bigoplus_{I \in \mathcal{I}} t_I(\vec{x}).$$

If $\mathcal{I} = \emptyset$, we identify this polynomial with the constant polynomial ∞ .

In other words, tropical polynomials are members of the $(\mathbb{Q}_\infty, \oplus, \odot)$ -linear space spanned by the monomials (for example, $1 \oplus x^{\odot 2} \odot y \oplus 2 \odot x$ and ∞ are tropical polynomials). One can treat f as a concave piecewise linear function.

Tropical addition and multiplication are correctly defined on tropical polynomials as $\infty \odot m = \infty$ and $a \odot m \oplus b \odot m = \min\{a, b\} \odot m$ for any monomial m , and thus we never need more than one term per monomial.

Complexity of tropical polynomials. We usually write the coefficients and the exponents in binary, so the bit-size of x^{2^n} is polynomial (which is important when we estimate the size of a proof that uses tropical polynomials). The degree of a tropical polynomial f , denoted by $\text{degtr}(f)$, is the maximal degree of its terms. Let $\mu(f)$ be the number of terms in f (it is strictly positive).

Min-plus polynomials and inequalities. A *min-plus polynomial* is a pair of tropical polynomials $(f(\vec{x}), g(\vec{x}))$. The degree of a min-plus polynomial is the maximum of the degrees of f and g . A point $\vec{a} \in \mathcal{R}^n$ is a root of this polynomial if the following equality holds: $f(\vec{a}) = g(\vec{a})$. We can apply tropical operations component-wise to min-plus polynomials, thus min-plus polynomials can be summed using the tropical addition \oplus and can be tropically multiplied by a tropical monomial using tropical multiplication \odot , and these operations preserve the common roots of the involved polynomials. Thus, the closure of a set of min-plus polynomials under these operations is a (tropical) ideal. One of the central issues in tropical algebra is a criterion for the existence of common roots for systems of min-plus polynomials $\{(f_1, g_1), \dots, (f_k, g_k)\}$. In classical algebra such a criterion is provided by Hilbert’s Nullstellensatz (over an algebraically closed field). In tropical algebra a criterion of solvability has been formulated as a Min-Plus Nullstellensatz [8, 17, 21, 26], further extended in [1].

In this paper we will deal with a more convenient (albeit equivalent) framework of the problem of the existence of common roots for systems of min-plus polynomial inequalities $\{f_1 \leq g_1, \dots, f_k \leq g_k\}$. A *min-plus polynomial inequality* is a pair of tropical polynomials f, g that we write as $f \leq g$ or (f, g) . A point $\vec{a} \in \mathcal{R}^n$ is a root of min-plus inequality $f \leq g$ if $f(\vec{a}) \leq g(\vec{a})$. Note that \vec{a} is a root of $f \leq g$ iff it is a root of $(f \oplus g, f)$. In what follows, we abuse the notation by writing $f = g$ instead of the two inequalities $f \leq g$ and $g \leq f$.

Note that while one can consider solving min-plus equations and inequalities over \mathbb{Q} or over \mathbb{Q}_∞ , tropical polynomials will always have coefficients in \mathbb{Q}_∞ , in particular, ∞ is a tropical polynomial equivalent to “the empty tropical polynomial”.

An order on tropical polynomials. For tropical polynomials L and R , let $L \succeq R$ denote the component-wise \geq of the coefficients of the respective monomials in L and R .

Let $L \succ R$ denote the component-wise $>$ of the coefficients of the respective monomials in L and R , where R may also contain extra monomials not present in L .

Note that if $L \succ R$, then it is impossible for $R \leq L$ to have finite roots.

We define \preceq and \prec similarly. The following lemma is easy to see.

► **Lemma 3** (\succ inside \oplus, \odot). *Let $\Gamma, \Delta, \Gamma', \Delta'$ be tropical polynomials.*

1. *If $\Gamma \succ \Delta$ and $\Gamma' \succ \Delta'$, then $\Gamma \oplus \Gamma' \succ \Delta \oplus \Delta'$.*
2. *For a tropical term $t \neq \infty$, if $\Gamma \succ \Delta$, then $\Gamma \odot t \succ \Delta \odot t$.*

4 Tropical proof systems

Similarly to the already classical “algebraic” proof systems Nullstellensatz and Polynomial Calculus [7, 10] based on Hilbert’s Nullstellensatz, we introduce proof systems that rely on the Min-Plus Nullstellensatz. The most general static proof system MP-NS (Min-Plus Nullstellensatz, Definition 7) for the language of unsolvable linear inequalities requires a proof that is a contradictory algebraic combination of the input inequalities and trivial axioms $0 \leq 0$, $f \leq \infty$. That is, for a system of min-plus inequalities $f_i \leq g_i$, the proof is a contradictory inequality $\bigoplus_{j=1}^K p_j \leq \bigoplus_{j=1}^K q_j$ for some $K \geq 1$, where for each $1 \leq j \leq K$ we have $(p_j, q_j) = (t_j \odot f_{i_j}, t_j \odot g_{i_j})$ for some term t_j and some $1 \leq i_j \leq k$. The contradiction must follow immediately from the coefficients of the inequality; namely, for every monomial present in the left-hand side, its coefficient must be strictly greater than the coefficient of the same monomial in the right-hand side (also, for technical reasons the right-hand side must have a finite constant term).

For example, the system of inequalities $\{x \leq y, y \leq z, z + 1 \leq x, 2x \leq 0\}$, which is written tropically as $\{x \leq y, y \leq z, z \odot 1 \leq x, x^{\odot 2} \leq 0\}$, can be refuted by tropically multiplying its inequalities by $x \odot \frac{1}{3}$, $x \odot \frac{2}{3}$, x , and $\frac{1}{3}$, respectively. This results in

$$\underline{x^{\odot 2} \odot \frac{1}{3}} \oplus \underline{y \odot x \odot \frac{2}{3}} \oplus \underline{z \odot x \odot 1} \leq \underline{y \odot x \odot \frac{1}{3}} \oplus \underline{z \odot x \odot \frac{2}{3}} \oplus \underline{x^{\odot 2} \odot 0} \oplus \underline{\frac{1}{3}}.$$

One can see that the requirement on the coefficients is satisfied.

Similarly to algebraic proof systems, we introduce a dynamic version of MP-NS: Min-Plus Polynomial Calculus (MP-PC), see Definition 11. It derives the contradiction of the same sort step by step by tropically adding inequalities, tropically multiplying them by terms, and substituting inequalities into other inequalities.

We also consider the additional *tropical resolution* rule

$$\frac{t \oplus f \leq 0 \quad t' \oplus f \leq 0}{(t \odot t') \oplus f \leq 0}, \text{ where } t, t' \text{ are terms,} \quad (\odot\text{RES})$$

which is a counterpart of (+RES) in Res(LP) and Res(CP). When we add this rule to our systems, we mention this explicitly. While this rule is not needed for the completeness of our tropical proof systems, on the one hand, and is looking very natural, on the other hand, its elimination from the system may be expensive, as shown in Theorem 27.

The proof systems MP-NS and MP-PC can be transformed into proof systems for UNSAT using several possibilities to encode the truth values, Boolean variables and Boolean clauses. In the “default” setting, we encode the truth values by $\{0, 1\}$, introduce the dual variable \bar{x} for every variable x , and transform a clause into the corresponding linear inequality (which, in tropical terms, is $1 \leq m$ for a multilinear monomial m). These proof systems are called MP-NSR and MP-PCR; the diagram of connections between them and known systems is given in Figure 1. One can also considered different encodings (without dual variables or with values in $\{0, \infty\}$), the detailed treatment of these is delayed to the full version of the paper.

4.1 The basic static proof system, MP-NS

► **Definition 4.** Consider a system of min-plus polynomials $F = \{(f_1, g_1), \dots, (f_k, g_k)\}$. An algebraic combination of F is a min-plus polynomial (f, g) that can be represented as

$$(f, g) = \left(\bigoplus_{j=1}^K p_j, \bigoplus_{j=1}^K q_j \right), \quad (1)$$

for some $K \geq 1$, where for each $1 \leq j \leq K$ we have $(p_j, q_j) = (t_j \odot f_{i_j}, t_j \odot g_{i_j})$ for some term t_j and some $1 \leq i_j \leq k$. We will abuse the language by calling an “algebraic combination” both the min-plus polynomial (f, g) and the composition (1), that is, t_j ’s.

We call a system of min-plus polynomials *symmetric* if it always includes (f_i, g_i) together with (g_i, f_i) . The possibility of refuting min-plus systems of equations (and inequalities) using min-plus proofs is based on the following theorem.

► **Theorem 5** (Min-Plus Nullstellensatz, [17, Theorem 3.8] over \mathbb{Q}_∞ without the degree claim). *Consider a symmetric system of min-plus polynomial equations F as in Def. 4 in n variables.*

The system F has no roots over the tropical semifield \mathbb{Q}_∞ iff we can construct an algebraic min-plus combination

$$(f, g) = \left(\bigoplus_{j=1}^K p_j, \bigoplus_{j=1}^K q_j \right)$$

of F such that for each monomial m occurring in f , and also for the constant monomial even if it is infinite, its coefficient in f is greater than the coefficient of this monomial in g (in particular, m must be present in g).

It can be easily observed that one direction of the theorem is trivial: indeed, if there is an algebraic combination (f, g) satisfying the conditions of the theorem (recall also that in terms of integer operations, the “coefficient” is the additive constant in a standard arithmetic linear combination of variables), so the system F is unsatisfiable. The finite constant term in g saves us from the parasite all- ∞ solution.

It is easy to see that in the case of systems of inequalities (which correspond to not necessarily symmetric systems of min-plus polynomials), a similar result holds as a corollary.

► **Theorem 6.** *Consider a system of min-plus polynomial inequalities S in n variables over \mathbb{Q}_∞ . Let $F = S \cup \{(0, 0)\} \cup \{(g_i, \infty) \mid (f_i, g_i) \in S\}$.*

The system S has no roots over the tropical semifield \mathbb{Q} iff we can construct an algebraic min-plus combination

$$(f, g) = \left(\bigoplus_{j=1}^K p_j, \bigoplus_{j=1}^K q_j \right)$$

of F (in terms of Def. 4) such that for each monomial $m \neq \infty$ occurring in f , its coefficient is greater than the coefficient of this monomial in g (in particular, m must be present in g).

Over the semifield \mathbb{Q}_∞ we need an additional property: the constant term in g is finite.

► **Definition 7** (Min-Plus Nullstellensatz, MP-NS). *We will call a min-plus algebraic combination (that is, $(t_j)_{j=1}^K$ in terms Def. 4) satisfying the conditions of Theorem 6 (over \mathbb{Q}_∞ , unless otherwise stated) a Min-Plus Nullstellensatz (MP-NS) refutation of S .*

► **Note 8** (The $0 \leq c$ “axiom”). In Theorem 6 we have added the axioms $0 \leq 0$ and $g \leq \infty$ to MP-NS. Note that, for any constant $c \geq 0$, the inequality $0 \leq c$ can be easily derived as a tropical sum of $0 \leq \infty$ and $0 \leq 0$ tropically multiplied by c . In what follows we will use it without further notice both for MP-NS and for our dynamic proof system described later.

In fact, the “last line” of the proof (that is, (f, g) in terms of the theorem, after combining similar terms) can be thought of as a refutation itself: the composition of this algebraic combination can be easily reconstructed, and its complexity parameters are bounded by a

polynomial in the complexity of (f, g) . (In what follows, when we speak about the size of a rational number, we mean the size of its nominator plus the size of its denominator; for ∞ this is zero.)

► **Proposition 9** (MP-NS derivation reconstruction). *Given a system of min-plus inequalities (f_i, g_i) and given their algebraic combination as two polynomials (f, g) , we can find the terms t_j 's of this algebraic combination in polynomial time, their number is bounded by a polynomial in the number of monomials in the system and (f, g) , their coefficient size is bounded by a polynomial in the size of coefficients in the system and (f, g) . and their degree does not exceed the degree of monomials in f and g .*

► **Remark 10.** Now we say a few words about the complexity of constructing an MP-NS refutation given a system of min-plus equations, or more generally, inequalities (including strict ones). First, one can estimate a bound on the degree of a refutation (algebraic combination) with the help of [17, Theorem 3.8] in the case of min-plus equations, which was extended to the case of min-plus inequalities in Theorem 3.1 [1]. For a given bound on the degree, an algebraic combination can be treated as a system of min-plus **strict linear** inequalities (whose unknowns are the rational coefficients of an algebraic combination). Solvability of a system of min-plus linear inequalities (including strict ones) is reduced in [5] (within polynomial complexity) to solvability of a system of min-plus linear equations (with integer coefficients and thereby, integer unknowns).

One can apply to a system of min-plus linear equations one of the algorithms to solve it (see e.g., [9, 2, 16]). The complexity of each of these algorithms is polynomial in the number of unknowns and equations and in the absolute values of integer coefficients of the system. Observe that this complexity bound is not polynomial in the size of the input because the complexity depends on the absolute values of the coefficients rather than on their bit-sizes. It is a longstanding open problem whether a system of min-plus linear equations is solvable within polynomial complexity. However, this problem belongs to the class $\mathbf{NP} \cap \mathbf{co-NP}$ (see e.g., [2, 16]).

4.2 The basic dynamic proof system, MP-PC

We also consider a dynamic version of MP-NS called the Min-Plus Polynomial Calculus (MP-PC). It has some informal resemblance to Krajíček's original quantifier-free propositional LK(CP) proof system [23]: it uses both sides of a “sequent” while Res(CP) used only one because of the presence of an efficient negation, which we are missing. However, the “sequents” of our system contain tropical terms (affine functions) and not inequalities.

We will sometimes use equations to abbreviate pairs of two opposite inequalities.

► **Definition 11** (Min-Plus Polynomial Calculus, MP-PC). *Consider a system of min-plus polynomial inequalities $S = \{f_1 \leq g_1, \dots, f_m \leq g_m\}$ in n variables. A Min-Plus PC (MP-PC) refutation of F is a list of min-plus inequalities*

$$p_1 \leq q_1, \dots, p_K \leq q_K$$

such that

1. In the last inequality, for each monomial $m = x_1^{\odot j_1} \odot \dots \odot x_n^{\odot j_n}$ in p_K there is a matching monomial in q_K , and the coefficient in the monomial in p_K is greater than the corresponding coefficient in q_K . Moreover, the constant term in q_K must be present and must be finite.

► **Note 12** (0/1 variables in \mathbb{Q}_∞). Later on, in our systems dealing with $\{0, 1\}$ variables, all monomials become bounded from the above and thus the requirement on the constant term can be satisfied automatically.

2. Each inequality $p_i \leq q_i$ is obtained from the previously derived inequalities using the following rules.

Axioms:

$$\frac{\emptyset}{f_j \leq g_j}, \quad \text{where } 1 \leq j \leq m,$$

$$\frac{\emptyset}{0 \leq 0},$$

$$\frac{\emptyset}{p \leq \infty}, \quad \text{for any tropical polynomial } p. \quad (\text{WEAK})$$

Minimum. We can take a minimum of two previously derived inequalities:

$$\frac{p \leq q \quad p' \leq q'}{p \oplus p' \leq q \oplus q'}.$$

Tropical multiplication:

$$\frac{p \leq q}{p \odot t \leq q \odot t},$$

where t is a term.

Transitivity of the order:

$$\frac{p \leq h \quad h \leq r}{p \leq r}.$$

► **Note 13** (Substitutions). It is easy to see that by combining transitivity with other rules we can substitute inequalities into each other on the left or on the right, for example,

$$\frac{p \oplus h \leq q \quad r \leq h}{p \oplus r \leq q} \quad \frac{p \leq q \oplus h \quad h \leq r}{p \leq q \oplus r}.$$

and do it even inside monomials by multiplying the substitution by an appropriate term. In what follows, we refer to these derivations as **substitutions**.

► **Note 14** (Tropical product). Note that we can take the tropical product (\odot) of two inequalities by first multiplying one of them by the left-hand-side of the other one and then applying the transitivity rule. We will discuss the complexity issues of constructing tropical products of inequalities later.

► **Note 15** (Natural weakening). Note also that we can weaken inequalities by dropping a summand from the right-hand side or adding a summand to the left-hand side. This is simulated using the (WEAK) axiom with the minimum rule and substitution on the right.

► **Note 16** (Systems based on equations instead of inequalities). Similarly to Theorem 5, it is possible to talk about symmetric systems and thus min-plus equations, even in the context of dynamic proof systems. For example, one could define a refutation system for symmetric systems with the same rules except for the axiom (WEAK), and with an additional rule to swap an equation ($f = g \rightarrow g = f$). This apparently weaker proof system turns out to be polynomially equivalent to MP-PC for symmetric systems (see full version of this paper). As inequalities provide a more natural framework and allow to refute more unsolvable systems, we stick to using inequalities.

4.3 The tropical resolution rule

As usual, when we consider stronger systems that include additional rules, we denote them by “system+rule”, for example, MP-PC+(\odot RES).

In what follows f denotes any tropical monomial.

The following rule can be viewed as the generalization of the resolution-like rule (+RES) in Res(CP)-like systems (though we limit it very much) or as tropical multiplication of two inequalities, each one being in the tropical sum.

$$\frac{t \oplus f \leq 0 \quad t' \oplus f \leq 0}{(t \odot t') \oplus f \leq 0}, \text{ where } t, t' \text{ are terms.} \quad (\odot\text{RES})$$

While this rule is looking very natural, we do not know how to eliminate its use at a polynomial cost. Moreover, we will show later that its direct simulation in the static proof systems is impossible.

4.4 Encoding of Boolean logic: MP-NSR and MP-PCR systems

In this extended abstract we concentrate on the following encoding. We use 0 for **false** and 1 for **true** and introduce a “dual variable” for the negation of each variable. Note that we use $\neg\Phi$ as the Boolean negation of a Boolean formula Φ (without distinguishing Φ from $\neg\neg\Phi$) while keeping the notation \bar{x} for dual variables. Recall that we denote literals (which are variables or the negations of variables) by $\ell, \ell_1, \ell_2, \dots$ without further notice.

We thus obtain from (the systems for refuting conjunctions of min-plus inequalities) MP-NS and MP-PC the propositional proof systems MP-NSR and MP-PCR, respectively. In order to do that we translate a formula in CNF into a conjunction of tropical inequalities. Namely, we translate each clause into an inequality and also add additional inequalities (axioms) to ensure that variables are Boolean.

Boolean axioms. We include the axioms

$$\frac{\emptyset}{x \odot \bar{x} = 1} \quad (01/\odot) \qquad \frac{\emptyset}{x \oplus \bar{x} = 0} \quad (01/\oplus)$$

to ensure that x and \bar{x} are dual and in $\{0, 1\}$.

► **Note 17.** For any binary variable x we can derive from $(01/\oplus)$ in MP-PC that $0 \leq x$ and $x \leq 1$. The first inequality can be derived from $0 \leq x \oplus \bar{x}$ by simplification. The latter inequality can be derived in the following way: from $0 \leq \bar{x}$ we can derive $x \leq x \odot \bar{x}$, from which we can derive $x \leq 1$ using $x \odot \bar{x} \leq 1$.

Translations of Boolean clauses. We can encode a Boolean clause $\ell_1 \vee \ell_2 \vee \dots \vee \ell_k$ using the equation

$$\bar{\ell}_1 \oplus \bar{\ell}_2 \oplus \dots \oplus \bar{\ell}_k \leq 0. \quad (\text{D})$$

Note that clauses are encoded in Res(Lin) [30] in exactly the same way (the absence of the dual variables does not matter as re-encoding is done by a simple linear substitution).

However, there is another possibility to encode a clause, which is used in CP and similar proof systems:

$$1 \leq \ell_1 \odot \ell_2 \odot \dots \odot \ell_k. \quad (\text{I})$$

It is not difficult to see that in the case of MP-PCR these encodings are equivalent. A formal proof of this statement can be found in the full version of the paper.

5 Preliminary lemmas and the equivalence of encodings

Before we come to the main results, we state several technical lemmas about derivations in tropical systems.

► **Lemma 18** (tropical product, treelike, no axioms). *For $1 \leq i \leq k$, let A_i, B_i be tropical terms. Then there is a treelike MP-PC derivation of all the inequalities*

$$\bigodot_{i=1}^j A_i \leq \bigodot_{i=1}^j B_i, \text{ for } j \leq k,$$

from the inequalities $A_i \leq B_i$ (each used once).

The derivation contains $O(k)$ (not necessarily different) terms and the bit-size of every coefficient (respectively, the tropical exponent) in the derivation is upper bounded by $O(k \cdot b)$, where b is the maximum bit-size of a coefficient (respectively, a tropical exponent) in any of the A_i, B_i . In particular, the coefficients (resp., tropical exponents) in the derivation are sums of the original coefficients (resp., tropical exponents).

Proof. Start with $A_1 \leq B_1$. Tropically multiply it by A_2 and tropically multiply $A_2 \leq B_2$ by B_1 to conclude $A_1 \odot A_2 \leq B_1 \odot B_2$ by transitivity.

Continue in the same way for $j = 3, \dots, k$, multiplying $A_j \leq B_j$ by $\bigodot_{i=1}^{j-1} A_i$, multiplying the previously derived $\bigodot_{i=1}^{j-1} A_i \leq \bigodot_{i=1}^{j-1} B_i$ by B_j , and applying the transitivity rule. ◀

► **Lemma 19** (powers of axioms, treelike). *For a variable x and an integer $b > 0$, there are treelike MP-PCR derivations from the axioms of the following inequalities:*

$$x^{\odot b} \odot \bar{x}^{\odot b} \leq b, \tag{2}$$

$$b \leq x^{\odot b} \odot \bar{x}^{\odot b}, \tag{3}$$

$$x^{\odot b} \oplus \bar{x}^{\odot i} \leq 0 \text{ (as well as of the symmetrical inequality), for } 1 \leq i \leq b. \tag{4}$$

The tropical degree of these derivations is $O(b)$. The derivations of (2), (3) contain $O(b)$ (not necessarily distinct) terms, all the coefficients in them and constants are $\leq b$. The derivation of (4) contains $O(b^3)$ (not necessarily distinct) terms, all the coefficients in it are zeroes.

Proof. The inequalities (2) and (3) follow from Lemma 18.

To show (4) for $i = 1$, proceed by induction on b (starting with $b = 1$). Tropically multiply the axiom $x \oplus \bar{x} \leq 0$ by $x^{\odot(b-1)}$ and substitute $\bar{x} \leq \bar{x} \odot x^{\odot(b-1)}$ (which is $0 \leq x^{\odot(b-1)}$, provided by Lemma 18, multiplied by \bar{x}) into its left-hand side getting

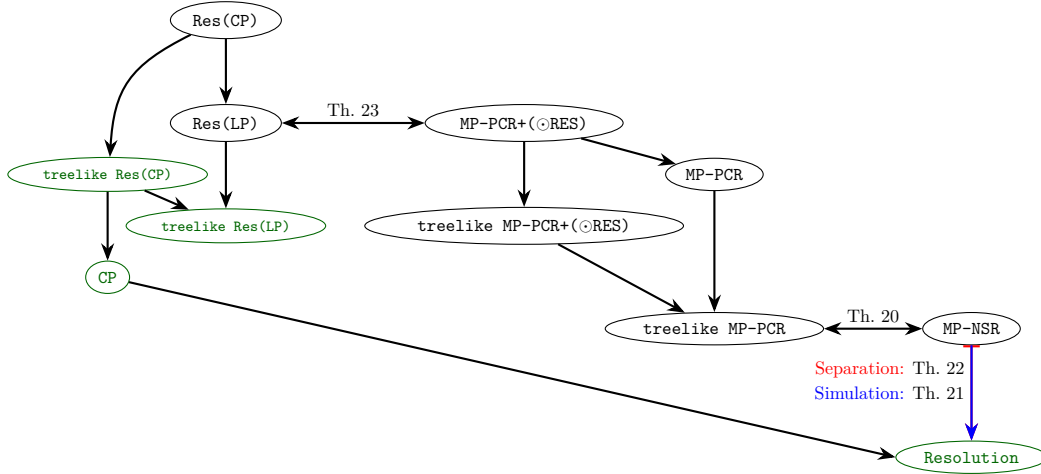
$$x^{\odot b} \oplus \bar{x} \leq x^{\odot(b-1)}.$$

Tropically add \bar{x} to both sides and substitute the induction hypothesis for $b - 1$ on the right obtaining the desired inequality.

The inequality $x^{\odot b} \oplus \bar{x} \leq 0$ provided by the previous argument is the starting point for deriving (4), now by the induction on i (where $i = 1$ is the base). Take it and tropically multiply it by $\bar{x}^{\odot(i-1)}$ obtaining

$$x^{\odot b} \odot \bar{x}^{\odot(i-1)} \oplus \bar{x}^{\odot i} \leq \bar{x}^{\odot(i-1)}.$$

Tropically add $x^{\odot b}$ to both sides, substitute the induction hypothesis on the right. Substitute $x^{\odot b} \leq x^{\odot b} \odot \bar{x}^{\odot(i-1)}$ (which is $0 \leq \bar{x}^{\odot(i-1)}$, provided by Lemma 18, multiplied by $x^{\odot b}$) on the left. ◀



■ **Figure 1** Map of tropical systems with $\{0, 1\}$ dual encoding. An arrow $\Pi \rightarrow \Psi$ means polynomial simulation of Π by Ψ . Proof systems known to be not polynomially bounded are shown in green.

6 Our results and methods

6.1 Static systems: Already stronger than Resolution

Static tropical proofs with dual variables over $\{0, 1\}$ turn out to be surprisingly powerful. We start with a natural statement that static tropical proofs are equivalent to treelike proofs:

► **Theorem 20.** *MP-NS polynomially simulates treelike MP-PC.*

Proof Sketch. Simulating a treelike tropical proof is done by combining the steps of the proof in a single algebraic combination with *decreasing coefficients*. Namely, when we go down (towards the root) the proof tree, which becomes now the formula tree of the algebraic combination, we tropically multiply subformulas by small positive coefficients. A similar idea is demonstrated in more detail in the proof of the next Theorem 21. ◀

Note that when we convert a treelike proof into a formula, the tropical multiplication of a tropical polynomial p applies to the whole subtree deriving p , thus this strategy does not work for simulating daglike tropical proofs (if p is multiplied by different terms t_i , we need to repeat it as many times).

However, we show that MP-NSR polynomially simulates *daglike* resolution. While the simulation of the treelike tropical proof is technical but intuitively straightforward, the simulation of the Resolution proof system is trickier due to its daglike nature.

► **Theorem 21.** *MP-NSR polynomially simulates Resolution.*

Proof. We simulate a resolution proof by putting it into the static proof step by step. For a disjunction $A = \ell_1 \vee \dots \vee \ell_k$, define its translation $[A] = 0 \odot \ell_1 \odot \dots \odot \ell_k$ with the meaning that it is true iff $[A] > 0$. In particular, every initial clause A is translated into $1 \leq [A]$, as we expect in MP-NSR.

Translate a Resolution proof into an MP-NSR proof as follows. Let s be the number of steps in the Resolution proof. We can assume that steps can be of two kinds: a resolution step

$$\frac{A \vee x \quad A \vee \neg x}{A}, \quad (5)$$

and a weakening step

$$\frac{A}{A \vee \ell}, \tag{6}$$

where A is a clause, x is a variable, and ℓ is a literal.

We now compose our algebraic combination.

For every initial clause A , we take its translation $1 \leq [A]$:

$$1 \leq 0 \odot [A]. \tag{7}$$

At step $i = 1, 2, \dots, s$, we do the following. Let $c_i = i/(s + 1)$.

- For a resolution step, multiply the axiom $x \oplus \bar{x} \leq 0$ by $c_i \odot [A]$ obtaining

$$c_i \odot x \odot [A] \oplus c_i \odot \bar{x} \odot [A] \leq c_i \odot [A]. \tag{8}$$

Observe that $[A \vee v] = [A] \odot v$ for every variable v , so the terms in (8) are exactly the translation of (5) multiplied by c_i .

- For a weakening step, we would like to multiply $0 \leq [\ell]$ by $c_i \odot [A]$ obtaining

$$c_i \odot [A] \leq c_i \odot [A] \odot [\ell]. \tag{9}$$

Observe that the terms in (9) are exactly the translation of (6) multiplied by c_i .

Strictly speaking, $0 \leq [\ell]$ is not an axiom, while $0 \leq [\neg\ell] \oplus [\ell]$ is. Formally, we must multiply the latter (rather than the former) by $c_i \odot [A]$. However, this leaves only extra terms in the right-hand side compared to (9), which cannot harm our MP-NSR refutation.

Note that the last step's right-hand side is $c_s \odot [\emptyset]$, that is, $1 - 1/(s + 1)$.

Our algebraic combination is a tropical sum of all inequalities (7) (for all the initial clauses A), (8), and (9) (for all steps of the Resolution proof).

The constant terms of the combination are 1 in the left-hand side, from the initial clauses, and $1 - 1/(s + 1)$ in the right-hand side, from the last clause; $1 > 1 - 1/(s + 1)$.

Every other monomial in the left-hand side has its counterpart in the right-hand side, from the previous steps of the proof. The coefficient is smaller in the simulation of the previous steps and in the initial clauses (thus, on the right-hand side).

It is clear that the total number of terms appearing in the proof is bounded by a polynomial in s , the degree of every monomial is bounded by the width of the resolution proof, and the nominators and denominators of the rational coefficients are also bounded by a polynomial in s . ◀

We also show that MP-NSR has polynomial-size proofs for *the propositional pigeonhole principle*, thus it is strictly stronger than Resolution. We consider the following translation of the pigeonhole principle:

$$1 \leq \bigodot_{j=1}^n x_{ij} \text{ for } 1 \leq i \leq m,$$

$$1 \leq \bar{x}_{ij} \odot \bar{x}_{ij} \text{ for } 1 \leq i \leq m, 1 \leq j \leq n.$$

- ▶ **Theorem 22.** *For any $n > m$, PHP_n^m has polynomial-size MP-NSR refutations.*

Proof. For the refutation of the propositional pigeonhole principle, we construct a treelike MP-PCR proof and then convert it into a static proof. We use the overall strategy for a treelike CP proof [22, Proposition 19.5]. The main step of this proof is the inductive

8:14 Tropical Proof Systems: Between R(CP) and Resolution

derivation of long inequalities $\sum_i x_{ij} \leq 1$ stating that every hole contains at most one pigeon, from short inequalities $x_{ij} + x_{i'j} \leq 1$. In CP this is done using the rounding rule, however, in MP-NSR we do not have it. Instead, we consider the cases for the newly added pigeon using tropical tools. More precisely, we will prove the following lemma

► **Lemma** (short to long inequalities). *Let $v_1 \dots v_k, \bar{v}_1, \dots, \bar{v}_k$ be variables. Given the Boolean axioms and the set of inequalities $v_i \odot v_{i'} \leq 1$ for $1 \leq i < i' \leq k$, one can construct a treelike MP-PCR derivation of $\bigodot_{i=1}^k v_i \leq 1$, which contains $O(k^4)$ terms, has tropical degree $O(k)$, and its coefficients are zeroes and ones.*

Proof of Lemma. Denote $V_j = \bigodot_{i=1}^j v_i$. We proceed by the induction on the number of variables constructing a treelike MP-PCR derivation of $V_j \leq 1$.

The base ($j = 2$) is trivial. The induction step comes in three stages.

Stage 1. Take the induction hypothesis for $j - 1$ and tropically multiply it by v_j getting

$$V_j \leq 1 \odot v_j. \quad (10)$$

Stage 2. For $i = 1, \dots, j - 1$, construct also the following derivation: tropically multiply the initial inequality $v_i \odot v_j \leq 1$ by \bar{v}_j and substitute its left-hand side by the axiom $1 \leq v_j \odot \bar{v}_j$ multiplied by v_i . Multiply the result by (-1) obtaining $v_i \leq \bar{v}_j$. Apply Lemma 18 to multiply these inequalities for $i = 1, \dots, j - 1$:

$$\bigodot_{i=1}^{j-1} v_i \leq \bar{v}_j^{\odot(j-1)}.$$

Further multiply it by v_j and substitute $v_j \odot \bar{v}_j \leq 1$ multiplied by $\bar{v}_j^{\odot(j-2)}$ into it obtaining

$$V_j \leq 1 \odot \bar{v}_j^{\odot(j-2)}. \quad (11)$$

Stage 3. Take the tropical sum of (10) and (11) and substitute its right-hand side with $v_j \oplus \bar{v}_j^{j-2} \leq 0$ (due to Lemma 19) multiplied by 1 eventually obtaining $V_j \leq 1$. ◀

Given the lemma, we apply it to x_{1j}, \dots, x_{mj} for each $j = 1, \dots, n$ separately. Multiply the results to get

$$\bigodot_{j=1}^n \bigodot_{i=1}^m x_{ij} \leq n.$$

On the other hand, by multiplying the initial inequalities, we get

$$m \leq \bigodot_{i=1}^m \bigodot_{j=1}^n x_{ij}$$

After substitution of the first equation into the right-hand side of the second equation, we arrive at the contradiction $m \leq n$. ◀

This shows that MP-NSR is strictly stronger than resolution; however, its relation to CP remains open. This makes MP-NSR a nice frontier proof system, for which (as a proof system for unsatisfiable formulas in CNF) we do not know any superpolynomial lower bounds.

6.2 Dynamic systems: Going up to Res(LP)

We do not know whether MP-PCR simulates Res(LP). The additional (\odot RES) rule strengthening MP-PCR is needed for that. We prove the following equivalence.

► **Theorem 23.** *MP-PCR+(\odot RES) with $\{0, 1\}$ encoding is polynomially equivalent to Res(LP).*

Proof Sketch. We simulate a Res(LP) proof step by step. A line of a Res(LP) refutation of the form $f_1 \geq 0 \vee f_2 \geq 0 \vee \dots \vee f_k \geq 0$ is translated into the min-plus inequality

$$[-f_1] \oplus [-f_2] \oplus \dots \oplus [-f_k] \leq 0,$$

where for a linear inequality $f \geq 0$, its translation $[-f]$ is given by the natural tropical term semantically equivalent to $-f$. To simulate the “main” rule (+RES) deriving a nonnegative linear combination, we split its coefficients into bits, simulate linear combinations for this easy case, and then sum everything together using (\odot RES).

In the other direction, a min-plus inequality $f_1 \oplus f_2 \oplus \dots \oplus f_t \leq g_1 \oplus g_2 \oplus \dots \oplus g_k$ is translated into the disjunctions of inequalities, one for each $1 \leq j \leq k$:

$$\bigvee_{i=1}^t \{f_i\} \leq \{g_j\},$$

where $\{\cdot\}$ again defines the natural semantically equivalent translation of tropical terms into linear inequalities. One can prove that a dynamic tropical proof can always be finished by a constant inequality $1 \leq 0$ (this is done in the same vein as simulating treelike proofs by static proofs, but now dynamically). Therefore, we do not need to deal with a complicated last line of the tropical proof in our simulation by Res(LP). ◀

Systems over $\{0, \infty\}$. One can consider systems that encode **false** and **true** by $\{0, \infty\}$ instead of $\{0, 1\}$. The axioms (01/ \odot) and (01/ \oplus) are then replaced by

$$\frac{\emptyset}{x \odot \bar{x} = \infty} \quad (0\infty/\odot), \quad \frac{\emptyset}{x \oplus \bar{x} = 0} \quad (0\infty/\oplus)$$

to ensure that x and \bar{x} are dual and in $\{0, \infty\}$. With this encoding, MP-PCR becomes equivalent to a different proof system (additional rules are not needed in this setting):

► **Theorem 24.** *MP-PCR that uses $\{0, \infty\}$ encoding with dual variables is polynomially equivalent to Res(∞), the unbounded version of the Res(k) proof system [24].*

The proof is similar to the $\{0, 1\}$ case; the main difference is that for $\{0, \infty\}$ tropical operations are essentially conjunction and disjunction, so it remains to process accurately statements like $f = c$ for a term f and a constant $c \in \mathbb{Q} \cup \{\infty\}$, and prove the corresponding translations of MP-PCR rules in Res(∞). The other direction goes almost literally by translating clauses composed of DNFs into the corresponding min-plus inequalities over $\{0, \infty\}$.

6.3 Lower bounds

We prove the lower bound k on the size of refutations of a much simplified tropical version $x^{\odot k} = c$ (where $c \in \mathbb{Q} \setminus \mathbb{N}_0$) of the (generalized) BINARY VALUE PRINCIPLE [4, 28] in MP-NSR.

► **Theorem 25.** *The size of MP-NSR refutations of a tropical binomial $x^{\odot k} = c$ for $c \in \mathbb{Q}$ is greater than k .*

In particular, for $x^{\odot 2^n} = -1$ this is an exponential lower bound when the coefficients and the degrees of tropical proofs are represented in binary.

The proof adheres to the following ideology. The MP-NS refutations are based on comparing coefficients in left- and right-hand sides in algebraic combinations. Having this in mind, when talking about size in MP-NSR, we construct a directed graph on monomials occurring in a tropical algebraic combination. We do it in a way such that some specific function on the vertices of the graph related to the coefficients is strictly monotone along any arrow. To establish the lower bound on the size of refutations of $x^{\odot k} = c$ in MP-NSR we prove that any cycle in this graph should contain at least k arrows.

Proof. Consider a refutation

$$(f, g) := \bigoplus_{i,j} x^{\odot i} \odot \bar{x}^{\odot j} \odot (a_{i,j} \odot (x \oplus \bar{x}, 0) \oplus b_{i,j} \odot (0, x \oplus \bar{x}) \\ \oplus d_{i,j} \odot (x \odot \bar{x}, 1) \oplus e_{i,j} \odot (1, x \odot \bar{x}) \oplus u_{i,j} \odot (x^{\odot k}, c) \oplus v_{i,j} \odot (c, x^{\odot k})),$$

where $a_{i,j}, b_{i,j}, d_{i,j}, e_{i,j}, u_{i,j}, v_{i,j} \in \mathbb{Q}_{\infty}$ and $f \succ g$.

We construct a directed graph H (See Fig. 2) whose vertices are tropical monomials m appearing in g . We distinguish 7 cases regarding from which summand in the right-hand side of the refutation a monomial m emerges in g :

$$m = x^{\odot i} \odot \bar{x}^{\odot j}, \quad c(m) = a_{i,j}; \quad (12)$$

$$m = x^{\odot(i+1)} \odot \bar{x}^{\odot j}, \quad c(m) = b_{i,j}; \quad (13)$$

$$m = x^{\odot i} \odot \bar{x}^{\odot(j+1)}, \quad c(m) = b_{i,j}; \quad (14)$$

$$m = x^{\odot i} \odot \bar{x}^{\odot j}, \quad c(m) = d_{i,j} + 1; \quad (15)$$

$$m = x^{\odot(i+1)} \odot \bar{x}^{\odot(j+1)}, \quad c(m) = e_{i,j}; \quad (16)$$

$$m = x^{\odot i} \odot \bar{x}^{\odot j}, \quad c(m) = u_{i,j} + c; \quad (17)$$

$$m = x^{\odot(i+k)} \odot \bar{x}^{\odot j}, \quad c(m) = v_{i,j} \quad (18)$$

for suitable i, j . If several cases apply to the same monomial, we choose one of them in an arbitrary way.

Draw an arrow in H from m to

$$x \odot m = x^{\odot(i+1)} \odot \bar{x}^{\odot j} \text{ in case (12);} \quad (19)$$

$$x^{\odot i} \odot \bar{x}^{\odot j} \text{ in case (13);} \quad (20)$$

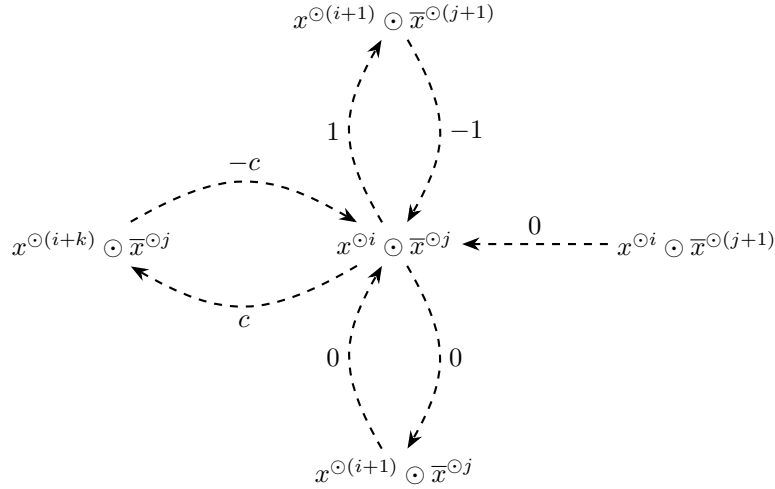
$$x^{\odot i} \odot \bar{x}^{\odot j} \text{ in case (14);} \quad (21)$$

$$x^{\odot(i+1)} \odot \bar{x}^{\odot(j+1)} \text{ in case (15);} \quad (22)$$

$$x^{\odot i} \odot \bar{x}^{\odot j} \text{ in case (16);} \quad (23)$$

$$x^{\odot(i+k)} \odot \bar{x}^{\odot j} \text{ in case (17);} \quad (24)$$

$$x^{\odot i} \odot \bar{x}^{\odot j} \text{ in case (18).} \quad (25)$$



■ **Figure 2** Possible arrows in a fragment of the graph H in Theorem 25 (not all arrows are present in a specific proof!). Coefficients decrease more than by the value shown on the arrows (according to (26)–(32)).

Since $f \succ g$, these arrows indeed correspond to strict inequalities on the coefficients:

$$c(x^{\odot(i+1)} \odot \bar{x}^{\odot j}) < c'(x^{\odot(i+1)} \odot \bar{x}^{\odot j}) \leq c(x^{\odot i} \odot \bar{x}^{\odot j}) \text{ in cases (12), (19);} \quad (26)$$

$$c(x^{\odot i} \odot \bar{x}^{\odot j}) < c'(x^{\odot i} \odot \bar{x}^{\odot j}) \leq c(x^{\odot(i+1)} \odot \bar{x}^{\odot j}) \text{ in cases (13), (20);} \quad (27)$$

$$c(x^{\odot i} \odot \bar{x}^{\odot j}) < c'(x^{\odot i} \odot \bar{x}^{\odot j}) \leq c(x^{\odot i} \odot \bar{x}^{\odot(j+1)}) \text{ in cases (14), (21);} \quad (28)$$

$$c(x^{\odot(i+1)} \odot \bar{x}^{\odot(j+1)}) < c'(x^{\odot(i+1)} \odot \bar{x}^{\odot(j+1)}) \leq c(x^{\odot i} \odot \bar{x}^{\odot j}) - 1 \quad (29)$$

in cases (15), (22);

$$c(x^{\odot i} \odot \bar{x}^{\odot j}) < c'(x^{\odot i} \odot \bar{x}^{\odot j}) \leq c(x^{\odot(i+1)} \odot \bar{x}^{\odot(j+1)}) + 1 \text{ in cases (16), (23);} \quad (30)$$

$$c(x^{\odot(i+k)} \odot \bar{x}^{\odot j}) < c'(x^{\odot(i+k)} \odot \bar{x}^{\odot j}) \leq c(x^{\odot i} \odot \bar{x}^{\odot j}) - c \text{ in cases (17), (24);} \quad (31)$$

$$c(x^{\odot i} \odot \bar{x}^{\odot j}) < c'(x^{\odot i} \odot \bar{x}^{\odot j}) \leq c(x^{\odot(i+k)} \odot \bar{x}^{\odot j}) + c \text{ in cases (18), (25).} \quad (32)$$

There exists a cycle Z in the graph H . To justify the required lower bound k when the numbers of arrows of types (24), (25) differ, note that the degree of x changes at most by one in all other types of arrows.

When they are equal, we observe that the number of arrows of type (23) in Z is greater than the number of arrows of type (22), because the coefficient at a tropical monomial increases (respectively, decreases) by 1 along an arrow of type (23) due to (16) (respectively, of type (22) due to (15)), while the coefficient does not change along arrows of types (19), (20), (21) due to (12), (13), (14), respectively. This leads to a contradiction since the tropical degree with respect to \bar{x} of a tropical monomial decreases (respectively, increases) by 1 along an arrow of type (23) (respectively, (22)), while this tropical degree does not increase along arrows of other types. ◀

6.4 Non-deducibility results

We also show two non-deducibility results. First we notice that one could encode Boolean logic without dual variables by replacing the axioms (01/·) and (01/⊕) by semantically equivalent

$$\frac{\emptyset}{x^{\odot 2} \oplus 1 = x} \quad (01/E)$$

We show that the inequality $0 \leq x$ is not derivable from (01/E) (thus showing the difference between two Boolean encodings). Formally, we prove the following theorem.

► **Theorem 26.** $\Gamma \oplus c \leq x \oplus \Delta$ is not derivable in MP-NS from (01/E) for any $c \in \mathbb{Q}_\infty$ and any $\Gamma \succ \Delta$.

After that we establish that the tropical resolution rule cannot be simulated directly in MP-PCR by providing an easy example of premises of these rules that cannot yield the conclusion through an algebraic combination with Boolean axioms (even with auxiliary polynomials remaining in the algebraic combination). More precisely, the next theorem states that there is no inference of the inequality $x^{\odot 2} \leq 0$ from the axioms in MP-NSR. Note that this demonstrates that one cannot infer the tropical resolution rule (\odot RES): namely, from $t \oplus f \leq 0$, $t' \oplus f \leq 0$ to infer $t \odot t' \oplus f \leq 0$, setting $t := t' := x$, $f := x^{\odot 2}$.

► **Theorem 27.** For any $\Gamma \succ \Delta$, there is no MP-NSR inference of $\Gamma \oplus x^{\odot 2} \leq 0 \oplus \Delta$ from the axioms of these systems.

This proof uses techniques similar to those of the size lower bounds. To prove non-derivability in MP-NSR, we again construct a directed graph on monomials occurring in a tropical algebraic combination, such that some specific function on the vertices of the graph related to the coefficients is strictly monotone along any arrow. Then we show the existence of a cycle in the graph, which leads to a contradiction.

7 Conclusion and Further Research

In this paper we introduced a new view of previously known proof systems by using tropical arithmetic. This view allowed us to isolate weaker fragments of Res(CP) (see Figure 1) so that we could hope for proving superpolynomial lower bounds on the proof size for them. The weakest of these fragments, static tropical proof systems, allow for different (and more elementary) methods of proving lower bounds. We provided several steps in this direction (though not for formulas in CNF). We view proving lower bounds for tropical proof systems as a promising direction.

The “knowledge border” for Boolean formulas in CNF lies between treelike Res(CP), where exponential lower bounds have been recently proved using quasipolynomial monotone interpolation [15], treelike Res(Lin) with semantic weakening, where exponential lower bounds are known for PHP [28], regular Res(\oplus), where exponential lower bounds for the binary pigeon-hole principle have been proved recently [13], on the one hand, and, on the other hand, Res(LP*) as well as Res(Lin) and Res(\oplus), where the question is so far open. In the non-CNF case, exponential lower bounds are known also for the Binary Value Principle in daglike Res(Lin) [28].

Tropical proof systems refine these borders. The static system MP-NSR lies between daglike Resolution and (through, for example, MP-PCR and Res(LP)) Res(CP). In the non-CNF case, we have shown an exponential lower bound on the refutations of a greatly simplified version of the Binary Value Principle in MP-NSR.

Several promising directions are (all questions concern $\{0, 1\}$ -variables encodings):

1. We were able to show the polynomial simulation of Res(LP) only after we added the rule (\odot RES) to MP-PCR.
 - a. It gives an additional hope to prove **lower bounds for MP-PCR** as it may be weaker than Res(LP).
 - b. Or maybe the two systems are polynomially equivalent even without this rule? (We only proved that it cannot be simulated directly in a rule-by-rule fashion.)

2. Fleming et al. [14] prove that CP quasipolynomially simulates `treelike Res(CP*)`. While `Res(CP*)` and `Res(LP*)` are polynomially equivalent, this is not necessarily true for their `treelike` versions. In fact, `treelike Res(LP)` has very limited ability to work with integer arithmetic at all, because it is unable to make rounding with big coefficients efficiently. Can we quasipolynomially simulate `treelike Res(LP)` in CP? Perhaps, we can quasipolynomially simulate MP-NSR in CP?
3. Relations between MP-NSR and CP are unclear, both for unary and binary coefficients, and even for `treelike` CP.
4. Relations between `treelike MP-PCR+(⊙RES)` and `treelike Res(LP)` are also unclear. Even polynomial simulation of MP-NSR in `treelike Res(LP)` does not seem to be trivial.

References

- 1 Marianne Akian, Antoine Béreau, and Stéphane Gaubert. The tropical Nullstellensatz and Positivstellensatz for sparse polynomial systems. In *Proc. ACM Intern. Symp. Symb. Algebr. Comput.*, pages 43–52, 2023. doi:10.1145/3597066.3597089.
- 2 Marianne Akian, Stéphane Gaubert, and Alexander Guterman. The correspondence between tropical convexity and mean payoff games. In *19 Intern. Symp. Math. Theory of Networks and Systems, Budapest*, pages 1295–1302, 2012.
- 3 Yaroslav Alekseev, Dima Grigoriev, and Edward A. Hirsch. Tropical proof systems. *Electron. Colloquium Comput. Complex.*, TR24-072, 2024. URL: <https://eccc.weizmann.ac.il/report/2024/072>.
- 4 Yaroslav Alekseev, Dima Grigoriev, Edward A. Hirsch, and Iddo Tzameret. Semialgebraic proofs, IPS lower bounds, and the τ -conjecture: Can a natural number be negative? *SIAM Journal on Computing*, 53(3):648–700, 2024. doi:10.1137/20M1374523.
- 5 Xavier Allamigeon, Uli Fahrenberg, Stéphane Gaubert, Ricardo D. Katz, and Axel Legay. Tropical Fourier-Motzkin elimination, with an application to real-time verification. *Int. J. Algebra Comput.*, 24(5):569–607, 2014. doi:10.1142/S0218196714500258.
- 6 Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing Planes. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2018.10.
- 7 Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert’s Nullstellensatz and propositional proofs. *Proc. London Math. Soc.*, 73(3):1–26, 1996. doi:10.1112/plms/s3-73.1.1.
- 8 Aaron Bertram and Robert Easton. The tropical Nullstellensatz for congruences. *Adv. Math.*, 308:36–82, 2017.
- 9 Peter Butkovic. *Max-linear systems: theory and algorithms*. Springer, 2010.
- 10 Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 174–183, New York, 1996. doi:10.1145/237814.237860.
- 11 Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979. doi:10.2307/2273702.
- 12 W. Cook, C.R. Coullard, and Gy. Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.
- 13 Klim Efremenko, Michal Garlik, and Dmitry Itsykson. Lower bounds for regular resolution over parities. ECCC TR23-187, 2023.
- 14 Noah Fleming, Mika Göös, Russell Impagliazzo, Toniann Pitassi, Robert Robere, Li-Yang Tan, and Avi Wigderson. On the power and limitations of branch and cut. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 6:1–6:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.6.

- 15 Max Gläser and Marc E. Pfetsch. Sub-exponential lower bounds for branch-and-bound with general disjunctions via interpolation. Technical Report 2308.04320, arXiv, 2023.
- 16 Dima Grigoriev. Complexity of solving tropical linear systems. *Comput. Complexity*, 22:71–88, 2013. doi:10.1007/S00037-012-0053-5.
- 17 Dima Grigoriev and Vladimir V. Podolskii. Tropical effective primary and dual Nullstellensätze. *Discrete and Computational Geometry*, 59(3):507–552, 2018. doi:10.1007/s00454-018-9966-3.
- 18 Armin Haken. The intractability of resolution. *Theoret. Comput. Sci.*, 39(2-3):297–308, 1985. doi:10.1016/0304-3975(85)90144-6.
- 19 Edward A. Hirsch and Arist Kojevnikov. Several notes on the power of Gomory-Chvátal cuts. *Ann. Pure Appl. Log.*, 141(3):429–436, 2006. doi:10.1016/j.apal.2005.12.006.
- 20 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Ann. Pure Appl. Log.*, 171(1), 2020. doi:10.1016/J.APAL.2019.102722.
- 21 Daniel Joo and Kalina Mincheva. Prime congruences of additively idempotent semirings and a Nullstellensatz for tropical polynomials. *Selecta Math.*, 24:2207–2233, 2018.
- 22 Stasys Jukna. *Boolean Function Complexity*. Springer, 2012.
- 23 Jan Krajíček. Discretely ordered modules as a first-order extension of the cutting planes proof system. *J. Symb. Log.*, 63(4):1582–1596, 1998. doi:10.2307/2586668.
- 24 Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1–3):123–140, 2001.
- 25 Jan Krajíček. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019.
- 26 Diane Maclagan and Felipe Rincon. Tropical ideals. *Compos. Math.*, 154:640–670, 2018.
- 27 Diane Maclagan and Bernd Sturmfels. *Introduction to Tropical Geometry*. Springer, 2015.
- 28 Fedor Part and Iddo Tzameret. Resolution with counting: Dag-like lower bounds and different moduli. *Comput. Complex.*, 30(1):2, 2021. doi:10.1007/S00037-020-00202-X.
- 29 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997. URL: <http://www.jstor.org/stable/2275583>, doi:10.2307/2275583.
- 30 Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. doi:10.1016/j.apal.2008.04.001.
- 31 Grigori Tseitin. On the complexity of derivations in propositional calculus. In *Studies in constructive mathematics and mathematical logic. Part II*, pages 115–125. Consultants Bureau, New-York-London, 1968.
- 32 Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

Improved Approximation Algorithms for (1,2)-TSP and Max-TSP Using Path Covers in the Semi-Streaming Model

Sharareh Alipour ✉ 

Department of Computer Science, Tehran Institute for Advanced Studies (TeIAS),
Khatam University, Tehran, Iran

Ermiya Farokhnejad ✉ 

Department of Computer Science, University of Warwick, Coventry, UK

Tobias Mömke ✉ 

Department of Computer Science, University of Augsburg, Germany

Abstract

We investigate semi-streaming algorithms for the Traveling Salesman Problem (TSP). Specifically, we focus on a variant known as the (1,2)-TSP, where the distances between any two vertices are either *one* or *two*. Our primary emphasis is on the closely related Maximum Path Cover Problem, which aims to find a collection of vertex-disjoint paths that covers the maximum number of edges in a graph. We propose an algorithm that, for any $\epsilon > 0$, achieves a $(\frac{2}{3} - \epsilon)$ -approximation of the maximum path cover size for an n -vertex graph, using $\text{poly}(\frac{1}{\epsilon})$ passes. This result improves upon the previous $\frac{1}{2}$ -approximation by Behnezhad et al. [3] in the semi-streaming model. Building on this result, we design a semi-streaming algorithm that constructs a tour for an instance of (1,2)-TSP with an approximation factor of $(\frac{4}{3} + \epsilon)$, improving upon the previous $\frac{3}{2}$ -approximation factor algorithm by Behnezhad et al. [3]¹.

Furthermore, we extend our approach to develop an approximation algorithm for the Maximum TSP (Max-TSP), where the goal is to find a Hamiltonian cycle with the maximum possible weight in a given weighted graph G . Our algorithm provides a $(\frac{7}{12} - \epsilon)$ -approximation for Max-TSP in $\text{poly}(\frac{1}{\epsilon})$ passes, improving on the previously known $(\frac{1}{2} - \epsilon)$ -approximation obtained via maximum weight matching in the semi-streaming model.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases (1,2)-TSP, Max-TSP, Maximum Path Cover, Semi-Streaming Algorithms, Approximation Algorithms, Graph Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.9

Related Version *Full Version:* <https://arxiv.org/abs/2501.04813> [2]

Funding *Tobias Mömke:* Partially supported by DFG Grant 439522729 (Heisenberg-Grant).

1 Introduction

The Traveling Salesman Problem (TSP) is a fundamental problem in combinatorial optimization. Given a graph $G = (V, E)$ with distances assigned to the edges, the objective is to find a Hamiltonian cycle with the lowest possible cost. The general form of TSP is known to be inapproximable unless $P = NP$ [22]. Consequently, research often focuses on specific types of

¹ Although Behnezhad et al. do not explicitly state that their algorithm works in the semi-streaming model, it is easy to verify.

distance functions, particularly the metric TSP, where distances satisfy the triangle inequality. Two notable metric versions of TSP are the graphic TSP, where distances correspond to the shortest path lengths in an unweighted graph, and (1,2)-TSP, a variant of TSP with distances restricted to either *one* or *two* [1, 4, 5, 6, 12, 14, 15, 17, 18, 19, 23, 24, 25].

Most research is conducted within the classic centralized model of computation. However, with the surge of large data sets in various real-world applications (as reviewed in [7]), there is a growing demand for algorithms capable of handling massive inputs. For very large graphs, classical algorithms are not only too slow but also suffer from excessive space complexity. When a graph's size exceeds the memory capacity of a single machine, algorithms that rely on random access to the data become impractical, necessitating alternative computational models. One such model that has gained significant attention recently is the graph stream model, introduced by Feigenbaum et al. [8, 9]. In this model, edges of the graph are not stored in memory but arrive sequentially in a stream, requiring processing in that order. The challenge is to design algorithms that use minimal space and ideally make only a small constant number of passes over the stream. A widely studied variant of this is the semi-streaming model. In the semi-streaming model, as outlined by Feigenbaum et al. [9], we consider a graph G with n vertices. The algorithm processes the graph's edges as they arrive in the stream and aims to compute results with minimal passes while using limited memory, constrained to $\tilde{O}(n) := O(n \cdot \text{polylog}(n))$.

It is straightforward to design a deterministic one-pass streaming algorithm to compute the cost of a Minimum Spanning Tree (MST) exactly, even in graph streams, which in turn immediately provides an $\tilde{O}(n)$ space algorithm to estimate TSP cost within a factor of 2. Thus, in the semi-streaming regime, the key challenge is to estimate TSP cost within a factor that is strictly better than 2. Recently, Chen, Khanna, and Tan [5] proposed a deterministic two-pass 1.96-approximation factor algorithm for metric TSP cost estimation in the semi-streaming model. For the case of (1,2)-TSP, using the approach of Behnezhad et al. [3], it is possible to provide a 1.5-approximation factor algorithm in the semi-streaming model. In [3], authors presented a sub-linear version of their algorithm; however, it is straightforward to implement their algorithm in the semi-streaming model.

In section 11 of [3], the authors showed a reduction from (1,2)-TSP to maximum matching and stated that achieving better approximation than 1.5 for (1,2)-TSP in the sub-linear model, solves an important open problem in sub-linear maximum matching. Considering the same reduction in semi-streaming model shows achieving non-trivial approximations for (1,2)-TSP in semi-streaming model is challenging. Since the maximum matching problem is studied in the semi-streaming model extensively, the following question naturally arises.

Question. What is the trade off between the approximation ratio and the number of passes for (1,2)-TSP in the semi-streaming model?

Maximum Path Cover. In an unweighted graph G , a subset of edges is called a *path cover* if it forms a union of vertex-disjoint paths. A maximum path cover (MPC²) in an unweighted graph is a path cover with the maximum number of edges (not paths) among all possible path covers in the graph. The problem of finding an MPC is known to be NP-complete. It is

² Throughout this paper we use this acronym for 'Maximum Path Cover'. Please note that we do **not** refer to the common abbreviation for 'Massively Parallel Computation'.

straightforward to see that a maximum matching provides a $1/2$ -approximation for MPC. Therefore, computing a maximal matching, which is a $1/2$ -approximation for maximum matching, yields a $1/4$ -approximate solution for MPC.

Behnezhad et al. [3] developed a $1/2$ -approximate MPC algorithm, which provides a 1.5 -approximate solution for $(1, 2)$ -TSP. Their algorithm can be implemented in one pass within the semi-streaming model using $\tilde{O}(n)$ space to return the cost, and in two passes if the approximate solution itself is required. Our primary contribution is an improvement in the approximation factor of their algorithm.

Result 1 (Formally as Theorem 8). For a given unweighted graph G , there is a semi-streaming algorithm that returns a $(\frac{2}{3} - \epsilon)$ -approximation of MPC in $\text{poly}(\frac{1}{\epsilon})$ passes.

(1, 2)-TSP. The classical problem $(1, 2)$ -TSP is well-studied and known to be NP-hard [15], and even APX-hard [20]. One can easily observe that in an instance of $(1, 2)$ -TSP, the optimal tour is almost the same as finding the MPC of the induced subgraph on edges with weight 1 and then joining their endpoints with edges with weight 2, except for a possible difference of 1 (in the case that there exists a Hamiltonian cycle all of whose edges have weight 1). A simple computation shows that if one can find a set of vertex-disjoint paths that is at least α times the optimal size ($\alpha \leq 1$), then one can also find a tour whose cost is no more than $(2 - \alpha)$ times the optimal cost for $(1, 2)$ -TSP. Thus, Result 1 implies the following result.

Result 2 (Formally as Theorem 10). For an instance of $(1, 2)$ -TSP, there is a semi-streaming algorithm that returns a $(\frac{4}{3} + \epsilon)$ -approximation of $(1, 2)$ -TSP in $\text{poly}(\frac{1}{\epsilon})$ passes.

In the second part of the paper, we examine Max-TSP in the semi-streaming model.

Max-TSP. For a given complete weighted graph G , the goal of Max-TSP is to find a Hamiltonian cycle such that the sum of the weights of the edges in this cycle is maximized.

It is evident that a maximum weighted matching provides a $\frac{1}{2}$ -approximation for the cost of Max-TSP. Consequently, the result of Huang and Saranurak [13], which computes a $(1 - \epsilon)$ -approximate maximum weight matching in the semi-streaming model, yields a $(\frac{1}{2} - \epsilon)$ -approximation for Max-TSP. In this paper, we improve this bound to $\frac{7}{12} - \epsilon$. Our result is as follows.

Result 3 (Formally as Theorem 16). For a given weighted graph G , there is a semi-streaming algorithm that returns a $(\frac{7}{12} - \epsilon)$ -approximation of Max-TSP in $\text{poly}(\frac{1}{\epsilon})$ passes.

To the best of our knowledge, this is the first non-trivial approximation algorithm for Max-TSP in the semi-streaming model.

Further related work

Our approach for computing MPC, $(1, 2)$ -TSP and Max-TSP mainly uses the subroutines for computing maximum matching in unweighted graphs and maximum weight matching in weighted graphs.

In the semi-streaming model, Fischer, Mitrovic and Uitto [10] gave a $(1 - \epsilon)$ -approximation for the maximum matching problem in $\text{poly}(1/\epsilon)$ passes. This result was an improvement over the $(1/\epsilon)^{O(1/\epsilon)}$ passes algorithm by McGregor [16].

For the maximum weight matching in the semi-streaming model, Paz and Schwartzman gave a simple deterministic single-pass $(1/2 - \epsilon)$ -approximation algorithm [21]. Gamlath, Kale, Mitrovic, and Svensson gave a $(1 - \epsilon)$ -approximation streaming algorithm that uses $O_\epsilon(1)$ passes and $O_\epsilon(n \cdot \text{poly}(\log n))$ memory. This was the first $(1 - \epsilon)$ -approximation streaming algorithm for weighted matching that uses a constant number of passes (only depending on ϵ) [11]. Also, Huang and Su in [13], gave a deterministic $(1 - \epsilon)$ -approximation for maximum weighted matching using $\text{poly}(1/\epsilon)$ passes in the semi-streaming model. When ϵ is smaller than a constant $O(1)$ but at least $1/\log^{o(1)} n$, their algorithm is more efficient than [11].

1.1 Notation

Let G be a simple graph. We denote the set of vertices and edges of G by $V(G)$ and $E(G)$ respectively. We also denote the maximum matching size in G by $\mu(G)$ and the size of the MPC in G by $\rho(G)$.

For a subset of edges $T \subseteq E(G)$, we denote G/T as the contraction of G on T , which is the graph derived by repeatedly removing edges of T (it is well-known that the order does not matter) from the graph and merging its endpoints to be a single node in the new graph. Note that after contraction the graph might have parallel edges, but this does not interfere with our algorithm. In the weighted case, if $w(e)$ is the weight of edge e , then we define $w(T)$ to be the sum of weights of the elements of T i.e. $\sum_{e \in T} w(e)$. Let $P = (u_1, u_2, \dots, u_k)$ be a path of length $k - 1$ ($u_i \in V$ for $1 \leq i \leq k$). We call u_1 and u_k *end points* of P and u_i for $2 \leq i \leq k - 1$ *middle points* of P .

2 Technical Overview and Our Contribution

We propose a simple algorithm that constructs a path cover with an approximation factor of almost $\frac{2}{3}$ for MPC. This new algorithm merely depends on basic operations and computing matching and approximate matching.

Our algorithm for MPC is as follows. Assume that $\text{MM}\epsilon$ is a $(1 - \epsilon)$ -approximation algorithm for computing maximum matching in an unweighted graph G . We use $\text{MM}\epsilon$ as a subroutine in our algorithm, which has two phases. In the first phase, we run $\text{MM}\epsilon$ to compute a $(1 - \epsilon)$ -approximate maximum matching, denoted by M_1 . In the next phase, we contract all edges in M_1 to obtain a new graph $G' = G/M_1$. Then, we compute a $(1 - \epsilon)$ -approximate maximum matching, denoted by M_2 , for G' using $\text{MM}\epsilon$. Finally, we return the edges of M_1 and M_2 as a path cover for G (see Algorithm 1).

We will show that the output of our algorithm is a collection of vertex-disjoint paths, i.e., a valid path cover (see Lemma 1).

The algorithm is simple, but proving that its approximation factor is $\frac{2}{3} - \epsilon$ is challenging. As a warm-up, it is straightforward to see that by computing a maximum matching in the first phase, we achieve a $\frac{1}{2}$ -approximate MPC. However, the challenge lies in the second phase, which helps to improve the approximation factor.

Now we explain the idea of our proof to find the approximation factor of Algorithm 1. For the matching M_1 in graph G , we provide a lower bound for $\mu(G/M_1)$. We show that if we consider a maximum path cover P^* and contract P^* on M_1 , the contracted graph becomes a particular graph in which we can find a lower bound on the size of its maximum

matching. Let M_2 be a maximum matching of G/M_1 , then this results in a lower bound for $|M_2|$. Finally, we exploit this lower bound for $|M_2|$ together with a lower bound for $|M_1|$, to come up with the approximation factor of Algorithm 1.

We explain how to implement this algorithm in the semi-streaming model, achieving an improved approximation factor for $(1, 2)$ -TSP within this model.

For the Max-TSP, we use a similar algorithm, except we compute maximum weight matching instead of maximum matching (see Algorithm 3). By computing the approximation factor of this algorithm, we provide a non-trivial approximation algorithm for Max-TSP in the semi-streaming model. Despite the extensive study of the weighted version of the maximum matching problem, Max-TSP has not been studied extensively in the literature within the semi-streaming model. One reason could be that it is not possible to extend the approaches for the unweighted version to the weighted version. Fortunately, we can extend our algorithm to the weighted version and improve the approximation factor of Max-TSP in the semi-streaming model. However, our method for analyzing the approximation factor of Algorithm 1 does not apply to the weighted version, so we present a different proof approach for computing the approximation factor of Algorithm 3.

3 Improved Approximation Factor Semi-Streaming Algorithm for MPC

In Algorithm 1, we presented our novel algorithm for MPC. This section provides an analysis of its approximation factor, followed by a detailed explanation of its streaming implementation.

■ **Algorithm 1** Approximating maximum path cover on a graph G .

-
- 1: Run MM_ϵ on G to find a matching M_1 .
 - 2: Contract G on M_1 to get a new graph $G' = G/M_1$.
 - 3: Run MM_ϵ on G' to find another matching M_2 .
 - 4: **return** $M_1 \cup M_2$.
-

We start by proving the correctness of this algorithm.

► **Lemma 1.** *If M_1 and M_2 are the matchings obtained in Algorithm 1, then $M_1 \cup M_2$ forms a path cover for G .*

Proof. We claim that $M_1 \cup M_2$ is a vertex-disjoint union of paths of length 1, 2 or 3. As a result, it is a path cover. Suppose $M_1 = \{u_1v_1, u_2v_2, \dots, u_kv_k\}$. Let us denote the vertices of G/M_1 by $\{(uv)_1, (uv)_2, \dots, (uv)_k, w_1, w_2, \dots, w_l\}$ where $(uv)_i$ represents the vertices u_i and v_i , merged in the contracted graph G/M_1 and w_j 's for $1 \leq j \leq l$ are the rest of the vertices. Let $xy \in M_2$ be an arbitrary edge. By symmetry between x and y , there are three cases as follows:

1. $x, y \in \{w_1, w_2, \dots, w_l\}$.
In this case, x and y are intact vertices after contraction, which means there are no edges in M_1 adjacent to x and y . Since $xy \in M_2$ and M_2 is a matching, there are no other edges in $M_1 \cup M_2$ adjacent to x and y in G . As a result, xy would be a path of length 1 in $M_1 \cup M_2$.
2. $x \in \{(uv)_1, (uv)_2, \dots, (uv)_k\}$ and $y \in \{w_1, w_2, \dots, w_l\}$.
In this case, $x = (uv)_i$ for some $1 \leq i \leq k$. As a result, xy would be u_iy or v_iy in G . By symmetry, assume that $xy = u_iy$ in G . Since M_1 is a matching, no other edges in M_1 are adjacent to u_i and v_i . No edge in M_1 is adjacent to y . Since M_2 is a matching, the only edge in M_2 adjacent to at least one of u_i, v_i and y in G is $xy = u_iy$. Finally, we can see that (v_i, u_i, y) is a path of length 2 in $M_1 \cup M_2$.

3. $x, y \in \{(uv)_1, (uv)_2, \dots, (uv)_k\}$.

In this case, let $x = (uv)_i$ and $y = (uv)_j$ and by symmetry, assume that xy is the edge connecting u_i to u_j in G . Since M_1 is a matching, the only edges in M_1 adjacent to at least one of u_i, u_j, v_i, v_j are $u_i v_i$ and $u_j v_j$. Since M_2 is a matching, the only edge in M_2 adjacent to at least one of u_i, u_j, v_i, v_j is $(uv)_i(uv)_j$. As a result, (v_i, u_i, u_j, v_j) would be a path of length 3 in $M_1 \cup M_2$.

So, $M_1 \cup M_2$ is a union of vertex-disjoint paths of length 1, 2 or 3. ◀

3.1 Analysis of the Approximation Factor of Algorithm 1

We start with a simple and basic lemmas that is crucial in our proof.

► **Lemma 2.** *Let G be an arbitrary graph. We have:*

$$\rho(G) \geq \mu(G) \geq \frac{1}{2}\rho(G).$$

Proof. Since every matching is a path cover, we have $\rho(G) \geq \mu(G)$. Also, given an MPC, we can select every other edge in this MPC to obtain a matching that contains at least half of its edges, which implies $\mu(G) \geq \frac{1}{2}\rho(G)$. ◀

► **Corollary 3.** *If M is a $(1 - \epsilon)$ -approximation of maximum matching in graph G , then*

$$|M| \geq \frac{1}{2}(1 - \epsilon)\rho(G).$$

We now present a lemma regarding the size of the maximum matching in a specific type of graph. This lemma may be of independent interest. In this paper we utilize this lemma on G/M_1 to derive a lower bound for $|M_2|$.

► **Lemma 4.** *Assume G is a graph without loops such that each vertex v of G has degree 1, 2, or 4. If $V_4(G)$ denotes the set of vertices of degree 4 in G , then we have*

$$\mu(G) \geq \frac{|E(G)| - |V_4(G)|}{3}.$$

Proof. The proof of this lemma is a bit long and technical. Here, we provide the proof sketch of the lemma due to page limit constraints and we refer the reader to the full version of the paper to see the complete proof.³ The main tools for the proof are induction and a charging scheme. The induction is on the number of edges between nodes of degree 2 or 4 in graph G denoted by $|E_{2,4}(G)|$.

In a high level, we provide some special local topological properties of the graph. If G does not satisfy at least one of the properties, then we do some delicate local changes on G to get G' such that $|E_{2,4}(G')| \leq |E_{2,4}(G)|$. Next, we improve a matching in G' to a matching for G with the desired size.

In the case that G satisfies all of the properties, we provide a charging scheme that takes some tokens for each edge in an arbitrary maximum matching and spread these tokens between incident edges, which proves the desired inequality. ◀

Using above lemma, we have the main lemma of this section as follows.

³ A full version of this extended abstract can be found at [2].

► **Lemma 5.** *If M is an arbitrary matching in a graph G , then*

$$\mu(G/M) \geq \frac{\rho(G) - |M|}{3}.$$

Proof. Assume P^* is a maximum path cover in G such that $P^* \cap M$ is maximal. We claim that every $e \in M \setminus P^*$ connects two middle points of P^* . The proof of this claim follows from a case by case argument. For the sake of contradiction, assume $e = uv \in M \setminus P^*$ does not connect two middle points of P^* . We have three cases for u and v as follows.

- None of u and v belong to P^* (case 1).
- Exactly one of them (say u) belongs to P^* . Then, u is an end point (case 2), or u is a middle point (case 3).
- Both u and v belong to P^* . Then, we have two sub cases.
 - u and v are on different paths. Then either they are both end points (case 4), or one is a middle point (say u) and the other one is an end point (case 5). Note that we have considered that both of u and v are not middle points at the same time.
 - u and v belong to the same path. Then, either they are both end points (case 6), or one is a middle point (say u) and the other one is an end point (case 7). Note that we have assumed both of them are not middle points at the same time.

Now, we explain each case in detail.

1. Neither u nor v belongs to \tilde{P} .
This case is impossible because $P^* + e$ is a path cover with a size larger than $|P^*|$, which is in contradiction with P^* being MPC (see Figure 1a).
2. u is an end point of a path in P^* and v is not contained in P^* .
Again, this case is impossible since $P^* + e$ is a path cover with a size larger than $|P^*|$, which is in contradiction with P^* being MPC (see Figure 1b).
3. u is a middle point of a path in P^* and v is not contained in P^* .
Let (p_1, p_2, \dots, p_k) be the path in P^* containing $u = p_i$. Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1c). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|\tilde{P} \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
4. u and v are end points of different paths in P^* .
In this case, let (p_1, p_2, \dots, p_k) and (q_1, q_2, \dots, q_l) be the paths in P^* containing $u = p_1$ and $v = q_1$, respectively. $P^* + e$ would be a path cover of size greater than $|P^*|$ which is in contradiction with P^* being MPC (see Figure 1d).
5. u and v are the middle and end points of different paths in P^* , respectively.
In this case, let (p_1, p_2, \dots, p_k) and (q_1, q_2, \dots, q_l) be the paths in P^* containing $u = p_i$ and $v = q_1$ respectively. Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1e). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
6. u and v are end points of the same path in P^* .
In this case, let (p_1, p_2, \dots, p_k) be the path in P^* containing $u = p_1$ and $v = p_k$. Replace P^* by $P^* - p_1p_2 + e$ which is an MPC of G (see Figure 1f). Since $e \in M$ we have $p_1p_2 \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
7. u and v are the middle and end points of the same path in P^* , respectively.
In this case, let (p_1, p_2, \dots, p_k) be the path in P^* containing $u = p_i$ and $v = p_1$ (since $e \notin P^*$, we have $2 < i$). Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1g). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.

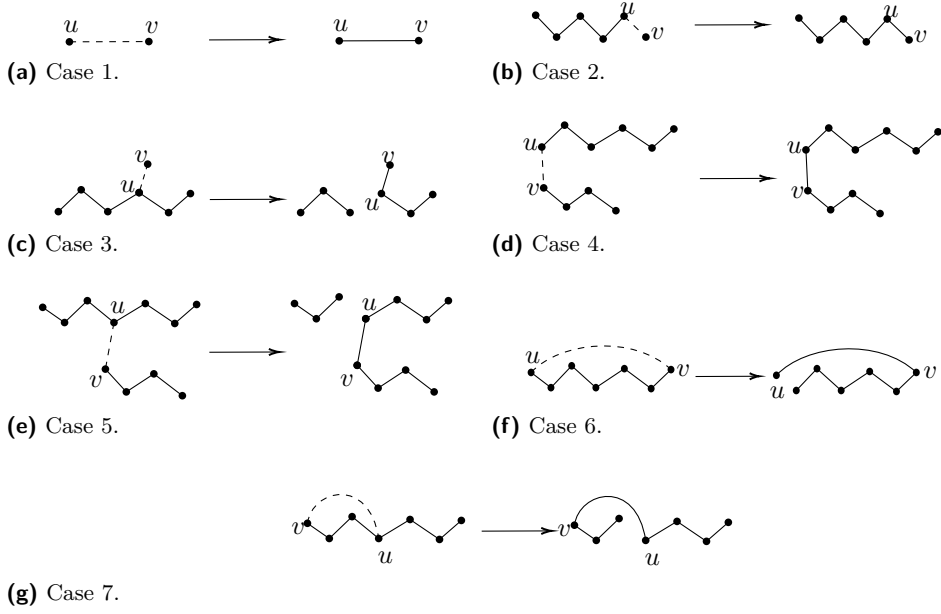


Figure 1 Different possible cases for u and v .

Each case leads to a contradiction, implying that every $e \in M \setminus P^*$ connects two middle points of P^* .

Now, contraction of each $e \in M \setminus P^*$ makes a vertex of degree 4 in P^*/M . Contraction of each $e \in M \cap P^*$ makes a vertex of degree 2 and decrements the number of edges in P^* . As a result, P^*/M is a graph whose vertices' degrees are 1,2 or 4, $|E(P^*/M)| = |P^*| - |P^* \cap M|$ and $|V_4(P^*/M)| = |M \setminus P^*|$. Finally, using Lemma 4 for P^*/M we have

$$\mu(P^*/M) \geq \frac{|E(P^*/M)| - |V_4(P^*/M)|}{3} = \frac{|P^*| - |P^* \cap M| - |M \setminus P^*|}{3} = \frac{|P^*| - |M|}{3}.$$

Since P^*/M is a subgraph of G/M we have

$$\mu(G/M) \geq \mu(P^*/M) \geq \frac{|P^*| - |M|}{3} = \frac{\rho(G) - |M|}{3}. \quad \blacktriangleleft$$

Using the above results, we compute the approximation factor of Algorithm 1.

► **Theorem 6.** *The approximation factor of Algorithm 1 is $\frac{2}{3}(1 - \epsilon)$, i.e.,*

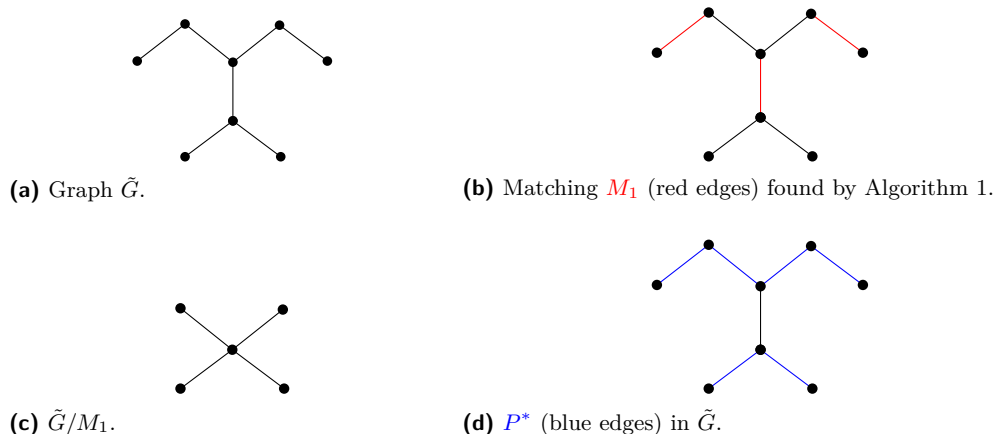
$$\rho(G) \geq |M_1 \cup M_2| \geq \frac{2}{3}(1 - \epsilon)\rho(G). \quad (1)$$

Proof. By Corollary 3 and, Lemma 5, we have

$$\begin{aligned} |M_1 \cup M_2| &= |M_1| + |M_2| \\ &\geq |M_1| + (1 - \epsilon)\mu(G/M_1) \\ &\geq |M_1| + \frac{1 - \epsilon}{3}(\rho(G) - |M_1|) \\ &\geq \frac{1 - \epsilon}{3}\rho(G) + \frac{2}{3}|M_1| \\ &\geq \frac{1 - \epsilon}{3}\rho(G) + \frac{1 - \epsilon}{3}\rho(G) = \frac{2}{3}(1 - \epsilon)\rho(G). \end{aligned}$$

Since $M_1 \cup M_2$ is a path cover, we have $\rho(G) \geq |M_1 \cup M_2|$. Hence, the approximation factor of Algorithm 1 is at least $\frac{2}{3}(1 - \epsilon)$. ◀

Now, we show that our analysis of the approximation factor of Algorithm 1 is tight. Consider the graph in Figure 2a and denote it by \tilde{G} . If we run Algorithm 1 on \tilde{G} , then the edges of M_1 could be the red edges shown in Figure 2b. After contracting \tilde{G} on M_1 , we have \tilde{G}/M_1 shown in Figure 2c. Finally, the second matching M_2 found by Algorithm 1 in \tilde{G}/M_1 contains at most one edge which implies $|M_1 \cup M_2| \leq 4$. On the other hand, maximum path cover P^* in \tilde{G} contains 6 edges shown in Figure 2d.



■ **Figure 2** An example of a graph \tilde{G} for which Algorithm 1 produces a path cover whose size is $\frac{2}{3}$ times the size of the MPC.

As a result, $\frac{|M_1 \cup M_2|}{|P^*|} \leq \frac{2}{3}$, so this example and Theorem 6 imply that the approximation factor of Algorithm 1 is $\frac{2}{3} - \epsilon$.

3.2 Implementation of Algorithm 1 in the Semi-Streaming Model

Now we explain how to implement Algorithm 1 in the semi-streaming model. We start with the following theorem by Fischer, Mitrovic and Uitto [10].

► **Theorem 7** (Theorem 1.1 in [10]). *Given a graph on n vertices, there is a deterministic $(1 - \epsilon)$ -approximation algorithm for maximum matching that runs in $\text{poly}(\frac{1}{\epsilon})$ passes in the semi-streaming model. Furthermore, the algorithm requires $n \cdot \text{poly}(\frac{1}{\epsilon})$ words of memory.*

To implement Algorithm 1 in the semi-streaming model, we proceed as follows: In the first phase, by applying Theorem 7, we compute a $(1 - \epsilon)$ -approximate matching for the graph G , denoted as M_1 . At the end of this phase, we have the edges of this matching. In the second phase, we again apply Theorem 7 to compute a matching for G/M_1 in the streaming model.

During the second phase, when we apply the algorithm of Theorem 7, while processing each edge (v_i, v_j) in the stream, we follow these rules: If $(v_i, v_j) \in M_1$, we ignore this edge. If $(v_i, v_j) \notin M_1$, but one of v_i or v_j is an endpoint of an edge in M_1 (e.g., $v_i, v_k \in M_1$), then since (v_i, v_j) is contracted, we consider v_i and v_j as a single vertex, v_{ij} . In this case, v_k is considered adjacent to the new vertex v_{ij} . Consequently, we can compute a $(1 - \epsilon)$ -approximation matching for G/M_1 in the next $\text{poly}(1/\epsilon)$ passes.

Thus, combining the results of Algorithm 1, Theorem 6, and Theorem 7, we have the main result of this section:

► **Theorem 8.** *Given an unweighted graph G on n vertices, there is a deterministic algorithm that returns a $(\frac{2}{3} - \epsilon)$ -approximate MPC in the semi-streaming model in $\text{poly}(\frac{1}{\epsilon})$ passes.*

4 (1, 2)-TSP

In this section, we present our algorithm for the (1, 2)-TSP, detailed in Algorithm 2, and analyze its approximation factor. We also provide an explanation of how to implement this algorithm in the semi-streaming model.

■ **Algorithm 2** Our algorithm for (1, 2)-TSP.

-
- 1: Let G_1 be the subgraph of G consisting of edges with weight 1.
 - 2: Run Algorithm 1 on G_1 to get a path cover \tilde{P} .
 - 3: Arbitrarily extend \tilde{P} to a Hamiltonian cycle \tilde{C} by adding edges between end points of \tilde{P} or/and existing vertices not in \tilde{P} .
 - 4: **return** \tilde{C} .
-

► **Theorem 9.** *The approximation factor of Algorithm 2 for (1, 2)-TSP is $\frac{4}{3} + \epsilon + \frac{1}{n}$.*

Proof. Let T^* be the optimal solution of (1,2)-TSP, ρ^* be the size of an MPC in G_1 , and n be the number of vertices of G . Since every Hamiltonian cycle contains n edges with weights 1 or 2, we have $n \leq T^* \leq 2n$. We also have $T^* = 2n - \rho^* - 1$ or $T^* = 2n - \rho^*$, where $T^* = 2n - \rho^* - 1$ occurs only when G contains a Hamiltonian cycle consisting solely of edges with weight 1.

Let $\tilde{\rho}$ be the size of the path cover obtained by Algorithm 1 in G_1 . The Hamiltonian cycle obtained by Algorithm 2 has a cost of at most $2(n - \tilde{\rho}) + \tilde{\rho} = 2n - \tilde{\rho}$. Let $\alpha \leq 1$ be the approximation factor of Algorithm 1, then we have $\alpha\rho^* \leq \tilde{\rho} \leq \rho^*$. As a result, $T^* \leq 2n - \rho^* \leq 2n - \tilde{\rho}$. We also have:

$$\begin{aligned}
 2n - \tilde{\rho} &\leq 2n - \alpha\rho^* = (2 - 2\alpha)n + \alpha(2n - \rho^* - 1) + \alpha \\
 &\leq (2 - 2\alpha)T^* + \alpha T^* + \alpha = (2 - \alpha)T^* + \alpha \\
 &\leq (2 - \alpha)T^* + 1 \\
 &\leq (2 - \alpha)T^* + \frac{T^*}{n} = \left(2 - \alpha + \frac{1}{n}\right)T^*. \tag{2}
 \end{aligned}$$

By Theorem 6, we have $\alpha \geq \frac{2}{3} - \epsilon$. Using Equation (2), we conclude that:

$$\begin{aligned}
 2n - \tilde{\rho} &\leq \left(2 - \alpha + \frac{1}{n}\right)T^* \\
 &\leq \left(2 - \left(\frac{2}{3} - \epsilon\right) + \frac{1}{n}\right)T^* = \left(\frac{4}{3} + \epsilon + \frac{1}{n}\right)T^*.
 \end{aligned}$$

Hence, $T^* \leq 2n - \tilde{\rho} \leq \left(\frac{4}{3} + \epsilon + \frac{1}{n}\right)T^*$. So, the approximation factor of our algorithm for (1,2)-TSP is $\frac{4}{3} + \epsilon + \frac{1}{n}$. ◀

4.1 Implementation of Algorithm 2 in the Semi-Streaming Model

For a given instance of (1, 2)-TSP in the streaming model, we compute an approximate MPC for the induced subgraph on the edges of weight 1 as explained in Theorem 8, then we add extra edges to connect these paths and vertices not in these paths arbitrarily to construct a Hamiltonian cycle, which gives us a $(\frac{4}{3} + \epsilon + \frac{1}{n})$ -approximate tour for (1, 2)-TSP. So, we have the main result of this section as follows.

► **Theorem 10.** *Given an instance of (1, 2)-TSP on n vertices, there is a deterministic algorithm that returns a $(\frac{4}{3} + \epsilon + \frac{1}{n})$ -approximate (1, 2)-TSP in the semi-streaming model in $O(\text{poly}(\frac{1}{\epsilon}))$ passes.*

5 Max-TSP

In this section, we introduce our algorithm for Max-TSP, which closely resembles our approach for MPC. The key difference is that, instead of using MM_ϵ , we employ a subroutine to compute an approximate maximum weight matching in a weighted graph.

Let MWM_ϵ be a subroutine for computing a $(1 - \epsilon)$ -approximate maximum weighted matching in a weighted graph G . First, we compute a matching M_1 for G using MWM_ϵ . Then, we contract the edges of M_1 to obtain another graph $G' = G/M_1$ and compute another matching, M_2 , for G' using MWM_ϵ again. We derive the union of the two weighted matchings, $M_1 \cup M_2$. Similar to Lemma 1, it is evident that $M_1 \cup M_2$ forms a union of vertex-disjoint paths in G . Finally, since the graph is complete, there can be only one vertex that is not in $M_1 \cup M_2$. In this case we connect this vertex to one of the paths in $M_1 \cup M_2$. Now, we add edges arbitrarily between the endpoints of the paths in $M_1 \cup M_2$ to obtain a Hamiltonian cycle C for G .

■ **Algorithm 3** Our algorithm for Max-TSP on a complete weighted graph G .

-
- 1: Run MWM_ϵ on G to find a matching M_1 .
 - 2: Contract G on M_1 to get a new graph $G' = G/M_1$.
 - 3: Run MWM_ϵ on G' to find another matching M_2 .
 - 4: Arbitrarily extend $M_1 \cup M_2$ to a Hamiltonian cycle C by adding edges between endpoints of $M_1 \cup M_2$ or/and existing vertices not in $M_1 \cup M_2$.
 - 5: **return** C .
-

Note that after contracting G on M_1 to obtain $G' = G/M_1$, this new graph might have parallel edges between vertices. Since we aim to find a maximum matching in G' , we can simply consider the edge with the largest weight for parallel edges and ignore the rest.

5.1 Analysis of the Approximation Factor of Algorithm 3

To analyze the approximation factor of Algorithm 3, we begin with a series of lemmas.

► **Lemma 11.** *Suppose C is a cycle of length k in a weighted graph G . Then, there exists a matching $M \subseteq C$ such that $w(M) \geq \frac{k-1}{2k}w(C)$.*

Proof. Assume that $e \in C$ is the edge with the minimum weight. Hence, $w(e) \leq w(C)/k$. Since $C - e$ is a path, there is a matching $M \subseteq C - e$ (which is also a subset of C) whose weight is at least $w(C - e)/2$. Finally,

$$w(M) \geq \frac{1}{2}w(C - e) = \frac{1}{2}(w(C) - w(e)) \geq \frac{1}{2} \left(w(C) - \frac{w(C)}{k} \right) = \frac{k-1}{2k}w(C). \quad \blacktriangleleft$$

► **Lemma 12.** *Suppose T is a path or a cycle in a weighted graph G . Then there exists a matching $M \subseteq T$ such that $w(M) \geq \frac{1}{3}w(T)$.*

Proof. We have two cases

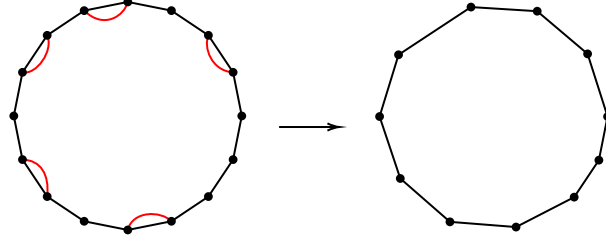
- T is a path. Enumerate the edges of T from one end point to the other. The odd numbered edges form a matching called M_{odd} . The same applies for even numbered edges, which form a matching called M_{even} . Since $T = M_{\text{odd}} \cup M_{\text{even}}$, at least one of these two matchings has weight no less than $w(T)/2$.
- T is a cycle. If it is a cycle of length 2 (i.e. T consists of two parallel edges), then obviously we can pick the edge e with bigger weight that satisfies $w(e) \geq \frac{1}{2}w(T) \geq \frac{1}{3}w(T)$. If the length of T is at least 3, then using Lemma 11, we conclude that there is a matching $M \subseteq T$ such that $w(M) \geq \frac{k-1}{2k}w(T) \geq \frac{1}{3}w(T)$. ◀

Now, we provide a lemma similar to Lemma 5 which works for the weighted version.

► **Lemma 13.** *Suppose M is a matching in a weighted graph G and C^* is a maximum weight Hamiltonian cycle of G . Then,*

$$\mu(G/M) \geq \frac{w(C^*) - w(M)}{6} - \frac{1}{3n}w(C^*).$$

Proof. We contract C^* on M in two steps. First, we contract C^* on the edges in $C^* \cap M$. Next, we contract the resulting graph on $M' = M \setminus C^*$. After the first step, $C' = C^*/(C^* \cap M)$ is a cycle with weight $w(C^*) - w(C^* \cap M)$ (see Figure 3).



■ **Figure 3** C^* remains a path cover after contraction on $C^* \cap M$ (red edges).

Here M' is a matching that connects some vertices of C' together (see Figure 4a, Figure 4b and Figure 4c).

Assume that the length of C' is k . Using Lemma 11, there is a matching $M^* \subseteq C'$ whose weight is at least $\frac{k-1}{2k}w(C')$ (see Figure 4d). Since the matching M_1 contains at most half of the edges of C^* , we conclude that $k \geq n/2$. As a result,

$$\begin{aligned} w(M^*) &\geq \frac{k-1}{2k}w(C') \geq \frac{n-2}{2n}(w(C^*) - w(C^* \cap M)) \\ &\geq \frac{w(C^*) - w(C^* \cap M)}{2} - \frac{1}{n}w(C^*). \end{aligned} \tag{3}$$

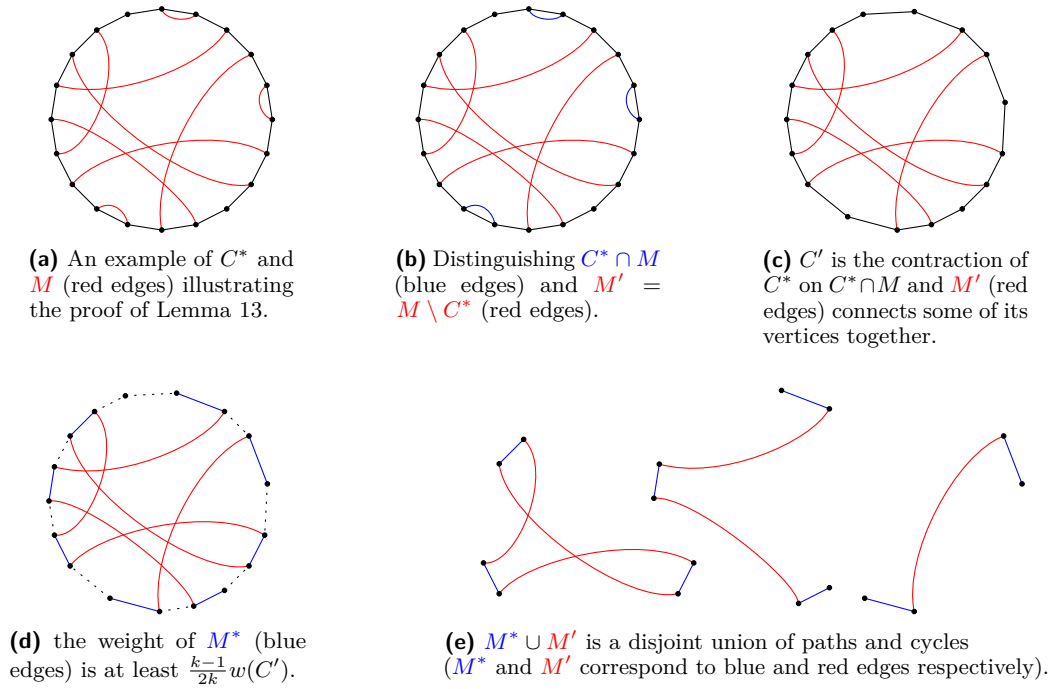
Since $M' \cap C' = \emptyset$, we conclude that $M' \cap M^* = \emptyset$. Because M^* and M' are matchings, it follows that $M^* \cup M'$ is a union of disjoint paths and cycles (see Figure 4e). As a result, after doing the second step of contraction, M^*/M' would also be a disjoint union of paths and cycles (whose number of edges is equal to $|M^*|$ since $M^* \cap M' = \emptyset$) in C'/M' . For instance, if $M^* \cup M'$ contains a cycle of length 4, then M^*/M' would contain a cycle of length 2 which contains parallel edges.

Now, consider each connected component of M^*/M' . This component is either a path or a cycle. Hence, by Lemma 12, We obtain a matching with a weight of at least one-third of the weight of the component.

Finally, since these components are vertex-disjoint, the union of obtained matching would be a matching whose weight is at least $w(M^*)/3$. Note that this matching is also a matching in G/M . Hence, using Equation (3), we have

$$\begin{aligned} \mu(G/M) &\geq \frac{w(M^*)}{3} \geq \frac{w(C^*) - w(C^* \cap M)}{6} - \frac{1}{3n}w(C^*) \\ &\geq \frac{w(C^*) - w(M)}{6} - \frac{1}{3n}w(C^*). \end{aligned} \quad \blacktriangleleft$$

So, we have the following theorem which is a lower bound for the approximation factor of Algorithm 3.



■ **Figure 4** An example of C^* and M illustrating the steps in the proof of Lemma 13.

► **Theorem 14.** *The approximation factor of Algorithm 3 is at least $(\frac{7}{12} - \frac{3}{4n})(1 - \epsilon)$.*

Proof. Let C^* be a maximum weight Hamiltonian cycle in G . By Lemma 11, there exists at least one matching $M \subseteq C^*$ whose weight is at least

$$\frac{n-1}{2n}w(C^*).$$

Since M_1 is a $(1 - \epsilon)$ -approximation of the maximum weighted matching in G we have

$$w(M_1) \geq \frac{(1 - \epsilon)(n-1)}{2n}w(C^*).$$

By using Lemma 13 for M_2 on G/M_1 , we have

$$\begin{aligned} w(M_1 \cup M_2) &= w(M_1) + w(M_2) \\ &\geq w(M_1) + (1 - \epsilon)\mu(G/M_1) \\ &\geq w(M_1) + \frac{1 - \epsilon}{6}(w(C^*) - w(M_1)) - \frac{1 - \epsilon}{3n}w(C^*) \\ &\geq (1 - \epsilon) \left(\frac{1}{6} - \frac{1}{3n} \right) w(C^*) + \frac{5}{6}w(M_1) \\ &\geq (1 - \epsilon) \left(\frac{1}{6} - \frac{1}{3n} \right) w(C^*) + \frac{5(1 - \epsilon)(n-1)}{12n}w(C^*) \\ &= \left(\frac{7}{12} - \frac{3}{4n} \right) (1 - \epsilon)w(C^*). \end{aligned}$$

Since the weight of the edges of G are nonnegative, we have

$$w(C) \geq w(M_1 \cup M_2) \geq \left(\frac{7}{12} - \frac{3}{4n} \right) (1 - \epsilon)w(C^*).$$

Finally, C is a Hamiltonian cycle which means $w(C^*) \geq w(C)$. Hence, the approximation factor of Algorithm 1 is at least $(\frac{7}{12} - \frac{3}{4n})(1 - \epsilon)$. ◀

5.2 Implementation of Algorithm 3 in the semi-streaming model

The implementation of Algorithm 3 in the semi-streaming model follows a similar approach as described in the previous section. Therefore, we omit a detailed explanation here. However, note that in this part, we should use a subroutine for computing a $(1 - \epsilon)$ -approximate maximum weight matching in the semi-streaming model. First, we recall the following theorem from [13]. We use the algorithm of this theorem as MWM_ϵ in our semi-streaming implementation of Algorithm 3.

► **Theorem 15** (Theorem 1.3 in [13]). *There exists a deterministic algorithm that returns a $(1 - \epsilon)$ -approximate maximum weight matching using $\text{poly}(\frac{1}{\epsilon})$ passes in the semi-streaming model. The algorithm requires $O(n \cdot \log W \cdot \text{poly}(\frac{1}{\epsilon}))$ words of memory where W is the maximum edge weight in the graph.*

Thus, Algorithm 3, Theorem 14, and Theorem 15 present the following theorem for Max-TSP in the semi-streaming model.

► **Theorem 16.** *Given an instance of Max-TSP on n vertices, there is an algorithm that returns a $(\frac{7}{12} - \frac{3}{4n})(1 - \epsilon)$ -approximate Max-TSP the semi-streaming model in $O(\text{poly}(\frac{1}{\epsilon}))$ passes. The algorithm requires $O(n \cdot \log W \cdot \text{poly}(\frac{1}{\epsilon}))$ words of memory where W is the maximum edge weight in the graph.*

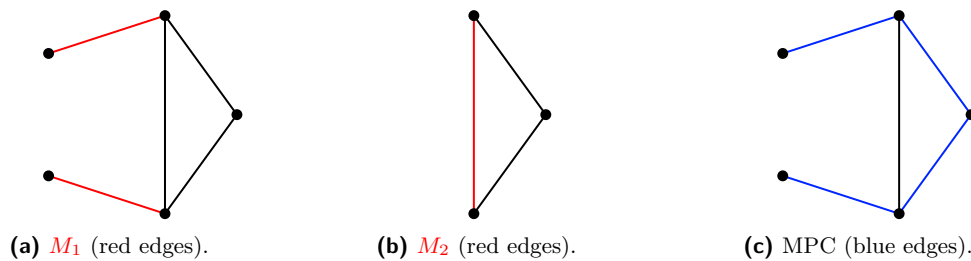
6 Future Work

As a future work we propose the following algorithm that can help to improve the approximation factor for MPC in the semi-streaming model. The algorithm improves Algorithm 1 by iteratively finding new matchings and contracting the graph over these matchings. It is crucial to ensure that this process preserves the path cover property. Hence, during the k th iteration of our loop, we must remove all the edges in G that are incident to a middle point of any path (connected component) within $\cup_{i=1}^k M_i$. This is because $(\cup_{i=1}^k M_i) \cup M_{k+1}$ must remain a path cover, which means M_{k+1} cannot include any edge incident to a middle point of a path in $\cup_{i=1}^k M_i$. See Algorithm 4.

■ **Algorithm 4** Extension of Algorithm 1.

-
- 1: Run MM_ϵ (or MWM_ϵ for weighted version) on G to find a matching M_1 .
 - 2: Let $i = 1$.
 - 3: while $M_i \neq \emptyset$:
 - 4: Let $G^{(i)} = G$.
 - 5: Remove all $e \in E(G^{(i)}) \setminus (\cup_{k=0}^i M_k)$ from $E(G^{(i)})$ that are incident to at least one middle point of a path (connected component) in $\cup_{k=0}^i M_k$.
 - 6: Contract $G^{(i)}$ on $\cup_{k=0}^i M_k$.
 - 7: Run MM_ϵ (or MWM_ϵ for weighted version) on $G^{(i)}$ to find a matching M_{i+1} .
 - 8: $i = i + 1$
 - 9: **return** $\cup_{k=1}^i M_k$.
-

We leave the computation of the approximation factor of Algorithm 4 as a challenging open problem. Currently, we know that the approximation factor is at most $3/4$. Consider the graph in Figure 5a: the algorithm may select the red edges as M_1 . After contraction, it might select the red edge in Figure 5b as M_2 . In the next iteration, the graph becomes empty, as we must remove any edge incident to a middle point of $M_1 \cup M_2$. Thus, the algorithm terminates with a path of length 3. However, the MPC has 4 edges (see Figure 5c).



■ **Figure 5** An example of a graph where Algorithm 4 terminates after two iterations. The algorithm produces a $\frac{3}{4}$ -approximation of the Maximum Path Cover (MPC).

The main bottleneck to find the approximation ratio of Algorithm 4 is that after the second iteration, there might be a lot of edges that we have to remove from the contracted graph in order to make sure that the union of matchings remains a path cover. More precisely, while running line 5 of Algorithm 4, a bunch of edges that are contained in every maximum path cover might be removed. We are not aware of any argument how to bound the number of these edges. This prevents us to provide an argument like Lemma 5 and Lemma 13. Finding the exact approximation ratio of Algorithm 4 seems to require clever new ideas, already for three matchings.

References

- 1 Anna Adamaszek, Matthias Mnich, and Katarzyna Paluch. New approximation algorithms for (1, 2)-tsp. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPICs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.9.
- 2 Sharareh Alipour, Ermiya Farokhnejad, and Tobias Mömke. Improved approximation algorithms for (1,2)-tsp and max-tsp using path covers in the semi-streaming model, 2025. arXiv:2501.04813.
- 3 Soheil Behnezhad, Mohammad Roghani, Aviad Rubinfeld, and Amin Saberi. Sublinear algorithms for TSP via path covers. *CoRR*, abs/2301.05350, 2023. doi:10.48550/arXiv.2301.05350.
- 4 Yu Chen, Sampath Kannan, and Sanjeev Khanna. Sublinear algorithms and lower bounds for metric TSP cost estimation. 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 30:1–30:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.30.
- 5 Yu Chen, Sanjeev Khanna, and Zihan Tan. Sublinear algorithms and lower bounds for estimating MST and TSP cost in general metrics. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 37:1–37:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.37.
- 6 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi:10.1145/1007352.1007386.

- 7 Doratha E. Drake and Stefan Hougardy. Improved linear time approximation algorithms for weighted matchings. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*, volume 2764 of *Lecture Notes in Computer Science*, pages 14–23. Springer, 2003. doi:10.1007/978-3-540-45198-3_2.
- 8 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 745–754. SIAM, 2005. URL: <http://dl.acm.org/citation.cfm?id=1070432.1070537>.
- 9 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/J.TCS.2005.09.013.
- 10 Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$ -approximate maximum matching with $\text{poly}(1/\epsilon)$ passes in the semi-streaming model and beyond. 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 248–260. ACM, 2022. doi:10.1145/3519935.3520039.
- 11 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.
- 12 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 550–559. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.80.
- 13 Shang-En Huang and Hsin-Hao Su. $(1-1/1013)$ -approximate maximum weighted matching in $\text{poly}(1/1013, \log n)$ time in the distributed and parallel settings. In Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu, editors, *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023*, pages 44–54. ACM, 2023. doi:10.1145/3583668.3594570.
- 14 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. 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 32–45. ACM, 2021. doi:10.1145/3406325.3451009.
- 15 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.
- 16 Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005. doi:10.1007/11538462_15.

- 17 Matthias Mnich and Tobias Mömke. Improved integrality gap upper bounds for traveling salesperson problems with distances one and two. *Eur. J. Oper. Res.*, 266(2):436–457, 2018. doi:10.1016/j.ejor.2017.09.036.
- 18 Tobias Mömke and Ola Svensson. Approximating graphic TSP by matchings. *CoRR*, abs/1104.3090, 2011. arXiv:1104.3090.
- 19 Marcin Mucha. 13/9-approximation for graphic TSP. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPICs*, pages 30–41. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPICs.STACS.2012.30.
- 20 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.
- 21 Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$ -approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG)*, 15(2):1–15, 2018. doi:10.1145/3274668.
- 22 Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976. doi:10.1145/321958.321975.
- 23 András Sebő and Jens Vygen. Shorter tours by nicer ears: $7/5$ -approximation for the graph-tsp, $3/2$ for the path version, and $4/3$ for two-edge-connected subgraphs. *Comb.*, 34(5):597–629, 2014. doi:10.1007/s00493-014-2960-3.
- 24 Xianghui Zhong. On the approximation ratio of the k-opt and lin-kernighan algorithm for metric and graph TSP. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 83:1–83:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.83.
- 25 Xianghui Zhong. On the approximation ratio of the 3-opt algorithm for the $(1, 2)$ -tsp. *CoRR*, abs/2103.00504, 2021. arXiv:2103.00504.

Monotone Weak Distributive Laws over the Lifted Powerset Monad in Categories of Algebras

Quentin Aristote  

Université Paris Cité, CNRS, Inria, IRIF, F-75013, Paris, France

Abstract

In both the category of sets and the category of compact Hausdorff spaces, there is a monotone weak distributive law that combines two layers of non-determinism. Noticing the similarity between these two laws, we study whether the latter can be obtained automatically as a weak lifting of the former. This holds partially, but does not generalize to other categories of algebras. We then characterize when exactly monotone weak distributive laws over powerset monads in categories of algebras exist, on the one hand exhibiting a law combining probabilities and non-determinism in compact Hausdorff spaces and showing on the other hand that such laws do not exist in a lot of other cases.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases weak distributive law, weak extension, weak lifting, iterated distributive law, Yang-Baxter equation, powerset monad, Vietoris monad, Radon monad, Eilenberg-Moore category, regular category, relational extension

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.10

Acknowledgements For sometimes small but always fruitful discussions on topics related to this work, the author thanks Gabriella Böhm, Victor Iwaniack, Jean Goubault-Larrecq, Alexandre Goy, Daniela Petrişan and Sam van Gool.

1 Introduction

In the study of the semantics of programming languages, and since the seminal work of Moggi [26], effectful computations are usually modeled with monads: an effectful function of type $X \multimap Y$ is interpreted as a function of type $X \rightarrow TY$, where the monadic structure on T allows for having identities and compositions of such effectful functions. When considering several effects at the same time, a natural question arises: given monads corresponding to two effects, is it possible to construct a monad that corresponds to the combination of these two effects? In particular, combining probabilities and non-determinism has been a very popular subject of study in the topological and (dually) domain-theoretic settings: see for instance the introduction of [21] for an extensive bibliography on the topic.

The most straightforward way to combine two monads S and T would be to compose their underlying functors, but unfortunately in general the resulting endofunctor ST may not carry the structure of a monad. For this to hold, a sufficient condition is the existence of a *distributive law* of T over S , i.e. a natural transformation $TS \Rightarrow ST$ satisfying four axioms involving the units and multiplications of the two monads [2]. Such a distributive law makes ST into a monad, and its data is equivalent to the data of a *lifting* of S to the Eilenberg-Moore category $\mathbf{EM}(T)$ of T -algebras or to the data of an *extension* of T to the Kleisli category $\mathbf{KI}(S)$ of free S -algebras.

Unfortunately, distributive laws turn out to be not so common: proving that some specific pairs of monads do not admit any distributive law between them has been the focus of several works [22, 29, 12], culminating in [31] where general techniques for proving the absence of distributive laws between monads on **Set**, so-called “no-go theorems”, are exhibited. Among the culprits are the powerset monad **P** and the probability distributions monad **D**: there is no distributive law $\mathbf{PP} \Rightarrow \mathbf{PP}$ nor $\mathbf{DP} \Rightarrow \mathbf{PD}$.



© Quentin Aristote;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 10; pp. 10:1–10:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



A next step in combining monads is thus to weaken the requirements asked for by distributive laws. In [3] a 2-categorical theory of such weakened distributive laws, where the axioms relating to the units of the monads are relaxed, is developed. This theory encompasses two orthogonal kinds of weakened laws, both called weak distributive laws: those of [28], extensively studied in [5, 6, 4], and those more recently put light on in [14] and given further instances of in [18, 19, 7]. This work focuses on the latter kind of laws: in the following a *weak distributive law* of \mathbf{T} over \mathbf{S} will thus be a natural transformation of type $\mathbf{TS} \Rightarrow \mathbf{ST}$ (again), where only three of the four axioms of distributive laws are required. Such a weak distributive law need not make \mathbf{ST} into a monad, but does so of a retract of it when the category is well-behaved. Weak laws $\mathbf{TS} \Rightarrow \mathbf{ST}$ are equivalent to *weak extensions* of \mathbf{T} to $\mathbf{KI}(\mathbf{S})$ – extensions of the semi-monad underlying \mathbf{T} , i.e. of its endofunctor and multiplication but not of its unit – or, when the category is well-behaved, *weak liftings* of \mathbf{S} to $\mathbf{EM}(\mathbf{T})$ – liftings of \mathbf{S} up to a retraction.

Weak extensions are a precious tool for building weak distributive laws, because *monotone extensions* to \mathbf{Rel} – the category of sets and relations, which also happens to be the category of free algebras of the powerset monad \mathbf{P} on \mathbf{Set} – admit a nice characterization: this makes it possible to find weak distributive laws without having to guess their formulæ anymore, and the monotonicity of the extension is a strong indicator that the resulting law will be semantically interesting. In the context of weak distributive laws, this strategy was first used by Garner [14], who exhibited a law $\mathbf{PP} \Rightarrow \mathbf{PP}$, but also a law $\mathbf{\beta P} \Rightarrow \mathbf{P\beta}$ combining the ultrafilter monad $\mathbf{\beta}$ and the powerset monad and whose corresponding weak lifting turned out to be the Vietoris monad \mathbf{V} of closed subsets on the category \mathbf{KHaus} of compact Hausdorff spaces (the algebras of $\mathbf{\beta}$ [24]). The same strategy was then used for instance in [18], where a law $\mathbf{DP} \Rightarrow \mathbf{PD}$, combining probabilities and non-determinism, is described.

In fact, most non-trivial weak distributive laws in the literature (trivial ones are described in [16, §2.2]) are laws of type $\mathbf{TP} \Rightarrow \mathbf{PT}$ built using this strategy. On the other hand, in [19] the goal was to find weak distributive laws in other categories than \mathbf{Set} , and it was reached as a law $\mathbf{VV} \Rightarrow \mathbf{VV}$, combining two layers of non-determinism in a continuous setting, was described. This last law was also built by constructing a monotone weak extension, and its formula is thus very close to that of the law $\mathbf{PP} \Rightarrow \mathbf{PP}$. In topological settings, non-determinism can also be combined with probabilities: weak distributive laws for this purpose are constructed by hand in a very recent pre-print [15].

The goal of the present work is to take over this program of finding non- \mathbf{Set} -based weak distributive laws: we focus here on categories of algebras, which fit in the general framework for monotone weak laws presented in [19]. In particular, we notice in Theorem 12 that the law $\mathbf{VV} \Rightarrow \mathbf{VV}$ is not only very similar to the law $\mathbf{PP} \Rightarrow \mathbf{PP}$, but is also actually some sort of weak lifting of it. We thus study whether there is a general framework for not only weakly lifting monads (as weak distributive laws do), but also weakly lifting weak distributive laws themselves: this framework should yield or simplify the construction of the law $\mathbf{VV} \Rightarrow \mathbf{VV}$, and hopefully generalize it to other categories of algebras, in particular $\mathbf{EM}(\mathbf{P})$ and $\mathbf{EM}(\mathbf{D})$ which have weakly lifted powerset monads thanks to the laws $\mathbf{PP} \Rightarrow \mathbf{PP}$ and $\mathbf{DP} \Rightarrow \mathbf{PD}$.

This question is largely related to the problem of composing weak distributive laws, which was investigated in [17] from the point of view of the Yang-Baxter equations – the usual tool for composing and lifting plain distributive laws [10]. We end up getting a general “no-go theorem” for monotone weak laws over weakly lifted powerset monads: in that sense this work is also close in spirit to [31], where general no-go theorems for (strict) distributive laws are given. While not restricted to monotone distributive laws, these theorems are unlikely to generalize to our setting because they are based on the correspondence between monads and algebraic theories, which is mostly restricted to \mathbf{Set} and does not have any obvious generalization to semi-monads – to which weak distributive laws are deeply related.

This article is organized as follows. In Section 2, we recall definitions and notations for and give examples of monads and weak distributive laws, and also recall the framework of [19] for monotone weak distributive laws in regular categories. This culminates in Theorem 12, where we notice that the law $\mathbf{VV} \Rightarrow \mathbf{VV}$ is some sort of weak lifting of the law $\mathbf{PP} \Rightarrow \mathbf{PP}$. The next two sections focus on lifting weak distributive laws: in Section 3 we study the approach of the Yang-Baxter equation, showing it indeed allows for weakly lifting weak laws but does not apply to the examples we consider; while in Section 4 we focus on the monotonicity of the laws, giving a simple characterization for the existence of monotone weak laws in categories of algebras and applying it to several examples. We conclude in Section 5.

Our main contributions are the following:

- we show that, while the law $\mathbf{VV} \Rightarrow \mathbf{VV}$ is a kind of weak lifting of the law $\mathbf{PP} \Rightarrow \mathbf{PP}$ (Theorem 18), this lifting does not come from a Yang-Baxter equation (Theorem 16), the usual approach to lifting laws: it is an instance of a general no-go theorem for Yang-Baxter equations involving the law $\mathbf{PP} \Rightarrow \mathbf{PP}$ (Theorem 15);
- we characterize the Kleisli category of the weak lifting \bar{S} of a monad S to the algebras of a monad T as the category of T -algebras and $\mathbf{KI}(S)$ -morphisms between them (Theorem 22);
- we give a characterization of the Kleisli categories of weakly lifted powerset monads as subcategories of relations (Theorem 29), in which a certain class of *decomposable morphisms* play a central role: it follows that monads must preserve these decomposable morphisms to have monotone weak distributive laws over weakly lifted powerset monads, and this is in fact a sufficient condition for monads that are themselves weak liftings (Theorem 37);
- concrete instances of this result are then easily derived: we recover independently the law combining probabilities and non-determinism in compact Hausdorff spaces and recently exhibited in [15] (Theorem 38), but we observe otherwise that monotone weak distributive laws over weakly lifted powerset monads in categories of algebras seem very rare (Table 1).

2 Preliminaries

In this section we first recall the theory of weak distributive laws from [14] and the tools that come along, especially the ones developed in [19]. The reader is assumed to be familiar with the basics of category theory, an introduction to which appearing for instance in [8].

2.1 Monads and (Weak) Distributive Laws

► **Definition 1** (monad). A monad on a category \mathcal{C} is the data (T, η^T, μ^T) , often abbreviated T , of an endofunctor $T: \mathcal{C} \rightarrow \mathcal{C}$ and natural transformations $\eta^T: \text{Id} \Rightarrow T$ and $\mu^T: TT \Rightarrow T$ satisfying the axioms $\mu^T \circ T\eta^T = \text{id} = \mu^T \circ \eta^T T$ and $\mu^T \circ T\mu^T = \mu^T \circ \mu^T T$.

► **Example 2** (monads). In this work we will be particularly concerned with the following monads on **Set**. Thereafter X and Y are sets, x is an element of X and $f: X \rightarrow Y$ is a function.

The powerset monad \mathbf{P} . $\mathbf{P}X = \{\text{subsets of } X\}$; for $e \subseteq X$, $(\mathbf{P}f)(e) = f[e] = \{f(x') \mid x' \in e\}$; $\eta_X^{\mathbf{P}}(x) = \{x\}$; for $E \subseteq \mathbf{P}X$, $\mu_X^{\mathbf{P}}(E) = \bigcup E$.

The probability distributions monad \mathbf{D} . $\mathbf{D}X$ is the set of finitely-supported probability distributions on X , i.e. functions $\varphi: X \rightarrow [0, 1]$ such that $\varphi^{-1}(0, 1]$ is finite and $\sum_{x \in X} \varphi(x) = 1$; $\mathbf{D}f$ is the pushforward along f given by $(\mathbf{D}f)(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x)$ for $\varphi \in \mathbf{D}X$ and $y \in Y$; $\eta_X^{\mathbf{D}}(x)$ is the Dirac δ_x for $x \in X$, i.e. the probability distribution such that $\delta_x(x) = 1$; and $\mu^{\mathbf{D}}$ computes the mean of a distribution of distributions, so that for $\Phi \in \mathbf{D}\mathbf{D}X$, $\mu^{\mathbf{D}}(\Phi)(x) = \sum_{\varphi \in \mathbf{D}X} \Phi(\varphi) \cdot \varphi(x)$.

The ultrafilter monad β . βX is the set of maximal filters on X , where filters are sets $E \in \mathbf{PX}$ that are non-empty, up-closed (for inclusion), stable under finite intersections and do not contain the empty set; for an ultrafilter $E \in \beta X$, $(\beta f)(E) = \{e' \supseteq f[e] \mid e \in E\}$; $\eta_X^\beta(x)$ is the principal filter $\{e \in \mathbf{PX} \mid x \in e\}$; and, for $\mathfrak{E} \in \beta\beta X$, $\mu_X^\beta(\mathfrak{E}) = \bigcup \{\bigcap \mathfrak{e} \mid \mathfrak{e} \in \mathfrak{E}\}$.

We will also be concerned with a topological analogue of the powerset monad, defined on the category \mathbf{KHaus} of compact Hausdorff spaces and continuous functions. Thereafter X and Y are compact Hausdorff spaces, x is an element of X , and $f: X \rightarrow Y$ is a continuous function.

The Vietoris monad \mathbf{V} . $\mathbf{V}X$ is the space of closed subsets of X equipped with the topology for which a subbase is given by the sets $\square u = \{c \in \mathbf{V}X \mid c \subset u\}$ and $\diamond u = \{c \in \mathbf{V}X \mid c \cap u \neq \emptyset\}$ where u ranges among all open sets of X ; $(\mathbf{V}f)(c) = f[c] = \{f(x) \mid x \in c\}$ for $c \in \mathbf{V}X$; $\eta_X^\mathbf{V}(x) = \{x\}$; and $\mu_X^\mathbf{V}(C) = \bigcup C$ for $C \in \mathbf{V}\mathbf{V}X$.

We also write \mathbf{P}_* and \mathbf{V}_* for the non-empty powerset and Vietoris monads, respectively obtained by removing the empty set from $\mathbf{P}X$ and $\mathbf{V}X$.

Weak distributive laws are a certain type of natural transformations involving monads.

► **Definition 3** ((weak) distributive law [14, Definition 9]). *A monad \mathbf{T} weakly distributes over a monad \mathbf{S} when there is a weak distributive law of \mathbf{T} over \mathbf{S} , i.e. a natural transformation $\rho: \mathbf{T}\mathbf{S} \Rightarrow \mathbf{S}\mathbf{T}$ such that the Diagrams (η^+) , (μ^-) , and (μ^+) below commute. A (strict) distributive law is a weak distributive law that moreover has the diagram (η^-) below commute.*

$$\begin{array}{ccc}
 & \mathbf{S} & \\
 \eta^{\mathbf{T}\mathbf{S}} \swarrow & & \searrow \eta^{\mathbf{S}\mathbf{T}} \\
 \mathbf{T}\mathbf{S} & \xrightarrow{\rho} & \mathbf{S}\mathbf{T} \\
 & (\eta^-) &
 \end{array}
 \qquad
 \begin{array}{ccccc}
 \mathbf{T}\mathbf{T}\mathbf{S} & \xrightarrow{\mathbf{T}\rho} & \mathbf{T}\mathbf{S}\mathbf{T} & \xrightarrow{\rho^{\mathbf{T}}} & \mathbf{S}\mathbf{T}\mathbf{T} \\
 \mu^{\mathbf{T}\mathbf{S}} \downarrow & & (\mu^-) & & \downarrow \mathbf{S}\mu^{\mathbf{T}} \\
 \mathbf{T}\mathbf{S} & \xrightarrow{\rho} & \mathbf{S}\mathbf{T} & &
 \end{array}$$

$$\begin{array}{ccc}
 & \mathbf{T} & \\
 \mathbf{T}\eta^{\mathbf{S}} \swarrow & & \searrow \eta^{\mathbf{S}\mathbf{T}} \\
 \mathbf{T}\mathbf{S} & \xrightarrow{\rho} & \mathbf{S}\mathbf{T} \\
 & (\eta^+) &
 \end{array}
 \qquad
 \begin{array}{ccccc}
 \mathbf{T}\mathbf{S}\mathbf{S} & \xrightarrow{\rho^{\mathbf{S}}} & \mathbf{T}\mathbf{S}\mathbf{S} & \xrightarrow{\mathbf{S}\rho} & \mathbf{S}\mathbf{S}\mathbf{T} \\
 \mathbf{T}\mu^{\mathbf{S}} \downarrow & & (\mu^+) & & \downarrow \mu^{\mathbf{S}\mathbf{T}} \\
 \mathbf{T}\mathbf{S} & \xrightarrow{\rho} & \mathbf{S}\mathbf{T} & &
 \end{array}$$

If $\rho: \mathbf{T}\mathbf{S} \Rightarrow \mathbf{S}\mathbf{T}$ is a distributive law, $\mathbf{S}\mathbf{T}$, $\eta^{\mathbf{S}\mathbf{T}} = \eta^{\mathbf{S}}\eta^{\mathbf{T}}$ and $\mu^{\mathbf{S}\mathbf{T}} = \mu^{\mathbf{S}}\mu^{\mathbf{T}} \circ \mathbf{S}\rho$ form a monad [2]. If it is only a weak distributive law this is not the case anymore, because $\mu^{\mathbf{S}\mathbf{T}} \circ \eta^{\mathbf{S}\mathbf{T}}\mathbf{S}\mathbf{T}$ is not the identity anymore, only an idempotent (its composite with itself is itself). But suppose now it is a split idempotent, i.e. that there is a functor $\mathbf{S} \bullet \mathbf{T}$ and natural transformations $p: \mathbf{S}\mathbf{T} \Rightarrow \mathbf{S} \bullet \mathbf{T}$ and $i: \mathbf{S} \bullet \mathbf{T} \Rightarrow \mathbf{S}\mathbf{T}$ such that $p \circ i = \text{id}$ and $i \circ p = \mu^{\mathbf{S}\mathbf{T}} \circ \eta^{\mathbf{S}\mathbf{T}}\mathbf{S}\mathbf{T}$. Then $\mathbf{S} \bullet \mathbf{T}$ can be made into a monad [14] with unit $\eta^{\mathbf{S} \bullet \mathbf{T}} = p \circ \eta^{\mathbf{S}}\eta^{\mathbf{T}}$ and multiplication $\mu^{\mathbf{S} \bullet \mathbf{T}} = p \circ \mu^{\mathbf{S}}\mu^{\mathbf{T}} \circ \mathbf{S}\rho$: the monad $\mathbf{S} \bullet \mathbf{T}$ is called the *weak composite* of \mathbf{S} and \mathbf{T} .

► **Example 4** (weak distributive laws). In this work we will be particularly concerned with the following weak distributive laws:

- in \mathbf{Set} , the weak distributive law $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$ given by Equation (1) below has for weak composite the monad $\mathbf{P} \bullet \mathbf{P}$ of sets of subsets closed under non-empty unions [14];
- in \mathbf{KHaus} , there is a weak distributive law $\lambda^{\mathbf{V}/\mathbf{V}}: \mathbf{V}\mathbf{V} \Rightarrow \mathbf{V}\mathbf{V}$ given by Equation (2) below [19];
- in \mathbf{Set} , the weak distributive law $\lambda^{\mathbf{D}/\mathbf{P}}: \mathbf{D}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{D}$ given by Equation (3) below has for weak composite the monad $\mathbf{P} \bullet \mathbf{D}$ of convex sets of finitely supported probability distributions [18].

$$\lambda_X^{\mathbf{P}/\mathbf{P}}(E) = \left\{ e' \in \mathbf{P}X \mid e' \subseteq \bigcup E \text{ and } \forall e \in E, e \cap e' \neq \emptyset \right\} \quad (1)$$

$$\lambda_X^{\mathbf{V}/\mathbf{V}}(C) = \left\{ c' \in \mathbf{V}X \mid c' \subseteq \bigcup C \text{ and } \forall c \in C, c \cap c' \neq \emptyset \right\} \quad (2)$$

$$\lambda_X^{\mathbf{D}/\mathbf{P}}(\Phi) = \left\{ (\mu_X^{\mathbf{D}} \circ \mathbf{D}f)(\Phi) \mid f: \mathbf{P}X \rightarrow \mathbf{D}X \text{ and } \forall e \in \mathbf{P}X, \Phi(e) \neq 0 \Rightarrow f(e) \in \mathbf{D}e \right\} \quad (3)$$

At this point two questions may come to the curious reader: how do we actually find these weak distributive laws – the formulas in Theorem 4 are not trivial – and how exactly is the monad structure on $\mathbf{S} \bullet \mathbf{T}$ constructed. These two questions will respectively be answered in the next two sections, which give two other equivalent presentations of weak distributive laws.

2.2 Regular Categories and Monotone (Weak) Extensions

Given a monad \mathbf{S} , one may form its *Kleisli category* $\mathbf{Kl}(\mathbf{S})$, which is intuitively the category of \mathbf{S} -effectful arrows of \mathbf{C} : its objects are those of \mathbf{C} , its arrows $X \dashrightarrow Y$ are those arrows $X \rightarrow \mathbf{S}Y$ in \mathbf{C} , the identity arrow $X \dashrightarrow X$ is given by $\eta_X^{\mathbf{S}}: X \rightarrow \mathbf{S}X$, and the composite of two arrows $f: X \dashrightarrow Y$ and $g: Y \dashrightarrow Z$ is given by the composition $X \xrightarrow{f} \mathbf{S}Y \xrightarrow{\mathbf{S}g} \mathbf{S}\mathbf{S}Z \xrightarrow{\mu_Z^{\mathbf{S}}} \mathbf{S}Z$ in \mathbf{C} . $\mathbf{Kl}(\mathbf{S})$ comes with an adjunction $\mathbf{F}_{\mathbf{S}}: \mathbf{C} \rightleftarrows \mathbf{Kl}(\mathbf{S}): \mathbf{U}_{\mathbf{S}}$, such that $\mathbf{U}_{\mathbf{S}}\mathbf{F}_{\mathbf{S}} = \mathbf{S}$: the left adjoint $\mathbf{F}_{\mathbf{S}}$ sends objects on themselves and an arrow $f: X \rightarrow Y$ to the *pure* effectful arrow $X \xrightarrow{f} Y \xrightarrow{\eta_Y^{\mathbf{S}}} \mathbf{S}Y$, and the right adjoint $\mathbf{U}_{\mathbf{S}}$ sends an object X on $\mathbf{S}X$ and an effectful arrow $f: X \rightarrow \mathbf{S}Y$ to the arrow $\mathbf{S}X \xrightarrow{\mathbf{S}f} \mathbf{S}\mathbf{S}Y \xrightarrow{\mu_Y^{\mathbf{S}}} \mathbf{S}Y$. The unit of the adjunction is $\eta^{\mathbf{S}}$ and its counit is the natural transformation $\varepsilon^{\mathbf{S}}: \mathbf{F}_{\mathbf{S}}\mathbf{U}_{\mathbf{S}} \Rightarrow \text{Id}$ (in $\mathbf{Kl}(\mathbf{S})$) with components the effectful arrows $\text{id}_{\mathbf{S}X}: \mathbf{S}X \rightarrow \mathbf{S}X$.

An endofunctor $\mathbf{T}: \mathbf{C} \rightarrow \mathbf{C}$ extends to $\mathbf{Kl}(\mathbf{S})$ if there is an endofunctor $\underline{\mathbf{T}}: \mathbf{Kl}(\mathbf{S}) \rightarrow \mathbf{Kl}(\mathbf{S})$ such that $\underline{\mathbf{T}}\mathbf{F}_{\mathbf{S}} = \mathbf{F}_{\mathbf{S}}\mathbf{T}$. If \mathbf{T} and \mathbf{T}' have extensions $\underline{\mathbf{T}}$ and $\underline{\mathbf{T}'}$, a natural transformation $\alpha: \mathbf{T} \Rightarrow \mathbf{T}'$ extends to $\mathbf{Kl}(\mathbf{S})$ if $\underline{\alpha}$, given by $\underline{\alpha}\mathbf{F}_{\mathbf{S}} = \mathbf{F}_{\mathbf{S}}\alpha$, is a natural transformation $\underline{\mathbf{T}} \Rightarrow \underline{\mathbf{T}'}$.

► **Definition 5** (extensions of monads). *Let \mathbf{S} be a monad on \mathbf{C} . A monad $(\mathbf{T}, \eta^{\mathbf{T}}, \mu^{\mathbf{T}})$ on \mathbf{C} weakly extends to $\mathbf{Kl}(\mathbf{S})$ when \mathbf{T} and $\mu^{\mathbf{T}}$ extend to $\mathbf{Kl}(\mathbf{S})$. It extends to $\mathbf{Kl}(\mathbf{S})$ when $\eta^{\mathbf{T}}$ also extends to $\mathbf{Kl}(\mathbf{S})$.*

Giving an extension of a monad \mathbf{T} to $\mathbf{Kl}(\mathbf{S})$ is equivalent to giving a distributive law $\rho: \mathbf{T}\mathbf{S} \Rightarrow \mathbf{S}\mathbf{T}$, just like giving a weak extension thereof is equivalent to giving a weak distributive law of the same type [14]: in both cases, the law ρ induces the extension which sends $f: X \rightarrow \mathbf{S}Y$ to $\mathbf{T}X \xrightarrow{\mathbf{T}f} \mathbf{T}\mathbf{S}Y \xrightarrow{\rho} \mathbf{S}\mathbf{T}Y$, and is computed from the extension as $\rho = \mathbf{U}_{\mathbf{S}}\underline{\mathbf{T}}\varepsilon^{\mathbf{S}}\mathbf{F}_{\mathbf{S}} \circ \eta^{\mathbf{S}}\mathbf{T}\mathbf{S}$. In particular Theorem 4 also yields examples of weak extensions.

Monotone extensions

If $\mathbf{Kl}(\mathbf{S})$ carries more structure than just that of a category then it is natural to ask that extensions of functors preserve this additional structure: intuitively, the more structure preserved, the more semantically canonical the resulting law should be. For instance, recall that $\mathbf{Kl}(\mathbf{P})$ is the category \mathbf{Rel} of sets and relations and that relations between two sets are ordered by inclusion. Laws arising from monotone extensions to \mathbf{Rel} are thus of particular interest.

► **Definition 6** (monotone (weak) distributive laws and extensions). *A (weak) distributive law $\mathbf{T}\mathbf{S} \Rightarrow \mathbf{S}\mathbf{T}$ and its (weak) extension $\underline{\mathbf{T}}$ to $\mathbf{Kl}(\mathbf{S})$ are called monotone when the sets of morphisms of $\mathbf{Kl}(\mathbf{S})$ admit a canonical order that is preserved by $\underline{\mathbf{T}}$.*

10:6 Monotone Weak Distributive Laws in Categories of Algebras

It turns out that such monotone extensions to relations can be defined and characterized in any *regular category*, as described in [19] and recalled now.

Let \mathbf{C} be a finitely complete category. The *kernel pair* of an arrow $f: X \rightarrow Y$ is the pullback $p_1, p_2: X \times_Y X \rightrightarrows X$ of the cospan $X \xrightarrow{f} Y \xleftarrow{f} X$. Assume \mathbf{C} has all the coequalizers of these kernel pairs: these coequalizers are called regular epimorphisms. \mathbf{C} is then called *regular* if these conditions are satisfied and if the regular epimorphisms are stable under pullbacks¹. Regular categories enjoy the fact that every arrow $f: X \rightarrow Y$ may be factored as a regular epimorphism (denoted with \twoheadrightarrow) followed by a monomorphism (denoted with \hookrightarrow), and this factorization is unique up to unique isomorphism – one should think of it as factoring an arrow through its image. **Set** is a canonical example of a regular category.

Regular categories are useful because they are categories where we can speak of relations: if \mathbf{C} is a regular category, a \mathbf{C} -relation between two objects X and Y is a subobject $r: R \hookrightarrow X \times Y$. Relations are preordered: $r: R \hookrightarrow X \times Y$ is smaller than $s: S \hookrightarrow X \times Y$, written $r \leq s$, when there is a monomorphism $m: R \hookrightarrow S$ such that $s \circ m = r$. One may form the category **Rel**(\mathbf{C}) with objects those of \mathbf{C} and arrows $X \rightsquigarrow Y$ the equivalence classes of relations between X and Y . The identity on X is the diagonal $\langle \text{id}_X, \text{id}_X \rangle: X \hookrightarrow X \times X$, and composition of relations $r = \langle r_X, r_Y \rangle: R \hookrightarrow X \times Y$ and $s = \langle s_Y, s_Z \rangle: S \hookrightarrow Y \times Z$, written $s \cdot r: X \rightsquigarrow Z$, is constructed as in Figure 1a: by considering the pullback $R \times_Y S \rightarrow R \times S$ of $R \xrightarrow{r_Y} Y \xleftarrow{s_Y} S$, and taking the monomorphism in the factorization of $R \times_Y S \rightarrow R \times S \xrightarrow{r_X \times s_Z} X \times Z$.

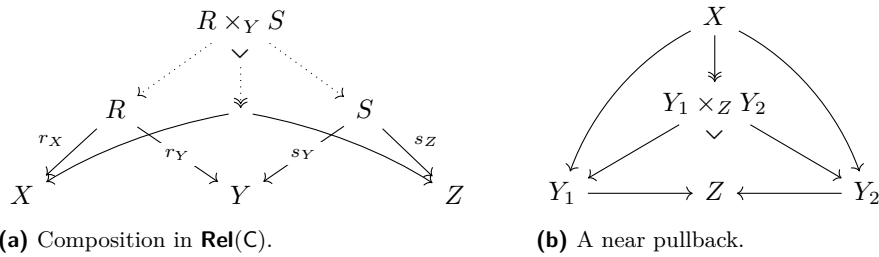


Figure 1 Diagrams involved in the definition of **Rel**(\mathbf{C}).

The graph functor $\mathbf{Graph}_{\mathbf{C}}: \mathbf{C} \rightarrow \mathbf{Rel}(\mathbf{C})$ sends objects to themselves and an arrow $f: X \rightarrow Y$ to the relation $\langle \text{id}_X, f \rangle: X \hookrightarrow X \times Y$. There is also a contravariant transpose functor $-\dagger: \mathbf{Rel}(\mathbf{C}) \rightarrow \mathbf{Rel}(\mathbf{C})$ that sends objects on themselves and a relation $\langle r_X, r_Y \rangle: R \hookrightarrow X \times Y$ to $\langle r_Y, r_X \rangle: R \hookrightarrow Y \times X$. All in all, a relation $\langle f, g \rangle: R \hookrightarrow X \times Y$ can also be written as the composite $\mathbf{Graph}_{\mathbf{C}}g \cdot (\mathbf{Graph}_{\mathbf{C}}f)^\dagger$: we will often omit **Graph** and write directly $g \cdot f^\dagger$.

If $\mathbf{KI}(\mathbf{S})$ is a wide² subcategory of **Rel**(\mathbf{C}) (with $\mathbf{Graph}_{\mathbf{C}}$ being the left adjoint in the Kleisli adjunction), an extension to $\mathbf{KI}(\mathbf{S})$ can be constructed by first finding a monotone extension to **Rel**(\mathbf{C}) and then restricting this extension to $\mathbf{KI}(\mathbf{S})$. This is a very useful technique because such monotone extensions to **Rel**(\mathbf{C}), called *relational extensions*, have a very nice characterization in terms of near pullbacks: a square is a near pullback when its limiting morphism into the corresponding pullback is a regular epimorphism – as in Figure 1b. A functor between regular categories is said to be *nearly cartesian* when it sends pullbacks

¹ a class of arrows is stable under pullbacks when for every pullback square $a \circ b = c \circ d$, if a is in the class then so is d
² a wide subcategory is a subcategory that contains all objects of the bigger category

on near pullbacks and preserves³ regular epimorphisms, or equivalently when it preserves near pullbacks, while a natural transformation $\alpha: F \Rightarrow G$ between nearly cartesian functors is *nearly cartesian* as well when its naturality squares $\alpha \circ Ff = Gf \circ \alpha$ are near pullbacks.

► **Theorem 7** ([19, Theorem 6]). *Let $F: C \rightarrow D$ be a functor between regular categories. F has a relational extension, i.e. an order-preserving functor $\mathbf{Rel}(F): \mathbf{Rel}(C) \rightarrow \mathbf{Rel}(D)$ such that $\mathbf{Rel}(F) \mathbf{Graph}_C = \mathbf{Graph}_D F$, if and only if F is nearly cartesian. In that case there is only one possible such $\mathbf{Rel}(F)$, given by $\mathbf{Rel}(F)(g \cdot f^\dagger) = (Fg) \cdot (Ff)^\dagger$.*

Let $F, G: C \rightrightarrows D$ be two such functors and $\alpha: F \Rightarrow G$ be a natural transformation between them. Then $\mathbf{Rel}(\alpha)$, given by $\mathbf{Rel}(\alpha) \mathbf{Graph}_D = \mathbf{Graph}_D \alpha$, is a natural transformation $\mathbf{Rel}(F) \Rightarrow \mathbf{Rel}(G)$, called the relational extension of α , if and only if α is nearly cartesian.

Just like we omit to write **Graph**, we will often omit to write the functor **Rel**, and instead write $F(g \cdot f^\dagger)$ for $\mathbf{Rel}(F)(g \cdot f^\dagger) = Fg \cdot (Ff)^\dagger$ and α for $\mathbf{Rel}(\alpha)$.

► **Example 8** (monotone extensions). **Set** is a regular category whose regular epimorphisms are the surjections and such that $\mathbf{Rel}(\mathbf{Set}) = \mathbf{Rel} \cong \mathbf{KI}(\mathbf{P})$. The endofunctors and the multiplications of the monads **P** and **D** are nearly cartesian, hence **P** and **D** have monotone weak extensions to $\mathbf{KI}(\mathbf{P})$: this is how the weak distributive laws $\mathbf{PP} \Rightarrow \mathbf{PP}$ and $\mathbf{DP} \Rightarrow \mathbf{PD}$ of Theorem 4 were constructed [14, 18]. **KHaus** is also a regular category: its regular epimorphisms are the surjective continuous functions and $\mathbf{Rel}(\mathbf{KHaus})$ is the category of compact Hausdorff spaces and closed relations, i.e. closed subsets of $X \times Y$, while $\mathbf{KI}(\mathbf{V})$ is the category of compact Hausdorff spaces and continuous relations, i.e. closed relations $r: X \leftrightarrow Y$ such that $r^{-1}[u]$ is open in X for every open u of Y . The endofunctor and multiplication of **V** are nearly cartesian hence they have a relational extension, which restricts to continuous relations: this yields a monotone weak extension of **V** to $\mathbf{KI}(\mathbf{V})$, and this is how the weak distributive law $\mathbf{VV} \Rightarrow \mathbf{VV}$ of Theorem 4 was constructed [19].

2.3 Weak Liftings

Let **T** be a monad over a category **C**. Its category of algebras $\mathbf{EM}(\mathbf{T})$ is the category whose objects are pairs (A, a) of a **C**-object A and a **C**-arrow $a: \mathbf{T}A \rightarrow A$, and whose arrows $(A, a) \rightarrow (B, b)$ are those **C**-arrows $A \rightarrow B$ such that $f \circ a = b \circ \mathbf{T}f$ – the identity morphism is the one with the identity as its underlying **C**-arrow, and composition of morphisms is done by composing the underlying **C**-arrows. Just like for $\mathbf{KI}(\mathbf{T})$, there is an adjunction $\mathbf{F}^\mathbf{T}: \mathbf{C} \rightleftarrows \mathbf{EM}(\mathbf{T}): \mathbf{U}^\mathbf{T}$ such that $\mathbf{U}^\mathbf{T} \mathbf{F}^\mathbf{T} = \mathbf{T}$: the left adjoint $\mathbf{F}^\mathbf{T}$ sends an algebra (A, a) to its carrier object A and a morphism $(A, a) \rightarrow (B, b)$ to the underlying arrow $A \rightarrow B$, while $\mathbf{U}^\mathbf{T}$ sends an object X to the free **T**-algebra on X , given by the pair $(\mathbf{T}X, \mu_X^\mathbf{T})$, and an arrow $f: X \rightarrow Y$ to the morphism $(\mathbf{T}X, \mu_X^\mathbf{T}) \rightarrow (\mathbf{T}Y, \mu_Y^\mathbf{T})$ with underlying **C**-arrow $\mathbf{T}f: \mathbf{T}X \rightarrow \mathbf{T}Y$. There is finally a natural transformation $\varepsilon^\mathbf{T}: \mathbf{F}^\mathbf{T} \mathbf{U}^\mathbf{T} \rightarrow \text{Id}$ (in $\mathbf{EM}(\mathbf{T})$) given by $\mathbf{U}^\mathbf{T} \varepsilon_{(A,a)}^\mathbf{T} = a: \mathbf{T}A \rightarrow A$.

► **Definition 9** (weak liftings [3, Definitions 4.1 and 4.2]). *A weak lifting of an endofunctor $S: C \rightarrow C$ to $\mathbf{EM}(\mathbf{T})$ is the data of an endofunctor $\bar{S}: \mathbf{EM}(\mathbf{T}) \rightarrow \mathbf{EM}(\mathbf{T})$ along with natural transformations $\pi_S^\mathbf{T}: \mathbf{S} \mathbf{U}^\mathbf{T} \Rightarrow \mathbf{U}^\mathbf{T} \bar{S}$ and $\iota_S^\mathbf{T}: \mathbf{U}^\mathbf{T} \bar{S} \Rightarrow \mathbf{S} \mathbf{U}^\mathbf{T}$ – also written π_S and ι_S when not ambiguous – such that $\pi_S^\mathbf{T} \circ \iota_S^\mathbf{T} = \text{id}$. The composite $\iota_S^\mathbf{T} \circ \pi_S^\mathbf{T}$ is then written $\kappa_S^\mathbf{T}$.*

³ throughout this work we will say that a functor preserves a class of diagrams whenever it sends any diagram in that class to a diagram in that class

10:8 Monotone Weak Distributive Laws in Categories of Algebras

Let $\alpha: \mathbb{S} \Rightarrow \mathbb{R}$ be a natural transformation between two \mathbb{C} -endofunctors with weak liftings $(\bar{\mathbb{S}}, \pi_{\bar{\mathbb{S}}}, \iota_{\bar{\mathbb{S}}})$ and $(\bar{\mathbb{R}}, \pi_{\bar{\mathbb{R}}}, \iota_{\bar{\mathbb{R}}})$. If a natural transformation $\bar{\alpha}: \bar{\mathbb{S}} \Rightarrow \bar{\mathbb{R}}$ has Diagram (π) , Diagram (ι) or both Diagrams (π) and (ι) below commute, it is respectively a weak π -, weak ι - or a weak lifting thereof.

$$\begin{array}{ccc} \mathbf{S}\mathbf{U}^{\mathbf{T}} & \xrightarrow{\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} \\ \alpha\mathbf{U}^{\mathbf{T}}\downarrow & (\pi) & \downarrow\mathbf{U}^{\mathbf{T}}\bar{\alpha} \\ \mathbf{R}\mathbf{U}^{\mathbf{T}} & \xrightarrow{\pi_{\bar{\mathbb{R}}}^{\mathbf{T}}} & \mathbf{U}^{\mathbf{T}}\bar{\mathbb{R}} \end{array} \quad \begin{array}{ccc} \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} & \xrightarrow{\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{S}\mathbf{U}^{\mathbf{T}} \\ \mathbf{U}^{\mathbf{T}}\bar{\alpha}\downarrow & (\iota) & \downarrow\alpha\mathbf{U}^{\mathbf{T}} \\ \mathbf{U}^{\mathbf{T}}\bar{\mathbb{R}} & \xrightarrow{\iota_{\bar{\mathbb{R}}}^{\mathbf{T}}} & \mathbf{R}\mathbf{U}^{\mathbf{T}} \end{array}$$

As discussed in [3], weak π - and weak ι -liftings are necessarily unique and given by $\mathbf{U}^{\mathbf{T}}\bar{\alpha} = \pi_{\bar{\mathbb{R}}}^{\mathbf{T}} \circ \alpha \circ \iota_{\bar{\mathbb{S}}}^{\mathbf{T}}$ if they exist. Their existence is moreover fully characterized:

► **Theorem 10** ([3, Proposition 4.3 and Theorem 4.4]). *Suppose idempotents split in \mathbb{C} . Then idempotents also split in $\mathbf{EM}(\mathbb{T})$, and having a weak lifting of $\mathbb{S}: \mathbb{C} \rightarrow \mathbb{C}$ to $\mathbf{EM}(\mathbb{T})$ is equivalent to having a law $\rho: \mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ that has Diagram (μ^+) commute.*

Fix now liftings $\bar{\mathbb{S}}$ and $\bar{\mathbb{R}}$ of \mathbb{S} and \mathbb{R} given by laws $\rho: \mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ and $\sigma: \mathbb{T}\mathbb{R} \Rightarrow \mathbb{R}\mathbb{T}$. Then, $\alpha: \mathbb{S} \Rightarrow \mathbb{R}$ respectively has a weak π -, weak ι - or weak lifting if and only if it makes Diagram (7), Diagram (8) or both Diagrams (7) and (8) below commute. The latter case holds equivalently if and only if $\sigma \circ \mathbb{T}\alpha = \alpha\mathbb{T} \circ \rho$.

$$\begin{array}{ccc} \mathbb{T}\mathbb{S} & \xrightarrow{\rho} & \mathbb{S}\mathbb{T} \\ \mathbb{T}\eta^{\mathbb{T}}\mathbb{S}\downarrow & (\gamma) & \downarrow\alpha\mathbb{T} \\ \mathbb{T}\mathbb{T}\mathbb{S} & \xrightarrow{\mathbb{T}\rho} \mathbb{T}\mathbb{S}\mathbb{T} \xrightarrow{\mathbb{T}\alpha\mathbb{T}} \mathbb{T}\mathbb{R}\mathbb{T} \xrightarrow{\sigma\mathbb{T}} \mathbb{R}\mathbb{T}\mathbb{T} \xrightarrow{\mathbb{R}\mu^{\mathbb{T}}} & \mathbb{R}\mathbb{T} \\ \eta^{\mathbb{T}}\mathbb{T}\mathbb{S}\uparrow & (\delta) & \uparrow\sigma \\ \mathbb{T}\mathbb{S} & \xrightarrow{\mathbb{T}\alpha} & \mathbb{T}\mathbb{R} \end{array}$$

This correspondence is moreover compositional: the composition of the weak (resp. weak π -, weak ι -) liftings of two functors is a weak (resp. weak π -, weak ι -) lifting of their composite.

In [14], Garner instantiates Theorem 10 to give another presentation of weak distributive laws: if idempotents split in \mathbb{C} , a weak distributive law $\mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ is equivalently given by a weak lifting of $(\mathbb{S}, \eta^{\mathbb{S}}, \mu^{\mathbb{S}})$ to $\mathbf{EM}(\mathbb{T})$, i.e. by weak liftings of \mathbb{S} , $\eta^{\mathbb{S}}$ and $\mu^{\mathbb{S}}$, respectively coming from Diagrams (μ^-) , (η^+) , and (μ^+) . That these two natural transformations weakly lift respectively means that Diagrams $(\pi \circ \eta)$ and $(\iota \circ \eta)$ and Diagrams $(\pi \circ \mu)$ and $(\iota \circ \mu)$ below commute. When Diagram (η^-) also commutes, ρ is a (strict) distributive law and this weak lifting is a (strict) lifting: $\pi_{\bar{\mathbb{S}}}$ and $\iota_{\bar{\mathbb{S}}}$ are both the identity.

$$\begin{array}{ccc} & \mathbf{U}^{\mathbf{T}} & \\ \eta^{\mathbb{S}}\mathbf{U}^{\mathbf{T}} \swarrow & & \searrow \mathbf{U}^{\mathbf{T}}\eta^{\bar{\mathbb{S}}} \\ \mathbf{S}\mathbf{U}^{\mathbf{T}} & \xrightarrow{\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} \end{array} \quad \begin{array}{ccc} \mathbf{S}\mathbf{S}\mathbf{U}^{\mathbf{T}} & \xrightarrow{S\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{S}\mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} \xrightarrow{\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}\bar{\mathbb{S}}} & \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}}\bar{\mathbb{S}} \\ \mu^{\mathbb{S}}\mathbf{U}^{\mathbf{T}}\downarrow & (\pi \circ \mu) & \downarrow\mathbf{U}^{\mathbf{T}}\mu^{\bar{\mathbb{S}}} \\ \mathbf{S}\mathbf{U}^{\mathbf{T}} & \xrightarrow{\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} \end{array}$$

$$\begin{array}{ccc} & \mathbf{U}^{\mathbf{T}} & \\ \mathbf{U}^{\mathbf{T}}\eta^{\bar{\mathbb{S}}} \swarrow & & \searrow \eta^{\mathbb{S}}\mathbf{U}^{\mathbf{T}} \\ \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} & \xrightarrow{\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{S}\mathbf{U}^{\mathbf{T}} \end{array} \quad \begin{array}{ccc} \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}}\bar{\mathbb{S}} & \xrightarrow{\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}\bar{\mathbb{S}}} & \mathbf{S}\mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} \xrightarrow{S\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{S}\mathbf{S}\mathbf{U}^{\mathbf{T}} \\ \mathbf{U}^{\mathbf{T}}\mu^{\bar{\mathbb{S}}}\downarrow & (\iota \circ \mu) & \downarrow\mu^{\mathbb{S}}\mathbf{U}^{\mathbf{T}} \\ \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}} & \xrightarrow{\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}} & \mathbf{S}\mathbf{U}^{\mathbf{T}} \end{array}$$

The weak composite monad $\mathbf{S} \bullet \mathbf{T}$ corresponding to a weak distributive law $\mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ can be retrieved from the weak lifting of \mathbb{S} to $\mathbf{EM}(\mathbb{T})$ by setting $\mathbf{S} \bullet \mathbf{T} = \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}}\mathbf{F}^{\mathbf{T}}$, $\eta^{\mathbf{S} \bullet \mathbf{T}} = \mathbf{U}^{\mathbf{T}}\eta^{\bar{\mathbb{S}}}\mathbf{F}^{\mathbf{T}} \circ \eta^{\mathbf{T}}$ and $\mu^{\mathbf{S} \bullet \mathbf{T}} = \mathbf{U}^{\mathbf{T}}\mu^{\bar{\mathbb{S}}}\mathbf{F}^{\mathbf{T}} \circ \mathbf{U}^{\mathbf{T}}\bar{\mathbb{S}}\varepsilon^{\mathbf{T}}\bar{\mathbb{S}}\mathbf{F}^{\mathbf{T}}$. $\kappa_{\bar{\mathbb{S}}}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}$ is moreover the idempotent $\mu^{\mathbf{S}\mathbf{T}} \circ \eta^{\mathbf{S}\mathbf{T}}\mathbf{S}\mathbf{T}: \mathbf{S}\mathbf{T} \Rightarrow \mathbf{S}\mathbf{T}$, and its splitting is thus given by $\pi_{\bar{\mathbb{S}}}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}: \mathbf{S}\mathbf{T} \Rightarrow \mathbf{S} \bullet \mathbf{T}$ and $\iota_{\bar{\mathbb{S}}}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}: \mathbf{S} \bullet \mathbf{T} \Rightarrow \mathbf{S}\mathbf{T}$.

► **Example 11** (weak liftings). All idempotents split in **Set** (they factor through their image). The algebras of **P** are the *complete join-semilattices* (we write $\mathbf{EM}(\mathbf{P}) \cong \mathbf{JSL}$): the weak lifting corresponding to the law $\mathbf{PP} \Rightarrow \mathbf{PP}$ is the monad of *subsets closed under non-empty joins* [19]. The algebras of **D** are the *barycentric algebras*, also called *convex spaces* (we write $\mathbf{EM}(\mathbf{D}) \cong \mathbf{Conv}$): the weak lifting corresponding to the law $\mathbf{DP} \Rightarrow \mathbf{PD}$ is the monad of *convex-closed subsets* [18]. Finally, there is also a weak distributive law $\mathbf{\beta P} \Rightarrow \mathbf{P\beta}$. Assuming the axiom of choice, the algebras of **\beta** are the compact Hausdorff spaces [24] (we write $\mathbf{EM}(\mathbf{\beta}) \cong \mathbf{KHaus}$): the corresponding weak lifting is the Vietoris monad **V**. In that case $\pi_{\mathbf{P}}$ computes the topological closure of a subset, while $\iota_{\mathbf{P}}$ embeds the set of closed sets into the set of all subsets.

3 Weakly Lifting Weak Distributive Laws

With Theorem 11 and the fact that **V** is a weak lifting of **P** to **KHaus** in mind, we may now notice that not only are the two laws $\mathbf{PP} \Rightarrow \mathbf{PP}$ (1) and $\mathbf{VV} \Rightarrow \mathbf{VV}$ (2) very similar, but the second one seems to be some kind of weak lifting to $\mathbf{KHaus} \cong \mathbf{EM}(\mathbf{\beta})$ of the first one:

► **Lemma 12.** $\mathbf{U}^{\beta} \lambda^{\mathbf{V}/\mathbf{V}} = \pi_{\mathbf{P}} \mathbf{V} \circ \mathbf{P} \pi_{\mathbf{P}} \circ \lambda^{\mathbf{P}/\mathbf{P}} \mathbf{U}^{\beta} \circ \mathbf{P} \iota_{\mathbf{P}} \circ \iota_{\mathbf{P}} \mathbf{V}$.

Proof. Recall from [14] that, if X is a compact Hausdorff space that we see as a **\beta**-algebra, the X -component of $\pi_{\mathbf{P}}$ is the function $\mathbf{P}X \rightarrow \mathbf{V}X$ that takes a subset of a compact Hausdorff space and outputs its closure, while the X -component of $\iota_{\mathbf{P}}$ is the function $\mathbf{V}X \rightarrow \mathbf{P}X$ that embeds closed subsets in the set of all subsets. Hence for C a closed set of closed sets,

$$\begin{aligned} \left(\mathbf{P} \pi_{\mathbf{P}} \circ \lambda^{\mathbf{P}/\mathbf{P}} \mathbf{U}^{\beta} \circ \mathbf{P} \iota_{\mathbf{P}} \circ \iota_{\mathbf{P}} \mathbf{V} \right)_X (C) &= \left\{ \bar{e} \mid e \subseteq \bigcup C \text{ and } \forall c \in C, c \cap e \neq \emptyset \right\} \\ &= \left\{ c \in \mathbf{V}X \mid c \subseteq \bigcup C \text{ and } \forall c' \in C, c \cap c' \neq \emptyset \right\} \\ &= \left(\iota_{\mathbf{P}} \mathbf{V} \circ \mathbf{U}^{\beta} \lambda^{\mathbf{V}/\mathbf{V}} \right)_X (C) \end{aligned}$$

(where \bar{e} denotes the closure of e). Indeed if $e \subseteq \bigcup C$ then $\bar{e} \subseteq \bigcup C$ because $\bigcup C = \mu^{\mathbf{V}}(C)$ is closed, and if $e \cap c \neq \emptyset$ then $\bar{e} \cap c \neq \emptyset$.

Because $\pi_{\mathbf{P}} \circ \iota_{\mathbf{P}} = \text{id}$, $\mathbf{U}^{\beta} \lambda^{\mathbf{V}/\mathbf{V}} = \pi_{\mathbf{P}} \mathbf{V} \circ \mathbf{P} \pi_{\mathbf{P}} \circ \lambda^{\mathbf{P}/\mathbf{P}} \mathbf{U}^{\beta} \circ \mathbf{P} \iota_{\mathbf{P}} \circ \iota_{\mathbf{P}} \mathbf{V}$. ◀

Is this just a coincidence, or is this an instance of Theorem 10? And in the latter case, can the weak distributivity of $\lambda^{\mathbf{V}/\mathbf{V}}$ be automatically derived from that of $\lambda^{\mathbf{P}/\mathbf{P}}$, and does this law $\mathbf{PP} \Rightarrow \mathbf{PP}$ also weakly lift to laws on other categories of algebras where the powerset monad weakly lifts, say $\mathbf{EM}(\mathbf{P})$ and $\mathbf{EM}(\mathbf{D})$?

In this section we thus consider three monads $(\mathbf{T}, \eta^{\mathbf{T}}, \mu^{\mathbf{T}})$, $(\mathbf{S}, \eta^{\mathbf{S}}, \mu^{\mathbf{S}})$ and $(\mathbf{R}, \eta^{\mathbf{R}}, \mu^{\mathbf{R}})$ on a category \mathbf{C} , and three weak distributive laws $\rho: \mathbf{TS} \Rightarrow \mathbf{ST}$, $\sigma: \mathbf{TR} \Rightarrow \mathbf{RT}$ and $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ (a mnemonic for which is which is that ρ does not involve \mathbf{R} , σ does not involve \mathbf{S} and τ does not involve \mathbf{T}). We assume that idempotents split in \mathbf{C} , so that the corresponding weak composite and weak liftings all exist.

3.1 The Yang-Baxter Equation for Weak Distributive Laws

The standard way to lift (strict) distributive laws is to use the so-called Yang-Baxter equation. The Yang-Baxter equation for the three laws ρ , σ and τ holds when diagram (YB) below commutes.

$$\begin{array}{ccccc} & & \mathbf{TRS} & \xrightarrow{\sigma^{\mathbf{S}}} & \mathbf{RTS} & & \\ & \nearrow^{\mathbf{T}\tau} & & & & \searrow^{\mathbf{R}\rho} & \\ \mathbf{TSR} & & & & & & \mathbf{RST} \\ & \searrow_{\rho^{\mathbf{R}}} & & & & \nearrow_{\tau^{\mathbf{T}}} & \\ & & \mathbf{STR} & \xrightarrow{\sigma^{\mathbf{S}}} & \mathbf{SRT} & & \end{array} \quad \text{(YB)}$$

10:10 Monotone Weak Distributive Laws in Categories of Algebras

If these laws are strict distributive laws, then it is well-known since [10] that the Yang-Baxter equation is enough to show that $R\rho \circ \sigma S: TRS \Rightarrow RST$ is a distributive law of T over RS , and that the distributive law $\tau: SR \Rightarrow RS$ lifts to a distributive law $\bar{\tau}: \bar{S}\bar{R} \Rightarrow \bar{R}\bar{S}$ in $\mathbf{EM}(T)$. The composition of weak distributive laws using the Yang-Baxter equation was investigated in [17]. A notable result is the following:

► **Proposition 13** ([17, Theorem 4.3]). *If the weak distributive laws $\rho: TS \Rightarrow ST$, $\sigma: TR \Rightarrow RT$ and $\tau: SR \Rightarrow RS$ have Diagram (YB) commute, then $\pi_{\bar{R}}^S \mathbf{F}^S T \circ R\rho \circ \sigma S \circ T \iota_{\bar{R}}^S \mathbf{F}^S$ is a weak distributive law $T(R \bullet S) \Rightarrow (R \bullet S)T$.*

The Yang-Baxter equation thus allows for weakly lifting $R \bullet S$ to $\mathbf{EM}(T)$. More importantly for our purpose, we show it also allows for weakly lifting the weak distributive law $SR \Rightarrow RS$:

► **Theorem 14.** *Weak distributive laws $\rho: TS \Rightarrow ST$, $\sigma: TR \Rightarrow RT$ and $\tau: SR \Rightarrow RS$ satisfy the Yang-Baxter equation if and only if $\tau: SR \Rightarrow RS$ weakly lifts to $\mathbf{EM}(T)$, i.e. if there is a natural transformation $\bar{\tau}: \bar{S}\bar{R} \Rightarrow \bar{R}\bar{S}$ such that Diagrams (14) and (15) commute.*

If this holds, $\bar{\tau}$ is a weak distributive law, and the weak composite $\bar{R} \bullet \bar{S}$ and the weak lifting $\overline{R \bullet S}$ (recall that $R \bullet S$ weakly lifts to $\mathbf{EM}(T)$ by Theorem 13) can be chosen to be equal (as monads).

$$\begin{array}{ccc}
 \text{SRU}^T \xrightarrow{S\pi_R} \text{SU}^T\bar{R} \xrightarrow{\pi_S\bar{R}} \text{U}^T\bar{S}\bar{R} & & \text{U}^T\bar{S}\bar{R} \xrightarrow{\iota_S\bar{R}} \text{SU}^T\bar{R} \xrightarrow{S\iota_R} \text{SRU}^T \\
 \tau\text{U}^T \downarrow & (14) & \downarrow \text{U}^T\bar{\tau} \\
 \text{RSU}^T \xrightarrow{R\pi_S} \text{RU}^T\bar{S} \xrightarrow{\pi_R\bar{S}} \text{U}^T\bar{R}\bar{S} & & \text{U}^T\bar{R}\bar{S} \xrightarrow{\iota_R\bar{S}} \text{RU}^T\bar{S} \xrightarrow{R\iota_S} \text{RSU}^T \\
 & & \downarrow \tau\text{U}^T
 \end{array}$$

When Theorem 14 holds it immediately follows that $\text{U}\bar{\tau} = \pi_{\bar{R}}\bar{S} \circ R\pi_S \circ \tau\text{U}^T \circ S\iota_R \circ \iota_S\bar{R}$, which is exactly the result we got in Theorem 12 for $\bar{\tau} = \lambda^{\mathbf{V}/\mathbf{V}}: \mathbf{V}\mathbf{V} \Rightarrow \mathbf{V}\mathbf{V}$ and $\tau = \lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$. It would thus be a reasonable conjecture that $\lambda^{\mathbf{B}/\mathbf{P}}: \mathbf{B}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{B}$, $\lambda^{\mathbf{B}/\mathbf{P}}: \mathbf{B}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{B}$ and $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$ satisfy the Yang-Baxter equation, from which we would immediately retrieve the weak distributivity of $\lambda^{\mathbf{V}/\mathbf{V}}: \mathbf{V}\mathbf{V} \Rightarrow \mathbf{V}\mathbf{V}$ but also learn that the weak composite $\mathbf{V} \bullet \mathbf{V}$ is a weak lifting of $\mathbf{P} \bullet \mathbf{P}$. Unfortunately, the Yang-Baxter equation does not hold in that case. A rather simple way to see this is to notice that $\lambda^{\mathbf{V}/\mathbf{V}}$ does not make Diagram (15) commute, i.e. it is not a ι -lifting of $\lambda^{\mathbf{P}/\mathbf{P}}$. More generally, we show the following no-go theorem for weak ι -liftings:

► **Proposition 15.** *Let $\lambda^{\mathbf{T}/\mathbf{P}}: \mathbf{T}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{T}$ be a weak distributive law with corresponding weak lifting $\bar{\mathbf{P}}$, and write $\bar{\mathbf{P}}\bar{\mathbf{P}}A = (\mathbf{P}\iota_{\bar{\mathbf{P}}} \circ \iota_{\bar{\mathbf{P}}}\mathbf{P}) \left[\text{U}^T\bar{\mathbf{P}}\bar{\mathbf{P}}(A, a) \right]$ when (A, a) is a T -algebra. If there is an (A, a) such that $\{A\} \in \bar{\mathbf{P}}\bar{\mathbf{P}}A$ and $\mathbf{P}_*A \notin \bar{\mathbf{P}}\bar{\mathbf{P}}A$, then $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$ does not have a weak ι -lifting to $\mathbf{EM}(T)$.*

► **Corollary 16.** *There is no weak ι -lifting (let alone weak liftings) of $\lambda^{\mathbf{P}/\mathbf{P}}$ to \mathbf{KHaus} , \mathbf{JSL} or \mathbf{Conv} .*

Proof sketch.

1. Given a compact Hausdorff space given as a \mathbf{B} -algebra (A, a) , $\bar{\mathbf{P}}\bar{\mathbf{P}}A$ is the set of all closed sets (in the Vietoris topology) of closed sets of (A, a) . $\{A\}$ is such a closed set of closed sets (all singletons and the whole set are always closed in a compact Hausdorff space) but \mathbf{P}_*A is not in general because it contains all non-empty sets, in particular non-closed sets if there are any (which is the case for the unit interval, for instance). ◀

► **Remark 17.** It is not hard to show that the Yang-Baxter equation holding for $\rho: \mathbf{TS} \Rightarrow \mathbf{ST}$, $\sigma: \mathbf{TR} \Rightarrow \mathbf{RT}$ and $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ is also equivalent to $\rho: \mathbf{TS} \Rightarrow \mathbf{ST}$ having an extension $\underline{\rho}: \underline{\mathbf{TS}} \Rightarrow \underline{\mathbf{ST}}$ to $\mathbf{KI}(\mathbf{R})$. In the case of $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{\beta P} \Rightarrow \mathbf{P\beta}$, $\underline{\lambda}$ is in fact a lax monad on \mathbf{Rel} whose algebras are the topological spaces [1]. Unfortunately, we have just shown that the Yang-Baxter equation does not hold for $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{\beta P} \Rightarrow \mathbf{P\beta}$, $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{PP} \Rightarrow \mathbf{PP}$ and $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{PP} \Rightarrow \mathbf{PP}$, and so do not get a way to weakly lift $\underline{\mathbf{P}}$ to topological spaces for free.

Theorem 14 does have some concrete instances: in [17], Goy gives a substantial number of examples of triples of weak distributive laws for which the Yang-Baxter equation holds, although these examples all involve at least one strictly distributive law out of the three.

3.2 The π -Yang-Baxter Equation

We still do not have an explanation for why $\lambda^{\mathbf{V}/\mathbf{V}}$ looks so much like $\lambda^{\mathbf{P}/\mathbf{P}}$. But $\lambda^{\mathbf{V}/\mathbf{V}}$ being a weak lifting of $\lambda^{\mathbf{P}/\mathbf{P}}$ is not a necessary condition for retrieving Theorem 12: in fact, $\lambda^{\mathbf{V}/\mathbf{V}}$ being only a weak ι - or π -lifting of $\lambda^{\mathbf{P}/\mathbf{P}}$ would be enough. We saw in Theorem 16 that the weak ι -lifting hypothesis was a dead-end: how about $\lambda^{\mathbf{V}/\mathbf{V}}$ being a weak π -lifting of $\lambda^{\mathbf{P}/\mathbf{P}}$? This turns out to be true, although the proof is of course more involved than that of the weaker Theorem 12.

► **Lemma 18.** $\lambda^{\mathbf{V}/\mathbf{V}}: \mathbf{VV} \Rightarrow \mathbf{VV}$ is a weak π -lifting of $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{PP} \Rightarrow \mathbf{PP}$.

Theorem 14 adapts to weak π -liftings, hence we immediately retrieve as a consequence of Theorem 18 that $\lambda^{\mathbf{V}/\mathbf{V}}: \mathbf{VV} \Rightarrow \mathbf{VV}$ is a weak distributive law.

► **Proposition 19.** Weak distributive laws $\rho: \mathbf{TS} \Rightarrow \mathbf{ST}$, $\sigma: \mathbf{TR} \Rightarrow \mathbf{RT}$ and $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ satisfy the π -Yang-Baxter equation, given by Diagram (π -YB), if and only if $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ weakly π -lifts to $\mathbf{EM}(\mathbf{T})$, i.e. if there is a natural transformation $\bar{\tau}: \bar{\mathbf{SR}} \Rightarrow \bar{\mathbf{RS}}$ such that Diagram (14) commutes.

If this holds, $\bar{\tau}$ is a weak distributive law.

$$\begin{array}{ccccc}
 \mathbf{TSR} & \xrightarrow{\mathbf{T}\tau} & \mathbf{TRS} & \xrightarrow{\sigma\mathbf{S}} & \mathbf{RTS} & \xrightarrow{\mathbf{R}\rho} & \mathbf{RST} \\
 \rho\mathbf{R} \downarrow & & & & & & \uparrow \mathbf{RS}\mu^{\mathbf{T}} \\
 \mathbf{STR} & & (\pi\text{-YB}) & & \mathbf{RSTT} & & \\
 \mathbf{S}\sigma \downarrow & & & & \uparrow \mathbf{R}\rho\mathbf{T} & & \\
 \mathbf{SRT} & \xrightarrow{\tau\mathbf{T}} & \mathbf{RST} & \xrightarrow{\eta^{\mathbf{T}}\mathbf{RST}} & \mathbf{TRST} & \xrightarrow{\sigma\mathbf{ST}} & \mathbf{RTST}
 \end{array}$$

We also retrieve that $\lambda^{\mathbf{V}/\mathbf{V}}$ is a monotone weak distributive law:

► **Proposition 20.** Consider a monotone weak distributive law $\lambda^{\mathbf{S}/\mathbf{P}}: \mathbf{SP} \Rightarrow \mathbf{PS}$ in \mathbf{Set} that has a weak π -lifting to $\mathbf{EM}(\mathbf{T})$. If the components of $\kappa_{\mathbf{P}}: \mathbf{PU}^{\mathbf{T}} \Rightarrow \mathbf{PU}^{\mathbf{T}}$ are monotone functions (they preserve inclusion of subsets) then $\bar{\lambda}^{\mathbf{S}/\mathbf{P}}: \bar{\mathbf{SP}} \Rightarrow \bar{\mathbf{PS}}$ is also a monotone weak distributive law.

Of course the point of Theorems 19 and 20 is that they make it easier to exhibit weak distributive laws in categories of algebras. Still, working with Diagram (π -YB) may be quite tedious, as it involves up to four composed layers of functors. In fact in Theorem 18 we did not use this π -Yang-Baxter equation at all, instead we directly proved that $\lambda^{\mathbf{V}/\mathbf{V}}$ was a weak π -lifting because we were already able to take for granted that its components were morphisms of $\mathbf{\beta}$ -algebras, i.e. continuous functions. Another problem with Theorem 19 is

10:12 Monotone Weak Distributive Laws in Categories of Algebras

that even if we manage to disprove its prerequisites for some examples, we only get that there is no weak π -lifting of the weak distributive law, but we do not learn anything about other possible meaningful weak distributive laws in the category of algebras.

For all of these reasons we do not try to apply Theorem 19 to weakly π -lift $\lambda^{\mathbf{P}/\mathbf{P}}: \mathbf{PP} \Rightarrow \mathbf{PP}$ to $\mathbf{EM}(\mathbf{P})$ and $\mathbf{EM}(\mathbf{D})$, and immediately turn towards another approach in Section 4 instead: we try to weakly lift the conditions for the existence of monotone weak distributive laws (described in Section 2.2). This is a reasonable strategy because monotone laws are easier to reason about (all non-trivial weak distributive laws described in the literature are monotone) and are closer to being fully characterized, meaning we should hopefully be able to prove no-go theorems for monotone weak distributive laws. In fact we will prove that there is no such law $\mathbf{PP} \Rightarrow \mathbf{PP}$ in $\mathbf{EM}(\mathbf{P})$ or $\mathbf{EM}(\mathbf{D})$, so that by Theorem 20 $\lambda^{\mathbf{P}/\mathbf{P}}$ cannot weakly π -lift to $\mathbf{EM}(\mathbf{P})$ nor to $\mathbf{EM}(\mathbf{D})$.

4 Weakly Lifting Monotone Weak Distributive Laws

Let \mathbf{T} be a monad on a regular category \mathbf{C} . It is folklore that, under mild conditions, $\mathbf{EM}(\mathbf{T})$ is regular as well. For instance on **Set**, all finitary monads, and even all monads if the axiom of choice is assumed to be true, have regular categories of algebras [9, Theorems 3.5.4 and 4.3.5]. Here we will assume that \mathbf{T} is a nearly cartesian functor, but the following result also holds for monads that preserve reflexive coequalizers.

► **Theorem 21** (categories of algebras are regular). *Let $(\mathbf{T}, \eta^{\mathbf{T}}, \mu^{\mathbf{T}})$ be a monad on a regular category \mathbf{C} such that \mathbf{T} is nearly cartesian. Then $\mathbf{EM}(\mathbf{T})$ is regular and $\mathbf{U}^{\mathbf{T}}$ creates finite limits and near pullbacks (a square is a near pullback in $\mathbf{EM}(\mathbf{T})$ if and only if its image by $\mathbf{U}^{\mathbf{T}}$ is so in \mathbf{C}).*

Consider weak distributive laws $\rho: \mathbf{TS} \Rightarrow \mathbf{ST}$, $\sigma: \mathbf{TR} \Rightarrow \mathbf{RT}$ and $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ on \mathbf{C} . Note that because \mathbf{C} is a regular category, idempotents split in \mathbf{C} by way of the factorization into regular epimorphisms followed by monomorphisms: in particular, all the weak composites and weak liftings corresponding to these weak distributive laws exist. When $\tau: \mathbf{SR} \Rightarrow \mathbf{RS}$ is a monotone weak distributive law thanks to the framework of [19], it is now natural to ask when there is also a monotone weak distributive law $\bar{\mathbf{S}}\bar{\mathbf{R}} \Rightarrow \bar{\mathbf{R}}\bar{\mathbf{S}}$ in $\mathbf{EM}(\mathbf{T})$ arising in the same way: it is for instance the case for $\mathbf{T} = \mathbf{\beta}$ and $\mathbf{S} = \mathbf{R} = \mathbf{P}$.

To apply the framework for monotone weak distributive laws of [19] to monads $\bar{\mathbf{S}}$ and $\bar{\mathbf{R}}$, we need to characterize $\mathbf{KI}(\bar{\mathbf{R}})$ as a subcategory of relations – we do this in Section 4.1 – and then prove that $\bar{\mathbf{S}}$ and $\mu^{\bar{\mathbf{S}}}$ are nearly cartesian and investigate when the extension of $\bar{\mathbf{S}}$ to $\mathbf{Rel}(\mathbf{EM}(\mathbf{T}))$ restricts to $\mathbf{KI}(\bar{\mathbf{R}})$ – we do this in Section 4.2. We finally apply our results in Section 4.3.

Because we strive to be as general as possible, in the following the assumptions that we use vary from result to result. In the propositions and theorems we thus recall every time all the assumptions that are necessary.

4.1 Kleisli Categories of Weakly Lifted Monads

Let us forget about regular categories and internal relations for an instant and first describe the Kleisli categories of a weakly lifted monad $\bar{\mathbf{R}}$ in terms of the Kleisli category of \mathbf{R} itself.

► **Proposition 22.** *Let $\sigma: \mathbf{TR} \Rightarrow \mathbf{RT}$ be a weak distributive law in a category \mathbf{C} where idempotents split, so that \mathbf{R} has a weak lifting $\bar{\mathbf{R}}$ to $\mathbf{EM}(\mathbf{T})$ and \mathbf{T} a weak extension $\underline{\mathbf{T}}$ to $\mathbf{KI}(\mathbf{R})$. Then $\mathbf{KI}(\bar{\mathbf{R}})$ -arrows $(A, a) \dashrightarrow (B, b)$ are in one-to-one correspondence with $\mathbf{KI}(\mathbf{R})$ -arrows $f: A \dashrightarrow B$ such that $f \circ \mathbf{F}_{\mathbf{R}}a = \mathbf{F}_{\mathbf{R}}b \circ \underline{\mathbf{T}}f$.*

Assume now that the framework of [19] applies: $\mathbf{KI}(\mathbf{R})$ is a wide subcategory of $\mathbf{Rel}(\mathbf{C})$ (and the left adjoint coincides with the graph functor $\mathbf{Graph}_{\mathbf{C}}: \mathbf{C} \rightarrow \mathbf{Rel}(\mathbf{C})$), \mathbb{T} and $\mu^{\mathbb{T}}$ are nearly cartesian and the weak extension of $(\mathbb{T}, \eta^{\mathbb{T}}, \mu^{\mathbb{T}})$ to $\mathbf{KI}(\mathbf{R})$ is the restriction of the relational extension of \mathbb{T} and $\mu^{\mathbb{T}}$ to $\mathbf{Rel}(\mathbf{C})$.

By Theorems 7 and 21, $\mathbf{U}^{\mathbb{T}}$ has a relational extension $\mathbf{Rel}(\mathbf{U}^{\mathbb{T}}): \mathbf{Rel}(\mathbf{EM}(\mathbb{T})) \rightarrow \mathbf{Rel}(\mathbf{C})$, and by Theorem 22 $\mathbf{KI}(\overline{\mathbf{R}})$ -morphisms $(A, a) \rightarrow (B, b)$ correspond to \mathbf{C} -relations $\psi: A \rightsquigarrow B$ that are in $\mathbf{KI}(\mathbf{R})$ and such that $\psi \cdot a = b \cdot \mathbb{T}\psi$. $\mathbf{KI}(\overline{\mathbf{R}})$ is thus itself a wide subcategory of $\mathbf{Rel}(\mathbf{EM}(\mathbb{T}))$:

► **Lemma 23.** *When the endofunctor \mathbb{T} is nearly cartesian, a \mathbf{C} -relation $\psi: A \rightsquigarrow B$ is the image of an $\mathbf{EM}(\mathbb{T})$ -relation $(A, a) \rightsquigarrow (B, b)$ by the faithful functor $\mathbf{Rel}(\mathbf{U}^{\mathbb{T}}): \mathbf{Rel}(\mathbf{EM}(\mathbb{T})) \rightarrow \mathbf{Rel}(\mathbf{C})$ if and only if $\psi \cdot a \geq b \cdot \mathbb{T}\psi$.*

We now describe in more concrete terms which $\mathbf{EM}(\mathbb{T})$ -relations are arrows in $\mathbf{KI}(\overline{\mathbf{P}})$. Decomposable \mathbb{T} -algebra morphisms play a central role in this description:

► **Definition 24** (decomposable morphisms of algebra). *Let \mathbb{T} be a monad on a category \mathbf{C} such that $\mathbf{EM}(\mathbb{T})$ is regular. A \mathbb{T} -algebra morphism $f: X \rightarrow Y$ is called decomposable when the square $f \circ \varepsilon_X^{\mathbb{T}} = \varepsilon_Y^{\mathbb{T}} \circ \mathbf{F}^{\mathbb{T}} \mathbf{U}^{\mathbb{T}} f$ is a near pullback.*

Given a jointly monic span $\langle \psi_X, \psi_Y \rangle$ in $\mathbf{EM}(\mathbb{T})$, the corresponding relation $\psi = \psi_Y \cdot \psi_X^{\dagger}$ is called decomposable when ψ_X is so.

Before looking at examples, let us give two lemmas that will make working with decomposability of morphisms and relations easier.

► **Lemma 25.** *If $\mathbf{U}^{\mathbb{T}}$ creates near pullbacks, a \mathbb{T} -morphism $f: (A, a) \rightarrow (B, b)$ is decomposable if and only if $\mathbf{U}^{\mathbb{T}} f \circ a = b \circ \mathbf{TU}^{\mathbb{T}} f$ is a near pullback in \mathbf{C} . In particular if \mathbb{T} is a monad on \mathbf{Set} , this holds if and only if for every $x \in A$ and $u \in \mathbf{T}B$ such that $f(x) = b(u)$, there is some $t \in \mathbf{T}A$ such that $(\mathbf{T}f)(t) = u$ and $a(t) = x$.*

► **Lemma 26.** *Suppose $\mathbf{U}^{\mathbb{T}}$ creates and \mathbb{T} preserves regular epimorphisms. If $g = h \circ e$ is decomposable and e is a regular epimorphism, then h is decomposable as well. In particular, if $f: R \rightarrow X$ is decomposable then for any $g: R \rightarrow Y$, $g \cdot f^{\dagger}$ is a decomposable relation.*

Decomposable morphisms have been studied in [11, Definition 3.1.1] in the setting of *monoidal topology*. There, these morphisms are called *open* as they generalize open maps between compact Hausdorff spaces, as the next example shows. We prefer the term “decomposable” here as we focus on other examples that feel more algebraic than topological:

► **Example 27** (decomposable morphisms and relations in categories of algebras over \mathbf{Set}). In $\mathbf{EM}(\beta) \cong \mathbf{KHaus}$, a continuous map is decomposable if and only if it is open (it preserves open sets), and decomposable relations are the continuous ones, i.e. those relations $\psi: X \rightsquigarrow Y$ such that $\psi^{-1}[u]$ is open in X for every open subset u of Y .

In $\mathbf{EM}(\mathbf{P}) \cong \mathbf{JSL}$, $\psi: X \rightsquigarrow Y$ is decomposable if and only if for every family $(x_i)_{i \in I}$ of elements of X and every $y \in Y$ such that $(\bigvee_{i \in I} x_i, y) \in \psi$, there is a family $(y_i)_{i \in I}$ of elements of Y such that $(x_i, y_i) \in \psi$ for all $i \in I$ and $\bigvee_{i \in I} y_i = y$.

In $\mathbf{EM}(\mathbf{D}) \cong \mathbf{Conv}$, $\psi: X \rightsquigarrow Y$ is decomposable if and only if for every $x \in X$, every *disintegration* of x as a barycenter $x = \sum_{i=1}^n \lambda_i x_i$ and every $y \in Y$ such that $(x, y) \in \psi$, y disintegrates as a barycenter $y = \sum_{i=1}^n \lambda_i y_i$ such that $(x_i, y_i) \in \psi$ for all $i \in I$.

A more general example of decomposable morphism is the following:

► **Lemma 28.** *When \mathbf{U}^T creates near pullbacks, μ^T is nearly cartesian if and only if every free algebra morphism $\mathbf{F}^T f: (\mathbf{T}X, \mu_X^T) \rightarrow (\mathbf{T}Y, \mu_Y^T)$ with $f: X \rightarrow Y$ is decomposable.*

We are now able to state the main result of this section. We first state it in full generality (Theorem 29), but in practice we will be especially concerned with the case $\mathbf{R} = \mathbf{P}$ on $\mathbf{C} = \mathbf{Set}$ (Theorem 30).

► **Theorem 29.** *Let $(\mathbf{T}, \eta^T, \mu^T)$ be a monad on a regular category \mathbf{C} . When the endofunctor \mathbf{T} is nearly cartesian, an $\mathbf{EM}(\mathbf{T})$ -relation $\psi: (A, a) \rightsquigarrow (B, b)$ is decomposable if and only if $\mathbf{U}^T \psi \cdot a = b \cdot \mathbf{T}\mathbf{U}^T \psi$. If μ^T is also nearly cartesian and $\mathbf{Rel}(\mathbf{T})$ restricts to $\mathbf{KI}(\mathbf{R}) \hookrightarrow \mathbf{Rel}(\mathbf{C})$ for some monad \mathbf{R} on \mathbf{C} , then the Kleisli category $\mathbf{KI}(\bar{\mathbf{R}})$ of the corresponding weakly lifted monad $\bar{\mathbf{R}}$ on $\mathbf{EM}(\mathbf{T})$ has for arrows $(A, a) \twoheadrightarrow (B, b)$ the decomposable relations $\psi: (A, a) \rightsquigarrow (B, b)$ in $\mathbf{EM}(\mathbf{T})$ such that $\mathbf{U}^T \psi$ is in $\mathbf{KI}(\mathbf{R})$.*

► **Corollary 30.** *If \mathbf{T} has a monotone weak distributive law over \mathbf{P} in \mathbf{Set} , the Kleisli category of the lifted powerset monad \mathbf{P} on $\mathbf{EM}(\mathbf{T})$ is the category of \mathbf{T} -algebras and decomposable relations between them.*

► **Remark 31** (subobject classifiers in categories of algebras). Recall that an elementary topos is a regular category such that the **Graph** functor has a right adjoint [13, §1.911]; the corresponding monad is called the powerset monad. If \mathbf{C} is an elementary topos with powerset monad \mathbf{P} , and if \mathbf{T} is a monad on \mathbf{C} such that the endofunctor \mathbf{T} and the natural transformation μ^T are nearly cartesian, then by Theorem 29 $\mathbf{EM}(\mathbf{T})$ is an elementary topos as soon as every \mathbf{T} -algebra morphism is decomposable. We retrieve for instance that the categories of group actions (algebras for monads $G \times -$ where G is a group) are toposes, because the corresponding morphisms of algebras are easily shown to all be decomposable.

This is not a necessary condition for a category of algebras to be an elementary topos: it is well known that categories of monoid actions (algebras for monads $M \times -$ where M is a monoid) are toposes, but there are equivariant morphisms that are not decomposable.

If $\mathbf{EM}(\mathbf{T})$ is an elementary topos as in Theorem 31, $\bar{\mathbf{P}}1$ classifies subobjects in the sense that subobjects $X \hookrightarrow Y$ are in one-to-one correspondence with morphisms $Y \rightarrow \bar{\mathbf{P}}1$, where 1 is the terminal object [13, §1.912].

This can be generalized when Theorem 29 holds as follows: $\bar{\mathbf{R}}1$ classifies decomposition-closed subobjects, in the sense that decomposable monomorphisms $X \hookrightarrow Y$ are in one-to-one correspondence with morphisms $Y \rightarrow \bar{\mathbf{R}}1$ (the correspondence comes from the adjunction $\mathbf{KI}(\bar{\mathbf{R}})(Y, 1) \cong \mathbf{EM}(\mathbf{T})(Y, \bar{\mathbf{R}}1)$). For instance, the Vietoris monad on $\mathbf{KHaus} \cong \mathbf{EM}(\beta)$ classifies clopen subsets of compact Hausdorff spaces, the non-empty-join-closed powerset monad on $\mathbf{JSL} \cong \mathbf{EM}(\mathbf{P})$ classifies downwards-closed subsets of join-semilattices, and the convex-closed powerset monad on $\mathbf{Conv} \cong \mathbf{EM}(\mathbf{D})$ classifies *walls*, i.e. subsets E such that if $x \in E$ and $\sum_{i=1}^n x_i = x$, $x_i \in E$ as well for all $1 \leq i \leq n$ (walls appear for instance in the structure theorem for convex algebras, which state that every convex algebra is a subalgebra of the *Plonka sum* of its walls [27, Theorem 4.5]).

4.2 Monotone Extensions to Kleisli Categories of Weakly Lifted Monads

In Section 4.1 we assumed \mathbf{T} had a monotone weak distributive law over \mathbf{R} coming from a relational extension of \mathbf{T} and μ^T , and described the Kleisli category of the corresponding weakly lifted monad $\bar{\mathbf{R}}$. Suppose now there is another monad \mathbf{S} that weakly lifts to $\mathbf{EM}(\mathbf{T})$ and that also has a monotone weak distributive law over \mathbf{R} coming from a relational extension of \mathbf{S} and μ^S . When do we also get a monotone weak distributive law of $\bar{\mathbf{S}}$ over $\bar{\mathbf{R}}$ coming from a relational extension of $\bar{\mathbf{S}}$ and $\mu^{\bar{\mathbf{S}}}$?

For the relational extension to exist, we need \bar{S} and $\mu^{\bar{S}}$ to be nearly cartesian. This always holds:

► **Lemma 32.** *Let C be a regular category where idempotents split, and suppose T is nearly cartesian. If $F: C \rightarrow C$ is nearly cartesian and weakly lifts to $\mathbf{EM}(T)$, then its weak lifting is nearly cartesian. If $\alpha: F \Rightarrow G$ between two such functors is nearly cartesian and weakly lifts to $\mathbf{EM}(T)$, then its weak lifting is nearly cartesian as well.*

\bar{S} thus has a relational extension $\mathbf{Rel}(\bar{S})$. By adapting Theorem 7, we can not only characterize when relational extensions restrict to $\mathbf{Kl}(\bar{R})$, but when any endofunctor or natural transformation has a monotone extension to $\mathbf{Kl}(\bar{R})$. We can even state this characterization more generally, without necessarily speaking of decomposable morphisms: we do this now.

► **Definition 33.** *Let Γ be a wide subcategory of a regular category C such that, in C ,*

- Γ -arrows are stable under pullbacks (in C);
- if $f \circ e$ is a Γ -arrow and e is a regular epimorphism (in C), f is a Γ -arrow.

Then we define $C \cdot \Gamma^\dagger$ to be the wide subcategory of $\mathbf{Rel}(C)$ whose arrows are the C -relations $\psi: X \rightsquigarrow Y$ given by jointly monic spans $\langle \psi_X, \psi_Y \rangle$ such that ψ_X is a Γ -arrow, and we write $\mathbf{Graph}_\Gamma: C \rightarrow C \cdot \Gamma^\dagger$ for the restriction of $\mathbf{Graph}_C: C \rightarrow \mathbf{Rel}(C)$ to $C \cdot \Gamma^\dagger$.

We also define a $\Gamma^\dagger \cdot C$ -square to be a square $a \circ b = c \circ d$ such that a or c is a Γ -arrow.

► **Definition 34.** *Let C and D be two regular categories with respective wide subcategories Γ and Δ as in Theorem 33. Let F be a functor $C \rightarrow D$. A (Γ, Δ) -relational extension of F is a functor $\underline{F}_{\Gamma, \Delta}: C \cdot \Gamma^\dagger \rightarrow D \cdot \Delta^\dagger$ such that $\underline{F}_{\Gamma, \Delta} \mathbf{Graph}_\Gamma = \mathbf{Graph}_\Delta \underline{F}_{\Gamma, \Delta}$. If $\alpha: F \Rightarrow G$ is a natural transformation between functors $C \rightarrow D$ with (Γ, Δ) -relational extensions $\underline{F}_{\Gamma, \Delta}$ and $\underline{G}_{\Gamma, \Delta}$, a (Γ, Δ) -relational extension is a (necessarily unique) natural transformation $\underline{\alpha}_{\Gamma, \Delta}: \underline{F}_{\Gamma, \Delta} \Rightarrow \underline{G}_{\Gamma, \Delta}$ such that $\underline{\alpha}_{\Gamma, \Delta} \mathbf{Graph}_\Gamma = \mathbf{Graph}_\Delta \underline{\alpha}_{\Gamma, \Delta}$.*

► **Theorem 35.** *Let C, D, Γ, Δ and $F: C \rightarrow D$ be as in Theorem 34. F has a monotone (Γ, Δ) -relational extension if and only if the following two conditions hold:*

- F restricts to a functor $\Gamma \rightarrow \Delta$;
- F sends near pullback $\Gamma^\dagger \cdot C$ -squares on near pullback (necessarily $\Delta^\dagger \cdot D$ -) squares (this is always true when F is nearly cartesian).

Such a monotone (Γ, Δ) -relational extension, if it exists, is necessarily unique and given by $\underline{F}_{\Gamma, \Delta}(g \cdot f)^\dagger = Fg \cdot (Ff)^\dagger$.

Let $\alpha: F \Rightarrow G$ be a natural transformation between two functors $C \rightarrow D$ having such monotone (Γ, Δ) -relational extensions. α has a (necessarily unique) (Γ, Δ) -relational extension if and only if it has near pullbacks for its naturality squares along Γ -morphisms (this is always true when α is nearly cartesian).

A first corollary, while not especially ground-breaking, is the following:

► **Corollary 36.** *In \mathbf{Set} , a monad (T, η^T, μ^T) whose endofunctor and multiplication are nearly cartesian also has a (necessarily unique) monotone weak distributive law over the monad \mathbf{P}_* of non-empty subsets, and has one over the monad \mathbf{P}_f of finite subsets if and only if T preserves functions with finite pre-images of elements.*

We could more generally characterize the existence of monotone weak distributive laws over these powerset monads for any monad on \mathbf{Set} . Still, Theorem 36 is already enough to prove that \mathbf{P} has monotone weak distributive laws over \mathbf{P}_f and \mathbf{P}_* , that \mathbf{D} has a monotone weak distributive law over \mathbf{P}_* but not over \mathbf{P}_f , and that \mathbf{b} has a monotone weak distributive law over \mathbf{P}_* . A more impactful corollary – we will apply it repeatedly in Section 4.3 – is the following:

► **Corollary 37.** *Let T be a monad on \mathbf{Set} equipped with a monotone weak distributive law $TP \Rightarrow PT$, and let S be a monad on $\mathbf{EM}(T)$. If there is a monotone weak distributive law $S\bar{P} \Rightarrow \bar{P}S$, S preserves decomposable T -algebra morphisms. Moreover if S is itself the weak lifting of a monad on \mathbf{Set} that has a monotone weak distributive law over \mathbf{P} , then the previous condition is not only necessary, but also sufficient.*

4.3 Monotone Weak Distributive Laws in Categories of Algebras

A first use of Theorem 37 is retrieving the monotone weak distributive law $\mathbf{V}\mathbf{V} \Rightarrow \mathbf{V}\mathbf{V}$: once \mathbf{V} and $\mu^{\mathbf{V}}$ are shown to be nearly cartesian, we only need to prove that decomposable morphisms are the open maps (Theorem 27) instead of proving that $\mathbf{KI}(\mathbf{V})$ -arrows are the continuous relations, and that \mathbf{V} preserves open maps, which is technically much simpler than proving that $\underline{\mathbf{V}}$ preserves continuous relations, as originally done in [19, Proposition 20]. We are also able to answer quite easily Goy's [16, Conjecture 7.31] on the weak distributivity of the Radon monad – the monad of Radon probability measures on a compact Hausdorff space – over the Vietoris monad:

► **Theorem 38.** *The Radon monad \mathbf{R} does not have a monotone weak distributive law over the Vietoris monad \mathbf{V} , but it has (a unique) one over the non-empty Vietoris monad \mathbf{V}_* .*

Let us stress the importance of this new weak distributive law: the question of how to combine probability and non-determinism has been the topic of numerous works (again, see the introduction of [21]), and this law provides an answer in \mathbf{KHaus} that is derived from a generic construction and thus comes with generic tools, e.g. generalized determinization and up-to techniques [18, 16]. In a very recent pre-print [15], Goubault-Larrecq also constructs this law $\mathbf{R}\mathbf{V}_* \Rightarrow \mathbf{V}_*\mathbf{R}$ as an instance of weak distributive laws between monads of continuous valuations and non-deterministic choice in more general categories of topological spaces: our result is more restricted, but we derive the law from generic categorical principles instead of building it by hand, exhibit its canonicity (it comes from a relational extension), and show why the non-empty version of the Vietoris monad is needed.

A second use of Theorem 37 is in proving the absence of monotone weak distributive laws. In Section 3 we were able to prove that the law $\mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$ does not weakly lift to $\mathbf{EM}(\mathbf{P})$ nor $\mathbf{EM}(\mathbf{D})$, but we were not able to say anything about the existence of other monotone weak distributive laws $\bar{\mathbf{P}}\mathbf{P} \Rightarrow \mathbf{P}\bar{\mathbf{P}}$. Now, thanks to the framework developed above and in particular Theorem 37, we are able to prove that such laws cannot exist. We start with $\mathbf{EM}(\mathbf{P})$:

► **Example 39.** In $\mathbf{EM}(\mathbf{P}) \cong \mathbf{JSL}$, let $\bar{\mathbf{P}}$ be the monad of subsets closed under non-empty joins: the join of a family $(E_i)_{i \in I}$ of non-empty-joins-closed subsets of X is the non-empty-joins-closed subset $\{\bigvee_{i \in I} x_i \mid x_i \in E_i\}$. Let $f: 4 \rightarrow 2$ (where $2 = \{0, 1\}$ and $4 = \{0, 1, 2, 3\}$) be the function given by $f(0) = f(2) = 0$ and $f(1) = f(3) = 1$. Then $\mathbf{F}^{\mathbf{P}}f$ is decomposable (by Theorem 28), but $\bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}f$ is not. Indeed, let $A \in \bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}4$ and $B, B_1, B_2 \in \mathbf{P}\mathbf{F}^{\mathbf{P}}2$ be as depicted in Figure 2a (page 17): $(\bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}f)(A) = B = B_1 \vee B_2$ but there are no $A_1, A_2 \in \bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}4$ such that $A = A_1 \vee A_2$ and $(\bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}f)(A_1) = B_1$ as well as $(\bar{\mathbf{P}}\mathbf{F}^{\mathbf{P}}f)(A_2) = B_2$.

Proof. Suppose indeed there are such A_1 and A_2 . Then $A_1, A_2 \subseteq A$ and thus $\{0, 1, 2, 3\} \in A_1 \cup A_2$ ($\{0, 1, 2, 3\}$ is join-irreducible in A). This would imply $\{0, 1\} \in B_1 \cup B_2$, which does not hold. ◀

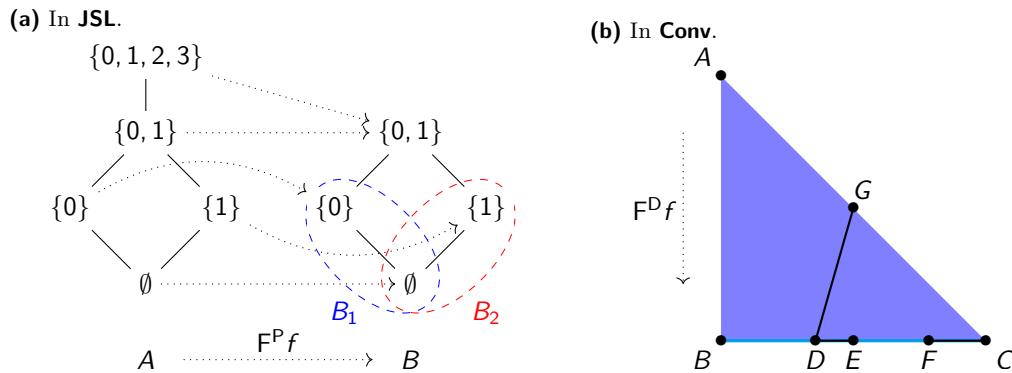
$\overline{\mathbf{P}}$ does not preserve decomposable \mathbf{P} -algebra morphisms, and thus there is no monotone weak distributive law $\overline{\mathbf{P}\mathbf{P}} \Rightarrow \overline{\mathbf{P}\mathbf{P}}$. In fact because the counter-example decomposable morphism is surjective and has finite pre-images, this also proves that there are no monotone weak distributive laws $\mathbf{P}\mathbf{P}_* \Rightarrow \mathbf{P}_*\mathbf{P}$ or $\mathbf{P}\mathbf{P}_f \Rightarrow \mathbf{P}_f\mathbf{P}$ in $\mathbf{EM}(\mathbf{P})$.

Because there is a morphism of monads $\mathbf{D} \Rightarrow \mathbf{P}$ (that sends a probability distribution to its support), there is a functor $\mathbf{EM}(\mathbf{P}) \rightarrow \mathbf{EM}(\mathbf{D})$ which allows us to transfer Theorem 39 to $\mathbf{EM}(\mathbf{D})$: there are no monotone weak distributive laws $\mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$ or $\mathbf{P}\mathbf{P}_* \Rightarrow \mathbf{P}_*\mathbf{P}$ in $\mathbf{EM}(\mathbf{D})$. But this argument is not entirely satisfying, as the resulting example is that of a morphism of convex algebras with a very unnatural structure, namely that of complete join-semilattices: one could imagine restricting to a full subcategory of $\mathbf{EM}(\mathbf{D})$ that does not contain these semilattices, and perhaps $\overline{\mathbf{P}}$ would preserve decomposable morphisms there. As shown by the following example, due to Harald Woracek and Ana Sokolova (private communication), this cannot be the case as soon as free convex algebras come in the picture.

► **Example 40** ([30]). In $\mathbf{EM}(\mathbf{D}) \cong \mathbf{Conv}$, let $\overline{\mathbf{P}}$ be the monad of convex subsets: a convex combination of some convex subsets of X is the convex set of the corresponding convex combinations of their points (in X). Let $f: \{A, B, C\} \rightarrow \{B, C\}$ be the function given by $f(A) = f(B) = B$ and $f(C) = C$. Then $\mathbf{F}^{\mathbf{D}}f$ is decomposable (by Theorem 28) but $\overline{\mathbf{P}}\mathbf{F}^{\mathbf{D}}f$ is not. Indeed, depicting $\mathbf{F}^{\mathbf{D}}\{A, B, C\}$ as the triangle depicted in Figure 2b (page 17), $\mathbf{F}^{\mathbf{D}}\{B, C\}$ is the line segment $[BC]$ and $\mathbf{F}^{\mathbf{D}}f$ is the vertical projection. Now $\frac{1}{2}\{B\} + \frac{1}{2}\{FC\} = [DE] = (\overline{\mathbf{P}}\mathbf{F}^{\mathbf{D}}f)([GD])$, but $[GD]$ itself cannot be disintegrated as the mean of two convex subsets of ABC , one above B and the other above $[FC]$.

Proof. If such a disintegration existed, then the convex subset above B would contain both B (because $D \in [DG]$) and A (because $G \in [DG]$), hence would be $[AB]$. The subset above $[FC]$ would contain at least one point, and hence the mean of these two subsets would have to contain a non-trivial vertical line segment, which is not the case of $[DG]$. ◀

Using similar arguments, we are able to prove the existence or absence of monotone weak distributive laws over lifted powerset monads in several categories of algebras: these results are gathered in Table 1. All the negative results in this table come from the non-preservation of decomposable morphisms, and thus the absence of monotone extensions of the endofunctors themselves. The topmost row indicates in which category we work. A monad in the second topmost row has a monotone weak distributive law over a monad in the left column if the corresponding cell is filled with \checkmark , otherwise it is filled with \times . In **Set**, $\overline{\mathbf{P}}$ and $\overline{\mathbf{P}}_*$ are the



■ **Figure 2** Counterexamples to preservation of decomposability.

■ **Table 1** Existence or absence of monotone weak distributed laws over weakly lifted powerset monads in categories of algebras.

	Set						KHaus		JSL	Conv	Mon				CMon			
	L	M	D	P	β	M_S	V	R	\bar{P}	\bar{P}	\bar{M}	\bar{D}	\bar{P}	\bar{M}_S	\bar{M}	\bar{D}	\bar{P}	
\bar{P}	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
\bar{P}_*	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

usual powerset monads \mathbf{P} and \mathbf{P}_* . \mathbf{L} is the monad of lists, \mathbf{M} that of multisets and \mathbf{M}_S that of modules for a semiring S satisfying the conditions of [7, Theorem 3.1]. $\mathbf{Mon} \cong \mathbf{EM}(\mathbf{L})$ is the category of monoids and $\mathbf{CMon} \cong \mathbf{EM}(\mathbf{M})$ that of commutative monoids. Linear theories distribute over commutative monads [25], hence \mathbf{L} and \mathbf{M} distribute over \mathbf{M} , \mathbf{P} , \mathbf{D} and \mathbf{M}_S when S is commutative, and so these four monads have liftings (in particular weak liftings) to \mathbf{Mon} and \mathbf{CMon} .

5 Conclusion

Noticing the similarity between the laws $\mathbf{V}\mathbf{V} \Rightarrow \mathbf{V}\mathbf{V}$ and $\mathbf{P}\mathbf{P} \Rightarrow \mathbf{P}\mathbf{P}$, we developed the theory for weakly lifting weak distributive laws and showed that it only applied partially in this case. We then focused on the monotonicity of the laws, and gave full characterizations of monotone weak distributive laws over weakly lifted powerset monads in categories of algebras by characterizing the Kleisli categories of the latter, a key notion appearing then being that of decomposable morphisms. We finally applied this result to exhibit a new law $\mathbf{R}\mathbf{V}_* \Rightarrow \mathbf{V}_*\mathbf{R}$ for combining probability and non-determinism in \mathbf{KHaus} , but also to show that in general these monotone weak distributive laws seem to be quite rare.

We leave for further work the development of a full 2-categorical theory for iterating weak distributive laws (in the vein of [10, 4]), which would complete Section 3 but would likely be scarce in new examples. With monotone laws over powerset-like monads fully characterized, another natural question is now whether this can be done in other settings, e.g. in \mathbf{Pos} -regular categories [23] or over other monads: the multiset monad for instance is a good candidate as its Kleisli category can also be described through spans. Finally, the author believes it would be interesting to transpose the results of Section 4 in the setting of monoidal topology [20], where categories of algebras for nearly cartesian monads generalize the category of compact Hausdorff spaces in a formal sense: perhaps for instance the weakly lifted powerset monads we study are a generalization of the Vietoris monads of topological spaces.

References

- 1 Michael Barr. Relational algebras. In S. MacLane, H. Applegate, M. Barr, B. Day, E. Dubuc, Phreilambud, A. Pultr, R. Street, M. Tierney, and S. Swierczkowski, editors, *Reports of the Midwest Category Seminar IV*, pages 39–55, Berlin, Heidelberg, 1970. Springer Berlin Heidelberg.
- 2 Jon Beck. Distributive laws. In H. Applegate, M. Barr, J. Beck, F. W. Lawvere, F. E. J. Linton, E. Manes, M. Tierney, F. Ulmer, and B. Eckmann, editors, *Seminar on Triples and Categorical Homology Theory*, Lecture Notes in Mathematics, pages 119–140, Berlin, Heidelberg, 1969. Springer. doi:10.1007/BFb0083084.

- 3 Gabriella Böhm. The weak theory of monads. *Advances in Mathematics*, 225(1):1–32, September 2010. doi:10.1016/j.aim.2010.02.015.
- 4 Gabriella Böhm. On the iteration of weak wreath products. *Theory and Applications of Categories*, 26(2):30–59, 2012. URL: <http://www.tac.mta.ca/tac/volumes/26/2/26-02abs.html>.
- 5 Gabriella Böhm, Stephen Lack, and Ross Street. On the 2-Categories of Weak Distributive Laws. *Communications in Algebra*, 39(12):4567–4583, December 2011. doi:10.1080/00927872.2011.616436.
- 6 Gabriella Böhm, Stephen Lack, and Ross Street. Idempotent splittings, colimit completion, and weak aspects of the theory of monads. *Journal of Pure and Applied Algebra*, 216(2):385–403, February 2012. doi:10.1016/j.jpaa.2011.07.003.
- 7 Filippo Bonchi and Alessio Santamaria. Convexity via Weak Distributive Laws. *Logical Methods in Computer Science*, Volume 18, Issue 4, November 2022. doi:10.46298/lmcs-18(4:8)2022.
- 8 Francis Borceux. *Handbook of Categorical Algebra: Volume 1: Basic Category Theory*, volume 1 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1994. doi:10.1017/CB09780511525858.
- 9 Francis Borceux. *Handbook of Categorical Algebra: Volume 2: Categories and Structures*, volume 2 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1994. doi:10.1017/CB09780511525865.
- 10 Eugenia Cheng. Iterated distributive laws. *Mathematical Proceedings of the Cambridge Philosophical Society*, 150(3):459–487, May 2011. doi:10.1017/S0305004110000599.
- 11 Maria Manuel Clementino, Eva Colebunders, and Walter Tholen. Lax algebras as spaces. In Dirk Hofmann, Gavin J. Seal, and Walter Tholen, editors, *Monoidal Topology: A Categorical Approach to Order, Metric, and Topology*, Encyclopedia of Mathematics and Its Applications, pages 375–466. Cambridge University Press, Cambridge, 2014. doi:10.1017/CB09781107517288.007.
- 12 Fredrik Dahlqvist and Renato Neves. Compositional semantics for new paradigms: Probabilistic, hybrid and beyond, April 2018. doi:10.48550/arXiv.1804.04145.
- 13 Peter John Freyd and Andre Scedrov. *Categories, Allegories*. North Holland, January 1990.
- 14 Richard Garner. The Vietoris Monad and Weak Distributive Laws. *Applied Categorical Structures*, 28(2):339–354, April 2020. doi:10.1007/s10485-019-09582-w.
- 15 Jean Goubault-Larrecq. Weak Distributive Laws between Monads of Continuous Valuations and of Non-Deterministic Choice, August 2024. doi:10.48550/arXiv.2408.15977.
- 16 Alexandre Goy. *On the Compositionality of Monads via Weak Distributive Laws*. PhD thesis, Université Paris-Saclay, October 2021. URL: <https://theses.hal.science/te1-03426949>.
- 17 Alexandre Goy. Weakening and Iterating Laws using String Diagrams. *Electronic Notes in Theoretical Informatics and Computer Science*, Volume 1 - Proceedings of MFPS XXXVIII, February 2023. doi:10.46298/entics.10482.
- 18 Alexandre Goy and Daniela Petrişan. Combining probabilistic and non-deterministic choice via weak distributive laws. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '20*, pages 454–464, New York, NY, USA, July 2020. Association for Computing Machinery. doi:10.1145/3373718.3394795.
- 19 Alexandre Goy, Daniela Petrişan, and Marc Aiguier. Powerset-Like Monads Weakly Distribute over Themselves in Toposes and Compact Hausdorff Spaces. In *DROPS-IDN/v2/Document/10.4230/LIPIcs.ICALP.2021.132*. Schloss-Dagstuhl - Leibniz Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.132.
- 20 Dirk Hofmann, Gavin J. Seal, and Walter Tholen, editors. *Monoidal Topology: A Categorical Approach to Order, Metric, and Topology*. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 2014. doi:10.1017/CB09781107517288.
- 21 Klaus Keimel and Gordon D. Plotkin. Mixed powerdomains for probability and nondeterminism. *Logical Methods in Computer Science*, Volume 13, Issue 1, January 2017. doi:10.23638/LMCS-13(1:2)2017.

- 22 Bartek Klin and Julian Salamanca. Iterated Covariant Powerset is not a Monad. *Electronic Notes in Theoretical Computer Science*, 341:261–276, December 2018. doi:10.1016/j.entcs.2018.11.013.
- 23 Alexander Kurz and Jiří Velebil. Quasivarieties and varieties of ordered algebras: Regularity and exactness. *Mathematical Structures in Computer Science*, 27(7):1153–1194, October 2017. doi:10.1017/S096012951500050X.
- 24 Ernest Manes. A triple theoretic construction of compact algebras. In H. Appelgate, M. Barr, J. Beck, F. W. Lawvere, F. E. J. Linton, E. Manes, M. Tierney, F. Ulmer, and B. Eckmann, editors, *Seminar on Triples and Categorical Homology Theory*, pages 91–118, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.
- 25 Ernie Manes and Philip Mulry. Monad compositions. I: General constructions and recursive distributive laws. *Theory and Applications of Categories*, 18:172–208, 2007. URL: <https://eudml.org/doc/128434>.
- 26 E. Moggi. Computational lambda-calculus and monads. In [1989] *Proceedings. Fourth Annual Symposium on Logic in Computer Science*, pages 14–23, June 1989. doi:10.1109/LICS.1989.39155.
- 27 Anna B. Romanowska and Jonathan D. H. Smith. On the Structure of Barycentric Algebras. *Houston Journal of Mathematics*, 16(3):431–448, 1990. URL: <https://www.math.uh.edu/~hjm/restricted/archive/v016n3/0431ROMANOWSKA.pdf>.
- 28 Ross Street. Weak distributive laws. *Theory and Applications of Categories*, 22(12):313–320, 2009. URL: <http://www.tac.mta.ca/tac/volumes/22/12/22-12abs.html>.
- 29 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, February 2006. doi:10.1017/S0960129505005074.
- 30 Harald Woracek and Ana Sokolova. Re: Decomposing convex subsets of simplices along free morphisms, June 2024.
- 31 Maaïke Zwart and Dan Marsden. No-Go Theorems for Distributive Laws. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, June 2019. doi:10.1109/LICS.2019.8785707.

Generalized Inner Product Estimation with Limited Quantum Communication

Srinivasan Arunachalam 

IBM Quantum, Almaden, CA, USA

Louis Schatzki  

Electrical and Computer Engineering, University of Illinois Urbana-Champaign, IL, USA

Abstract

In this work, we consider the fundamental task of distributed inner product estimation when allowed limited communication. Suppose Alice and Bob are given k copies of an unknown n -qubit quantum state $|\psi\rangle, |\phi\rangle$ respectively, are allowed to send q qubits to one another, and the task is to estimate $|\langle\psi|\phi\rangle|^2$ up to constant additive error. We show that $k = \Theta(\sqrt{2^{n-q}})$ copies are essentially necessary and sufficient for this task (extending the work of Anshu, Landau and Liu (STOC'22) who considered the case when $q = 0$). Additionally, we also consider the task when the goal of the players is to estimate $|\langle\psi|M|\phi\rangle|^2$, for arbitrary Hermitian M . For this task we show that certain norms on M determine the sample complexity of estimating $|\langle\psi|M|\phi\rangle|^2$ when using only classical communication.

2012 ACM Subject Classification Theory of computation \rightarrow Quantum information theory

Keywords and phrases Quantum property testing, Quantum Distributed Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.11

Related Version *Full Version:* <https://arxiv.org/pdf/2410.12684> [4]

Funding We acknowledge support from IBM through the Illinois-IBM Discovery Accelerator Institute.

1 Introduction

The construction of quantum networks is a major goal in quantum information science [22, 27, 24, 15]. These networks are envisioned to consist of disjoint quantum processing nodes with quantum communication interlinks, likely implemented through photonics. While the nodes may consist of the same type of qubit, this is not a requirement, and work has gone into designing heterogeneous networks where the nodes have different physical compositions [19]. With this in mind it is of interest to develop distributed protocols in a platform-independent framework.

In this paper we consider a natural *distributed* variant of the inner product estimation problem: suppose there are two spatially separated parties Alice and Bob who are each given copies of an unknown d -dimensional quantum state $|\psi\rangle$ and $|\phi\rangle$ respectively and their goal is to compute $|\langle\psi|M|\phi\rangle|^2$ for some Hermitian operator M . We will be interested in the difficulty of this task depending on how much entanglement Alice and Bob share. The core motivation for our setup is that distributing entanglement between nodes in a network is not a free resource and adds time and difficulty to a protocol. Thus, it is useful to minimize the amount of entanglement required. Similar concerns arise in various papers on distributed quantum computing [28, 18, 1].

We call this the *generalized distributed inner product estimation* problem (GIPE). The setting of $M = \mathbb{I}$ in GIPE (which we refer to as *distributed inner product estimation* DIPE) is one of the important steps in cross-platform verification proposed in [16] where the goal is to test if two quantum computers (say built on different platforms) are preparing the same quantum state; a fundamental problem for near-term devices. A natural constraint in these papers is how the quantum computers communicate with one another and in these works they



© Srinivasan Arunachalam and Louis Schatzki;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 11; pp. 11:1–11:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



consider either only classical communication or limited quantum communication (the latter being the focus of our work). Apart from verification, on a theoretical level, understanding the complexity of a question, as fundamental as inner product estimation, in a distributed manner is a natural question.

The seminal work of Buhrman, Cleve, Watrous and Wolf [9] on *quantum fingerprinting* introduced the so-called *swap test*. This subroutine takes as input one copy of unknown quantum states $|\psi\rangle$ and $|\phi\rangle$ and outputs a bit whose bias equals $|\langle\psi|\phi\rangle|^2$; allowing us to estimate the *inner product* between two unknown quantum states without knowing anything about the states themselves. This subroutine has found applications in several areas in quantum computing from complexity theory, learning theory, entanglement theory, to optimization. Although so widely used, there still remain some open fundamental questions about estimating inner products, which is the topic of this work.

There are two techniques to solve GIPE under different settings: (i) If allowed quantum communication, then Alice can send a few copies of the d -dimensional quantum state over to Bob, then Bob could perform a variant of the swap test to estimate $|\langle\psi|M|\phi\rangle|^2$; (ii) if allowed only classical communication, i.e., Alice and Bob only send classical messages, then a recent work of Anshu et al. [3] showed that $\Theta(\sqrt{d})$ copies of $|\psi\rangle, |\phi\rangle$ are necessary and sufficient in order to estimate $|\langle\psi|\phi\rangle|^2$. This setting is often referred to as *local operations and classical communication* (LOCC) and has received a lot of attention in quantum information theory [13, 7, 17]. Although (ii) is surprising in that, the sample complexity is quadratically better than full-state tomography, the exponential (since we typically work with $d = 2^n$, where n is the number of qubits) nature of the complexity seems rather large in practice. Their work raises two natural questions, which will be the focus here:

1. With the advent of near-term quantum devices, sending the full-quantum state (as in the protocol (i) above) might be hard, but it is plausible that one could send a *few* qubits of quantum message. If this were possible, then what is the complexity of DIPE when allowed q -qubit messages?
2. The work of Anshu et al. [3] showed how to solve GIPE when $M = \mathbb{I}$, but other natural properties of quantum states are captured by letting M be an arbitrary operator, for example, predicting few-body properties, such as two-point correlation functions, entanglement entropy of small subsystems, expectation values of observables, projector onto a subspace or perhaps a Pauli string, then M may be different than \mathbb{I} . In this case, understanding the complexity of GIPE for arbitrary M is relevant.

These questions were explicitly raised by Anshu et al. [3] (and as open question 21 in [2]) and in this work we answer both.

1.1 Results

Our first result considers the setup where Alice and Bob may communicate a few qubits in an interactive protocol but not necessarily enough to teleport copies of their states. For simplicity assume that Alice and Bob each have n -qubit states and can transmit in total q qubits of communication. As we mentioned above, the sample complexity when $q = 0$ was shown to be $\sqrt{2^n}$ by [3] and when $q = n$, the sample complexity is $O(1)$. Is there some saving in sample complexity when allowed q qubits of communication? Our first result shows a smooth interpolation between both these settings.

► **Theorem 1 (Informal).** *Suppose Alice and Bob are given k copies of n -qubit states $|\psi\rangle$ and $|\phi\rangle$ respectively. They can communicate $\Theta(q)$ qubits along with performing LOCC. Then it is necessary and sufficient for them to obtain*

$$k = \Theta\left(2^{(n-q)/2}\right),$$

*copies of their states to estimate $|\langle\phi|\psi\rangle|^2$ up to constant accuracy with high probability.*¹

The upper bound $k = \mathcal{O}\left(2^{(n-q)/2}\right)$ shows that quantum communication may help. However, to reach sub-exponential scaling, a large amount of quantum communication is necessary i.e., unless $q = n - \text{polylog}(n)$, our lower bound is still exponential in n . Informally, entanglement helps, but not too much.

Our next result concerns the sample complexity, under LOCC, of GIPE, i.e., for arbitrary Hermitian operators M . A natural possibility when estimating $|\langle\psi|M|\phi\rangle|^2$ is that, perhaps Alice and Bob only need to confirm that their states have a large overlap in one of several smaller subspaces “within M ”. Then, it is conceivable that this task would be much easier than full inner product estimation (i.e., when $M = \mathbb{I}$). We show that this is indeed the case! Let M_ε be M restricted to subspaces with eigenvalue of magnitude at least $\varepsilon/2$. We prove the following near-optimal results for general GIPE.

► **Theorem 2 (Informal).** *For every M with $\|M\| \leq 1$ and $\varepsilon > 0$, to estimate $|\langle\phi|M|\psi\rangle|^2$ to error ε , it is sufficient to obtain*

$$k = \mathcal{O}(\max\{1/\varepsilon^2, \|M_\varepsilon\|_2/\varepsilon\})$$

copies of $|\psi\rangle, |\phi\rangle$ and

$$k = \Omega(\max\{1/\varepsilon^2, \|M_\varepsilon\|_2/\sqrt{\varepsilon}\})$$

copies are necessary.

Interestingly, this implies that M having both positive and negative eigenvalues may not render estimation harder than definite operators. Take Pauli strings as an example, an important set of operators in quantum information. These have eigenspaces of eigenvalues ± 1 each of dimension 2^{n-1} . Then, in evaluating $|\langle\phi|M|\psi\rangle|^2$ there may be cancellations between components of the states lying in eigenspaces of different signs. However, our results show that estimating such an M is no harder than estimating \mathbb{I} , which is positive definite and basis independent. While our upper and lower bounds do not exactly match, this is partially due to choice of presentation. The lower bound we prove is actually $\Omega(\|M_\varepsilon\|_1/\varepsilon\sqrt{d_\varepsilon})$, where d_ε is the dimension of the support of M_ε . Converting 1 to 2 norm yields the lower bound as presented above. However, in some cases this more precise bound is tight. Take, for example, M a projector onto a subspace or a Pauli string. Then, we obtain the lower bound $\Omega(\|M_\varepsilon\|_2/\varepsilon)$, *exactly* matching the upper bound.

Proof sketch. In [3], they prove their lower bound using a very interesting connection to quantum cloning: in particular, they show that if one could solve a decision-version of DIPE, then one could have cloned the states “well-enough”. One of the main challenges in proving our lower bound was that the cloning-based lower bounds do not work when

¹ We will state the constants in this theorem more clearly when proving this theorem.

allowed even few qubits of communication in the cloning channel (which is the setting in which we would like lower bounds). Instead, here we showed that one could use the notion of *robustness of entanglement*, a concept from entanglement theory and split a protocol with a quantum channel into a sum of LOCC protocols. Proving this is a small technical lemma which we combine with the lower bound of [3] to obtain our overall lower bound of $\sqrt{2^{n-q}}$. In order to prove our upper bound, we use ideas from the celebrated Johnson-Lindenstrauss lemma. Our main idea is to perform projections onto random subspaces, i.e., Alice and Bob pick random subspaces and project their states onto these subspaces, maintaining the inner product between their states with high probability. Further, this protocol is completely agnostic to the states they started with. Now holding smaller-dimensional projections of their states, they can use classical communication to detect when their subspace-projections have collisions, use their shared quantum channel and perform SWAP tests to estimate the inner product within the subspace. With some analysis, we are able to show that the overall sample complexity for this scales as $\sqrt{2^{n-q}}$ as well, matching our lower bound.

For estimating bilinear forms, our protocol is fairly similar to the one in [3] except that we need to deal with some technicalities due to the Hermitian operator M . The key observation we first make is that $|\langle \phi | M | \psi \rangle|^2$ and $|\langle \phi | M_\varepsilon | \psi \rangle|^2$ can only differ by at most $\varepsilon/2$. So it suffices for Alice and Bob to restrict themselves to the support of M_ε . Then, they can simultaneously measure the magnitude of the components of their states in this subspace as well as the weighted inner product of said components. The calculations here are similar to the one in [3] wherein one needs to compute the mean variance of the estimator, but a little bit more technical in order to bound all the terms in terms of $\|M_\varepsilon\|_2$. The lower bound follows from realizing that M_ε when restricted to its support, is not too far off from identity and then we can invoke the lower bounds for inner product estimation can be applied. We remark that the lower bound here is much more subtle than the one in [3] since one needs to project onto the eigenspaces of M carefully in order to get the optimal dependence on $\|M_\varepsilon\|_2$.

1.2 Related works and open questions

Like we mentioned earlier, our work answers two open questions raised by Anshu et al. [3]. There have been a few more papers since their work: Hinsche et al. [21] look at specific instantiations of DIPE (when the protocol used by Alice and Bob are Pauli sampling and the quantum states are structured) and prove some positive and negative results to this end. Chen et al. [10] extend the work of [3] by proving better bounds for non-trace-preserving quantum channels, Chen et al. [11] also looked at quantum property testing in the LOCC model.

Distinguishability under LOCC has a rich history in the quantum information literature. Bennett, et al. demonstrated that there exist product bases which are easily distinguished, but are indistinguishable under LOCC [7]. Enough entanglement renders this task easy as the two parties can simply teleport states. Subsequent works showed that much less entanglement may suffice for this task [14, 29]. Unlike our framework, these papers concerned themselves with measurements of a single copy and further identifying states from some known ensemble, rather than learning some property.

A natural open question. A tantalizing open question left open by [3] and also this work is the analogue of DIPE in the mixed case setting. Here the goal is as follows: Alice has copies of ρ , Bob has copies of σ and they engage in LOCC. They need to distinguish if $\rho = \sigma$ or $\|\rho - \sigma\|_{tr} \geq \varepsilon$, promised one of them is the case. Using the bounds obtained in [3], we have an upper bound of $O(d^2/\varepsilon^4 + d^{1.5}/\varepsilon^2)$. As for lower bounds, [5] showed that if Alice also had copies of σ (instead of Bob having them), then one can solve this property testing task using

$\mathcal{O}(d/\varepsilon^2)$ copies (this procedure requires a highly-entangling operation between copies of ρ and σ). Furthermore, if we restricted the measurements to be single-copy measurements and if we fix $\sigma = \mathbb{I}/d$, then $\Omega(d^{3/2})$ copies are necessary [12]. Despite several attempts in making progress on this question, we have not been able to obtain a lower bound better than $\Omega(d)$ and believe that the sample complexity of mixed state DIPE should be $\mathcal{O}(d)$. We leave this as an open question.

2 Preliminaries

2.1 Quantum States and Measurements

Pure quantum states are unit vectors in \mathbb{C}^d . A qubit is a state in \mathbb{C}^2 and n -qubits is a state in $(\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$. Mixed states, also known as density matrices, ρ , are positive semi-definite operators on \mathbb{C}^d with unit trace. Pure states correspond to rank 1 mixed states and we will routinely use the notation $|\psi\rangle$ to refer to the density matrix corresponding to a pure state $|\psi\rangle$. Measurements of quantum systems are described by positive operator valued measures (POVMs), ensembles of positive semi-definite operators $\{M_i\}_i$ such that $\sum_i M_i = \mathbb{I}$. The probability of observing an outcome i is given by $\text{Tr}[M_i\rho]$. By $\|M\|$ we denote the operator norm of an operator.

In quantum computation states are manipulated via unitary evolution. That is, $|\psi\rangle$ is mapped to $U|\psi\rangle$ for some unitary U . An important unitary we will make repeated usage of is the SWAP gate on a bipartite state. This has the action $\text{SWAP}|\psi\rangle \otimes |\phi\rangle = |\phi\rangle \otimes |\psi\rangle$.

Operators on two Hilbert spaces \mathcal{H}_A and \mathcal{H}_B lie in the tensor product space $\mathcal{B}(\mathcal{H}_A \otimes \mathcal{H}_B)$. A positive semi-definite operator M acting on these tensor product space is said to be separable if it admits a decomposition $M = \sum_i A_i \otimes B_i$, where each A_i and B_i are positive semi-definite. We will be interested in separable POVMs, where each aspect M_i is separable. These are measurements that cannot create entanglement between distributed parties. However, such measurements may still go beyond those achievable with classical communication. With this in mind, one can define local operations and classical communication (LOCC) [13]. This can roughly be defined as all protocols where Alice performs some measurement in her lab, communicates the result to Bob, who then performs a measurement in his lab and communicates the result to Alice and so on. However, the set of LOCC measurements is mathematically unwieldy and it is generally easier to prove results regarding separable measurements. An e -bit is a standard resource of entanglement in quantum information. This is a shared two qubit resource state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. With n e -bits and classical communication, two distributed parties can teleport an n qubit state [23].

2.2 Subroutines

We will make repeated reference to the SWAP test, which can be used to estimate the overlap between two states $|\psi\rangle$ and $|\phi\rangle$ [6, 9].

► **Definition 3 (SWAP test).** *Given two mixed states ρ and σ and an ancilla qubit initialized in the state $|0\rangle_E$, the SWAP test performs the unitary $(H_E \otimes \mathbb{I}_{AB})(|0\rangle\langle 0| \otimes \mathbb{I}_{AB} + |1\rangle\langle 1| \otimes \text{SWAP}_{AB})(H_E \otimes \mathbb{I}_{AB})$ and measures the ancilla in the computation basis. The measurement probabilities are given by*

$$p(0) = \frac{1 + \text{Tr}[\rho\sigma]}{2}, \quad p(1) = \frac{1 - \text{Tr}[\rho\sigma]}{2}.$$

11:6 Generalized Inner Product Estimation with Limited Quantum Communication

Via standard amplification arguments, $\mathcal{O}(1/\varepsilon^2)$ trials are sufficient to estimate $\text{Tr}[\psi\phi]$ to error ε . Using block-encodings, this can be extended to estimating $|\langle\phi|M|\psi\rangle|^2$ for an arbitrary hermitian M such that $\|M\| \leq 1$. Again, standard amplification arguments show that $\mathcal{O}(1/\varepsilon^2)$ trials suffices to estimate $|\langle\phi|M|\psi\rangle|^2$. We will also use the standard POVM on the symmetric subspace. The symmetric group S_k has a natural action on the state space of $(\mathbb{C}^d)^{\otimes k}$ given by

$$\pi \bigotimes_{i=1}^k |\psi_i\rangle = \bigotimes_{i=1}^k |\psi_{\pi^{-1}(i)}\rangle \quad \forall \pi \in S_k . \quad (1)$$

► **Definition 4** (Symmetric Subspace). *The k -copy symmetric subspace of a vector space $V \cong \mathbb{C}^d$, is given by*

$$\vee^k \mathbb{C}^d = \left\{ |\psi\rangle \in (\mathbb{C}^d)^{\otimes k} \mid \pi |\psi\rangle = |\psi\rangle \forall \pi \in S_k \right\} .$$

The symmetric subspace has a natural spanning set given by $\left\{ |\phi\rangle^{\otimes k} \mid |\phi\rangle \in \mathbb{C}^d \right\}$ [20]. Due to $\vee^k \mathbb{C}^d$ being an irreducible representation of the unitary group (under the action $U^{\otimes k}$), we can define a uniform POVM on the symmetric subspace, which is known as the standard POVM.

► **Definition 5** (Standard POVM on $\vee^k \mathbb{C}^d$). *The standard POVM on $\vee^k \mathbb{C}^d$ is the continuous POVM with elements*

$$\left\{ \binom{d+k-1}{k} |\varphi\rangle\langle\varphi|^{\otimes k} d\varphi \mid |\varphi\rangle \in \mathbb{C}^d \right\} . \quad (2)$$

We require one last fact about the symmetric subspace:

► **Fact 6** (Haar moments). *If $|\psi\rangle$ is drawn from the Haar measure on \mathbb{C}^d , then*

$$\mathbb{E}_\psi[|\psi\rangle] = \frac{\mathbb{I}}{d} , \quad \mathbb{E}_\psi[|\psi\rangle\langle\psi|^{\otimes 2}] = \frac{1}{d(d+1)} (\mathbb{I} + \text{SWAP}) . \quad (3)$$

3 Estimating Inner Product

In this section we consider the following task: Alice and Bob each have copies of unknown d -dimensional states $|\phi\rangle$ and $|\psi\rangle$. They may use any amount of classical communication and some restricted quantum communication. Their goal is to estimate $|\langle\phi|\psi\rangle|^2$ using as few samples of $|\psi\rangle$ and $|\phi\rangle$ as possible. Here we will assume the desired measurement precision is some constant.

► **Theorem 7.** *Suppose Alice and Bob share a q -qubit entangled state, can perform measurements on k copies of their respective d -dimensional states $|\psi\rangle$ and $|\phi\rangle$ respectively and engage in unbounded classical communication. If they are able to estimate $|\langle\psi|\phi\rangle|^2$ to accuracy ε with high probability, then $\Omega(\max\{1/\varepsilon^2, \sqrt{d/q}/\varepsilon\})$ are necessary.*

► **Theorem 8.** *Suppose Alice and Bob share a $10q$ -qubit entangled state, can perform measurements on k copies of their respective d -dimensional states $|\psi\rangle$ and $|\phi\rangle$ respectively and engage in unbounded classical communication. It suffices to obtain $k = O(\sqrt{d/q}/\varepsilon^2)$ copies, in order to estimate $|\langle\psi|\phi\rangle|^2$ to accuracy ε with high probability.*

We remark that there is a constant-factor difference in the amount of entanglement bounds in the upper and lower bound. This largely seems to stem from the sample complexity still being greater than 1 even when $q = n$: say that Alice is able to transmit her state exactly and Bob does the swap test. Even then, in order to compute the inner product $|\langle\psi|\phi\rangle|^2$ to error 0.1, they need to repeat the swap test constantly many times. Our lower bound for this setting would imply a constant lower bound as well (matching the upper bound, but not “exactly” since it would be a constant-factor off). Regardless, our lower bound can be understood as saying that entanglement does not help for inner product estimation unless the shared entanglement dimension scales with d .

3.1 Lower bound in main theorem

To obtain a lower bound we consider the decision problem constructed in [3] (which is called the DIPE, *distributed inner product estimation* problem). Alice and Bob are promised to be in one of the following two scenarios. Their version restricts Alice and Bob to LOCC protocols. However, here we allow them to share some resource state σ as well.

- **Definition 9** (Distributed Inner Product Estimation, Decision Version). *Alice and Bob each have access to k copies of the states $|\phi\rangle$ and $|\psi\rangle$ in \mathbb{C}^d respectively and are asked to decide, using LOCC and perhaps a resource state σ , which of the following two scenarios they are in:*
- **YES:** $|\phi\rangle = |\psi\rangle$ and they are Haar random
 - **NO:** $|\phi\rangle, |\psi\rangle$ are Haar random.

Say that Alice and Bob are able to transmit a r dimensional quantum message. Because of quantum teleportation, this communication channel is equivalent, under LOCC, to sharing a r -dimensional maximally entangled state. Going forward, we let $\sigma = \frac{1}{r} \sum_{i,j=1}^r |ii\rangle\langle jj|$ be the state Alice and Bob share.

We also introduce a measure of entanglement called the *robustness of entanglement*.

- **Definition 10** (Robustness of entanglement [26]). *Any quantum state $\sigma \in \mathcal{B}(\mathcal{H}_A \otimes \mathcal{H}_B)$ can be decomposed as $\sigma = (1+s)\sigma^+ - s\sigma^-$, where σ^+ and σ^- are both separable states and $s \in \mathbb{R}_{\geq 0}$. The minimum value of s over all such decompositions is called the robustness of entanglement. We denote this minimum value by $E(\sigma)$.*

In particular, we will use the following lemma.

- **Lemma 11** ([26]). *If σ is a pure bipartite state, then the robustness of entanglement is $E(\sigma) = (\sum_i \lambda_i)^2 - 1$, where $\{\lambda_i\}_i$ are the Schmidt coefficients.*

Thus, the robustness of entanglement of σ is $E(\sigma) = r - 1$ (since all of its Schmidt coefficients are $1/\sqrt{r}$). In [3] they show the following result:

- **Theorem 12** ([3]). *Let M be a separable measurement, then*

$$|\mathbb{E}_{\phi,\psi} [M \cdot \text{Tr}[\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k})]]| \leq e^{k^2/d} - 1. \quad (4)$$

With this we now prove our main theorem lower bound.

Proof of lower bound of Thm 7. Say that there exists a separable protocol \mathcal{A} that uses k copies of the states $|\phi\rangle$ and $|\psi\rangle$ and the resource state σ and returns an estimate of $|\langle\psi|\phi\rangle|^2$ to accuracy ε with probability at least $3/4$. Because of anti-concentration of the Haar measure, Alice and Bob will be able to use this protocol to solve problem 9 with high probability. Let $B([\phi], \varepsilon)$ be the ball of radius $\varepsilon \leq \pi/2$ around $[\phi]$ in $\mathbb{C}\mathbb{P}(d)$ with respect to the Fubini-Study

metric, which we will denote as $d([\phi, \psi]) = \arccos |\langle \psi | \phi \rangle|$. Then, it is known that, under the uniform distribution, the volume of $B([\phi], \varepsilon)$ for $\varepsilon \leq \pi/2$ is given by $\sin^{2d-2} \varepsilon$ [8]. If $|\langle \psi | \phi \rangle|^2 > \varepsilon$, then $d([\psi, \phi]) \leq \arccos \sqrt{\varepsilon}$ (and $\arccos \sqrt{\varepsilon} \leq \pi/2$ for $\varepsilon \in [0, 1]$). Thus, for constant ε it follows that $|\langle \phi | \psi \rangle|^2 \leq \varepsilon$ with exponentially (in d) high probability in case two of 9. Thus, if Alice and Bob use the protocol \mathcal{A} , they will solve problem 9 with probability at least $2/3$ as well. This implies that there is some separable POVM $\{M, \mathbb{I} - M\}$ such that

$$1/3 \leq |\mathbb{E}_{\phi, \psi} \text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma)]|. \quad (5)$$

We now show that the RHS of the inequality above can be upper bounded by $(1 + 2E(\sigma))(e^{k^2/d} - 1)$. To see this, split σ into $\sigma = (1 + E(\sigma))\sigma^+ - E(\sigma)\sigma^-$, where σ^+ and σ^- are both separable states. Then, it follows that

$$\begin{aligned} \text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma)] &= (1 + E(\sigma))\text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma^+)] \\ &\quad + E(\sigma)\text{Tr}[M(\phi^{\otimes k} \otimes (\psi^{\otimes k} - \phi^{\otimes k}) \otimes \sigma^-)]. \end{aligned} \quad (6)$$

Note that the sign of σ^- has been absorbed into the trace in the second term above. Now, because σ^\pm are separable, each term above could be replaced by a separable measurement on $\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k})$ without any reference state. To see this, decompose σ^+ into $\sigma^+ = \sum_i p_i \rho_i^A \otimes \rho_i^B$. Using spectral decompositions $\rho_i^A = \sum_k \lambda_{i,k} |u_{i,k}\rangle \langle u_{i,k}|$ and $\rho_i^B = \sum_k \nu_{i,k} |v_{i,k}\rangle \langle v_{i,k}|$, we arrive at

$$\text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma^+)] \quad (7)$$

$$= \sum_i p_i \sum_{j,k} \lambda_{i,k} \nu_{i,j} \text{Tr}[M(\phi^{\otimes k} \otimes \psi^{\otimes k} \otimes |u_{i,j}\rangle \langle u_{i,j}| \otimes |v_{i,k}\rangle \langle v_{i,k}|)]. \quad (8)$$

As a separable measurement, we have $M = \sum_t A_t \otimes B_t$. Now define a new measurement

$$M' := \sum_t \sum_i p_i \left(\sum_j \lambda_{i,j} A_t^{i,j,j} \right) \otimes \left(\sum_k \nu_{i,k} B_t^{i,k,k} \right), \quad (9)$$

where $A_t = \sum_{j,j'} A_t^{i,j,j'} \otimes |u_{i,j}\rangle \langle u_{i,j'}|$ and $B_t = \sum_{k,k'} B_t^{i,k,k'} \otimes |v_{i,k}\rangle \langle v_{i,k'}|$. As a sum of positive operators, this is positive. As a convex combination of operators majorized by \mathbb{I} , this is also majorized by \mathbb{I} . It then follows that M' is a separable POVM. From Thm 12 it then follows that

$$\begin{aligned} |\mathbb{E}_{\phi, \psi} [\text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma^+)]]| &= |\mathbb{E}_{\phi, \psi} [\text{Tr}[M'(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma^+)]]| \\ &\leq e^{k^2/d} - 1. \end{aligned} \quad (10)$$

The same proof shows that

$$|\mathbb{E}_{\phi, \psi} [\text{Tr}[M(\phi^{\otimes k} \otimes (\psi^{\otimes k} - \phi^{\otimes k}) \otimes \sigma^-)]]| \leq e^{k^2/d} - 1. \quad (11)$$

Thus, we have that

$$|\mathbb{E}_{\phi, \psi} [\text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma)]]| \leq (1 + 2E(\sigma))(e^{k^2/d} - 1). \quad (12)$$

Putting the above along with our lower bound in Eq.(5) shows

$$1/3 \leq |\mathbb{E}_{\phi, \psi} \text{Tr}[M(\phi^{\otimes k} \otimes (\phi^{\otimes k} - \psi^{\otimes k}) \otimes \sigma)]| \leq (1 + 2E(\sigma))(e^{k^2/d} - 1). \quad (13)$$

Rearranging, this implies that

$$k = \Omega\left(\sqrt{\frac{d}{\log E(\sigma)}}\right). \quad (14)$$

So, if Alice and Bob share q Bell pairs, then $E(\sigma) = 2^q - 1$ and we get the desired lower bound.

Using techniques similar to [3, Section 5.3], wherein they showed how to use standard concentration techniques to reduce the decision version of DIPE (for which we have a $\Omega(\sqrt{d/q})$ lower bound) to prove a lower bound for estimating the inner product between $|\phi\rangle$ and $|\psi\rangle$ with a factor of $1/\varepsilon$. In particular, they prove that if there exists a protocol that solves the ε -version of inner product estimation then it can be used to solve DIPE with high probability. Then, the lower bound on DIPE follows from their reduction. \blacktriangleleft

3.2 Upper bound in main theorem

We now give a protocol which achieves the promised sample complexity of Thm 8. At a high level, the protocol works by randomly projecting $|\phi\rangle$ and $|\psi\rangle$ into small-dimensional subspaces. Since such a projection maintains the inner product between two states with high probability, Alice can simply send Bob these smaller dimensional states and Bob can perform a swap test. The reader may notice that this assumes that Alice can transmit an arbitrary state in a q -dimensional subspace of a d -dimensional space. This, however, does not lead to any issues since Alice *knows* the subspace the state lies in. Then, any maximally entangled state on a q -dimensional subspace can be converted to a maximally entangled state on the desired subspace via a unitary transformation. From there, Alice and Bob simply perform a teleportation protocol.

► **Theorem 13.** *Let $q = \Omega(\log d)$. Using $k = \mathcal{O}(\sqrt{d/q})$ copies of $|\psi\rangle$ and $|\phi\rangle$, there is a protocol that uses $\Theta(1)$ r -dimensional quantum messages, single copy measurements, and returns $|\langle\psi|\phi\rangle|^2$ up to constant error with high probability*

Proof. Let $\psi = |\psi\rangle\langle\psi|$ and $\phi = |\phi\rangle\langle\phi|$. Fix some unitary U . For convenience of notation we will assume that r divides d (working with qubits, if Alice and Bob may communicate $\log(r)$ qubits then this holds). Divide the d -dimensional Hilbert space into d/r subspaces each of dimension r . Let this decomposition be given by a collection of orthogonal projectors $\{P_i\}_{i=1}^{d/r}$. Alice and Bob perform the protocol in Figure 1.

Our theorem will follow by repeating the protocol above constantly many times, each time by sending a r -dimensional quantum message. To see the correctness, we will require the following fact.

► **Fact 14** ([25, Fact 2]). *Let $1 \leq q \leq d$. Let $v \in \mathbb{C}^d$ be a unit vector and P_i be an arbitrary projector onto a r -dimensional subspace. Let U be drawn from the unitary Haar measure. Then,*

$$\Pr_U \left[\|P_i U v\|_2 \notin (1 \pm \Delta) \sqrt{\frac{r}{d}} \right] \leq 4 \exp \left\{ -\frac{\Delta^2 r}{16} \right\}, \quad (15)$$

By linearity this extends to arbitrary vectors in \mathbb{C}^d , as the following corollary demonstrates.

► **Corollary 15.** *Let $1 \leq r \leq d$. Let $u \in \mathbb{C}^d$ be any vector and P_i an arbitrary projector onto a r -dimensional subspace. Let U be drawn from the unitary Haar measure. Then,*

$$\Pr_U \left[\|P_i U u\|_2 \notin (1 \pm \Delta) \sqrt{\frac{r}{d}} \|u\|_2 \right] \leq 4 \exp \left\{ -\frac{\Delta^2 r}{16} \right\}, \quad (16)$$

11:10 Generalized Inner Product Estimation with Limited Quantum Communication

Input: Alice gets $|\psi\rangle^{\otimes k}$, Bob gets $|\phi\rangle^{\otimes k}$

Output: ε -approximation of $|\langle\psi|\phi\rangle|^2$

1. Using shared randomness, Alice and Bob sample a unitary U from Haar measure.
2. Alice and Bob apply the POVM $\{UP_iU^\dagger\}_{i=1}^{\lceil d/r \rceil}$ on each of the k copies and record which copies were projected onto which subspaces.
3. Using a two-way classical communication they determine which copies lie now in the same subspaces. Alice sends Bob a constant number of such projected states which can be paired with a post-projection state of Bob's. Say the number of such pairs is m .
4. Bob performs a SWAP test between the post-projected pairs of states lying in the same subspaces. We say a SWAP test succeeds if Bob measures $|0\rangle$ on the ancilla register. Let the number of successes be s .
5. Bob declares the inner product between $|\psi\rangle$ and $|\phi\rangle$ to be $2s/m - 1$.

■ **Figure 1** Protocol to estimate inner product using shared entanglement.

Proof. This follows readily from Fact 14 via linearity. Apply Fact 14 to $\frac{u}{\|u\|_2}$ while multiplying both $\|P_iUu/\|u\|_2\|_2$ and $(1 \pm \Delta)\sqrt{\frac{r}{d}}$ by $\|u\|_2$. ◀

For a fixed unitary U , let $|\psi'_i\rangle = \frac{P_iU|\psi\rangle}{|P_iU|\psi|}$ and $|\phi'_i\rangle = \frac{P_iU|\phi\rangle}{|P_iU|\phi|}$ be the normalized projections of $|\psi\rangle$ and $|\phi\rangle$ into the subspace given by P_i . We will now show that, with high probability over the choice of U , $|\langle\psi'_i|\phi'_i\rangle|^2 \approx |\langle\psi|\phi\rangle|^2$.

► **Lemma 16.** *Let $\Delta > 0$. Then,*

$$\Pr_U \left[\left| |\langle\phi'_i|\psi'_i\rangle|^2 - |\langle\psi|\phi\rangle|^2 \right| \leq 16\Delta \text{ for every } i \in [d/r] \right] \geq 1 - O(d/r \cdot \exp(-\Delta^2 r/16)). \quad (17)$$

Proof. Using Fact 14, with probability at least $1 - 16 \exp\{-\frac{\Delta^2 r}{16}\}$, the following four conditions hold

$$\|P_iU|\psi\rangle\|_2, \|P_iU|\phi\rangle\|_2 \in (1 \pm \Delta)\sqrt{\frac{r}{d}}, \quad (18)$$

$$\|P_iU(|\psi\rangle - |\phi\rangle)\|_2 \in (1 \pm \Delta)\sqrt{\frac{r}{d}}\|\psi - \phi\|_2, \quad (19)$$

$$\|P_iU(|\psi\rangle - i|\phi\rangle)\|_2 \in (1 \pm \Delta)\sqrt{\frac{r}{d}}\|\psi - i\phi\|_2. \quad (20)$$

We now follow steps similar to those of [25, Theorem 1] to obtain

$$|\langle\phi_i|\psi_i\rangle - \langle\psi|\phi\rangle| \leq 8\Delta. \quad (21)$$

Let $x, y \in \mathbb{C}$ be such that $|x - y| \leq \Delta$. Then,

$$|x|^2 - |y|^2 = (|x| - |y|)(|x| + |y|) \leq \Delta(|x| + |y|). \quad (22)$$

In our case, this implies that

$$|\langle\phi_i|\psi_i\rangle|^2 \in |\langle\psi|\phi\rangle|^2 \pm 16\Delta. \quad (23)$$

Taking a union bound over all d/r subspaces completes the proof. ◀

Now, after receiving m pairs of post-projected states, Bob then performs m many SWAP tests. A SWAP test on the pair $|\psi'_i\rangle$ and $|\phi'_i\rangle$ succeeds with probability $\frac{1}{2} + \frac{1}{2}|\langle\phi'_i|\psi'_i\rangle|^2 \in \frac{1}{2} + \frac{1}{2}|\langle\phi|\psi\rangle| \pm 8\Delta$. The expected value of the sample average s/m then satisfies

$$\left| \mathbb{E} \left[\frac{s}{m} \right] - \frac{1}{2} - \frac{1}{2}|\langle\psi|\phi\rangle|^2 \right| \leq 8\Delta, \quad (24)$$

which implies $2\mathbb{E} \left[\frac{s}{m} \right] - 1 \in |\langle\phi|\psi\rangle|^2 \pm 16\Delta$, and by a Hoeffding bound, we get

$$2 \left| \frac{s}{m} - \mathbb{E} \left[\frac{s}{m} \right] \right| \leq \delta, \quad (25)$$

with probability at least $1 - 2\exp(-2m\delta^2)$. In this case, the total error of the estimator is at most $16\Delta + \delta$. For constant error, $m = O(1)$ suffices.

It remains to determine k needs to be to obtain $m = O(1)$ projected pairs with high probability. To this end, let $E_{a,b}$ be an indicator variable for if Alice's a th projection falls into the same subspace as Bob's b th projection. The expected number of collisions is given by

$$\sum_{a,b} \mathbb{E}[E_{a,b}] = k^2 \sum_i \text{Tr}[P_i U \psi U^\dagger] \text{Tr}[P_i U \phi U^\dagger] \geq k^2 (1 - \Delta)^4 \frac{r}{d}, \quad (26)$$

where the inequality follows from $\|P_i U |\psi\rangle\|, \|P_i U |\phi\rangle\| \geq (1 - \Delta)\sqrt{\frac{r}{d}}$. By a Hoeffding bound, it suffices to take $k = \Omega(\sqrt{d/r})$ to obtain a collision with high probability. As we only require a constant number of pairs, this can simply be repeated a constant number of times to obtain a constant number of collisions with high probability. Thus, the protocol succeeds for $k = \Omega(\sqrt{d/r})$. It remains to pick Δ and δ such that the claims go through: we require that $\frac{d}{r} \exp(-\Delta^2 r/16) = O(1)$, which can be achieved by letting $\Delta = O(1)$ and $r = \Omega(\log d)$. ◀

4 Generalized Distributed Inner Product Estimation

Now we turn to bilinear forms $f : \mathbb{C}^d \otimes \mathbb{C}^d \rightarrow \mathbb{C}$. Any such form can be expressed as $f(u, v) = u^\dagger M v$ for a matrix M . Here we will assume that M is Hermitian, which implies that $f(u, v) = \overline{f(v, u)}$. Without loss of generality we will assume M to be normalized such that $\|M\| = 1$. Due to the unphysical nature of global phases, $f(|\psi\rangle, |\phi\rangle)$ is less meaningful than $|f(|\psi\rangle, |\phi\rangle)|^2$ and we concern ourselves entirely with this value instead. Of course, the special case of $M = \mathbb{I}$ corresponds to inner product estimation. Fixing such an M , let P_ε be a projector which annihilates all eigenspaces of M of eigenvalue less than $\varepsilon/2$. We define d_ε to be the dimension of the support of P_ε . Now define $M_\varepsilon = P_\varepsilon M P_\varepsilon$. The intuition behind this is that we can drop eigenspaces with small weights in estimating $|\langle\phi|M|\psi\rangle|^2$. The following lemma formalizes this.

► **Lemma 17.** *Let M be a Hermitian operator and $M_\varepsilon = P_\varepsilon M P_\varepsilon$, then*

$$\left| |\langle\phi|M|\psi\rangle|^2 - |\langle\phi|M_\varepsilon|\psi\rangle|^2 \right| \leq \varepsilon/2. \quad (27)$$

Proof.

$$|\text{Tr}[M\psi M\phi] - \text{Tr}[M_\varepsilon\psi M_\varepsilon\phi]| = |\text{Tr}[\psi(M\phi M - M_\varepsilon\phi M_\varepsilon)]| \quad (28)$$

$$= |\text{vec}(\psi)^\dagger (M \otimes M - M_\varepsilon \otimes M_\varepsilon)^T \text{vec}(\phi)| \quad (29)$$

$$\leq \|M \otimes M - M_\varepsilon \otimes M_\varepsilon\| \leq \varepsilon/2, \quad (30)$$

where the second line uses the linear map $\text{vec} : \mathcal{B}(\mathcal{H}) \rightarrow \mathcal{H} \otimes \mathcal{H}$ defined by $\text{vec}(|i\rangle\langle j|) = |i\rangle \otimes |j\rangle$ and the identity $\text{vec}(AXB) = (A \otimes B)^T \text{vec}(X)$. ◀

Input: An operator M . Alice gets $|\psi\rangle^{\otimes k}$, Bob gets copies of $|\phi\rangle^{\otimes k}$.

Output: An ε -approximation of $\langle\psi|M|\phi\rangle^2$.

1. Alice and Bob each perform the two-outcome measurement $\{P_\varepsilon, \mathbb{I} - P_\varepsilon\}$ on each of the k copies of their respective states and obtain s_a and s_b copies of the states projected into one of the two subspace. If $s_a = 0$ or $s_b = 0$ then Alice and Bob simply output 0.
2. Alice and Bob independently perform the standard POVM in the symmetric subspace on the support of M_ε , obtaining the classical outcomes $|u\rangle$ and $|v\rangle$.
3. Alice communicates $|u\rangle$ and s_a to Bob who outputs

$$w := \frac{(d_\varepsilon + s_a)(d_\varepsilon + s_b)}{k^2} |\langle u|M_\varepsilon|v\rangle|^2 - \frac{\text{Tr}[M^2]}{k^2}. \quad (31)$$

■ **Figure 2** Protocol to estimate $|\langle\phi|M_\varepsilon|\psi\rangle|^2$ using only LOCC.

Using this lemma, if Alice and Bob can estimate $|\langle\phi|M_\varepsilon|\psi\rangle|^2$ up to precision $\varepsilon/2$, that directly implies an ε -approximation to the quantity $|\langle\phi|M|\psi\rangle|^2$. For the remainder of this section, we will be primarily focused on upper and lower bounds for estimating $|\langle\phi|M_\varepsilon|\psi\rangle|^2$.

With this notation, in this section our main result will be the following theorem, proving close-to-optimal upper and lower bounds for estimating bilinear forms on $|\psi\rangle, |\phi\rangle$.

► **Theorem 18.** *For every M with $\|M\| \leq 1$, with only LOCC, it is sufficient to obtain*

$$k = O(\max\{1/\varepsilon^2, \|M_\varepsilon\|_2/\varepsilon\})$$

copies of $|\psi\rangle, |\phi\rangle$ to estimate $|\langle\phi|M|\psi\rangle|^2$ to error ε with high prob. and it necessary to obtain

$$k = \Omega(\max\{1/\varepsilon^2, \|M_\varepsilon\|_2/\sqrt{\varepsilon}\})$$

copies to produce an ε -approximation to $|\langle\phi|M|\psi\rangle|^2$.

4.1 Upper Bound

In Figure 2 we outline a protocol that estimates $|\langle\phi|M_\varepsilon|\psi\rangle|^2$. Recall that the standard POVM on the symmetric subspace, def 5, has continuous aspects $\left\{ \binom{d+k-1}{k} \varphi^{\otimes k} \mid |\varphi\rangle \in \mathbb{C}^d \right\}$. This can be extended to $\vee^k W$ for arbitrary subspaces $W \subseteq \mathbb{C}^d$:

$$\left\{ \binom{\dim W + k - 1}{k} \varphi^{\otimes k} \mid |\varphi\rangle \in W \right\}. \quad (32)$$

The second step of the protocol of Figure 2 has Alice and Bob implementing this POVM for $\text{Im}P_\varepsilon \subseteq \mathbb{C}^d$. Since they may not have k copies after the projection step, this is a POVM on $\vee^{s_a} \text{Im}P_\varepsilon$ and $\vee^{s_b} \text{Im}P_\varepsilon$, respectively. First note that if $P_\varepsilon|\psi\rangle = 0$ or $P_\varepsilon|\phi\rangle = 0$ then they always output 0, which must be a good estimate in this case. Thus, we assume that $|\psi_\varepsilon\rangle = P_\varepsilon|\psi\rangle / \|P_\varepsilon|\psi\rangle\|$ and $|\phi_\varepsilon\rangle = P_\varepsilon|\phi\rangle / \|P_\varepsilon|\phi\rangle\|$ both exist. The technical lemmas that we prove are the mean and variance of our estimators, whose proofs we defer to the full version [4].

► **Lemma 19.** *The expected value of the estimator w given in Figure 2 is*

$$\mathbb{E}[w] = |\langle\phi|M_\varepsilon|\psi\rangle|^2 + \frac{\text{Tr}[M_\varepsilon^2\phi_\varepsilon]}{k} \text{Tr}[P_\varepsilon\psi] + \frac{\text{Tr}[M_\varepsilon^2\psi_\varepsilon]}{k} \text{Tr}[P_\varepsilon\phi]. \quad (33)$$

► **Lemma 20.** *The variance of the estimator of Figure 2 is upper bounded by*

$$\text{Var}(w) = \mathcal{O}\left(\frac{1}{k} + \frac{\|M_\varepsilon\|_2^2}{k^2} + \frac{\|M_\varepsilon\|^4}{k^4}\right). \quad (34)$$

Our estimator is biased, but the bias is at most $2/k$. Taking

$$k = \Omega\left(\max\left\{\frac{1}{\varepsilon^2}, \frac{\|M_\varepsilon\|_2^2}{\varepsilon}\right\}\right),$$

ensures that both the variance and bias are small enough that Alice and Bob's estimate is within $\varepsilon/2$ of $|\langle\phi|M_\varepsilon|\psi\rangle|^2$ and thus within ε of $|\langle\phi|M|\psi\rangle|^2$ (with high probability).

4.2 Lower Bound

We now prove the claimed lower bounds in Theorem 18. The first, $\Omega(1/\varepsilon^2)$ follows from lemma 13 in [3]. We restate their argument here slightly adapted to our setup.

► **Lemma 21.** *Say there is an algorithm acting on $\rho^{\otimes k} \otimes \sigma^{\otimes k}$ that outputs an estimate of $\text{Tr}[M\rho M\sigma]$ to accuracy ε with high probability. Then, $k = \Omega(1/\varepsilon^2)$.*

Proof. Let $|0\rangle$ be such that $M|0\rangle = \pm 1$, which exists for since $\|M\| = 1$ and the unit sphere is compact. Let $|1\rangle$ be an eigenvector of M orthogonal to $|0\rangle$. Consider the states

$$|\psi_0\rangle = \sqrt{\frac{1}{2} - \varepsilon}|0\rangle + \sqrt{\frac{1}{2} + \varepsilon}|1\rangle, \quad |\psi_1\rangle = \sqrt{\frac{1}{2} + \varepsilon}|0\rangle + \sqrt{\frac{1}{2} - \varepsilon}|1\rangle. \quad (35)$$

Now, say that Alice is given k copies of $|\psi_0\rangle$ or $|\psi_1\rangle$ and Bob is given k copies of $|0\rangle$. We have that

$$|\langle\psi_0|M|0\rangle|^2 = \frac{1}{2} - \varepsilon, \quad |\langle\psi_1|M|0\rangle|^2 = \frac{1}{2} + \varepsilon. \quad (36)$$

Thus, if Alice and Bob can estimate $|\langle\psi|M|\phi\rangle|^2$ to accuracy ε , they can distinguish between these two states. However, the fidelity between these states is given by

$$F(\psi_0^{\otimes k}, \psi_1^{\otimes k}) = (1 - 4\varepsilon^2)^k \geq 1 - 4k\varepsilon^2, \quad (37)$$

which implies that $k = \Omega(1/\varepsilon^2)$. ◀

Before proving the second lower bound, we give some intuition. Say that $M \geq 0$. If $|\psi\rangle$ and $|\phi\rangle$ are drawn independently from the Haar measure, then

$$\mathbb{E}[\text{Tr}[M\psi M\phi]] = \text{Tr}[M^2]/d^2. \quad (38)$$

If they are identical, that is $\psi = \phi$ always, then instead the expected value is

$$\mathbb{E}[\text{Tr}[M\psi M\psi]] = \frac{1}{d(d-1)} (\text{Tr}[M^2] + \text{Tr}[M]^2). \quad (39)$$

The difference between these two is roughly $\text{Tr}[M^2]/d^2$. Thus, if Alice and Bob are able to estimate $|\langle\psi|M|\phi\rangle|^2$ to this order, then they can solve DIPE (that we defined as Problem 9). However, $\text{Tr}[M^2]/d^2$ may be quite small. Restricted to the support of M_ε , we instead have that $\varepsilon^2/4 \leq \text{Tr}[M_\varepsilon^2]/d_\varepsilon^2 \leq 1$. We will show that if Alice and Bob can estimate $|\langle\phi|M_\varepsilon|\psi\rangle|^2$, then they can solve the following decision problem, which is known to require $k = \Omega(\sqrt{\dim \mathcal{H}}/\varepsilon)$ samples [3] (when $\varepsilon \leq 0.01$).

11:14 Generalized Inner Product Estimation with Limited Quantum Communication

► **Definition 22** (Inner product estimation, ε decision version). *Let $\varepsilon > 0$. Alice and Bob are given k copies each of pure states $|\psi\rangle, |\phi\rangle \in \mathcal{H}$, for some finite dimensional Hilbert space \mathcal{H} . They are asked to decide, using LOCC, which of the following two scenarios they are in:*

■ **YES instance:**

$$|\psi\rangle = \sqrt{1 - \varepsilon}e^{i\theta} |0\rangle + \sqrt{\varepsilon} |\chi\rangle, \quad |\phi\rangle = \sqrt{1 - \varepsilon}e^{i\theta'} |0\rangle + \sqrt{\varepsilon} |\chi\rangle, \quad (40)$$

where $|\chi\rangle$ is drawn from the Haar measure on the orthogonal complement of $|0\rangle$ and θ and θ' are independently and uniformly drawn from $[0, 2\pi)$.

■ **NO instance:**

$$|\psi\rangle = \sqrt{1 - \varepsilon}e^{i\theta} |0\rangle + \sqrt{\varepsilon} |\chi\rangle, \quad |\phi\rangle = \sqrt{1 - \varepsilon}e^{i\theta'} |0\rangle + \sqrt{\varepsilon} |\varphi\rangle, \quad (41)$$

where $|\chi\rangle$ and $|\varphi\rangle$ are drawn independently from the Haar measure on the orthogonal complement of $|0\rangle$ and θ and θ' are independently and uniformly drawn from $[0, 2\pi)$.

We will require the two following technical lemmas in proving the lower bound. Here Lipschitz constants are taken to be with respect to the Euclidean norm $\|\cdot\|_2$.

► **Fact 23** (Levy's Lemma). *If $f : \mathbb{S}^d \rightarrow \mathbb{R}$ is λ -Lipschitz, then*

$$\Pr_u[|f(u) - \mathbb{E}[f(u)]| \geq t] \leq 2e^{-\frac{dt^2}{2\lambda^2}}, \quad (42)$$

where u is drawn from the Haar measure on \mathbb{S}^d .

► **Lemma 24.** *Let $c < 1$ be a sufficiently small constant. Say there is a protocol acting on $\rho^{\otimes k} \otimes \sigma^{\otimes k}$ via LOCC that outputs an estimate of $\text{Tr}[M\rho M\sigma]$ to accuracy $c\varepsilon$ with high probability. Then, $k = \Omega(\|M_\varepsilon\|_1/\varepsilon\sqrt{d_\varepsilon})$.*

Proof. Let $M|0\rangle = |0\rangle$ be stabilized by M . Such a vector exists by considering either M or $-M$, and we choose the appropriate sign without loss of generality. Let $W = \{|\psi\rangle \in \mathbb{C}^d \mid \langle 0|\psi\rangle = 0\} \cap \text{Im}P_\varepsilon$ and $\tilde{M}_\varepsilon = M_\varepsilon|_W$. Without loss of generality we assume that \tilde{M}_ε is a (semi)definite matrix. Indeed, let $\tilde{M}_\varepsilon = \tilde{M}_\varepsilon^+ - \tilde{M}_\varepsilon^-$ then $\|\tilde{M}_\varepsilon\|_2^2 = \|\tilde{M}_\varepsilon^+\|_2^2 + \|\tilde{M}_\varepsilon^-\|_2^2$. Of course, this means that $\|\tilde{M}_\varepsilon^+\|_2 \geq \|\tilde{M}_\varepsilon\|_2/\sqrt{2}$ or $\|\tilde{M}_\varepsilon^-\|_2 \geq \|\tilde{M}_\varepsilon\|_2/\sqrt{2}$. Thus, we restrict ourselves further and consider the subspace, again labeled by W , in the support of the operator with a larger norm.

We will now show that estimating $|\langle \phi|M|\psi\rangle|^2$ suffices to solve problem 22 with high probability where $\mathcal{H} = \mathbb{C}|0\rangle \oplus W$. Let $\tilde{d} := \dim W$. Set the precision parameter in problem 22 to be $\delta := \varepsilon\tilde{d}/200\text{Tr}[\tilde{M}_\varepsilon]$ and define $\vartheta := \theta - \theta'$. Since $\text{Tr}[\tilde{M}_\varepsilon] \geq \varepsilon\tilde{d}/2$, $\delta \leq 0.01$ as required. Let f be Alice and Bob's estimate of $|\langle \phi|M|\psi\rangle|^2$, and say that it is within $c \cdot \varepsilon$ of the true value with probability at least 0.99. Then, if f lies in the range $[(1 - \delta)^2 - 3c \cdot \varepsilon, (1 - \delta)^2 + c \cdot \varepsilon]$, they REJECT. Otherwise, they ACCEPT. We split the proof into the two cases.

YES instance: in this case

$$|\langle \phi|M|\psi\rangle|^2 = (1 - \delta)^2 + \delta^2\text{Tr}[M\chi]^2 + 2\delta(1 - \delta)\cos\vartheta\text{Tr}[M\chi]. \quad (43)$$

Thus,

$$\Pr[f \in [(1 - \delta)^2 - \frac{\delta}{8}, (1 - \delta)^2 + x]] \quad (44)$$

$$\leq \Pr[\delta^2\text{Tr}[M\chi]^2 + 2\delta(1 - \delta)\cos\vartheta\text{Tr}[M\chi] \in [-\frac{\delta}{8} - c\varepsilon, \frac{\delta}{8} + c\varepsilon]] + 0.01. \quad (45)$$

$\text{Tr}[M\chi]$ is 2-Lipschitz:

$$|\langle\psi|M|\psi\rangle - \langle\phi|M|\phi\rangle| = |(\langle\psi| - \langle\phi|)M|\psi\rangle - \langle\phi|M(|\phi\rangle - |\psi\rangle)| \quad (46)$$

$$\leq |(\langle\psi| - \langle\phi|)M|\psi\rangle| + |\langle\phi|M(|\phi\rangle - |\psi\rangle)| \quad (47)$$

$$\leq 2\| |\psi\rangle - |\phi\rangle \|, \quad (48)$$

where the final inequality follows from $\|M\| = 1$ and Cauchy-Schwarz. Then, using fact 23, it holds that

$$\Pr \left[|\text{Tr}[M\chi] - \mathbb{E}_\chi[\text{Tr}[M\chi]]| \leq \frac{7}{\sqrt{\tilde{d}}} \right] > 0.99. \quad (49)$$

We now have that $\frac{7}{\sqrt{\tilde{d}}} \leq \frac{\text{Tr}[\tilde{M}_\varepsilon]}{4\tilde{d}}$, since $28/\sqrt{\tilde{d}} \leq \varepsilon$. But notice that if $\varepsilon = o(1/\sqrt{\tilde{d}})$, then the lower bound $k = \Omega(1/\varepsilon^2) = \Omega(d)$ dominates since $\|M_\varepsilon\|_2/\sqrt{\varepsilon} \leq \sqrt{\tilde{d}}/\sqrt{\varepsilon} = o(1/\varepsilon^2)$. The above bounds imply that we can take $\text{Tr}[M\chi]^2 \leq 25\text{Tr}[\tilde{M}_\varepsilon]^2/16\tilde{d}^2$. We are then interested in the probability

$$\Pr \left[2\delta(1-\delta)\frac{3\text{Tr}[\tilde{M}_\varepsilon]}{4\tilde{d}} \cos \vartheta \in \left[-\frac{\delta}{8} - c\varepsilon - \delta^2\frac{25\text{Tr}[\tilde{M}_\varepsilon]^2}{16\tilde{d}^2}, \frac{\delta}{8} + c\varepsilon - \delta^2\frac{\text{Tr}[\tilde{M}_\varepsilon]^2}{\tilde{d}^2} \right] \right]. \quad (50)$$

Now, by construction $\delta \leq 0.01$ and thus $2(1-\delta) \geq 99/50$. It follows that

$$2\delta(1-\delta)\frac{3\text{Tr}[\tilde{M}_\varepsilon]}{4\tilde{d}} \geq \frac{297}{40000}\varepsilon. \quad (51)$$

Then, the above probability is upper bounded by

$$\Pr \left[\cos \vartheta \in \left[-\frac{1}{2} - \frac{25\varepsilon}{4752}, \frac{1}{2} + \frac{\varepsilon}{297} \right] \right] \leq \Pr [\cos \vartheta \in [-0.51, 0.51]] \leq 0.34. \quad (52)$$

In total, they accept in the YES case with probability at least 0.66.

NO instance: in this case

$$|\langle\varphi|M|\chi\rangle|^2 = (1-\delta)^2 + \delta^2\text{Tr}[M\varphi M\chi] + 2\delta(1-\delta)\text{Re}(e^{i\vartheta}\langle\varphi|M|\chi\rangle). \quad (53)$$

By symmetry, $\mathbb{E}[\text{Re}(e^{i\vartheta}\langle\varphi|M|\chi\rangle)] = 0$. Now, fixing $|\varphi\rangle$, $|\langle\varphi|M|\chi\rangle|$ is 1-Lipschitz in $|\chi\rangle$. This implies that $|\langle\varphi|M|\chi\rangle| \leq 8/\sqrt{\tilde{d}}$ with probability at least 0.999. This then implies that $\text{Tr}[M\varphi M\chi] \leq 84/\tilde{d}$. Then, we have that

$$|\langle\varphi|M|\chi\rangle|^2 - (1-\delta)^2 \leq \frac{\delta}{\sqrt{\tilde{d}}} \left(\frac{84}{\sqrt{\tilde{d}}} + 16 \right). \quad (54)$$

Further, $\delta/\sqrt{\tilde{d}} \leq 1/100\sqrt{\tilde{d}}$. We assume that $84/\sqrt{\tilde{d}} < 1$ (as otherwise the lower bound is constant anyways) and obtain

$$|\langle\varphi|M|\chi\rangle|^2 - (1-\delta)^2 \leq \frac{17}{100\sqrt{\tilde{d}}}. \quad (55)$$

We require that this is less than $c\varepsilon/3$. This requires that $\varepsilon = \Omega(1/\sqrt{\tilde{d}})$, but, as previously stated, we assume this as otherwise the other lower bound dominates. The total probability for this instance to be rejected is then at least 0.9.

Thus, Alice and Bob are able to solve problem 22 with high probability. Then, it must be that $k = \Omega(\sqrt{\tilde{d}}/\delta) = \Omega(\text{Tr}[\tilde{M}_\varepsilon]/\varepsilon\sqrt{\tilde{d}})$. ◀

► **Corollary 25.** *Let $c < 1$ be a sufficiently small constant. Say there is a protocol acting on $\rho^{\otimes k} \otimes \sigma^{\otimes k}$ via LOCC that outputs an estimate of $\text{Tr}[M\rho M\sigma]$ to accuracy $c\varepsilon$ with high probability. Then, $k = \Omega(\|M_\varepsilon\|_2/\sqrt{\varepsilon})$.*

Proof. Lemma 24 yields a lower bound of $k = \Omega(\text{Tr}[\tilde{M}_\varepsilon]/\varepsilon\sqrt{\tilde{d}})$. Using $\text{Tr}[\tilde{M}_\varepsilon] \geq \|\tilde{M}_\varepsilon\|_2^2$ and $\|\tilde{M}_\varepsilon\|_2 \geq \sqrt{\varepsilon\tilde{d}/2}$, we arrive at our claimed lower bound of $k = \Omega(\|M_\varepsilon\|_2/\sqrt{\varepsilon})$. ◀

References

- 1 Pablo Andres-Martinez, Tim Forrer, Daniel Mills, Jun-Yi Wu, Luciana Henaut, Kentaro Yamamoto, Mio Muraio, and Ross Duncan. Distributing circuits over heterogeneous, modular quantum computing network architectures. *Quantum Science and Technology*, 9(4):045021, 2024.
- 2 Anurag Anshu and Srinivasan Arunachalam. A survey on the complexity of learning quantum states. *Nature Reviews Physics*, 6(1):59–69, 2024.
- 3 Anurag Anshu, Zeph Landau, and Yunchao Liu. Distributed quantum inner product estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 44–51, 2022. doi:10.1145/3519935.3519974.
- 4 Srinivasan Arunachalam and Louis Schatzki. Distributed inner product estimation with limited quantum communication. *arXiv preprint*, 2024. doi:10.48550/arXiv.2410.12684.
- 5 Costin Bădescu, Ryan O’Donnell, and John Wright. Quantum state certification. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 503–514, 2019. doi:10.1145/3313276.3316344.
- 6 Adriano Barenco, Andre Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello. Stabilization of quantum computations by symmetrization. *SIAM Journal on Computing*, 26(5):1541–1557, 1997. doi:10.1137/S0097539796302452.
- 7 Charles H Bennett, David P DiVincenzo, Christopher A Fuchs, Tal Mor, Eric Rains, Peter W Shor, John A Smolin, and William K Wootters. Quantum nonlocality without entanglement. *Physical Review A*, 59(2):1070, 1999.
- 8 Michael Brannan. Alice and bob meet banach: The interface of asymptotic geometric analysis and quantum information theory, 2021.
- 9 Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. Quantum fingerprinting. *Physical review letters*, 87(16):167902, 2001.
- 10 Kean Chen, Qisheng Wang, Peixun Long, and Mingsheng Ying. Unitarity estimation for quantum channels. *IEEE Transactions on Information Theory*, 69(8):5116–5134, 2023. doi:10.1109/TIT.2023.3263645.
- 11 Kean Chen, Qisheng Wang, and Zhicheng Zhang. Local test for unitarily invariant properties of bipartite quantum states. *arXiv preprint*, 2024. doi:10.48550/arXiv.2404.04599.
- 12 Sitan Chen, Jerry Li, Brice Huang, and Allen Liu. Tight bounds for quantum state certification with incoherent measurements. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1205–1213. IEEE, 2022. doi:10.1109/FOCS54457.2022.00118.
- 13 Eric Chitambar, Debbie Leung, Laura Mančinska, Maris Ozols, and Andreas Winter. Everything you always wanted to know about locc (but were afraid to ask). *Communications in Mathematical Physics*, 328:303–326, 2014.
- 14 Scott M Cohen. Understanding entanglement as resource: Locally distinguishing unextendible product bases. *Physical Review A – Atomic, Molecular, and Optical Physics*, 77(1):012304, 2008.
- 15 Jacob P Covey, Harald Weinfurter, and Hannes Bernien. Quantum networks with neutral atom processing nodes. *npj Quantum Information*, 9(1):90, 2023.

- 16 Andreas Elben, Benoît Vermersch, Rick van Bijnen, Christian Kokail, Tiff Brydges, Christine Maier, Manoj K Joshi, Rainer Blatt, Christian F Roos, and Peter Zoller. Cross-platform verification of intermediate scale quantum devices. *Physical review letters*, 124(1):010504, 2020.
- 17 Heng Fan. Distinguishability and indistinguishability by local operations and classical communication. *Physical Review Letters*, 92(17):177905, 2004.
- 18 Ranjani G Sundaram, Himanshu Gupta, and CR Ramakrishnan. Efficient distribution of quantum circuits. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 19 Giovanni Guccione, Tom Darras, Hanna Le Jeannic, Varun B Verma, Sae Woo Nam, Adrien Cavallès, and Julien Laurat. Connecting heterogeneous quantum networks by hybrid entanglement swapping. *Science advances*, 6(22):eaba4508, 2020.
- 20 Aram W Harrow. The church of the symmetric subspace. *arXiv preprint*, 2013. [arXiv:1308.6595](https://arxiv.org/abs/1308.6595).
- 21 Marcel Hinsche, Marios Ioannou, Sofiene Jerbi, Lorenzo Leone, Jens Eisert, and Jose Carrasco. Efficient distributed inner product estimation via pauli sampling. *arXiv preprint*, 2024. [arXiv:2405.06544](https://arxiv.org/abs/2405.06544).
- 22 Christopher Monroe, Robert Raussendorf, Alex Ruthven, Kenneth R Brown, Peter Maunz, L-M Duan, and Jungsang Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89(2):022317, 2014.
- 23 Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.
- 24 A Pirker, J Wallnöfer, and W Dür. Modular architectures for quantum networks. *New Journal of Physics*, 20(5):053054, 2018.
- 25 Pranab Sen. A quantum johnson-lindenstrauss lemma via unitary t-designs. *arXiv preprint*, 2018. [arXiv:1807.08779](https://arxiv.org/abs/1807.08779).
- 26 Guifré Vidal and Rolf Tarrach. Robustness of entanglement. *Physical Review A*, 59(1):141, 1999.
- 27 Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):eaam9288, 2018.
- 28 Jun-Yi Wu, Kosuke Matsui, Tim Forrer, Akihito Soeda, Pablo Andrés-Martínez, Daniel Mills, Luciana Henaut, and Mio Muraō. Entanglement-efficient bipartite-distributed quantum computing. *Quantum*, 7:1196, 2023. [doi:10.22331/q-2023-12-05-1196](https://doi.org/10.22331/q-2023-12-05-1196).
- 29 Zhi-Chao Zhang, Fei Gao, Tian-Qing Cao, Su-Juan Qin, and Qiao-Yan Wen. Entanglement as a resource to distinguish orthogonal product states. *Scientific reports*, 6(1):30493, 2016.

Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

Christine Awofeso  

Birkbeck, University of London, UK

Patrick Greaves  

Birkbeck, University of London, UK

Oded Lachish  

Birkbeck, University of London, UK

Felix Reidl  

Birkbeck, University of London, UK

Abstract

We study the property of H -freeness in graphs with known bounded average degree, i.e. the property of a graph not containing some graph H as a subgraph. H -freeness is one of the fundamental graph properties that has been studied in the property testing framework.

Levi [10] showed that triangle-freeness is testable in graphs of bounded *arboricity*, which is a superset of e.g. planar graphs or graphs of bounded degree. Complementing this result is a recent preprint [7] by Eden *et al.* which shows that, for every $r \geq 4$, C_r -freeness is not testable in graphs of bounded arboricity.

We proceed in this line of research by using the r -admissibility measure that originates from the field of structural sparse graph theory. Graphs of bounded 1-admissibility are identical to graphs of bounded arboricity, while graphs of bounded degree, planar graphs, graphs of bounded genus, and even graphs excluding a fixed graph as a (topological) minor have bounded r -admissibility for any value of r [12].

In this work we show that H -freeness is testable in graphs with bounded 2-admissibility for all graphs H of diameter 2. Furthermore, we show the testability of C_4 -freeness in bounded 2-admissible graphs directly (with better query complexity) and extend this result to C_5 -freeness. Using our techniques it is also possible to show that C_6 -freeness and C_7 -freeness are testable in graphs with bounded 3-admissibility. The formal proofs will appear in the journal version of this paper.

These positive results are supplemented with a lower bound showing that, for every $r \geq 4$, C_r -freeness is not testable for graphs of bounded $(\lfloor r/2 \rfloor - 1)$ -admissibility. This lower bound will appear in the journal version of this paper. This implies that, for every $r > 0$, there exists a graph H of diameter $r + 1$, such that H -freeness is not testable on graphs with bounded r -admissibility. These results lead us to the conjecture that, for every $r > 4$, and $t \leq 2r + 1$, C_t -freeness is testable in graphs of bounded r -admissibility, and for every $r > 2$, H -freeness for graphs H of diameter r is testable in graphs with bounded r -admissibility.

2012 ACM Subject Classification Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Property Testing, Sparse Graphs, Degeneracy, Admissibility

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.12

1 Introduction

A *graph property* is a class of graphs closed under isomorphisms. A property-tester for a property P is a probabilistic algorithm that receives as input the size of a graph G , a distance parameter $\epsilon > 0$ (among potentially other parameters), and oracle access to the graph G . The algorithm accepts with probability at least $2/3$ any input from P and rejects with probability at least $2/3$ an input that it is ϵ -far from the property P . The term “ ϵ -far” is a notion of distance that depends on the exact problem setting and discuss it further



© Christine Awofeso, Patrick Greaves, Oded Lachish, and Felix Reidl;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 12; pp. 12:1–12:16



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



below. The complexity measure of a property-tester is a function that bounds above the total number of queries to the oracle the algorithm uses, as a function of the input parameters ϵ , size of the graph and any other input parameters provided. A property is *testable* if it has a property-tester whose query complexity is independent of the size of the input graph.

We study here the property of H -freeness, where H is a fixed known graph, i.e. the property of graphs that do not have a subgraph isomorphic to H , which is one of the fundamental graph properties that has been studied in the property testing framework. H -freeness was studied in the dense, sparse and general graph models. In the dense model it was shown implicitly in [1] that H -freeness is testable, for a more explicit result see [2]. Goldreich and Ron [8], showed that H -freeness is testable in the bounded degree graph model. In an effort to move towards larger sparse¹ graph classes, Czumaj *et al.* [4] showed that H -freeness is testable in sparse graphs when H is a tree. This result is most likely tight for sparse graphs as Alon *et al.* [3] showed that triangle-freeness is not testable in sparse graphs.

While this settles the question for the most general notion of sparse graphs, the question is still open for a plethora of sparse graph subclasses which are more structured. Possibly the most famous among them is the class of planar graphs. Czumaj *et al.* [5] showed that H -freeness is testable for this class. Proceeding in this line of research, and moving to a much larger class of sparse graphs, Levi [10] showed that triangle-freeness is testable in graphs of bounded *arboricity*, which is a superset of the family of planar graphs. In Eden *et al.* [7] it is shown that, for every $r \geq 4$, C_r -freeness is not testable in graphs of bounded arboricity. Specifically, it is shown that, in graphs of bounded arboricity, the query complexity of C_4 -freeness and C_5 -freeness is $\tilde{\Theta}(n^{1/4})$, the query complexity of C_6 -freeness is $\tilde{O}(n^{1/2})$, and for every $k \geq 6$, the query complexity of C_k -freeness is $O(n^{1-1/\lfloor k/2 \rfloor})$ and $\Omega(n^{1/3})$. These results also leave open the question of which sparse classes – somewhere between bounded arboricity and planar graphs – and which values of r is C_r -freeness testable.

In order to identify a suitable notion of sparseness, we draw inspiration from the field of structural sparse graph theory and propose the r -admissibility measure of graphs as a parameter that governs the testability of H -freeness. 2-admissibility was originally defined in [9], where it was simply referred to as admissibility. The more general notion of r -admissibility² for natural values of r was first defined in [6]. Strictly speaking, r -admissibility is a family of measures where r governs how “deep” into the graph we look. We remark that, graph classes with bounded 1-admissibility are equivalent to graphs with bounded arboricity (both measures lie within a constant factor of each other). Many well-known sparse classes, like planar graphs, graphs of bounded genus, graphs excluding a fixed (topological) minor and graphs of bounded degree have bounded r -admissibility [12, 14, 6] meaning that the r -admissibility of any member of such a class can be bounded by a function which only depends on r and the class itself.

We show that C_4 -freeness is testable in graphs with bounded 2-admissibility, and that H -freeness, for every H of diameter 2, is also testable in graphs with bounded 2-admissibility and in particular C_5 -freeness is testable in graphs with bounded 2-admissibility. Using our techniques it is possible to show that C_6 and C_7 are testable in graphs with bounded 3-admissibility. This result will appear in the journal version of this paper that will also contain a lower bound which shows that, for every $r \geq 4$, C_r -freeness is not testable in

¹ Here sparse should be understood as having a linear number of edges or equivalently bounded average degree

² If you are interested in a more formal definition in this stage, we suggest that you proceed to Section 3, and read up to Proposition 3 before you proceed here.

graphs of bounded $(\lfloor r/2 \rfloor - 1)$ -admissibility. This implies that for every $r > 0$, there exists a graph H of diameter $r + 1$ such that H -freeness is not testable on graphs with bounded r -admissibility. The above leads us to conjecture that, for every $r \in \mathcal{N}$ and $t \leq 2r + 1$, C_t -freeness is testable in graphs with bounded r -admissibility and furthermore for every $r > 2$, H -freeness, for H of diameter r , is testable in graphs with bounded r -admissibility.

Techniques

All the lower bounds use Yao’s Minimax Principle [13] and the construction of suitable input instances. The graph constructions used for the lower bounds, unsurprisingly, are very similar to the ones used in [7] to prove a lower bound on testing C_4 -freeness in bounded arboricity graphs. We chose to provide our lower bounds here and not refer to [7], to demonstrate how they apply to testing C_r -freeness of graphs of bounded $\lfloor r/2 \rfloor - 1$ -admissibility, for natural values of r . This is not covered in [7]. In addition, the graphs provided in the proof also demonstrate that, for every natural value of r and large enough n , there are bounded r -admissibility graphs with n vertices some of which have degree $\Omega(\sqrt{n})$. It should be noted that these graphs are also “far” from being planar.

All upper bounds are based on the same algorithm and can be seen as a variation of the “standard” BFS (breadth first search) based testers for the bounded degree graph model. Many examples of “standard” BFS testers can be found in [8]. In such testers, initially a small subset of the vertices of the graph is selected uniformly at random (u.a.r.) and then a fixed depth BFS is performed (using oracle queries) from every vertex in the selected set. The tester presented here (Algorithm 1) differs from the “standard” testers at the BFS stage as follows: While the “standard” testers queries proceeds with the BFS by querying all neighbours of a vertex v , Algorithm 1 randomly selects size at most $\min\{\alpha, \deg_G(v)\}$ subset of $[\deg_G(v)]$, where α is a parameter provided to the algorithm and $\deg_G v$ is the degree of the vertex v in the input graph G , and queries the identity of the i th neighbour of the vertex, for every i in the selected set. We note Algorithm 1 behaves like the “standard” BFS based tester if the input graph has maximum degree α .

Algorithm 1 has a one-sided error, i.e. it only rejects if it detects an H -subgraph (a subgraph that is isomorphic to H), therefore, to prove its correctness, it is sufficient to show that, given oracle access to a bounded admissibility graph that is ϵ -far from being H -free, Algorithm 1 rejects with high probability.

H-subgraph

To prove the required rejection probability we show the following. Given an input graph G with bounded admissibility, we can remove edges from the graph in a process we refer to as *trimming* to obtain a new auxiliary graph \tilde{G} . We relate this graph \tilde{G} to G in two ways: first we show that if G is far from H -freeness, then so is \tilde{G} . Then we show that if there exists an H -subgraph in \tilde{G} , then this subgraph includes a vertex with low degree, such that if Algorithm 1 starts its search from this vertex in G (not in \tilde{G}), then with high probability it will discover an H -subgraph (though not necessarily the one we just referred to). The “high probability” is proved to be sufficiently large by using the properties of the graph \tilde{G} . Note that the construction of \tilde{G} is only a tool for this proof, it is not actually constructed by the testing algorithm.

Finally, we show that when \tilde{G} is far from being H -free, then there are many such low degree vertices in G . This implies that with sufficiently high probability Algorithm 1 initially selects such a vertex.

2 Preliminaries

$\mathcal{N}, \mathcal{R}, [k]$ We use \mathcal{N} for the set of integer numbers, \mathcal{R} for the set of real numbers, \mathcal{R}^+ for the set of positive real numbers. For an integer k , we use $[k]$ as a shorthand for the set $\{1, 2, \dots, k\}$. In this paper, all graphs are simple. For a graph G we use $V(G)$ and $E(G)$ to refer to its vertex- and edge-set, respectively. For a vertex $u \in V(G)$ we use the notation $N_G(u)$ to denote the set of all of u 's neighbours in G . We omit the subscript, when clear for context. The diameter of a graph is the maximum of the distance between u and v over all pairs of vertices u and v in $V(G)$.

Heavy We use the notation $\text{Heavy}_\alpha(G)$, where $\alpha \in \mathcal{N}$ to denote the set of vertices in $V(G)$ that have degree larger than α . We write Heavy_α when G is clear from context. In some places we will use the notation without initially stating the value of α , in those cases α is calculated further down when its concrete value is required.

xPy For sequences of vertices x_1, x_2, \dots, x_ℓ , in particular paths, we use notations like x_1Px_ℓ , x_1 or x_1P or Px_ℓ to indicate subpaths P that are part of the larger path. For example, the notation uPv for a path u, a, b, c, v would mean that P is the subpath a, b, c , whereas uP in the same context would mean that P is the subpaths a, b, c, v . All the paths in this paper are simple. Though paths here are undirected, we often treat them as directed by specifying a start and end vertex.

Property testing

We work only with properties of (p, r) -admissible graphs, which we formally define in the next section. At this stage it is enough to know that both p and r take integer values that are strictly positive and that a (p, r) -admissible graph of n vertices can have at most pn edges.

graph property, far, close A *graph property* (or simply *property* in this paper) is a class of graphs closed under isomorphism and we say that a graph has the property if it is contained in this class. The distance of a graph G from a property of (p, r) -admissible graphs is the minimum number of edges that must be removed/added to G in order to arrive at a (p, r) -admissible graph with the property. We say a graph is ϵ -far from a property of (p, r) -admissible graph, if the graph's distance to the property is at least ϵpn (an ϵ portion of the maximum number of edges possible in (p, r) -admissible graphs) and otherwise we say that it is ϵ -close to the property.

Property tester A property tester is a randomised algorithm that receives oracle access to a graph as part of its input. An oracle can answer the following queries for vertices $u, v \in V(G)$:

- the degree $\deg(v)$ of a vertex v (degree query),
- the i th neighbour of v in G (neighbour query),
- whether $\{u, v\}$ is an edge in G (adjacency query).

By combining these queries it can in particular reveal the whole neighbourhood of a vertex v using $1 + \deg(v)$ queries. The oracle returns the special symbol " \perp " when asked out of range neighbour queries, e.g. when asked to return the 10th neighbour of a vertex with less than 10 neighbours.

Formally, a *property tester* for a property P of (p, r) -admissible graphs, is a randomized algorithm \mathcal{A} that receives as input parameters $n \in \mathcal{N}$, $p, r \in [n]$, $\epsilon > 0$ and oracle access to a (p, r) -admissible graph G with n vertices. If the graph G is ϵ -far from P , then \mathcal{A} rejects with probability at least $2/3$. If the graph G is in P , then \mathcal{A} accepts with probability 1 (if the tester is *one-sided*) or with probability at least $2/3$ (if the tester is *two-sided*). The

complexity measure of a property tester is a function depending on n, p, r , and ϵ which bounds the maximum number of queries the tester makes on an input graph with those parameters.

In this paper, we study the property of (p, r) -admissible graphs that are H -free. That is, graphs that are (p, r) -admissible and do not have any H -subgraph (a subgraph isomorphic to H). In some cases, H may be a specific graph, for example, H may be a C_i (cycle of length i), for some $i \geq 3$ and we then refer to the problem as C_i -freeness.

In further sections, we use the term *knowledge graph* to refer to the graph that includes all the vertices, edges and anti-edges (learned from negative answers to neighbour queries) that the algorithm discovered via queries.

H-free,
H-subgraph

Knowledge graph

3 Graph degeneracy and admissibility, related notations and necessary lemmas

An *ordered graph* is a pair $\mathbb{G} = (G, \leq)$ where G is a graph and \leq is a total order relation on $V(G)$. We write $\leq_{\mathbb{G}}$ to denote the ordering of \mathbb{G} and extend this notation to derive relations $<_{\mathbb{G}}, >_{\mathbb{G}}, \geq_{\mathbb{G}}$. For simplicity we will call \mathbb{G} an *ordering of G* and we write $\pi(G)$ for the set of all possible orderings of G .

\mathbb{G} , *ordered graph,*
 $\pi(G)$

We use the same notations for graphs and ordered graphs, additionally we write $N_{\mathbb{G}}^-(u) := \{v \in N(u) \mid v <_{\mathbb{G}} u\}$ for the *before neighbourhood* and $N_{\mathbb{G}}^+(u) := \{v \in N(u) \mid v >_{\mathbb{G}} u\}$ for the *after neighbourhood* of a vertex $u \in \mathbb{G}$. We omit the graphs in the subscripts if clear from the context. We further use $\deg_{\mathbb{G}}^-(u) := |N_{\mathbb{G}}^-(u)|$ and $\deg_{\mathbb{G}}^+(u) := |N_{\mathbb{G}}^+(u)|$, as well as $\Delta^-(\mathbb{G}) := \max_{u \in \mathbb{G}} \deg_{\mathbb{G}}^-(u)$ and $\Delta^+(\mathbb{G}) := \max_{u \in \mathbb{G}} \deg_{\mathbb{G}}^+(u)$. We omit subscripts if clear from the context.

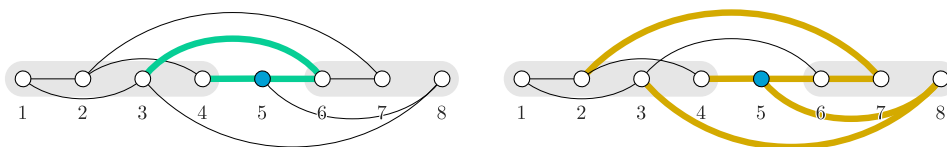
before and after
neighbourhood,
 $N_{\mathbb{G}}^-(u), N_{\mathbb{G}}^+(u),$
 $\Delta_{\mathbb{G}}^+(u), \Delta^-(\mathbb{G})$

► **Definition 1 (Degeneracy).** A graph G is γ -degenerate if there exists an ordering \mathbb{G} (a degeneracy ordering) such that $\Delta^-(\mathbb{G}) \leq \gamma$.

Degeneracy

In particular, for every vertex v in a γ -degenerate graph G and every degeneracy ordering \mathbb{G} of the graph it holds that $\deg_{\mathbb{G}}^-(v) \leq \gamma$. Consequently, the number of edges in a γ -degenerate graph is bounded by γn . A degeneracy ordering of a graph can be computed in time $O(n+m)$ and $O(\gamma n)$ for γ -degenerate graphs [11].

Admissibility



■ **Figure 1** On the left, the green highlighted edges form an example of a maximal 2-admissible path packing of size 2 that is rooted at the blue vertex in position 5. On the right, the yellow highlighted edges form an example of a maximal 3-admissible path packing of size 3 that is rooted at the blue vertex in position 5. In both packings the only vertex common to all the paths is their root. Also in both packing every path’s end vertex is smaller than the root and all the path’s internal vertices are all larger than the root.

Let $\mathbb{G} = (G, \leq)$ and $v \in V(G)$. A path vPx is r -admissible in \mathbb{G} if its length $\|vPx\| \leq r$, $x <_{\mathbb{G}} v$ and $\min P >_{\mathbb{G}} v$. That is, the path goes from v to x using only vertices w such that $w >_{\mathbb{G}} v$ and x satisfies $v >_{\mathbb{G}} x$.

r-admissible path

12:6 Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

Target For every integer $i > 0$ we let $\text{Target}_{\mathbb{G}}^i(v)$ be the set of all vertices $u \in V(G)$ such that $u <_{\mathbb{G}} v$ and u is reachable from v via an r -admissible path vPu of length exactly i . We omit the subscript \mathbb{G} when it is clear from context.

(r, \mathbb{G}) -admissible path packing An r -admissible path packing is a collection of paths $\{vP_i x_i\}_i$ with joint root v and the additional properties that every path $vP_i x_i$ is r -admissible and the subpaths $P_i x_i$ are all pairwise vertex-disjoint (cf. Figure 1). In particular, all endpoints $\{x_i\}_i$ are distinct. We write $\text{pp}_{\mathbb{G}}^r(v)$ to denote maximum size of any r -admissible path packing rooted at v .

► **Definition 2** (Admissibility). *The r -admissibility of an ordered graph \mathbb{G} , denoted $\text{adm}_r(\mathbb{G})$ and the admissibility of an unordered graph G , denoted $\text{adm}_r(G)$ are³*

$$\text{adm}_r(\mathbb{G}) := \max_{v \in \mathbb{G}} \text{pp}_{\mathbb{G}}^r(v) \quad \text{and} \quad \text{adm}_r(G) := \min_{\mathbb{G} \in \pi(G)} \text{adm}_r(\mathbb{G}).$$

Admissibility ordering If \mathbb{G} is an ordering of G such that $\text{adm}_r(\mathbb{G}) = \text{adm}_r(G)$, then we call \mathbb{G} an *admissibility ordering* of G . The 1-admissibility of a graph coincides with its degeneracy and therefore such orderings are easily computable in linear time. For $r \geq 2$ an optimal ordering can also be computed in linear time in n , albeit the machinery required is much more complicated, see [6].

(p, r) -admissible, adm_r -bounded We say that a graph G is (p, r) -admissible if $\text{adm}_r(G) = p$. Note that, by definition, if a graph G is (p, r) -admissible it is also (p, r') -admissible for all $r' \leq r$. We call a graph class adm_r -bounded if all of its members are (p, r) -admissible for some finite value p .

The following is a well-known result in the field of sparse graphs, we replicate it here using our notation for completeness:

► **Proposition 3.** *Let r and p be natural numbers, and $\mathbb{G} = (G, \leq)$ such that $\text{adm}_r(G) = p$, then for every $v \in V(G)$, and $h \in [r]$ it holds that $|\text{Target}_{\mathbb{G}}^h(v)| \leq p(p-1)^{h-1}$ and in particular $|N_{\mathbb{G}}^-(v)| \leq p$.*

Proof. Let v be an arbitrary vertex in $V(G)$, and $h \in [r]$. Note first that, by construction, $N_{\mathbb{G}}^-(v) = \text{Target}_{\mathbb{G}}^1(v)$, and hence we only need to prove the bound on $|\text{Target}_{\mathbb{G}}^h(v)|$.

For every $u \in \text{Target}_{\mathbb{G}}^h(v)$, let $vP_u u$ be an r -admissible path of length h ; such a path exists by the definition of $\text{Target}_{\mathbb{G}}^h(v)$. Let W be a subgraph of G defined as follows: $V(W)$ includes exactly the vertex v , all the vertices in $\text{Target}_{\mathbb{G}}^h(v)$ and all the vertices in P_u , for every $u \in \text{Target}_{\mathbb{G}}^h(v)$; and $E(W)$ includes every edge of G participating in a path $vP_u u$ for some $u \in \text{Target}_{\mathbb{G}}^h(v)$.

By construction, the set of vertices $\text{Target}_{\mathbb{G}}^h(v)$ is independent in W and for every $w \in \text{Target}_{\mathbb{G}}^h(v)$, $\deg_W(w) = 1$. Also by construction, the distance in W of v from any vertex in $\text{Target}_{\mathbb{G}}^h(v)$ is at most h . Hence, we can find a rooted tree T in W with the following properties: v is the root of T , the set $\text{Target}_{\mathbb{G}}^h(v)$ is the set of leaves of T and the depth of T is at most h .

We next show that the degree of every vertex in the tree is at most p . This implies that indeed $|\text{Target}_{\mathbb{G}}^h(v)| \leq p(p-1)^{h-1}$ and in particular $|N_{\mathbb{G}}^-(v)| \leq p$.

Let p_v be the degree of v in W . Since T is a tree, there are at most p_v edge disjoint paths from v to the leaves of T (the vertices in $\text{Target}_{\mathbb{G}}^h(v)$). These paths have length at most h , they share only the vertex v , and they correspond to r -admissible paths of \mathbb{G} . The last part holds since $u >_{\mathbb{G}} v$, for every internal vertex u (non-leaf or root vertex) of T and,

³ Note that some authors choose to define the admissibility as $1 + \max_{v \in \mathbb{G}} \text{pp}_{\mathbb{G}}^r(v)$ as this matches with some other, related measures.

for every $w \in \text{Target}_{\mathbb{G}}^h(v)$ (the leaves of T), it holds that $w <_{\mathbb{G}} v$. Therefore, these paths are an r -admissible packing of \mathbb{G} , and hence their number $p_v \leq p$, and in particular the degree of v in T is at most p .

We now show that this applies also to every internal vertex y of T . We notice that it also holds that $w <_{\mathbb{G}} y$, for every $w \in \text{Target}_{\mathbb{G}}^h(v) \cup \{v\}$. However, it is not necessarily the case that $u <_{\mathbb{G}} y$ when u is an internal vertex of T . This is resolved by noticing that for any $x \in \text{Target}_{\mathbb{G}}^h(v)$ and path yPx in T , if P has a vertex that is smaller than y , then instead of taking the path yPx , we take its shortest subpath $yP'x'$ such that $x' <_{\mathbb{G}} y$. Now, the same reasoning we used for v implies that the degree of y in T is at most p . ◀

4 Upper bounds strategy and the testing algorithm

In this section we present Algorithm 1 which is used for all upper bounds presented. Algorithm 1 receives as an input a graph H , a set of parameters including the parameter p and oracle access to a graph G . We note that for each upper bound shown in this paper, the parameters provided to Algorithm 1, in addition to p , are dependent on the graph H .

■ **Algorithm 1** The PBFS.

Input: $n \in \mathcal{N}$, $p \in [n]$, $\alpha, \tau \in \mathbb{R}$, fixed graph H and oracle access to a graph G

Set $S_0 = \emptyset$

Repeat $\lceil 4\alpha/(\epsilon p) \rceil$ **times**

 Add to S_0 an independently and u.a.r selected vertex from $V(G)$

for $i = 1, 2, \dots, \tau$ **do**

 Set $S_i = \emptyset$

for $v \in S_{i-1} \setminus \bigcup_{j=0}^{i-2} S_j$ **do**

 Query the degree of v

 Set $X = \emptyset$

Repeat $\lceil 2\alpha \rceil$ **times**

 Add to X an independently and u.a.r selected integer k from $[\text{deg}(v)]$

if $\text{deg}(v) \leq \lceil \alpha \rceil$ **then**

 Set $X = [\text{deg}(v)]$

for $k \in X$ **do**

 query the identity of the k 'th neighbour of v and add the answer to S_i

if the knowledge graph contains a H -subgraph **then**

 Reject

else

 Accept

Algorithm 1 can be seen as a variation of the “standard” BFS (breadth first search) based testers for properties of bounded degree graphs. In such testers, initially a small subset of the graph’s vertices is randomly selected and then a fixed depth BFS is performed (using oracle queries) from every vertex in the selected set.

Algorithm 1 differs from the “standard” testers, at the BFS stage: In the “standard” tester the search is expanded to all neighbours of a discovered vertex (until the fixed depth is reached). In contrast, Algorithm 1 only queries a subset of neighbours, specifically for

12:8 Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

a vertex v it randomly selects a size at most $\min\{\deg(v), \lceil \alpha \rceil\}$ subset $X \subseteq [\deg_G(v)]$ and then it queries the identity of every i th neighbour for $i \in X$. We call this type of search *PBFS* *pseudo-BFS* and refer to it with the acronym *PBFS*.

► **Lemma 4.** *On input $n \in \mathcal{N}$, $p \in [n]$, $\alpha, \tau \in \mathbb{R}$, $\epsilon > 0$, fixed graph H , the query complexity of Algorithm 1 is $O((\alpha/(\epsilon p))\alpha^\tau)$.*

Proof. The proof follows directly from the algorithm. ◀

Upper bound strategy

Algorithm 1 has a one-sided error, i.e. it only rejects if it discovers an H -subgraph. That is, once Algorithm 1 finished its last query, its knowledge graph has an H -subgraph). Therefore, to prove its correctness, it is sufficient to show that, with high constant probability, it rejects a bounded admissibility graph that is ϵ -far from being H -free.

To prove the required rejection probability we proceed as follows: given an input graph G with bounded admissibility, a new graph \tilde{G} is constructed by initially setting $\tilde{G} = G$, and then removing edges from \tilde{G} , in a process we refer to as *trimming*. We call \tilde{G} the auxiliary graph.

We relate the auxiliary graph \tilde{G} to G in two ways: first we show that if G is far from H -freeness, then so is \tilde{G} . Then we show that if there exists an H -subgraph in \tilde{G} , then this subgraph includes a vertex with low degree, such that assuming Algorithm 1 starts its search from this vertex in G (not in \tilde{G}), then with high probability it will discover an H -subgraph of G (though not necessarily the one we just referred to). The “high probability” is proved to be sufficiently large by using the properties of the graph \tilde{G} . Finally, we show that when \tilde{G} is far from being H -free, then there are many such low degree vertices in G . This implies that with sufficiently high probability Algorithm 1 initially selects such a vertex in the sample set S_0 .

The “trimming” process is problem dependent. The simplest case is for C_4 -freeness in adm_2 -bounded graphs and we provide some intuition here before the full formal treatment in the next section. Suppose that $v \in \text{Heavy}_\alpha$ and $u \in \text{Target}_\mathbb{C}^2(v)$. Suppose also that the set X of their common neighbours, not including those in Heavy_α , is small relative to the degree of v in G . In the trimming process we then remove all edges that are incident to v as well as some vertex in X . We can then show that (i) we did not remove too many edges and (ii) if u and v participate in a C_4 -subgraph (a subgraph isomorphic to a C_4) in \tilde{G} , then a large enough portion of the neighbours of v , are also neighbours of u and are not in Heavy_α . Therefore, if v is encountered in the first iteration of the external loop of Algorithm 1 (guaranteed to happen if S_0 has a neighbour of v that is not in Heavy_α), then with high probability the knowledge graph will include two edges vw_1 and vw_2 , where w_1 and w_2 are not in Heavy_α and are common neighbours of both u and v . Since the PBFS continues with these common neighbour and they are both not in Heavy_α , the edges uw_1 and uw_2 will also appear in the knowledge graph. Thus, the knowledge graph contains a C_4 -subgraph. Similar – but more involved – ideas work for H -freeness, when H has diameter 2 and for C_6 -freeness and C_7 -freeness.

It is important to note that proving the above properties (i) and (ii) hold relies on the graph G being adm_r -bounded (for some problem-dependent value of r), and we make extensive use of Proposition 3.

5 Testing C_4 -freeness in adm_2 -bounded graphs

In this section we fix some $\epsilon > 0$, an integer $p > 0$, $\alpha = 32p^2/\epsilon$. The input graph G is $(p, 2)$ -admissible and we let \mathbb{G} be an ordering of G with $\text{adm}_2(\mathbb{G}) = p$.

Trimming. In the trimming procedure we construct \tilde{G} from G . We begin with $\tilde{G} = G$ and then remove edges from $E(\tilde{G})$ as follows:

1. For every $v \in \text{Heavy}_\alpha(G)$, and $u \in N^-(v)$, the edge uv is removed from $E(\tilde{G})$.
2. For every $v \in V(G)$ and $u \in \text{Target}_{\mathbb{G}}^2(v)$, if $|N_{\tilde{G}}(u) \cap N_{\tilde{G}}(v)| \leq \deg_G(v)/(\alpha/2)$, then for every $w \in N_{\tilde{G}}(u) \cap N_{\tilde{G}}(v)$, the edge vw is removed from $E(\tilde{G})$.
3. The previous step is repeated until it does not result in the removal of any edges from $E(\tilde{G})$.

We first show that this trimming procedure preserves farness:

► **Lemma 5.** *If G is ϵ -far from being C_4 -free, then \tilde{G} is $\epsilon/2$ -far from being C_4 -free.*

Proof. Initially $E(\tilde{G}) = E(G)$ and then edges are removed from $E(\tilde{G})$ in Steps 1 and 2 of the trimming. We next show that, in Step 1 of the trimming, at most $\epsilon|E(G)|/4$ edges are removed from $E(\tilde{G})$, and that the same applies to Step 2 of the trimming. This implies the lemma.

In Step 1 of the trimming, for every $v \in \text{Heavy}_\alpha(G)$ we remove $|N_{\tilde{G}}^-(v)|$ edges. Thus, the total number of edges removed in this step is at most $\sum_{v \in \text{Heavy}_\alpha(G)} |N_{\tilde{G}}^-(v)|$. By Proposition 3, for every $u \in V(G)$, it holds that $|N_{\tilde{G}}^-(u)| \leq p$. Hence, the preceding sum is at most $\sum_{v \in \text{Heavy}_\alpha(G)} p = p \cdot |\text{Heavy}_\alpha(G)|$ and

$$p \cdot |\text{Heavy}_\alpha(G)| = \frac{p \cdot |\text{Heavy}_\alpha(G)|}{|E(G)|} |E(G)| \leq \frac{p \cdot |\text{Heavy}_\alpha(G)|}{\alpha |\text{Heavy}_\alpha(G)|/2} |E(G)| = \frac{2p}{\alpha} |E(G)| < \frac{\epsilon pn}{4},$$

where the first inequality follows since $|E(G)| \geq \alpha |\text{Heavy}_\alpha(G)|/2$ (the sum of degrees is twice the number of edges), and the last equality holds because $\alpha = 32p^2/\epsilon$.

In Step 2 of the trimming, for every $v \in V(G)$ and $u \in \text{Target}_{\mathbb{G}}^2(v)$ at most $\deg_G(v)/(\alpha/2)$ edges are removed. Thus, in this step, at most $\sum_{v \in V(G)} \sum_{u \in \text{Target}_{\mathbb{G}}^2(v)} \deg_G(v)/(\alpha/2)$ edges are removed. By Proposition 3, for every $u \in V(G)$, it holds that $|\text{Target}_{\mathbb{G}}^2(u)| < p^2$. Hence, the preceding sum is strictly less than $\sum_{v \in V(G)} p^2 \deg_G(v)/(\alpha/2) = (\epsilon/8) \sum_{v \in V(G)} \deg_G(v) \leq \epsilon pn/4$, where the first equality follows because $\alpha = 32p^2/\epsilon$. ◀

► **Lemma 6.** *Every C_4 -subgraph C of \tilde{G} that has more than one vertex from $\text{Heavy}_\alpha(G)$, has exactly two vertices w_1 and w_2 from $\text{Heavy}_\alpha(G)$, all the other two vertices of C are neighbours of both w_1 and w_2 and, there exists $i \in \{1, 2\}$, such that $|N_G(w_1) \cap N_G(w_2)| \geq \deg_G(w_i)/(\alpha/2)$.*

Proof. Let C be a C_4 -subgraph of \tilde{G} such that $|V(C) \cap \text{Heavy}_\alpha(G)| > 1$ and w_1 and w_2 be two vertices in $V(C) \cap \text{Heavy}_\alpha(G)$. Assume without loss of generality that $w_1 >_{\mathbb{G}} w_2$, and otherwise rename the vertices accordingly.

According to Step 1 of the trimming, $w_1 w_2 \notin E(\tilde{G})$ and hence w_1 and w_2 are not adjacent in C . By the same reasoning w_1 and w_2 are the only vertices of $\text{Heavy}_\alpha(G)$ in $V(C)$. We conclude that C consists of the two vertices w_1 and w_2 in $\text{Heavy}_\alpha(G)$ and two vertices that are not in $\text{Heavy}_\alpha(G)$ and are neighbours of both w_1 and w_2 .

Let $v \in V(C) \setminus \text{Heavy}_\alpha(G)$ and $Y = N_{\tilde{G}}(w_1) \cap N_{\tilde{G}}(w_2)$. By the same reasoning as before we may conclude that $v >_{\mathbb{G}} w_1$. Thus, as $w_1 >_{\mathbb{G}} w_2$, $w_2 \in \text{Target}_{\mathbb{G}}^2(w_1)$. Consequently, by Step 1 of the trimming, $|Y| > \deg_G(w_1)/(\alpha/2)$, since otherwise $Y = \emptyset$ in contradiction to $v \in Y$

12:10 Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

(because v is adjacent to both w_1 and w_2). As \tilde{G} is a subgraph of G and $Y \cap \text{Heavy}_\alpha(G) = \emptyset$ (because of Step 1 of the trimming), we can conclude that also in the graph G it holds that $|(N_G(w_1) \cap N_G(w_2)) \setminus \text{Heavy}_\alpha(G)| > \deg_G(w_1)/(\alpha/2)$. ◀

The next two lemmas show that if the set S_0 selected by Algorithm 1 contains specific types of vertices from $V(G) \setminus \text{Heavy}_\alpha(G)$, then it rejects with sufficient probability to prove the main Theorem of this section.

► **Lemma 7.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters p , $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$ and let S_0 be the set selected by the algorithm in the first step. Assume there exists a vertex $v \in S_0 \setminus \text{Heavy}_\alpha(G)$ such that v is in a C_4 -subgraph C of \tilde{G} and C satisfies $|V(C) \cap \text{Heavy}_\alpha(G)| \leq 1$. Then Algorithm 1 rejects with probability 1.*

Proof. Let v be as in the statement of the lemma. Since C is a C_4 -subgraph that includes, together with v , three vertices from $V(C) \setminus \text{Heavy}_\alpha(G)$ in G , it follows that C includes a path P of length 2 that consists of v and two other vertices from $V(C) \setminus \text{Heavy}_\alpha(G)$.

Now, as Algorithm 1 will execute a PBFS of depth 3 (we set $\tau = 3$) it is guaranteed that the knowledge graph constructed by Algorithm 1 will include *all* vertices of P and *all* edges incident to these vertices. In particular, the knowledge graph eventually has C as a subgraph with probability 1 and the claim follows. ◀

► **Lemma 8.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters, p , $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$. Let S_0 be the set selected by the algorithm in the first step. Assume there exists a vertex $v \in S_0 \setminus \text{Heavy}_\alpha(G)$ such that v is in a C_4 -subgraph C of \tilde{G} and C satisfies $|V(C) \cap \text{Heavy}_\alpha(G)| \geq 2$. Then Algorithm 1 rejects with probability at least $5/6$.*

Proof. Let v be as in the statement of the lemma. By Lemma 6, C has exactly two vertices w_1 and w_2 from $\text{Heavy}_\alpha(G)$, the other two vertices of C are neighbours of both w_1 and w_2 and are not in $\text{Heavy}_\alpha(G)$. Since $\deg_G(v) < \alpha$, Algorithm 1 queries all of v 's edges, in the first iteration of the PBFS. Thus after the first iteration of the PBFS the knowledge graph already contains the edges vw_1 and vw_2 .

We show next that with with probability at least $5/6$ Algorithm 1 queries a neighbour $u \neq v$ of w_1 that is not in $\text{Heavy}_\alpha(G)$ and is also a neighbour of w_2 , so after the second iteration of the PBFS the knowledge graph also contains the edge uw_1 . We note that this implies the lemma, because in the third iteration of the PBFS (this is the last iteration, since $\tau = 3$) Algorithm 1 will discover all the edges incident to u , since $\deg_G(u) \leq \alpha$, and in particular it will discover the edge uw_2 , which implies that after the end of the PBFS from v the knowledge graph will contain a C_4 -subgraph and hence Algorithm 1 will reject.

Let $Y = (N_G(w_1) \cap N_G(w_2)) \setminus \text{Heavy}_\alpha(G)$. According to Lemma 6, there exists $i \in \{1, 2\}$ such that $|Y| \geq \deg_G(w_i)/(\alpha/2)$. Without loss of generality assume that $i = 1$ and otherwise rename w_1 and w_2 accordingly. Since $\deg_G(w_1) \geq \alpha$, we can conclude that $|Y| \geq 2$ and hence we can assume that at least $1/2$ of the vertices in Y are not v . Thus, $|Y \setminus \{v\}| \geq \deg_G(w_1)/(2\alpha/2)$. Algorithm 1 selects a vertex from $N(w_1)$ independently and u.a.r. 2α times and hence the probability that none of them are from $|Y \setminus \{v\}|$ is at most $(1 - 1/\alpha)^{2\alpha} < e^{-2} < 1/6$. So with probability at least $5/6$ Algorithm 1 finds a vertex u in $Y \setminus \{v\}$ and the claim follows. ◀

► **Theorem 9.** *Suppose that Algorithm 1 is executed with oracle access to G and parameters, p , $H = C_4$, $\alpha = 32p^2/\epsilon$, $\tau = 3$, then (i) if G is C_4 -free, then Algorithm 1 accepts with probability 1; and (ii) if G is ϵ -far from being C_4 -free, then Algorithm 1 rejects with probability at least $2/3$. Algorithm 1 uses at most $O(p^7/\epsilon^5)$ queries.*

Proof. The query complexity of the algorithm follows from Lemma 4 and the values of α and τ .

If G is C_4 -free, then the knowledge graph constructed by Algorithm 1 will not have a C_4 -subgraph and hence Algorithm 1 accepts with probability 1. So assume in the sequel that G is ϵ -far from C_4 -freeness.

Let U be the set of all vertices in $V(G) \setminus \text{Heavy}_\alpha(G)$ that participate in a C_4 -subgraph of \tilde{G} . Since the degree of all vertices in $V(G) \setminus \text{Heavy}_\alpha(G)$ is less than α , if we remove every edge that is incident to a vertex from U , then the total number of edges we removed is bounded above by $\alpha|U|$ and the resulting graph does not have a C_4 -subgraph. Now, by Lemma 5, \tilde{G} is $\epsilon/2$ -far from C_4 -freeness it must be the case that $\alpha|U| \geq \epsilon np/2$. Consequently, $|U| \geq \epsilon np/(2\alpha)$.

Algorithm 1 selects $\lceil 4\alpha/(\epsilon p) \rceil$ vertices u.a.r. for the set S_0 . The probability that none of them are from U is at most $(1 - \epsilon p/(2\alpha))^{\lceil 4\alpha/(\epsilon p) \rceil} < e^{-2} < 1/6$. So with probability at least $5/6$, the set S_0 includes a vertex $v \in V(G) \setminus \text{Heavy}_\alpha(G)$ that participates in a C_4 -subgraph of \tilde{G} .

Let C be a C_4 -subgraph of \tilde{G} that includes a vertex $v \in V(G) \setminus \text{Heavy}_\alpha(G)$. One of two cases applies to C : Either (a) C includes at most one vertex from $\text{Heavy}_\alpha(G)$ or (b) C includes more than one vertex from $\text{Heavy}_\alpha(G)$. We now show that given $v \in S_0$, with probability at least $5/6$, the knowledge graph of Algorithm 1 eventually has a C_4 -subgraph. By using the union bound we can conclude that Algorithm 1 rejects with probability at least $2/3$.

According to Lemma 7, if case (a) occurs, then Algorithm 1 rejects with probability 1. According to Lemma 8, if case (b) occurs, then Algorithm 1 rejects with probability at least $5/6$. We conclude that Algorithm 1 rejects with probability at least $(5/6)^2 > 2/3$, as claimed. \blacktriangleleft

6 Testing H -freeness in adm_2 -bounded graphs when H has diameter 2

In this section we fix $\epsilon > 0$, $p \in \mathcal{N}$, $\alpha = 3|V(H)|\lceil \epsilon^{-1}2^{2|V(H)|+4p^2 \log p} \rceil$. As before, G is an arbitrary $(p, 2)$ -admissible graph and \mathbb{G} is an ordering of G with $\text{adm}_2(\mathbb{G}) = p$. Finally, H is an arbitrary diameter 2 graph.

The trimming process in this section depends on the structure of H and requires an extra construct (**H**). Let us first provide some intuition why this is required.

Suppose that \tilde{H} is an H -subgraph of \tilde{G} and h is the largest vertex in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$. We show below that the number of vertices like h in \tilde{G} is large enough to ensure that with high enough probability one of them will be in the set S_0 selected by Algorithm 1.

Ideally, in \tilde{H} , every vertex $u \in V(\tilde{H})$, is reachable from h via vertices not from $\text{Heavy}_\alpha(G)$. This is an ideal case, because of the following. The depth of the PBFs used is $|V(H)|$. Thus, as the PBFs queries all the neighbours of vertices $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, all these vertices will be discovered and also all edges incident to them. This means that the algorithm discovers all the edges of \tilde{H} except those that are incident on two vertices from $V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$. The trimming we use ensures that there are no edges between heavy vertices, so in particular, this latter case cannot occur and Algorithm 1 will discover \tilde{H} .

If we do not find ourselves in the ideal case, \tilde{H} has a separator \tilde{K} that consists only of vertices from $\text{Heavy}_\alpha(G)$. By similar reasoning to the ideal case, if h is included in the set S_0 that Algorithm 1 selects, it discovers all the vertices in \tilde{H} that can be reached from h via vertices in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, this includes all the vertices of \tilde{K} . Let us denote the subgraph found so far by \tilde{H}_1 . The algorithm now still needs to find the remaining vertices of \tilde{H} that are “behind” the separator, let us call denote this subgraph by \tilde{H}_2 .

12:12 Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

The problem here is that the vertices \tilde{K} of on the boundary between \tilde{H}_1 and \tilde{H}_2 have a high degree, therefore the probability to discover \tilde{H}_2 may be arbitrarily small. We therefore design a trimming process that will ensure that there are many “useful” \tilde{H}_2 -subgraphs \tilde{H}'_2 isomorphic to \tilde{H}_2 in \tilde{G} that can serve to complete \tilde{H}_1 into a graph isomorphic to H . Specifically, a subgraph \tilde{H}'_2 is useful if $\tilde{K} \subset V(\tilde{H}'_2)$ and there exists an isomorphism from \tilde{H}'_2 to \tilde{H}_2 which maps every vertex of \tilde{K} to itself.

Let Q' be a maximum family of vertex disjoint useful \tilde{H}_2 -subgraphs with respect to the fixed graph \tilde{H}_1 . If Q' is small then we can remove them all simply by removing all edges between members of Q' and \tilde{K} . If this is not possible, Q' must be sufficiently large and the PBFS can discover one of its members with sufficiently high probability.

The above is a simplification, since \tilde{H} might disconnect into more than two components when removing the separator \tilde{K} . In this case we also need to ensure that the members of Q' are sufficiently disjoint.

Finally, in the preceding discussion we fixed a single \tilde{H} with separator \tilde{K} , however \tilde{H} is of course not known in advance. We therefore enumerate all possible sets $K \subset H$ that can take the role of \tilde{K} (not that, by Proposition 3, we only need to look at sets of size $< p^2$). Since K consists only of heavy vertices and our trimming removes edges between heavy vertices, we can further assume K to be independent.

Kernel, \mathbf{H} ► **Definition 10 (Kernel and \mathbf{H}).** A kernel K of the graph H is an independent subset of $V(H)$ of size less than p^2 for which H has two subgraphs H_1 and H_2 such that

1. both H_1 and H_2 are induced and connected,
2. $V(H_1) \cap V(H_2) = K$,
3. $V(H_1) \cup V(H_2) = V(H)$,
4. every edge of H that is incident to a vertex from $V(H_1)$ and $V(H_2)$, is incident to an edge from K (K is a vertex separator in H),
5. the induced subgraph of H_1 on the vertices $V(H_1) \setminus K$ is connected.

We define \mathbf{H} to be the family of ordered pairs (H_2, K) over all kernels K of H .

► **Definition 11 (Sibling subgraphs).** Let L be a subset of $V(\tilde{G})$, and R_1 and R_2 be two subgraphs of \tilde{G} , both including the set of vertices L . We say that R_1 and R_2 are siblings if $V(R_1) \cap V(R_2) = L$ and there exists an isomorphism ϕ from R_1 to R_2 , such that, for every $\ell \in L$, $\phi(\ell) = \ell$.

We say a set of graphs is a set of sibling graphs by L , if every pair of graphs in the set are siblings by L .

Trimming **Trimming.** In the trimming procedure we construct \tilde{G} from G . We begin with $G = \tilde{G}$ and then remove edges from $E(\tilde{G})$ as follows:

1. For every $v \in \text{Heavy}_\alpha(G)$, and $u \in N^-(v)$, the edge uv is removed from $E(\tilde{G})$.
2. For every $v \in \text{Heavy}_\alpha(G)$, $(H_2, K) \in \mathbf{H}$ and size $|K|$ subset $M \subseteq \text{Target}_{\mathbb{G}}^2(v)$, if a maximum set of vertex disjoint H_2 -subgraphs of \tilde{G} that are siblings by M , and are isomorphic to H_2 so that the vertices of M are mapped to the vertices of K , has size at most $2|V(H)| \deg_G(v)/\alpha$, then, for every vertex w in a subgraph of this set, we remove every edge incident to w and a vertex in $\text{Target}_{\mathbb{G}}^2(v)$.
3. The previous steps are repeated until it does not result in the removal of any edges from $E(\tilde{G})$.

► **Proposition 12.** For every $v \in \text{Heavy}_\alpha(G)$, $(H_2, K) \in \mathbf{H}$ and size $|K|$ subset $M \subseteq \text{Target}_{\mathbb{G}}^2(v)$, if in some execution of Step 2, for a maximum family of vertex disjoint H_2 -subgraphs that are siblings by M , and are isomorphic to H_2 so that the vertices of M are

mapped to the vertices of K , the following holds: for every vertex w in a subgraph of this set, we remove every edge incident to w and a vertex in $\text{Target}_{\mathbb{G}}^2(v)$, then after the specific execution of Step 2, there does not exist any H_2 -subgraph that is a sibling by M of a subgraph in the maximum family.

Proof. Recall that H has diameter 2, so if it has a separator K , then all the vertices of H that are not in the separator must be adjacent to some vertex in the separator. Therefore in Step 2, when for every w in a subgraph of the set all edges between w and $\text{Target}_{\mathbb{G}}^2(v)$ are removed, we have that every such vertex w cannot participate in any subgraph like the one in the set. Hence, as the family is of maximum size the proposition follows. ◀

► **Lemma 13.** *If G is ϵ -far from being H -free, then \tilde{G} is $\epsilon/2$ -far from being H -free.*

Proof. Note that initially $E(\tilde{G}) = E(G)$ and then edges are removed from $E(\tilde{G})$ in Steps (1) and (2) of the trimming. Given that the value of α here is larger than the value used in Section 5, and that Step 1 of the trimming here is the same as Step 1 in Section 5, by the same considerations as in Lemma 5, at most $\epsilon pn/4$ edges are removed from $E(\tilde{G})$, at Step 1 of the trimming. So, to complete the proof, we only need to show that in Step 1 of the trimming also at most $\epsilon pn/4$ edges are removed from $E(\tilde{G})$.

According to Step 2 of the trimming, the total number of edges removed from $E(\tilde{G})$, for every vertex in $\text{Heavy}_{\alpha}(G)$, is bounded above by the product of the following values:

- $|\mathbf{H}|$, and
- $(p^2)!$, which is an upper bound on the number of options to choose (while considering order) a size $|K|$ subset of $\text{Target}_{\mathbb{G}}^2(v)$ (where, for every $v \in V(G)$, by Proposition 3, $|\text{Target}_{\mathbb{G}}^2(v)| < p^2$), and
- $2|V(H)| \deg_G(v)/\alpha$, the threshold for edge removal for number of subgraphs in a maximum set of sibling subgraphs, and
- $p^2|V(H)|$, the number of edges incident to a vertex from $\text{Target}_{\mathbb{G}}^2(v)$ and vertices in a subgraph of a maximum set of sibling subgraphs.

The size of \mathbf{H} is the number of subsets $V(H)$ with size less than p^2 . Hence, $|\mathbf{H}| < 2^{|V(H)|}$. So, the total number of edges removed from $E(\tilde{G})$ is at most

$$\sum_{v \in \text{Heavy}_{\alpha}(G)} 2^{|V(H)|} \cdot (p^2)! \cdot (2|V(H)| \deg_G(v)/\alpha) \cdot p^2|V(H)| < \frac{\epsilon}{8} \sum_{v \in \text{Heavy}_{\alpha}(G)} \deg_G(v) \leq \frac{\epsilon pn}{4},$$

where the first equality follows because $\alpha = 3|V(H)| \lceil \epsilon^{-1} 2^{2|V(H)| + 4p^2 \log p} \rceil$. ◀

► **Proposition 14.** *Let \tilde{H} be a H -subgraph of \tilde{G} . $\text{Heavy}_{\alpha}(G)$ is an independent set in \tilde{G} , the largest vertex in \tilde{H} is not in $\text{Heavy}_{\alpha}(G)$ and if \tilde{H} has a separator $U \subseteq \text{Heavy}_{\alpha}(G)$, then $U = \text{Heavy}_{\alpha}(G) \cap V(\tilde{H})$ and U is the only separator in \tilde{H} consisting of only vertices from $\text{Heavy}_{\alpha}(G)$.*

Proof. According to Step 1 of the trimming, for every $v \in \text{Heavy}_{\alpha}(G)$, if $vu \in E(\tilde{G})$, then $u >_{\mathbb{G}} v$. Thus, as one vertex is greater than the other for every pair of vertices in $\text{Heavy}_{\alpha}(G)$ there cannot be an edge in \tilde{G} that is incident to both. Therefore, $\text{Heavy}_{\alpha}(G)$ is an independent set. By the same reasoning, the largest vertex in an H -subgraph of \tilde{G} is not in $\text{Heavy}_{\alpha}(G)$, since it is adjacent to vertices smaller than it (H is connected, because it has diameter 2).

Finally, for the last part the claim, in the proof of Proposition 12, we noticed that for every separator of H , every vertex of H that is not in the separator must be adjacent to some vertex in the separator. The same applies to \tilde{H} . So, if a separator of \tilde{H} is a subset

12:14 Results on H -Freeness Testing in Graphs of Bounded r -Admissibility

of $\text{Heavy}_\alpha(G)$, then every vertex that is adjacent to some vertex in the separator is not in $\text{Heavy}_\alpha(G)$ and in particular all the vertices of \tilde{H} that are not in the separator. This also implies that U is the only separator in \tilde{H} consisting of only vertices from $\text{Heavy}_\alpha(G)$. ◀

► **Lemma 15.** *Let \tilde{H} be an H -subgraph of \tilde{G} and $U = V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$ and suppose that U is a separator of \tilde{H} , then $|U| < p^2$.*

Proof. Let v be the largest vertex in U . Since H is of diameter 2, it has a common neighbour with every vertex in U . By Step 1, v does not have any neighbours smaller than it and hence together with the previous we can conclude that $U \setminus \{v\} \subseteq \text{Target}_{\mathbb{G}}^2(u)$. By Proposition 3, $\text{Target}_{\mathbb{G}}^2(u) \leq p(p-1)$ and the lemma follows. ◀

► **Theorem 16.** *If Algorithm 1 is executed with oracle access to G and parameters, p , H , $\alpha = 3|V(H)|\lceil \epsilon^{-1}2^{2|V(H)|+4p^2 \log p} \rceil$, $\tau = |H|$, then (i) if G is H -free, then Algorithm 1 accepts with probability 1; and (ii) if G is ϵ -far from being H -free, then Algorithm 1 rejects with probability at least $2/3$. Algorithm 1 uses at most $O(2^{(|V(H)|+1)(2|V(H)|+4p^2 \log p)})$ queries.*

Proof. The query complexity of the algorithm follows from Lemma 4 and the values of α and τ .

If G is H -free, then knowledge graph of Algorithm 1, will never have an H -subgraph and hence, Algorithm 1 accepts with probability 1. So, from here on in this proof, assume that G , is ϵ -far from H -freeness.

Let U be the set of all vertices in $V(G)$ that are the largest vertices in a H -subgraph of \tilde{G} . By Proposition 14, $U \cap \text{Heavy}_\alpha(G) = \emptyset$ and therefore, by the same reasoning as in Theorem 9, it holds that $|U| > \epsilon np / (2\alpha)$ and, with probability at least $5/6$ that $S_0 \cap U \neq \emptyset$. So, assume that v is a vertex in S_0 that is the largest vertex in a H -subgraph \tilde{H} of \tilde{G} .

One of two cases applies to \tilde{H} : (a) every vertex $u \in V(\tilde{H})$, is reachable from v via the vertices in $V(\tilde{H}) \setminus \text{Heavy}_\alpha(G)$, and (b) there exists a set M such that $M = V(\tilde{H}) \cap \text{Heavy}_\alpha(G)$ that is a separator of \tilde{H} .

If case (a) applies, then as we already described previously in this section, with probability 1, the knowledge graph of Algorithm 1 will eventually have an H -subgraph. Thus, Algorithm 1 will reject with probability 1. So, from here on we assume that case (b) applies.

So, assume that \tilde{H} has a separator M consisting only of vertices from $\text{Heavy}_\alpha(G)$ (by Proposition 14, this separator includes all the vertices of $\text{Heavy}_\alpha(G) \cap V(\tilde{H})$). Since $v \notin \text{Heavy}_\alpha(G)$, the diameter of H is 2 and $\text{Heavy}_\alpha(G)$ is an independent set, for every vertex in M , v is either its neighbour or shares a neighbour x with it, where $x \notin \text{Heavy}_\alpha(G)$. This ensures that, with probability 1, after two steps of the PBFS all the vertices in M are discovered. By similar consideration to the previous case, with probability 1, all vertices reachable from v via vertices not in $\text{Heavy}_\alpha(G)$ are added to the knowledge graph. For every one these vertices that is not in $\text{Heavy}_\alpha(G)$ also all the edges incident on them are also in the knowledge graph. This implies that all the edges between the vertices of M and the vertices added to the knowledge graph as described are also in the knowledge graph.

It remains to show that with sufficiently high probability, the edges required to complete the above to H -subgraph are included in the knowledge graph. Let ℓ be the largest vertex in M . Let \tilde{H}' be an induced subgraph of \tilde{H} that includes all the vertices of M and all other vertices of \tilde{H} that are separated from v by M . By Step 2, of the trimming we are guaranteed that there exists a family \mathcal{Q} , of size greater than $2|V(H)| \deg_G(v) / \alpha$, that consists of vertex disjoint subgraphs of \tilde{G} , where every graph in the family is either \tilde{H}' or its sibling by M .

Let be the graph \tilde{H}^* induced by \tilde{H}' on $V(\tilde{H}') \setminus M$ If we were guaranteed that \tilde{H}^* is connected, then we would only need to show that with high probability Algorithm 1 will discover an edge incident to ℓ and a vertex x of this subgraph (or similar subgraph of one of

its siblings in M , that does not share any vertices with the part of the H -subgraph already discovered). This holds because, every vertex in \tilde{H}^* is not in $\text{Heavy}_\alpha(G)$, and there are enough steps of the PBFS so that the PBFS reaches all the vertices in \tilde{H}^* and discovers all the edges adjacent to them (the PBFS has $|V(H)|$ steps, and the vertices on a shortest path from ℓ to x are not in \tilde{H}^*), thus the PBFS will discover all the vertices of \tilde{H}^* and the edges incident on them.

However, the above guarantee does not hold. So \tilde{H}^* may contain almost $|V(H)|$ connected components. Regardless, if for every one of these connected components, the knowledge graph has it as a subgraph or has its isomorphic equivalent in one of the subgraphs in \mathcal{Q} , and if none of the isomorphic equivalent shares a vertex with other vertices of \tilde{H}' in the knowledge graph, the knowledge graph has an H -subgraph.

For each connected component, there are at least $3|V(H)| \deg_G(\ell)/\alpha \geq 3|V(H)|\alpha/\alpha = 3|V(H)|$ vertex disjoint copies (where the inequality follows because $\ell \in \text{Heavy}_\alpha(G)$). So, at least two thirds of the copies of such connected component do not include any other vertices from \tilde{H} .

The probability of not discovering one such component is $(1 - 4|V(H)|/(3\alpha))^{2\alpha} \leq e^{-|V(H)|/6} < (6|V(H)|)^{-1}$. By the union bound, with probability at least $5/6$, the knowledge graph has all the subgraphs required so that it has a H -subgraph. Thus, the proof is complete. \blacktriangleleft

7 Testing C_6 and C_7 -freeness in adm_3 -bounded graphs

The proof of the following theorem will appear in the journal version of this paper.

► **Theorem 17** (\star). *If Algorithm 1 is executed with oracle access to G and parameters, $H = C_7$, $\alpha = \lceil 2^{12}p^4/\epsilon \rceil$, $\tau = 7$, then (i) if G is C_7 -free, then Algorithm 1 accepts with probability 1; and (ii) if G is ϵ -far from being C_7 -free, then Algorithm 1 rejects with probability at least $2/3$. Algorithm 1 uses at most $O(p^{27}/\epsilon^8)$ queries.*

8 Lower bounds for testing C_r -freeness for $r \geq 4$

The proof of the following theorem will appear in the journal version of this paper.

► **Theorem 18** (\star). *For every integer $r \geq 4$ and sufficiently large integer n , every two-sided Property-Tester for the C_r -freeness, has query complexity $\Omega(n^{1/4})$, on $(2\lfloor r/2 \rfloor - 1)$ input graphs of size n .*

References

- 1 Noga Alon, Richard A. Duke, Hanno Lefmann, Vojtech Rödl, and Raphael Yuster. The algorithmic aspects of the regularity lemma. *J. Algorithms*, 16(1):80–109, 1994. doi:10.1006/JAGM.1994.1005.
- 2 Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Comb.*, 20(4):451–476, 2000. doi:10.1007/S004930070001.
- 3 Noga Alon, Tali Kaufman, Michael Krivelevich, and Dana Ron. Testing triangle-freeness in general graphs. *SIAM Journal on Discrete Mathematics*, 22(2):786–819, 2008. doi:10.1137/07067917X.
- 4 Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014. doi:10.1002/RSA.20462.

- 5 Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (it's all about forbidden subgraphs). In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1525–1548. IEEE, 2019. doi:10.1109/FOCS.2019.00089.
- 6 Zdeněk Dvořák. Constant-factor approximation of the domination number in sparse graphs. *European Journal of Combinatorics*, 34(5):833–840, July 2013. doi:10.1016/j.ejc.2012.12.004.
- 7 Talya Eden, Reut Levi, and Dana Ron. Testing c_k -freeness in bounded-arboricity graphs. 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 60:1–60:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.60.
- 8 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 406–415, 1997. doi:10.1145/258533.258627.
- 9 H. A. Kierstead and W. T. Trotter. Planar graph coloring with an uncooperative partner. *Journal of Graph Theory*, 18(6):569–584, October 1994. doi:10.1002/jgt.3190180605.
- 10 Reut Levi. Testing triangle freeness in the general model in graphs with arboricity $O(\sqrt{n})$. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 93:1–93:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.93.
- 11 David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983. doi:10.1145/2402.322385.
- 12 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.
- 13 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE Computer Society, 1977.
- 14 Xuding Zhu. Colouring graphs with bounded generalized colouring number. *Discrete Mathematics*, 309(18):5562–5568, 2009. doi:10.1016/J.DISC.2008.03.024.

Hyperbolic Random Graphs: Clique Number and Degeneracy with Implications for Colouring

Samuel Baguley  

Hasso Plattner Institute, University of Potsdam, Germany

Yannic Maus  

TU Graz, Austria

Janosch Ruff  

Hasso Plattner Institute, University of Potsdam, Germany

George Skretas  

Hasso Plattner Institute, University of Potsdam, Germany

Abstract

Hyperbolic random graphs inherit many properties that are present in real-world networks. The hyperbolic geometry imposes a scale-free network with a strong clustering coefficient. Other properties like a giant component, the small world phenomena and others follow. This motivates the design of simple algorithms for hyperbolic random graphs.

In this paper we consider threshold hyperbolic random graphs (HRGs). Greedy heuristics are commonly used in practice as they deliver a good approximations to the optimal solution even though their theoretical analysis would suggest otherwise. A typical example for HRGs are degeneracy-based greedy algorithms [Bläsius, Fischbeck; Transactions of Algorithms '24]. In an attempt to bridge this theory-practice gap we characterise the parameter of degeneracy yielding a simple approximation algorithm for colouring HRGs. The approximation ratio of our algorithm ranges from $(2/\sqrt{3})$ to $4/3$ depending on the power-law exponent of the model. We complement our findings for the degeneracy with new insights on the clique number of hyperbolic random graphs. We show that degeneracy and clique number are substantially different and derive an improved upper bound on the clique number. Additionally, we show that the core of HRGs does not constitute the largest clique.

Lastly we demonstrate that the degeneracy of the closely related standard model of geometric inhomogeneous random graphs behaves inherently different compared to the one of hyperbolic random graphs.

2012 ACM Subject Classification Theory of computation → Random network models

Keywords and phrases hyperbolic random graphs, scale-free networks, power-law graphs, cliques, degeneracy, vertex colouring, chromatic number

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.13

Related Version *Full Version:* <https://arxiv.org/abs/2410.11549> [1]

Funding This research was partially funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – project number 390859508, and by Austrian Science Fund (FWF) <https://doi.org/10.55776/P36280>, <https://doi.org/10.55776/I6915>, and <https://doi.org/10.55776/D0C183>.

Acknowledgements The authors would like to thank Thomas Bläsius for enriching discussions and Maximilian Katzmann for running experiments indicating a gap between core and maximum inner-neighbourhood during a research visit of JR in Karlsruhe.

1 Introduction

Many real-world networks have a heterogeneous degree distribution, close to a power-law, as well as a constant clustering coefficient. The *hyperbolic random graph* model (HRG) introduced by Krioukov et. al. [29] combines both properties [19], which has led to considerable interest



© Samuel Baguley, Yannic Maus, Janosch Ruff, and George Skretas;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 13; pp. 13:1–13:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



in recent years. Various aspects of HRGs have been studied, including clique size [8, 9, 16], treewidth [10], minimum vertex cover size [4, 5], and diameter [17, 24, 33]. A hyperbolic random graph is a graph embedded in the hyperbolic plane where pairs of vertices have an edge if they are close according to the hyperbolic distance. It is generated by randomly throwing n vertices on a disk of radius R (dependent on n). Since hyperbolic space grows exponentially, most vertices are of small degree and located close to the boundary of the disk, while few vertices have large degree and lie close to the centre. This distribution of vertices leads to the power-law degree distribution of HRGs.

In this paper, we study the vertex colouring problem on HRGs, along with the related concepts of clique number and degeneracy. The k -colouring problem asks to colour the vertices of a graph with k colours, while assigning different colours to adjacent vertices. The *chromatic number* $\chi(G)$ is the minimum number of colours needed to colour a graph in such a way. For $k \geq 3$ the problem is one of the original NP-hard problems [13, 21]. On general graphs, even loosely approximating the chromatic number is particularly hard [36].

The answer to the colouring problem is closely related to the clique number $\omega(G)$ and the degeneracy of a graph. The *clique number* is the number of vertices of the largest clique of the graph and it serves as a natural lower bound to the chromatic number. The *degeneracy* $\kappa(G)$ is the minimum integer k' for which there exists an ordering of the vertex set of G , $V = (v_1, v_2, \dots, v_n)$, such that for every index $i \in [n - 1]$, v_i has at most k' neighbours with greater index. Any graph G can be *easily* coloured with $\kappa(G) + 1$ colours by iterating through the vertices in reverse order and simply colouring a vertex with a colour not used by any of its higher ranked neighbours.

We aim to study these structural parameters that are not only fundamental to the model but also in general for algorithm design in various models of computation [2, 12, 18]. The most prominent large clique in an HRG is formed by the vertices in the graph's core [8]. Simply put, the core emerges among polynomially many vertices of distance at most $R/2$ from the centre of the disk, which due to the triangle inequality form a clique. We denote the size of this clique by $\sigma(G)$. At this point one may wonder whether the core forms the largest clique of the graph, a statement that we disprove in Proposition 16. Nevertheless we show that the largest clique can at most be small constant factor larger than the core.

► **Theorem** (Simplified version of Theorem 20). *There exists a constant $\delta > 0$ such that for any threshold HRG G , $\sigma(G) + 1 \leq \omega(G) \leq \sqrt{4/3 - \delta} \cdot \sigma(G)$ holds w.e.h.p.¹*

This upper bound improves on prior work [8], which showed that there exists some constant $c > 1$ such that $\omega(G) \leq c \cdot \sigma(G)$ holds w.e.h.p., but without providing any upper bound on c .

We can now see that the core and clique are not the same. Nonetheless, (i) this theorem shows that the largest clique size and the core size are closely related, and (ii) the core of HRGs is a very well understood object, whereas the largest clique is not. Thus the natural approach to bound the degeneracy is to use the core. We show the following theorem.

► **Theorem** (Simplified version of Theorem 9). *There exist constants $\delta_1, \delta_2 > 0$ such that for any threshold HRG G , $(1 + \delta_1) \cdot \sigma(G) \leq \kappa(G) \leq (4/3 - \delta_2) \cdot \sigma(G)$ holds w.e.h.p.*

The main surprise of this theorem is that the degeneracy is bounded away from the coresize by a constant factor. As the chromatic number is lower bounded by the core size, the upper bound on the degeneracy in this theorem implies a simple algorithm colouring with at most

¹ An event holds *with extremely high probability* (w.e.h.p.), if for every $c > 1$, there exists an n_0 such that for every $n \geq n_0$ the event holds with probability at least $1 - n^{-c}$.

$(4/3 - \delta_2)\chi(G)$ colours. The approximation guarantee of this algorithm ranges from $2/\sqrt{3}$ to $4/3$ depending on further model parameters, see Section 2 and Theorem 11 for details. In any case, this improves on the previously best approximation ratio of 2 [7, Lemma 7]. The algorithm iteratively removes vertices of degree at most $(4/3 - \delta_2)\sigma(G) - 1$ and then colours them in the reverse order. The next thing one would hope is to be able colour the graph with $\omega(G)$ colours using the same process. In [3], the authors conducted experiments where they were iteratively removing the vertex with the smallest degree of the graph, up to vertices with residual degree equal to $\omega(G)$. In their findings, this process did not remove every vertex, implying $\omega(G) < \kappa(G)$ for their generated graphs. We substantiate their findings by providing a rigorous proof demonstrating that the clique number is, in fact, a constant factor smaller than the degeneracy.

► **Theorem** (Simplified version of Theorem 13). *There exists a constant $\varepsilon > 0$ such that for any threshold HRG G , $\omega(G) \leq (1 - \varepsilon) \cdot \kappa(G)$ holds w.e.h.p.*

Our final contribution is to study the degeneracy of *geometric inhomogeneous random graphs* (GIRGs) [23], a sibling to HRGs. The GIRGs also combine heterogeneity and high clustering. For most properties GIRGs and HRGs exhibit the same behaviour. Perhaps the first paper to find a difference between them is [9], where the authors show that the minimum number of maximal cliques in the two models differ. We show a significant discrepancy for the degeneracy of GIRGs compared to that of HRGs, see Figure 1 and Corollary 24.

Outline. See Figure 1 for a table with our results, as well as a plot comparing the bounds of our theorems for various model parameters. In Section 1.1 we provide a detailed discussion of our results and techniques. Section 3 contains bounds on the degeneracy of HRGs (Theorem 9). In Section 4, we show the gap between clique number and degeneracy (Theorem 13), as well as bounds on the clique number (Theorem 20). Finally, Section 5 contains results about the degeneracy of GIRGs. Statements where proofs or details are omitted due to space constraints can be found in the full version [1].

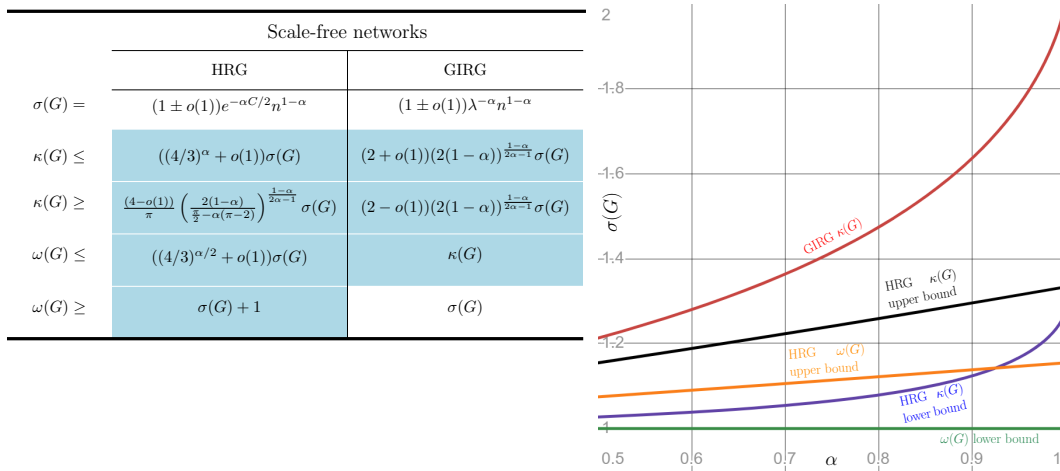
1.1 Discussion of our Results and Techniques

HRGs have a power-law degree distribution [19, 34], that is, the probability that a vertex has degree k is given by $\sim k^{-(2\alpha+1)}$. The model parameter $\alpha \in (1/2, 1)$ controls the power-law exponent and all our results, particularly the size of the aforementioned constant factor gap depends on the choice of α . For the ease of presentation this overview largely omits this dependence, but the summary of our results in Figure 1 plots it in detail.

Upper bound on degeneracy (Theorem 9). One consequence of generating a graph in hyperbolic space is that vertices tend to have fewer neighbours with increasing radius, i.e., the expected number of neighbours of a vertex decreases with the distance from the centre of the hyperbolic disc. This produces the power-law degree distribution that is valuable in modelling real-world networks. It also leads to a simple approach for upper bounding degeneracy: instead of removing vertices ordered by (increasing) degree, we remove them by (decreasing) radius. If k is such that each vertex has at most k neighbours of smaller radius, then k is an upper bound on the degeneracy.

The notion governing this approach is the *inner-neighbourhood* of a vertex, see Figure 2. The inner-neighbourhood of a vertex u with radius r , denoted $\Gamma(u)$, is the set of vertices of distance at most R from u , i.e., they are neighbours of u , and with radius at most r , i.e., they are closer to the centre of the disc than u . The size $|\Gamma(u)|$ of the inner-neighbourhood

13:4 Hyperbolic Random Graphs: Clique Number and Degeneracy



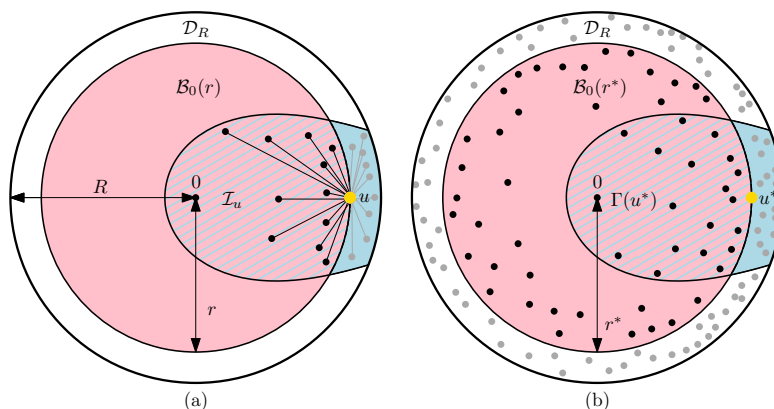
■ **Figure 1** Results on the degeneracy $\kappa(G)$ and the size of the largest clique $\omega(G)$ in hyperbolic random graphs (HRG) and geometric inhomogeneous random graphs (GIRG). The bounds hold w.e.h.p. and are stated in comparison to the core size $\sigma(G)$. Each curve represents the multiplicative factor in front of $\sigma(G)$ for $\kappa(G)$ and $\omega(G)$ (y-axis) depending on the parameter $\alpha \in (1/2, 1)$ (x-axis). Prior work is listed with white background, whereas our results are listed with blue background.

is called the *inner-degree* of u . The expected value of $|\Gamma(u)|$ scales with the area of u 's *inner-ball* $\mathcal{I}(r) = \mathcal{B}_u(r) \cap \mathcal{B}_0(r)$. More precisely, $\mathbb{E}[|\Gamma(u)|] = (n-1) \cdot \mu(\mathcal{I}(r))$. See Figure 2 a for a visual representation. The vertex of largest inner-degree is denoted u^* ; the choice of u^* and the value of $|\Gamma(u^*)|$ depend on the random distribution of the vertices.

If vertices are removed from outside to inside, then each vertex at the time of its removal will have degree less or equal to the inner-degree of u^* . To bound the degeneracy, we derive a probabilistic upper bound for $|\Gamma(u^*)|$. We do this by finding the radius that maximises $\mu(\mathcal{I}(r))$, which upper bounds $\mathbb{E}[|\Gamma(u)|]$ (for every vertex u). Since $|\Gamma(u)|$ is concentrated we can apply a Chernoff and a union bound to obtain a high probability upper bound for $|\Gamma(u^*)|$. Despite the simplicity of the inner-neighbourhood, we elaborate on this key concept as it is not only crucial to our upper bound on degeneracy but also for most of our other results discussed later on.

Lower bound on degeneracy (Theorem 9). In Lemma 6 we show that the maximum inner-degree also produces a lower bound on the degeneracy $\kappa(G)$, in the sense that $\kappa(G) \geq (1 - o(1))|\Gamma(u^*)|$ w.e.h.p. This yields asymptotically tight bounds on $\kappa(G)$. We prove the lower bound of $\kappa(G)$ by considering the subgraph G' induced by the vertices that have smaller radius than u^* (see Figure 2 b). Note that G' contains vertices that are not neighbours of u^* . We show that every vertex u in G' has, w.e.h.p., at least $(1 - o(1))|\Gamma(u^*)|$ neighbours in G' ; this statement uses the choice of u^* and is not true if u was an arbitrary vertex. Now, in any ordering of the vertices of G' , the first vertex has at least $(1 - o(1))|\Gamma(u^*)|$ neighbours of greater index, implying $(1 - o(1))|\Gamma(u^*)| \leq \kappa(G') \leq \kappa(G)$. While this provides a lower bound that asymptotically matches our upper bound, a lower order gap remains.

Gap between clique number and degeneracy (Theorem 13). The most immediate lower bound for degeneracy is the clique number [2], because in every ordering of the vertices of G , the vertex of the clique with the lowest index has at least $\omega(G) - 1$ neighbours of



■ **Figure 2** Illustration of the inner-neighbourhood. (a) The pink area is the ball $\mathcal{B}_0(r)$. The hatched area $\mathcal{I}_u = \mathcal{B}_u(R) \cap \mathcal{B}_0(r)$ is the inner-ball of u . The vertices $V \cap \mathcal{I}_u$ form the inner-neighbourhood $\Gamma(u)$. (b) Sketch of proof of Lemma 6. Vertex u^* is the vertex with the largest inner-degree. Each vertex $v \in U = V \cap \mathcal{B}_0(r^*)$ has at least nearly as many neighbours within $\mathcal{B}_0(r^*)$ as $|\Gamma(u^*)|$ (w.e.h.p.).

higher index. We prove that the lower bounds on the degeneracy that are derived from the clique number are strictly worse than the bounds discussed above obtained via analysing the inner-degree. Our approach to show this is the following: we take an arbitrary clique K and the vertex u with the largest radius in K . Let G'' be the subgraph induced by $\Gamma(u)$. Next, we partition G'' into three sets of vertices, each containing at least a constant fraction of the vertices of G'' w.e.h.p. See Figure 3 b for an illustration of this partition. Lastly, we use purely geometric arguments to show that K cannot contain vertices from all three sets. The gap then follows as the left out set contains a constant fraction of u 's inner neighbourhood. This gap between $\omega(G)$ and $\kappa(G)$ makes approaches for computing the clique number via degeneracy, like that of Walteros and Buchanan [35], unsuitable for HRGs.

Clique number and the core (Theorem 20). The upper bound on the clique number is proven via a geometric approach. Let u, v, w be any three vertices. If the vertices are far apart they cannot be contained in a clique. Otherwise their pairwise distance is at most R . Using the hyperbolic version of Jung's theorem [20, 14, 15] implies that they are contained in a ball \mathcal{B} of small radius and we show that \mathcal{B} also has a small area. Hence the expected number of vertices in \mathcal{B} is small as well. As this expectation is well concentrated whenever the area is significantly large, this bound holds with extremely high probability when introducing lower order deviations from the expectation. Via a union bound over all possible $\binom{n}{3}$ triples of vertices we rule out that any clique contained in such a covering ball is large. The main claim now follows because any clique has to be contained in one of these coverings balls. The question of whether $\omega(G)/\sigma(G) \rightarrow 1$ as $n \rightarrow \infty$ remains open.

2 Preliminaries

Hyperbolic Random Graphs. We follow the formalisation of hyperbolic random graphs introduced in [34], which is known as the *native representation*. We denote by $\mathbb{H}^2 = [0, \infty) \times [0, 2\pi)$ the hyperbolic plane in the polar coordinate system, where a point $x \in \mathbb{H}^2$ is parametrised by a radius $r(x)$ and an angle $\varphi(x)$. We equip \mathbb{H}^2 with a metric $d_h(x, y)$ characterised by

13:6 Hyperbolic Random Graphs: Clique Number and Degeneracy

$$\cosh(d_h(x, y)) = \cosh(r(x)) \cosh(r(y)) - \sinh(r(y)) \sinh(r(x)) \cos(\varphi(x) - \varphi(y)). \quad (1)$$

This metric is what gives \mathbb{H}^2 a hyperbolic geometry, of curvature -1, as opposed to the Euclidean metric. We equip \mathbb{H}^2 with the topology induced by d_h .

The geometric space of most importance in this work is the bounded disk in \mathbb{H}^2 defined by $\mathcal{D}_R = [0, R] \times [0, 2\pi)$, where $R = 2 \log(n) + C$ with $C \in \Theta(1)$. We refer to point $(0, 0)$ as the *centre of this disk*. The space \mathcal{D}_R inherits the topology of \mathbb{H}^2 , and from now on we shall only consider subsets of this space – thus, for example, a ball around a point $x \in \mathcal{D}_R$ is defined by the set $\mathcal{B}_x(\varepsilon) = \{y \in \mathcal{D}_R : d_h(x, y) \leq \varepsilon\} \subseteq \mathcal{D}_R$.

We now introduce a probability measure μ on \mathcal{D}_R , which is parametrised by the model parameter $\alpha \in (1/2, 1)$, and was first defined by Papadopoulos et. al. [34]. For measurable $\mathcal{S} \subseteq \mathcal{D}_R$, define

$$\mu(\mathcal{S}) = \int_{\mathcal{S}} \rho(x) dx, \quad \rho(x) = \frac{\alpha \sinh(\alpha x)}{2\pi(\cosh(\alpha R) - 1)},$$

where ρ is the density of μ with respect to the Lebesgue measure on \mathcal{D}_R . This measure differs from the uniform probability measure on \mathcal{D}_R in that it puts more mass at the centre of the disk; both measures coincide at $\alpha = 1$. The benefit of μ lies in the properties it induces in our central object of study, the hyperbolic random graph.

Threshold hyperbolic random graph (HRG). A (*threshold*) *hyperbolic random graph* or *HRG* is a pair $G = (V, E)$ defined by the following procedure. First, n vertices are sampled independently at random in \mathcal{D}_R according to μ . Then any two vertices $u, v \in V$ are connected by an edge if and only if their distance $d_h(u, v)$ is at most R . We write $G \sim \mathcal{G}(n, \alpha, C)$ to denote a graph generated in this way. A vertex $u \in V$ is identified by its point coordinates in \mathbb{H}^2 and we write $V \cap \mathcal{A}$ to denote the set of vertices that are located in an area $\mathcal{A} \subseteq \mathcal{D}_R$.

The use of μ to distribute vertices in \mathcal{D}_R has the effect of giving G a power-law degree distribution, as was shown in [19, 34]. It is sometimes convenient to characterise connection of vertices in terms of their *angular distance*, and to that end we define

$$\theta_R(r_1, r_2) = \arccos\left(\frac{\cosh(r_1) \cosh(r_2) - \cosh(R)}{\sinh(r_1) \sinh(r_2)}\right),$$

which per (1) yields the following observation.

► **Observation 1.** *Two vertices u and v are connected if and only if their angular distance is less than $\theta_R(r(u), r(v))$.*

We also make use of the following expression of the distribution of the radius of a vertex u .

$$\mathbb{P}(r(u) \leq r) = \mu(\mathcal{B}_0(r)) = \int_0^r \int_{-\pi}^{\pi} \rho(x) d\theta dx = \frac{\cosh(\alpha r) - 1}{\cosh(\alpha R) - 1} = (1 - o(1))e^{-\alpha(R-r)} \quad (2)$$

We briefly note that a variant of HRGs exists in which vertices are not connected purely according to whether their distance is below a threshold, but rather with probability $p(u, v) = (1 + \exp(\frac{1}{2T}(d_h(u, v) - R)))^{-1}$ determined by both distance and a “temperature” parameter T (see e.g. [30, §3.1]).

Degeneracy, clique number, chromatic number and core. For a graph $G = (V, E)$, the *degeneracy* $\kappa(G)$ is the minimum integer k for which there exists an ordering of the vertex set of G , $V = (v_1, v_2, \dots, v_n)$, such that for every index $i \in [n - 1]$, v_i has at most k neighbours with greater index. The *clique number* $\omega(G)$ is the size of the largest clique of G . The *chromatic number* $\chi(G)$ is the smallest number of colours required so that a conflict-free vertex colouring is possible for G . The *core* of a hyperbolic random graph is the set of vertices with radius at most $R/2$ and we denote its size by $\sigma(G)$. Since for any points $u, v \in \mathcal{B}_0(R/2)$ the distance is at most $d_h(u, v) \leq R$, the core forms a clique. Finally, since the core is a clique, any vertex of a clique needs a different colour in a conflict-free colouring, and $\chi(G) \leq \kappa(G) + 1$ (see e.g. [31, Lemma 4]), we have the following chain of inequalities.

► **Observation 2.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then,*

$$\sigma(G) \leq \omega(G) \leq \chi(G) \leq \kappa(G) + 1.$$

Concentration bounds. We use the following Chernoff bounds [32, Theorem 4.4].

► **Theorem 3 (Chernoff bound).** *For $i \in [k]$, let $X_i \in \{0, 1\}$ be independent random variables and $X = \sum_i X_i$. Then for $\varepsilon \in (0, 1)$,*

$$\mathbb{P}(X \geq (1 + \varepsilon)\mathbb{E}[X]) \leq e^{-\varepsilon^2 \cdot \mathbb{E}[X]/3} \text{ and } \mathbb{P}(X \leq (1 - \varepsilon)\mathbb{E}[X]) \leq e^{-\varepsilon^2 \cdot \mathbb{E}[X]/2}.$$

3 Degeneracy of Hyperbolic Random Graphs

A tool we make use of several times is the *inner-ball* of a point $x \in \mathcal{D}_R$, defined by $\mathcal{I}_x = \mathcal{B}_x(R) \cap \mathcal{B}_0(r(x))$. The inner-ball of a vertex is the inner-ball of the point using the vertex' coordinates. The *inner-neighbourhood* of a vertex u is the set of vertices (excluding u) contained in its inner-ball, that is, the neighbours of u that have a smaller radius than u (see Figure 2 a), and is denoted $\Gamma(u)$.

We upper bound the degeneracy $\kappa(G)$ via the inner-neighbourhood by using the following informal process. Consider a graph G and order its vertices (v_1, v_2, \dots, v_n) by decreasing radius, so $r(v_i) \geq r(v_{i+1})$, and iteratively remove vertices from G one-by-one, from lower to higher index. Note that the set of neighbours of v_i that have greater index than i coincide with $\Gamma(v_i)$. This implies that the degree of each vertex v_i at the time of its removal is $|\Gamma(v_i)|$. Let u^* be the vertex v_k that maximises $|\Gamma(v_k)|$. As the largest degree of a vertex at the time of its removal is given by $|\Gamma(u^*)|$, we get the following upper bound for the degeneracy.

► **Observation 4.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG and let u^* be the vertex of G with the largest inner-degree in G . Then $\kappa(G) \leq |\Gamma(u^*)|$.*

We will now show that the largest inner-degree does not only yield an immediate upper bound on the degeneracy, but also a lower bound. Informally, this lower bound follows from the following argument. Order the vertices of the graph in descending order of their radius, that is, (v_1, v_2, \dots, v_n) such that $r(v_i) \geq r(v_{i+1})$. For $i \in [n]$, let $G_i = G \setminus \{v_1, v_2, \dots, v_i\}$. Let v_k be the node with the maximum inner neighbourhood in G . We show (with a probabilistic guarantee) that the graph G_k has minimum degree $(1 - o(1))|\Gamma(v_k)|$. Before we make this formal, we introduce a slightly modified version of [19, Lemma 3.3], that implies the following: the closer a vertex is to the origin, the more neighbours it has in expectation. This can be derived via the fact that the angle $\theta_R(r, x)$ is monotonically decreasing in x .

► **Corollary 5.** *Let $r, s, t \in [0, R)$ with $s < t$. Then $\mu(\mathcal{B}_s(R) \cap \mathcal{B}_0(r)) \geq \mu(\mathcal{B}_t(R) \cap \mathcal{B}_0(r))$.*

13:8 Hyperbolic Random Graphs: Clique Number and Degeneracy

Corollary 5 tells us that any vertex with radius at most r has in expectation at least as many neighbours up to radius r , as the expected inner-degree of a vertex with radius exactly r . In order to derive a high-probability bound on $|\Gamma(u^*)|$ it now suffices to show concentration around the expectation of all considered neighbourhoods.

Since the size of every clique K is upper bounded by the inner-degree of the vertex in $u \in K$ that has the largest radius, $\omega(G)$ is a lower bound for the maximum inner-degree. We can now lower bound the degeneracy $\kappa(G)$ based on the largest inner-degree. Note that, in contrast to the upper bound of Observation 4, the lower bound is not deterministic.

► **Lemma 6.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then $\kappa(G) \geq (1 - o(1))|\Gamma(u^*)|$ w.e.h.p.*

Proof. For any subgraph $H \subseteq G$, it is clear that $\kappa(G) \geq \min_{v \in V(H)} \deg_H(v)$, where $\deg_H(v) = \sum_{w \in V \setminus \{v\}} \mathbb{1}\{\{v, w\} \in E(H)\}$. We let H be the (random) subgraph of G created by restricting G to vertices that land in $\mathcal{B}_0(r)$, that is, that have radius at most r , and keeping the same edges. Then for any $v \in V$,

$$\begin{aligned} \mathbb{E}[\deg_H(v) | r(v)] &= \sum_{w \in V \setminus \{v\}} \mathbb{P}(w \in \mathcal{B}_0(r) \cap \mathcal{B}_{r(v)}(R) | r(v)) \mathbb{1}\{r(v) \leq r\} \\ &= (n-1)\mu(\mathcal{B}_0(r) \cap \mathcal{B}_{r(v)}(R)) \mathbb{1}\{r(v) \leq r\} \\ &\geq (n-1)\mu(\mathcal{B}_0(r) \cap \mathcal{B}_r(R)) \mathbb{1}\{r(v) \leq r\}. \end{aligned} \quad (\text{Corollary 5})$$

We write $\gamma := (n-1)\mu(\mathcal{B}_0(r) \cap \mathcal{B}_r(R))$. Since $\deg_H(v)$ is a sum of Bernoulli random variables that are all independent under $\mathbb{P}(\cdot | r(v))$, a Chernoff bound (Theorem 3) gives

$$\begin{aligned} \mathbb{P}\left(\min_{v \in V(H)} \deg_H(v) < (1-\varepsilon)\gamma\right) &= \mathbb{P}\left(\bigcup_{v \in V} \{\deg_H(v) < (1-\varepsilon)\gamma, r(v) \leq r\}\right) \\ &\leq n\mathbb{P}(\deg_H(v) < (1-\varepsilon)\gamma, r(v) \leq r) \quad (\text{Union bound}) \\ &= n\mathbb{E}[\mathbb{P}(\deg_H(v) < (1-\varepsilon)\gamma | r(v)) \mathbb{1}\{r(v) \leq r\}] \quad (\text{Tower rule}) \\ &\leq n\mathbb{E}\left[e^{-\varepsilon^2 \mathbb{E}[\deg_H(v) | r(v)]/2} \mathbb{1}\{r(v) \leq r\}\right] \quad (\text{Chernoff bound}) \\ &\leq ne^{-\gamma\varepsilon^2/2} \mathbb{P}(r(v) \leq r) \leq ne^{-\gamma\varepsilon^2/2}. \end{aligned}$$

Taking r to be the argmax of γ yields $\gamma(r) \geq \mathbb{E}[\Gamma(u^*)]$. By Observation 2 and applying (2) at $R/2$ we have that $\gamma(r) \geq \mathbb{E}[\Gamma(u^*)] \geq \mathbb{E}[\sigma(G)] \in n^{\Omega(1)}$. Thus, choosing $\varepsilon = 1/\log(n)$, we obtain for any constant c as long as n is large enough

$$\mathbb{P}(\kappa(G) < (1-\varepsilon)|\Gamma(u^*)|) \leq \mathbb{P}(\kappa(G) < (1-\varepsilon)\gamma(r)) \leq ne^{-\gamma(r)\varepsilon^2/2} \leq ne^{-n^{\Omega(1)}/\log(n)} \leq n^{1-c},$$

that is, $\kappa(G) \geq (1 - o(1))|\Gamma(u^*)|$ w.e.h.p. ◀

In the rest of this section we derive bounds for the largest inner-degree on HRGs that hold w.e.h.p. which, by Observation 4 and Lemma 6, translate into results for the degeneracy. Since a vertex v belongs to the inner-neighbourhood of a vertex u , if and only if it resides in the inner-ball of u , we can use the measure of an inner-ball \mathcal{I}_u to bound the maximum inner-degree of a graph. Moreover, the measure of the inner-ball is invariant under rotation around the origin, which is why we write $\mu(\mathcal{I}(r))$ instead of $\mu(\mathcal{I}_u)$ for $r = r(u)$. We sum up our results for the area of the inner-ball in the following technical lemma.

► **Lemma 7** (Volume of the inner-ball). *Let $\Delta \in \Theta(1)$ and let $u \in \mathcal{D}_R$ with radius $r = R/2 + \Delta$. Then, depending on Δ , there exist constants γ, η , such that*

$$\begin{aligned} \mu(\mathcal{I}(r)) &\geq (1 + \Theta(e^{-\alpha R})) \frac{\alpha e^{-\alpha r}}{(\alpha - 1/2)} \left(\frac{2}{\pi} e^{\frac{1}{2}(2\alpha-1)(2r-R)} - \left(\frac{2}{\pi} - \frac{(\alpha - 1/2)}{\alpha} \right) \right) \quad \text{and} \\ \mu(\mathcal{I}(r)) &\leq (1 + \Theta(e^{-\alpha R})) \frac{\alpha e^{-\alpha r}}{\alpha - 1/2} \left(\gamma e^{\frac{1}{2}(2\alpha-1)(2r-R)} - \eta \right), \end{aligned}$$

where

$$\gamma, \eta = \begin{cases} 1, \frac{1}{2\alpha} & \text{for } \Delta \geq 0, \\ \frac{4}{3\sqrt{3}}, \frac{1}{2\alpha} - \left(1 - \frac{4}{3\sqrt{3}}\right) \left(\frac{4}{3}\right)^{(\alpha-1/2)} & \text{for } \Delta \geq \log(\sqrt{4/3}), \\ \frac{1}{\sqrt{2}}, \frac{1}{2\alpha} - \left(1 - \frac{4}{3\sqrt{3}}\right) \left(\frac{4}{3}\right)^{(\alpha-1/2)} - \left(\frac{4}{3\sqrt{3}} - \frac{1}{\sqrt{2}}\right) 2^{(\alpha-1/2)} & \text{for } \Delta \geq \log(\sqrt{2}), \\ \frac{2}{3}, \frac{1}{2\alpha} - \left(1 - \frac{4}{3\sqrt{3}}\right) \left(\frac{4}{3}\right)^{(\alpha-1/2)} - \left(\frac{4}{3\sqrt{3}} - \frac{1}{\sqrt{2}}\right) 2^{(\alpha-1/2)} - \left(\frac{1}{\sqrt{2}} - \frac{2}{3}\right) 2^{(2\alpha-1)} & \text{for } \Delta \geq \log(2). \end{cases}$$

► **Lemma 8.** *Let r^* be the radial coordinate of the point in \mathcal{D}_R that maximises the measure of the inner-ball. Then $r^* = R/2 + \log\left(\frac{\alpha\eta}{\gamma(1-\alpha)}\right)/(2\alpha - 1)$.*

Proof. Using the expression of $\mu(\mathcal{I}(r))$ in Lemma 7,

$$\frac{d}{dr} \mu(\mathcal{I}(r)) = (1 + \Theta(e^{-\alpha R})) \left(\frac{\eta \alpha^2 e^{-\alpha r}}{\alpha - 1/2} - \frac{\gamma \alpha e^{-\alpha r}}{\alpha - 1/2} \left((1 - \alpha) e^{\frac{1}{2}(2\alpha-1)(2r-R)} \right) \right).$$

Setting this equal to 0 and solving for r yields $r^* = R/2 + \log\left(\frac{\alpha\eta}{\gamma(1-\alpha)}\right)/(2\alpha - 1)$. ◀

We use Lemma 7 to upper and lower bound the degeneracy. The lower bound tells us that there exists a constant $\delta_\alpha > 0$ such that $\kappa(G) \geq (1 + \delta_\alpha)\sigma(G)$ w.e.h.p. The constant δ_α is increasing with increasing $1/2 < \alpha < 1$, see Figure 1.

► **Theorem 9** (Bounds on the degeneracy). *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then w.e.h.p. its degeneracy $\kappa(G)$ satisfies*

$$\frac{(4 - o(1))}{\pi} \left(\frac{2(1 - \alpha)}{\frac{\pi}{2} - \alpha(\pi - 2)} \right)^{\frac{1-\alpha}{2\alpha-1}} \sigma(G) \leq \kappa(G) \leq ((4/3)^\alpha + o(1))\sigma(G).$$

Proof sketch. The upper bound is obtained by using r^* as stated in Lemma 8 for the upper bound of the inner-ball with $(n - 1)\mu(\mathcal{I}(r^*))$, and using a Chernoff bound along with a union bound which gives an upper bound for $|\Gamma(u^*)|$ w.e.h.p. Observation 4 then yields the upper bound for the degeneracy. For the lower bound, we first show that there exists a vertex with a radius \tilde{r} close in value to r^* w.e.h.p. Then $(1 - o(1))(n - 1)\mu(\mathcal{I}(\tilde{r}))$ lower bounds $|\Gamma(u^*)|$ w.e.h.p. This gives a lower bound for the degeneracy, due to Lemma 6. ◀

Applying Observation 2 and Theorem 9, the following is immediate.

► **Corollary 10** (Bounds on the chromatic number). *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then w.e.h.p. its chromatic number is $\sigma(G) \leq \chi(G) \leq ((4/3)^\alpha + o(1))\sigma(G)$.*

Our structural results directly produce algorithmic applications. The small gap between degeneracy and core translates into an efficient approximation algorithm to colour a HRG.

► **Theorem 11.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then an approximate vertex colouring of G can be computed in time $\mathcal{O}(n)$ with approximation ratio $((4/3)^\alpha + o(1))$ w.e.h.p.*

Proof sketch. Using a *smallest-last* vertex ordering [31] the number of colours required is upper-bounded by $\kappa(G)$. Computing the smallest-last vertex ordering, and then using the ordering to colour the graph, both takes linear as the giant component is sparse w.e.h.p. by [25, Corollary 17]. The approximation ratio is achieved by comparing the lower bound of the chromatic number in Corollary 10 to the upper bound of the degeneracy in Theorem 9. ◀

4 Clique Number of Hyperbolic Random Graphs

Recall that for any graph G , the clique number $\omega(G)$, chromatic number $\chi(G)$, and degeneracy $\kappa(G)$ are related via the inequalities $\omega(G) \leq \chi(G) \leq \kappa(G)+1$. For this reason we are interested in the relationship between clique number and degeneracy for hyperbolic random graphs. In Section 4.1 we show that the two differ and that for HRGs, the degeneracy is strictly larger than the clique number by a constant multiplicative constant. In Section 4.2 we give new insights about where in the hyperbolic disk the largest clique is formed. We then conclude the section by providing a new upper bound for the clique number in Section 4.3 that states a leading constant in front of the size of the core, and that is increasing in α .

4.1 The gap between Clique Number and Degeneracy

Because of the centralising effect of hyperbolic geometry, one might hope to show that the clique contained in the core of the disk is the largest, and that $\omega(G) = (1 - o(1))\kappa(G)$. This would achieve two things on HRGs: first, it would imply a tight bound for the chromatic number $\chi(G)$, sandwiching it between clique number and degeneracy. Moreover, it would also imply a linear time $(1 + o(1))$ -approximation algorithm for the two NP-complete problems clique number and chromatic number using a smallest-last vertex ordering (see [31]).

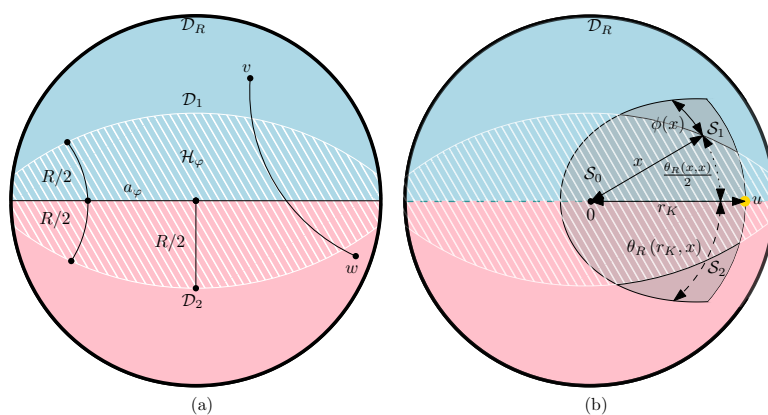
In this section we disprove these claims. We show that there exists a constant gap between clique number and degeneracy; this is the content of Theorem 13. Before embarking on the details of the proof, we first sketch its idea. For any clique K , its size is bounded by the inner-degree $|\Gamma(u)|$, where u is the vertex with largest radius among the vertices of K . For $r(u) \leq R/2 + o(1)$ and $r(u) \in R/2 + \omega(1)$, $|\Gamma(u)|$ is already smaller by a multiplicative constant than the lower bound for the degeneracy given in Theorem 9. What remains is to extend the result to $r(u) \in R/2 + \Theta(1)$, which requires more intricate arguments and is addressed in the following lemma.

► **Lemma 12.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG, let $\Delta \in \Theta(1)$ and let K be any clique where $u \in K$ is the vertex with largest radius $r_K = R/2 + \Delta$. Then w.e.h.p. there exists a constant $\varepsilon \in (0, 1)$ such that $|K| \leq (1 - \varepsilon)|\Gamma(u)|$.*

Proof. Let G' be the inner-neighbourhood of u , i.e., the induced subgraph $G' = G[\Gamma(u)]$. Then $K \subseteq V(G')$, and thus $\omega(G') \geq |K|$. We show that $\omega(G') < (1 - \varepsilon)|\Gamma(u)|$ w.e.h.p.

We accomplish this by proving that there exists a colouring for G' with $(1 - \varepsilon)|V(G')| = (1 - \varepsilon)|\Gamma(u)|$ many colours. This implies an upper bound for the chromatic number $\chi(G')$, which also serves as an upper bound for $\omega(G')$. We do this by partitioning the inner-neighbourhood \mathcal{I}_u into three disjoint sub-regions $\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2$ such that no vertex in \mathcal{S}_1 is adjacent to any vertex in \mathcal{S}_2 . Thus, \mathcal{S}_0 separates \mathcal{S}_1 from \mathcal{S}_2 and we can colour the set of vertices $V \cap \mathcal{S}_1$ with the same colours as $V \cap \mathcal{S}_2$. Thus if $\min\{|V \cap \mathcal{S}_1|, |V \cap \mathcal{S}_2|\} \geq \varepsilon|\Gamma(u)|$ w.e.h.p., our desired statement will be proven.

To find such a separator \mathcal{S}_0 , we follow the lines drawn by Bläsius et. al. [10] where *hypercycles* for hyperbolic random graphs were introduced (see Figure 3 a). A hypercycle \mathcal{H}_φ (of radius $R/2$) is defined as follows: let a_φ denote the line whose points have angle φ



■ **Figure 3** Illustration of a separator. (a) The line a_φ partitions disk \mathcal{D}_R into two halfdisks \mathcal{D}_1 (blue) and \mathcal{D}_2 (pink). The hypercycle \mathcal{H}_φ (hatched area) is defined by the line a_φ . Points located in different halfdisks and outside the hatched area have distance at least R . (b) The separator (hatched area) separates the inner-neighbourhood (grey area) of a vertex u of radius r_K into three sub-areas \mathcal{S}_0 , \mathcal{S}_1 and \mathcal{S}_2 . Any vertex located in \mathcal{S}_1 has no edge to a vertex in \mathcal{S}_2 .

and $\varphi + \pi$. Then $\mathcal{H}_\varphi := \{u \in \mathcal{D}_R : d_h(u, a_\varphi) \leq R/2\}$, i.e. the set of points with distance at most $R/2$ to line a_φ . Consider the point $u = (r_K, \varphi)$ and let $\mathcal{S}_0 := \mathcal{I}_u \cap \mathcal{H}_\varphi$. To define \mathcal{S}_1 and \mathcal{S}_2 separate \mathcal{D}_R into the two disjoint *halfdisks* $\mathcal{D}_1 = \{x \in \mathcal{D}_R : \varphi(x) - \varphi \leq \pi\}$ and $\mathcal{D}_2 = \{x \in \mathcal{D}_R : \varphi(x) - \varphi \geq \pi\}$ (see Figure 3 a). Then we define $\mathcal{S}_1 := (\mathcal{I}_u \cap \mathcal{D}_1) \setminus \mathcal{H}_\varphi$ and symmetrically $\mathcal{S}_2 := (\mathcal{I}_u \cap \mathcal{D}_2) \setminus \mathcal{H}_\varphi$.

We observe that any point $w \in \mathcal{S}_1$ has distance at least R to any point $v \in \mathcal{S}_2$. This can be shown for example by observing that the geodesic between w and v must pass through some point $x \in a_\varphi$, and so $d_h(w, v) = d_h(w, x) + d_h(x, v) \geq d_h(w, a_\varphi) + d_h(a_\varphi, v) > R$. This ensures our objective of separating the two regions via \mathcal{S}_0 .

Next, we show that there exists a constant $\varepsilon > 0$ small enough, such that $\mu(\mathcal{S}_1) = \mu(\mathcal{S}_2) \geq \varepsilon n^{-\alpha}$ (a sketch of the idea is given in Figure 3 b). Setting $\phi(x) = \max(0, \theta_R(r_K, x) - \theta_R(x, x)/2)$ we derive by symmetry

$$\mu(\mathcal{S}_1) = \mu(\mathcal{S}_2) = \int_{R/2}^r \int_0^{\phi(x)} \rho(x) d\phi dx = \int_{R/2}^{R/2+\Delta} \phi(x) \rho(x) dx.$$

Now we choose another constant $\Delta' < \Delta$ that fulfills $2\theta_R(R/2 + \Delta, R/2 + \Delta') - \theta_R(R/2 + \Delta', R/2 + \Delta') =: c \in \Theta(1)$. This is possible because $\Delta \in \Omega(1)$. By our choice of Δ' we then obtain

$$\begin{aligned} \mu(\mathcal{S}_1) &\geq \int_{R/2+\Delta'}^{R/2+\Delta} \phi(x) \rho(x) dx = \int_{R/2+\Delta'}^{R/2+\Delta} \left(\theta_R(R/2 + \Delta, x) - \frac{\theta_R(x, x)}{2} \right) \rho(x) dx \\ &\geq c \int_{R/2+\Delta'}^{R/2+\Delta} \rho(x) dx = \frac{c(\cosh(\alpha(R/2 + \Delta)) - \cosh(\alpha(R/2 + \Delta')))}{\cosh(\alpha R) - 1} \in \Omega(n^{-\alpha}), \end{aligned}$$

where the last line follows since $\theta_R(\cdot, \cdot)$ is monotonically decreasing and since $\rho(x) = \frac{\alpha \sinh(\alpha x)}{\cosh(\alpha R) - 1}$, $|\Delta - \Delta'| > 0$ and $R = 2 \log(n) + C$. Therefore $\mu(\mathcal{S}_1) \in \Omega(\mathbb{E}[\|\Gamma(u)\|] / n)$ w.e.h.p., since

$$\liminf_{n \rightarrow \infty} \frac{n\mu(\mathcal{S}_1)}{\mathbb{E}[\|\Gamma(u)\|]} = \liminf_{n \rightarrow \infty} \frac{\mu(\mathcal{S}_1)}{n^{-\alpha}} \frac{n^{1-\alpha}}{\mathbb{E}[\|\Gamma(u)\|]} > 0,$$

13:12 Hyperbolic Random Graphs: Clique Number and Degeneracy

where $\mathbb{E}[|\Gamma(u)|] \leq n\mu(\mathcal{B}_0(R/2 + \Delta)) \in \mathcal{O}(n^{1-\alpha})$ by Equation (2), because $\Delta \in \mathcal{O}(1)$. Thus there exists some $\varepsilon > 0$ for which, for n large enough,

$$\mathbb{E}[|V \cap \mathcal{S}_2|] = \mathbb{E}[|V \cap \mathcal{S}_1|] = n\mu(\mathcal{S}_1) \geq (1 - 1/\log(n))^{-2}\varepsilon\mathbb{E}[|\Gamma(u)|];$$

since $|V \cap \mathcal{S}_1| \leq |\Gamma(u)|$ a.s. and is strictly smaller with positive probability, then $\varepsilon < 1$.

Using a Chernoff bound for both $|V \cap \mathcal{S}_1|$ and $|V \cap \mathcal{S}_2|$, we obtain that neither random variable is smaller than $(1 - 1/\log(n))\mathbb{E}[|V \cap \mathcal{S}_1|] \geq (1 - 1/\log(n))^{-1}\varepsilon\mathbb{E}[|\Gamma(u)|]$ w.e.h.p. On the other hand, another application of a Chernoff bound reveals $|\Gamma(u)| \leq (1 + 1/\log(n))\mathbb{E}[|\Gamma(u)|]$ w.e.h.p., since for $r(u) \in R/2 + \Theta(1)$ we have $\mathbb{E}[|\Gamma(u)|] \in \Theta(n^{(1-\alpha)})$ using Lemma 7. A union bound then shows that w.e.h.p., $\min\{|V \cap \mathcal{S}_1|, |V \cap \mathcal{S}_2|\} \geq \frac{\varepsilon\mathbb{E}[|\Gamma(u)|]}{1 - 1/\log(n)} \geq \varepsilon|\Gamma(u)|$. As argued above, a naïve colouring that colours the vertices of \mathcal{S}_1 and \mathcal{S}_2 with the same set of colours yields the upper bound. \blacktriangleleft

► **Theorem 13 (Clique-degeneracy-gap).** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then w.e.h.p. there exists a constant $\varepsilon \in (0, 1)$ such that $\kappa(G)/\omega(G) > 1 + \varepsilon$.*

Proof. Let K be the largest clique of G , and let u be the vertex of K with maximal radius $r_K := r(u)$. We assume the following cases for r_K which cover all possibilities:

Case 1 [$r_K \in R/2 + \omega(1)$]: Observe that $K \subseteq V \cap \mathcal{I}(r_K)$. Hence, $|K| \leq |V \cap \mathcal{I}(r_K)|$ and

$$\mu(\mathcal{I}(r_K)) \leq (1 + \Theta(e^{-\alpha R})) \frac{\alpha e^{-\alpha r_K}}{\alpha - 1/2} \left(\gamma e^{\frac{1}{2}(2\alpha-1)(2r_K-R)} - \eta \right) \in \Theta(1)n^{-\alpha}e^{-(1-\alpha)\omega(1)},$$

by Lemma 7 since γ and η are both constants. Taking the expectation and using a Chernoff bound we have $|K| \in o(n^{1-\alpha})$ w.e.h.p. Recall that $\kappa(G) > (1 + \delta)e^{-\alpha C/2}n^{1-\alpha}$ w.e.h.p. by Theorem 9. It follows $\kappa(G)/|K| \in \omega(1)$ w.e.h.p.

Case 2 [$r_K \leq R/2 + o(1)$]: Observe that the size of any clique $K \subseteq V \cap \mathcal{B}_0(r)$ is upper bounded by $X = |V \cap \mathcal{B}_0(r)|$. By Equation (2) we have $\mu(\mathcal{B}_0(r_K)) \leq (1 + o(1))n^{-\alpha}e^{-\alpha C/2}$. Hence, we get $\mathbb{E}[X] \leq (1 + o(1))n^{1-\alpha}e^{-\alpha C/2}$. Since X is a binomial random variable we can apply a Chernoff bound, which yields $|K| \leq X \leq (1 + o(1))n^{1-\alpha}e^{-\alpha C/2}$ w.e.h.p. Taking $\varepsilon = \delta/(1 + \delta)$ with δ as in Theorem 9, we have

$$\frac{|K|}{1 - \varepsilon} \leq (1 + o(1))(1 + \delta)n^{1-\alpha}e^{-\alpha C/2} < (1 + o(1))\kappa(G) \text{ w.e.h.p.}$$

Case 3 [$r_K \in R/2 + \Theta(1)$]: Let $\xi \in o(1)$. Using Lemmas 6 and 12, we get w.e.h.p. that

$$\omega(G) = |K| \leq \frac{1 - \varepsilon}{1 - \xi} |\Gamma(u)| \leq \frac{1 - \varepsilon}{1 - \xi} |\Gamma(u^*)| \leq (1 - \varepsilon)\kappa(G),$$

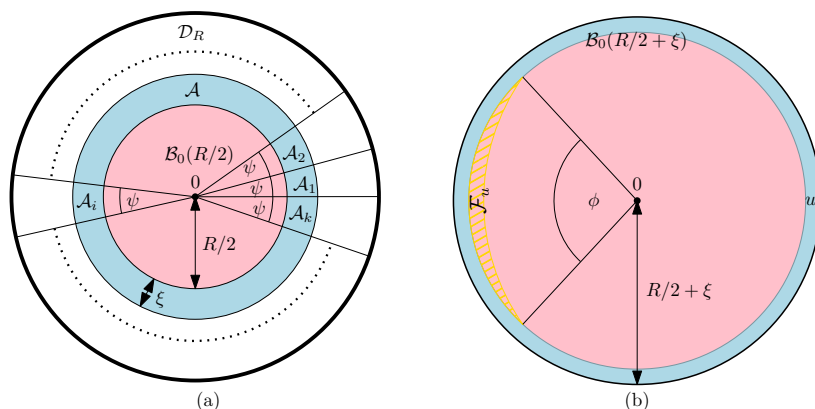
for an adequate choice of ξ . \blacktriangleleft

► **Corollary 14.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then there exists a constant $\varepsilon \in (0, 1)$ such that w.e.h.p., $\omega(G) \leq (4(1 - \varepsilon)/3)^\alpha \sigma(G)$.*

Proof. This follows from Theorems 9 and 13. \blacktriangleleft

4.2 Cliques larger than the Core

In this section, we show that there exist a super-constant number of unique cliques that contain the core, but are strictly larger than it. The overall argument goes as follows. We consider a set of vertices with radial coordinates slightly outside the core. We show that any vertex of this set has a constant probability to be adjacent to all vertices belonging to the core, and thus induces a clique that is larger than the core itself. To this end, the following lemma concerned about points close to the core proves useful.



■ **Figure 4** Sketch of the proof idea for Proposition 16. (a) Illustration of the set of points \mathcal{A} (blue) of width ξ slightly outside of the core (pink). The intersection with a sector of angle ψ and the band \mathcal{A} forms a box \mathcal{A}_i that contains a vertex w.e.h.p. The number of non-intersecting boxes is $k = 2\pi/\psi \in \omega(\log(n))$. (b) A vertex u located on the boundary of the area \mathcal{A} . The hatched area \mathcal{F}_u with angle at most ϕ is the corresponding forbidden area of u . Any point in \mathcal{F}_u has distance at least R to u . An adequate choice for the width ξ yields that this area is empty with constant probability.

► **Lemma 15.** *Let $k \in \mathbb{N} \setminus \{0\}$ and $\xi_k = \log(1 + \frac{\log^k(n)}{n^{1-\alpha}}) \in o(1)$. Consider two points with radial coordinates $r = R/2 + \xi_k$ and $x = R/2$. Then $\theta_R(r, x) \geq \pi - 2\sqrt{\log^k(n)n^{\alpha-1}}$.*

Lemma 15 lower bounds the angle distance such that two points have distance at most R .

► **Proposition 16.** *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then w.e.h.p. there exist $\omega(\log(n))$ cliques that are larger than $\sigma(G)$.*

Proof. For the proof we consider the Poissonized version of the HRG model (see e.g. [26, 27]). The upshot of this model is that it allows us to analyse disjoint areas in the hyperbolic disk independently. Since the final result holds w.e.h.p. this directly carries over to the uniform model w.e.h.p. ([22, Lemma 3.9]). We start by defining an area \mathcal{A} close to the core. Let $\xi = \log(1 + \frac{\log^3(n)}{n^{1-\alpha}}) \in o(1)$ and consider a band of points $\mathcal{A} := \mathcal{B}_0(R/2 + \xi) \setminus \mathcal{B}_0(R/2)$. We see via $R = 2 \log(n) + C$ that

$$\begin{aligned}
 \mu(\mathcal{A}) &= \int_{R/2}^{R/2+\xi} \frac{\sinh(\alpha x)}{\cosh(\alpha R) - 1} = \frac{\cosh(\alpha(R/2 + \xi)) - \cosh(\alpha R/2)}{\cosh(\alpha R) - 1} \\
 &= (1 + \Theta(e^{-\alpha R}))(e^{-\alpha(R/2-\xi)} - e^{-\alpha R/2}) \\
 &= (1 + \Theta(e^{-\alpha R}))(n^{-\alpha} e^{-\alpha C/2} (1 + \log^3(n)n^{\alpha-1}) - n^{-\alpha} e^{-\alpha C/2}) \\
 &= (1 + \Theta(e^{-\alpha R}))(e^{-\alpha C/2} \log^3(n)n^{-1}) \in \Theta(\log^3(n)/n). \tag{3}
 \end{aligned}$$

Let $A = V \cap \mathcal{A}$. Then $\mathbb{E}[|A|] \in \Theta(\log^3(n))$. We further partition the band \mathcal{A} into $k = \lceil \log^{3/2}(n) \rceil$ sectors $\mathcal{A}_1, \dots, \mathcal{A}_k$, each of equal size (see Figure 4 a), and $A_i = V \cap \mathcal{A}_i$. Since $\mathbb{E}[|A|] \in \Theta(\log^3(n))$ and $k = \log^{3/2}(n)$, we have for any $i \in [k]$ that $\mathbb{E}[|A_i|] = \frac{\mathbb{E}[|A|]}{k} \in \omega(\log(n))$. Since we are in the Poissonized model we get $\mathbb{P}(|A_i| = 0) = e^{-\mathbb{E}[|A_i|]} \in n^{-\omega(1)}$. Subsequently, a union bound yields that there is no sector \mathcal{A}_i that is empty w.e.h.p.

In the second step, we show that for any vertex $u \in A$, the probability that there exists a vertex in its *forbidden area* $\mathcal{F}_u := \{x \in \mathcal{B}_0(R/2) : d_h(u, x) > R\}$ (see Figure 4 b) is strictly less than 1. Since $r(u) \leq R/2 + \xi$, we have that $u \in A$ has distance at most R (and thus, an

13:14 Hyperbolic Random Graphs: Clique Number and Degeneracy

edge) to any vertex in the area $\mathcal{B}_0(R/2 - \xi)$. Hence, we have $\mathcal{F}_u \subset \mathcal{B}_0(R/2) \setminus \mathcal{B}_0(R/2 - \xi)$. Now, for $R/2 \geq x \geq R/2 - \xi$ we seek to find the angle size $\phi = 2(\pi - \theta_R(r(u), x))$ in order to upper bound $\mu(\mathcal{F}_u)$. Since $\theta_R(r, x)$ is monotonically decreasing in x we have $\phi \leq 2(\pi - \theta_R(R/2 + \xi, R/2))$ and we obtain

$$\begin{aligned} \mu(\mathcal{F}_u) &= \int_{R/2-\xi}^{R/2} \int_0^\phi \rho(x) \, d\phi \, dx \leq 2(\pi - \theta_R(R/2 + \xi, R/2)) \int_{R/2}^{R/2+\xi} \rho(x) \, dx \\ &\leq 2(\pi - \theta_R(R/2 + \xi, R/2))\mu(\mathcal{A}), \end{aligned}$$

where we used $\int_{R/2-\xi}^{R/2} \rho(x) \, dx \leq \int_{R/2}^{R/2+\xi} \rho(x) \, dx = \mu(\mathcal{A})$. By our choice of $\xi = \log(1 + \log^3(n)n^{\alpha-1}) \in o(1)$, we can apply Lemma 15 and obtain $\theta_R(R/2 + \xi, R/2) \geq \pi - 2\sqrt{\frac{\log^3(n)}{n^{1-\alpha}}}$. In Equation (3) we established that $\mu(\mathcal{A}) \in \Theta(\log^3(n)/n)$. Combining the two leads to $\mu(\mathcal{F}_u) \in \mathcal{O}\left(\sqrt{\log^9(n)n^{\alpha-3}}\right)$.

Notice that for $\alpha \in (1/2, 1)$, the measure of the forbidden area of a vertex $u \in A$ is $\mu(\mathcal{F}_u) \in o(1/n)$. Hence, writing $F = V \cap \mathcal{F}_u$, we get $\mathbb{E}[|F|] \in o(1)$, i.e., the expected number of vertices in \mathcal{F}_u is vanishing. Applying Markov's inequality then gives us $\mathbb{P}(|F| \geq 1) \in o(1)$. Thus $p := \mathbb{P}(|F| < 1) \in \Omega(1)$, so the forbidden area is empty with constant probability.

We now establish our third and final desired property. Namely, we construct a subset $U \subset A$, consisting of vertices whose forbidden areas are empty and disjoint and for which $\mathbb{E}[|U|] \in \omega(\log(n))$.

Recall that we partitioned \mathcal{A} into $k = \log^{3/2}(n) \in \omega(\log(n))$ sectors. Let X_i be the indicator that there exists a vertex $u \in \mathcal{A}_i$ whose forbidden area \mathcal{F}_u is empty. Then $X = \sum_{i=1}^{\lfloor k/2 \rfloor} X_{2i}$ is a loose lower bound on the number of sectors with this property. By linearity of expectation, p constant and $k = \log^{3/2}(n)$ we then obtain

$$\mathbb{E}[X] \geq \mathbb{E}\left[\sum_{i=1}^{\lfloor k/2 \rfloor} X_{2i}\right] \geq p \cdot \Theta(1) \log^{3/2}(n) \in \omega(\log(n)).$$

We proceed by showing independence among the random variables X_i and X_j for $|j - i| > 1$. To this end, observe that the angle ψ (see Figure 4 b) spanned by any sector \mathcal{A}_i is $\psi = 2\pi/k \geq \left\lfloor 2\pi \log^{-3/2}(n) \right\rfloor$. In contrast, we recall $\phi \leq 2(\pi - \theta_R(R/2 + \xi, R/2)) \leq 4\sqrt{\log^3(n)n^{\alpha-1}}$. Since $\sqrt{\log^3(n)n^{\alpha-1}} \in o(\log^{-3/2}(n))$ we conclude $\phi \in o(\psi)$. This implies that the forbidden areas \mathcal{F}_u and \mathcal{F}_v of any $u \in \mathcal{A}_i, v \in \mathcal{A}_j$ are disjoint. Thus X_i and X_j are independent.

To wrap things up, recall that in the first step we established that each sector \mathcal{A}_i contains a vertex w.e.h.p. Moreover, since the X_i are independent, we have by a Chernoff bound that $X \in \omega(\log(n))$ w.e.h.p. Though these two events are not independent, we can apply the union bound to their complements to obtain that w.e.h.p. $\omega(\log(n))$ vertices outside of $\mathcal{B}_0(R/2)$ are adjacent to all vertices in $\mathcal{B}_0(R/2)$, which finishes the proof. \blacktriangleleft

4.3 Upper Bound on the Clique Number

Recall that two vertices u, v are adjacent if and only if $d_h(u, v) \leq R$ and that we call the clique formed in $\mathcal{B}_0(R/2)$ the core whose size is $\sigma(G) = (1 - o(1))e^{-\alpha C/2}n^{1-\alpha}$ w.e.h.p. The core size is a lower bound for the clique number $\omega(G)$. We have established that the largest clique is smaller than the degeneracy w.e.h.p. (Theorem 13), and in this section we further investigate an upper bound for $\omega(G)$. We note that the upper bound we derive in this section implies Theorem 13 for α large enough (see Figure 1). However, for smaller α , the upper bound for $\omega(G)$ is larger than the lower bound (Theorem 9) for $\kappa(G)$ in the HRG model, and thus does not directly imply Theorem 13 for these values for α .

Before going into details, we lay out our proof strategy. We aim to bound the region where a clique can be located. Since vertices are adjacent if and only if their (hyperbolic) distance is at most R , this can be done by characterising a shape that covers any hyperbolic region of diameter R . A classic result by Jung [20] answers the question of how large the radius of a ball in Euclidean space needs to be at most, so that its interior can contain an entire set of points of fixed diameter. The hyperbolic version of this result was discovered by Dekster [14, 15] nearly a century later. He extended Jung's result to (among other geometries) hyperbolic space and we apply it as follows: we identify $\mathcal{O}(n^3)$ many balls where one of these balls contains the clique of largest size $\omega(G)$. This clique (and all the other identified cliques) needs to be located in a ball $B_x(r)$ with radius r large enough. We use the hyperbolic variant of Jung's theorem to upper bound r which, in turn, allows us to upper bound the area of this ball. This yields an upper bound for the amount of vertices one such ball $B_x(r)$ could contain w.e.h.p., leading to an upper bound for $\omega(G)$. Since we only need to consider at most $\mathcal{O}(n^3)$ balls, a union bound is sufficient to derive the same bound for the worst case. We work with the following version of Jung's theorem for hyperbolic geometry.

► **Theorem 17.** [14, Theorem 2] *Let $\mathcal{K} \subset \mathbb{H}^d$ be compact and suppose that for any $y, z \in \mathcal{K}$, $d_h(y, z) \leq D$. Then there exists $x \in \mathbb{H}^d$ such that $\mathcal{K} \subseteq \mathcal{B}_x(r)$ for r satisfying*

$$D \geq 2 \sinh^{-1} \left(\sqrt{\frac{d+1}{2d}} \sinh(r) \right).$$

In the hyperbolic plane \mathbb{H}^2 , this simplifies to the following.

► **Corollary 18.** *Let $\mathcal{K} \subset \mathbb{H}^2$ be compact and suppose that for any $y, z \in \mathcal{K}$, $d_h(y, z) \leq R$. Then there exists $x \in \mathbb{H}^2$ such that $\mathcal{K} \subseteq \mathcal{B}_x(r)$ for r satisfying $r \leq R/2 + \log(2/\sqrt{3})$.*

Proof. Using that $d = 2$, we directly get from Theorem 17 for diameter R that $\frac{R}{2} \geq \sinh^{-1}(\sqrt{3/4} \sinh(r))$. Rearranging and using for $x \in \mathbb{R}$ that $\sinh(x) = \frac{1}{2}e^x(1 - e^{-2x})$ yields

$$R/2 - r \geq \log(\sqrt{3/4}) + \log \left(\frac{(1 - e^{-2r})}{(1 - e^{-R})} \right).$$

Solving for r and using that $r \geq R/2 \geq \log(n) + C/2$ in conjunction with recalling that $C \in \Theta(1)$, it follows that $r \leq R/2 + \log(2/\sqrt{3})$. ◀

Our next observation follows from the definition of μ , and formalises the intuition that μ puts more mass at the centre of the disk.

► **Observation 19.** *Let $0 < r \leq R$ and $u, v \in \mathcal{D}_R$ with $r(u) \geq r(v)$. Then $\mu(\mathcal{B}_u(r)) \leq \mu(\mathcal{B}_v(r))$.*

Recall that $\sigma(G)$ denotes the core size $|V \cap \mathcal{B}_0(R/2)|$ which is a lower bound for the clique number $\omega(G)$ (see Observation 2), and that $\sigma(G) = (1 - o(1))e^{-\alpha C/2}n^{1-\alpha}$ w.e.h.p. We state our upper bound relative to this lower bound.

► **Theorem 20** (Clique upper bound). *Let $G \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG with $\alpha \in (1/2, 1)$. Then w.e.h.p.*

$$\omega(G) \leq \left((4/3)^{\alpha/2} + o(1) \right) \sigma(G).$$

13:16 Hyperbolic Random Graphs: Clique Number and Degeneracy

Proof. Consider any triplet of vertices $u, v, w \in V$ with pairwise distance at most R , so that they are pairwise adjacent. Since u, v, w are a.s. in general position, there is a unique ball $\mathcal{B}_x(r)$ such that u, v, w lie on the boundary $\partial\mathcal{B}_x(r)$. By Corollary 18, $r \leq R/2 + \log(\sqrt{4/3})$ since $\max(d_h(u, v), d_h(v, w), d_h(u, w)) \leq R$. Over all possible triplets $u, v, w \in V$ this gives us a set B of at most $\binom{n}{3}$ closed balls. Any clique must be contained in one of these balls, and therefore so is the largest clique. Thus upper bounding the number of vertices for each individual ball $\mathcal{B} \in B$ yields an upper bound on the size of the largest clique.

We now upper bound the expected number of vertices in one ball B . To this end we fix a ball $\mathcal{B} = \mathcal{B}_x(r) \in B$ and let $Y_{\mathcal{B}}$ be the random variable counting the number of vertices in \mathcal{B} . The balls in B are identically (though clearly not independently) distributed. Since vertices are thrown independently according to μ , we have that $Y_{\mathcal{B}} - 3 \sim \text{Bin}(n - 3, \mu(\mathcal{B}))$, and so

$$\begin{aligned} \mathbb{E}[Y_{\mathcal{B}}] &= 3 + (n - 3)\mu(\mathcal{B}) \leq 3 + (n - 3)\mu(\mathcal{B}_0(r)) && \text{(Observation 19)} \\ &\leq 3 + (n - 3)\mu(\mathcal{B}_0(R/2 + \log(2/\sqrt{3}))) && \text{(Corollary 18)} \\ &\leq ((2/\sqrt{3})^\alpha + o(1))e^{-\alpha C/2}n^{1-\alpha}. && \text{(Equation (2))} \end{aligned}$$

Thus $((4/3)^{\alpha/2} + o(1))\sigma(G) \geq (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]$ w.e.h.p. This is relevant to the bound in the theorem statement because it implies that

$$\begin{aligned} \mathbb{P}\left(\omega(G) > ((4/3)^{\alpha/2} + o(1))\sigma(G)\right) &\leq \mathbb{P}\left(\max_{\mathcal{B} \in B} Y_{\mathcal{B}} > ((4/3)^{\alpha/2} + o(1))\sigma(G)\right) \\ &\leq \mathbb{P}\left(\max_{\mathcal{B} \in B} Y_{\mathcal{B}} > (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]\right) + n^{-c} \end{aligned}$$

for arbitrary c . Thus to finish the proof we need to show concentration, which via a union bound over all triplets u, v, w will yield the result. To show concentration we apply a Chernoff bound Theorem 3. Using $\varepsilon = (1/\log(n))$ we obtain

$$\mathbb{P}(Y_{\mathcal{B}} > (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]) \leq e^{-\mathbb{E}[Y_{\mathcal{B}}]/(3(1/\log(n))^2)} \leq e^{-\Theta(1)(n^{1-\alpha})/\log^2(n)} \leq n^{-c}$$

for any choice of c , since for $\alpha < 1$, $\Theta(1)(n^{1-\alpha}) \in n^{\Theta(1)}$ and $\liminf_{n \rightarrow \infty} \frac{n^{\Theta(1)}}{\log^2(n)} \in \omega(\log(n))$. Finally, to show that this holds w.e.h.p. for all balls in B , we use that $|B| \leq \binom{n}{3} < n^3$, so that

$$\begin{aligned} \mathbb{P}\left(\max_{\mathcal{B} \in B} Y_{\mathcal{B}} \geq (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]\right) &\leq \sum_{\mathcal{B} \in B} \mathbb{P}(Y_{\mathcal{B}} \geq (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]) \\ &\leq n^3 \mathbb{P}(Y_{\mathcal{B}} \geq (1 + 1/\log(n))\mathbb{E}[Y_{\mathcal{B}}]) \leq n^{-c+3}. \quad \blacktriangleleft \end{aligned}$$

A further refinement of the ‘‘clique covering’’ argument of Theorem 20 should be possible. Any clique has by definition a diameter of at most R , and so the shape in \mathbb{H}^2 of diameter R with maximal area would provide an improved upper bound via a similar covering argument. It is not clear what a tight bound would be, and $\omega(G) \leq (1 + o(1))\sigma(G)$ may be possible.

5 Geometric Inhomogeneous Random Graphs

Geometric Inhomogeneous Random Graphs or *GIRGs* were introduced in [11] as an alternative model to HRGs that capture many of the same properties, in particular the power-law degree distribution. In their most general form, GIRGs strictly generalise HRGs, but they are more often studied in a slightly restricted form; comparisons are made in [6, 28]. In this restricted form, called the *standard* GIRG model by [16], any HRG G can be coupled with two GIRGs H_1 and H_2 such that $H_1 \subseteq G \subseteq H_2$, where \subseteq denotes graph inclusion.

Because of this relationship, GIRGs are used as proxies for HRGs in some theoretical and experimental works. This is partly done because GIRGs are (by design) far more tractable than HRGs. It is therefore valuable to understand differences between the two models. In [6] experimental evidence was given to suggest that the “sandwiching” of an HRG by two standard GIRGs is not tight. In Corollary 24 we provide a theoretical result demonstrating a difference between the two models.

► **Definition 21** (Standard GIRG model). *Let $\beta \in (2, 3)$, $\lambda \in \Theta(1)$, and $n \in \mathbb{N}$. A geometric inhomogeneous random graph $G \sim \mathcal{G}(n, \beta, \lambda)$ is a random graph with vertex set $V = \{v_1, \dots, v_n\}$ satisfying the following properties.*

1. *Every $u \in V$ is equipped with a random tuple (w_u, x_u) , where weight $w_u \in [1, \infty)$ has density $f(y) = (\beta - 1)y^{-\beta}$ and coordinate x_u is drawn uniformly at random from $[0, 1]$;*
2. *Any pair of vertices $u, v \in V$ are connected if and only if $\min\{|x_u - x_v|, 1 - |x_u - x_v|\} \leq t(u, v)$, where $t(u, v) = \frac{1}{2} \left(\frac{\lambda w_u w_v}{n} \right)$.*

One way of thinking of a GIRG is that vertices are being thrown uniformly at random onto the 1-dimensional torus \mathbb{T}^1 , and connected according to whether their distance is below their threshold $t(u, v)$. The weights are drawn according to a Pareto distribution. Analogously to HRGs, for a vertex u of a GIRG we define the inner-degree of u to be $|\Gamma(u)| = |\{v \in V \mid u \text{ and } v \text{ are connected and } w_v \geq w_u\}|$. The proofs of Observation 4 and Lemma 6 can be adapted to the GIRG model to characterise the degeneracy via the largest inner-degree.

► **Corollary 22.** *Let $G \sim \mathcal{G}(n, 2\alpha + 1, \lambda)$ be a standard GIRG. Consider the vertex u^* with the largest inner-degree in G . Then w.e.h.p. $\kappa(G) = (1 - o(1))|\Gamma(u^*)|$.*

Corollary 22 allows us to state a tight bound for the degeneracy in comparison to the core of the GIRG, which is defined to contain all vertices of weight $\hat{w} \geq \sqrt{n/\lambda}$, and has size $\sigma(G) = (1 \pm o(1))\lambda^{-\alpha}n^{1-\alpha}$ w.e.h.p. This is analogous to the core of an HRG, which is the clique formed by vertices of radius at most $R/2$, regardless of their angular coordinates.

► **Theorem 23.** *Let $G \sim \mathcal{G}(n, 2\alpha + 1, \lambda)$ be a threshold GIRG. The degeneracy is w.e.h.p.*

$$\kappa(G) = (2 \pm o(1))(2(1 - \alpha))^{(1-\alpha)/(2\alpha-1)}\sigma(G).$$

Proof. We bound the maximal inner-degree $|\Gamma(u^*)|$; the statement then follows from Corollary 22. Notice that, independent of the geometric distance, a vertex with weight w is adjacent to any vertex with weight w' if $w' \geq \frac{n}{w\lambda}$ since $t(w, w') = \frac{\lambda w w'}{2n} \geq \frac{\lambda w \frac{n}{w\lambda}}{2n} = \frac{1}{2}$ which is the maximal distance between two points in the unit torus. Thus, using $\beta = 2\alpha + 1$, the probability that a vertex v is in the inner-neighbourhood of a vertex u with weight w is

$$\begin{aligned} \mathbb{P}(v \in \Gamma(u)) &= \mathbb{P}(\{\{u, v\} \in E\} \cap \{W_v \geq w\}) \\ &= \mathbb{P}\left(W_v \geq \frac{n}{w\lambda}\right) + 2 \int_w^{\frac{n}{w\lambda}} t(y, w) 2\alpha y^{-(2\alpha+1)} dy && (t(w, n/w\lambda) = 1/2) \\ &= \left(\frac{n}{w\lambda}\right)^{-2\alpha} + \frac{2\alpha\lambda w}{n} \int_w^{\frac{n}{w\lambda}} y^{-2\alpha} dy && (\text{Pareto and threshold}) \\ &= \left(\frac{n}{w\lambda}\right)^{-2\alpha} + \frac{2\alpha\lambda w}{n(2\alpha-1)} \left(w^{1-2\alpha} - \left(\frac{n}{w\lambda}\right)^{1-2\alpha}\right) && \left(\int y^{-2\alpha} dy = \left[\frac{y^{1-2\alpha}}{1-2\alpha}\right]\right) \\ &= \left(\frac{n}{w\lambda}\right)^{-2\alpha} + \frac{\alpha}{\alpha-1/2} \left(\frac{\lambda w^{2(1-\alpha)}}{n} - \left(\frac{n}{w\lambda}\right)^{-2\alpha}\right). && (4) \end{aligned}$$

13:18 Hyperbolic Random Graphs: Clique Number and Degeneracy

Next, we calculate the value w^* , which maximises the expected inner-degree. To this end, we consider the probability measure of the inner-neighbourhood, take its derivative with respect to w and set it equal to 0. Differentiating yields

$$\frac{d}{dw} \mathbb{P}(v \in \Gamma(u)) = \frac{2(nw)^{-(2\alpha+1)}(1-\alpha)\alpha(n^{2\alpha}w^2\lambda - nw^{4\alpha}\lambda^{2\alpha})}{2\alpha-1},$$

and solving for the maximum reveals $w^* = (2(1-\alpha))^{\frac{1}{4\alpha-2}} \sqrt{\frac{n}{\lambda}} \in \Theta(\sqrt{n})$.

We plug in w^* for the weight of u denoted by u^* into $\mathbb{P}(v \cap \Gamma(u^*))$ and get by (4) that

$$\mathbb{P}(v \in \Gamma(u^*)) = 2(2(1-\alpha))^{(1-\alpha)/(2\alpha-1)}(n\lambda)^{-\alpha}.$$

Recalling that $\sigma(G) = (1 \pm o(1))\lambda^{-\alpha}n^{1-\alpha}$ w.e.h.p., the upper bound now follows from the expectation of $|\Gamma(u^*)|$ and applying a Chernoff bound in conjunction with a union bound. The lower bound is established by showing that there exists a vertex within the range of weights $\tilde{w} = [w^*(1+n^{\alpha-1}\log^2(n))^{-1/(2\alpha)}, w^*]$ w.e.h.p. and lower bound the inner-degree of such vertex. Using the Pareto distribution and $w^* \in \Theta(\sqrt{n})$, we calculate the probability for a vertex u to belong to the range of weights \tilde{w} . We obtain

$$\begin{aligned} \mathbb{P}(W_u \in \tilde{w}) &= \mathbb{P}\left(w^*(1+n^{\alpha-1}\log^2(n))^{-1/(2\alpha)} \leq W_u \leq w^*\right) && \text{(Range of } \tilde{w}\text{)} \\ &= \mathbb{P}(W_u \leq w^*) - \mathbb{P}\left(W_u \leq w^*(1+n^{\alpha-1}\log^2(n))^{-1/(2\alpha)}\right) \\ &= 1 - (w^*)^{-2\alpha} - (1 - (w^*(1+n^{\alpha-1}\log^2(n))^{-1/(2\alpha)})^{-2\alpha}) && \text{(Pareto)} \\ &= (w^*(1+n^{\alpha-1}\log^2(n))^{-1/(2\alpha)})^{-2\alpha} - (w^*)^{-2\alpha} \\ &= (w^*)^{-2\alpha}n^{\alpha-1}\log^2(n) \\ &= \Theta(1)\frac{\log^2(n)}{n}. && (w^* \in \Theta(\sqrt{n})) \end{aligned}$$

By this we have $\mathbb{E}[|V \cap \tilde{w}|] \in \omega(\log(n))$. Using a Chernoff bound there exists a vertex within the desired weight range \tilde{w} w.e.h.p. To conclude the proof we lower bound the inner-degree of a vertex \tilde{u} included in the weight range \tilde{w} . Note that $\tilde{w} = (1-o(1))w^* = (2-o(1)(1-\alpha))^{\frac{1}{4\alpha-2}}\sqrt{\frac{n}{\lambda}}$. We then have via Equation (4)

$$\mathbb{E}[|\Gamma(\tilde{u})|] = (n-1)\mathbb{P}(v \cap \Gamma(\tilde{u})) \geq (2-o(1))(2(1-\alpha))^{(1-\alpha)/(2\alpha-1)}\lambda^{-\alpha}n^{1-\alpha}.$$

A final application of a Chernoff bound then ensures the concentration to finish the proof. ◀

Comparing the lower bound of the degeneracy for GIRGs given in Theorem 23 to the upper bound of a HRG we obtained in Theorem 9 we draw the conclusion that the degeneracy-to-core ratio between the two models is fundamentally different.

► **Corollary 24** (GIRG-HRG degeneracy difference). *Fix an $\alpha \in (1/2, 1)$. Let $G \sim \mathcal{G}(n, 2\alpha+1, \lambda)$ be a standard GIRG and $H \sim \mathcal{G}(n, \alpha, C)$ be a threshold HRG. Then w.e.h.p.*

$$\left| \frac{\kappa(G)}{\sigma(G)} - \frac{\kappa(H)}{\sigma(H)} \right| \in \Theta(1).$$

6 Conclusion

We have shown that the clique number, degeneracy, and chromatic number of HRGs are asymptotically (with small differences in the leading O -notation constants) as large as the core, though the clique number and degeneracy differ significantly. Our upper bound on

the degeneracy provides a constant factor approximation algorithm for the graph colouring problem. The approximation ratio ranges from $2/\sqrt{3}$ to $4/3$ and depends on the model parameter α . This raises several open questions and future research directions.

- Is the chromatic number bounded away from the degeneracy, the clique number, or both?
- Can HRGs be coloured optimally in polynomial time or is it NP-complete?
- What are the asymptotics of $\omega(G)/\sigma(G)$? Is the clique number a constant factor larger than the core and has similar behaviour as the degeneracy?

There are further directions of research such as determining other differences between HRGs and GIRGs or designing colouring algorithms for HRGs in various models of computation.

References

- 1 Samuel Baguley, Yannic Maus, Janosch Ruff, and George Skretas. Hyperbolic random graphs: Clique number and degeneracy with implications for colouring. *CoRR*, 2024. doi:10.48550/arXiv.2410.11549.
- 2 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Morgan & Claypool Publishers, 2013. doi:10.1007/978-3-031-02009-4.
- 3 Thomas Bläsius and Philipp Fischbeck. On the external validity of average-case analyses of graph algorithms. *ACM Trans. Algorithms*, 2024. doi:10.1145/3633778.
- 4 Thomas Bläsius, Philipp Fischbeck, Tobias Friedrich, and Maximilian Katzmann. Solving vertex cover in polynomial time on hyperbolic random graphs. *Theory Comput. Syst.*, 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*, 2023. doi:10.1007/S00453-023-01143-X.
- 6 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently Generating Geometric Inhomogeneous and Hyperbolic Random Graphs. In *ESA*, 2019. doi:10.4230/LIPIcs.ESA.2019.21.
- 7 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, and Daniel Stephan. Strongly Hyperbolic Unit Disk Graphs. In *STACS*, 2023. doi:10.4230/LIPIcs.STACS.2023.13.
- 8 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Cliques in hyperbolic random graphs. *Algorithmica*, 2017. doi:10.1007/s00453-017-0323-3.
- 9 Thomas Bläsius, Maximilian Katzmann, and Clara Stegehuis. Maximal cliques in scale-free random graphs. *Network Science*, 2024. doi:10.1017/nws.2024.13.
- 10 Thomas Bläsius, Tobias Friedrich, and Anton Krohmer. Hyperbolic random graphs: Separators and treewidth. In *ESA*, 2016. doi:10.4230/LIPIcs.ESA.2016.15.
- 11 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theoretical Computer Science*, 2019. doi:10.1016/j.tcs.2018.08.014.
- 12 Aleksander Bjørn Grodt Christiansen, Krzysztof Nowicki, and Eva Rotenberg. Improved dynamic colouring of sparse graphs. In *STOC*, 2023. doi:10.1145/3564246.3585111.
- 13 Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, 1971. doi:10.1145/800157.805047.
- 14 B. V. Dekster. The Jung theorem for spherical and hyperbolic spaces. *Acta Mathematica Hungarica*, 1995. doi:10.1007/bf01874495.
- 15 B. V. Dekster. The Jung theorem in metric spaces of curvature bounded above. *Proceedings of the American Mathematical Society*, 1997. doi:10.1090/s0002-9939-97-03842-2.
- 16 Tobias Friedrich, Andreas Göbel, Maximilian Katzmann, and Leon Schiller. Cliques in high-dimensional geometric inhomogeneous random graphs. *SIAM Journal on Discrete Mathematics*, 2024. doi:10.1137/23m157394x.
- 17 Tobias Friedrich and Anton Krohmer. On the Diameter of Hyperbolic Random Graphs. *SIAM Journal on Discrete Mathematics*, 2018. doi:10.1137/17M1123961.

- 18 Mohsen Ghaffari and Christoph Grunau. Dynamic α (arboricity) coloring in polylogarithmic worst-case time. In *STOC*, 2024. doi:10.1145/3618260.3649782.
- 19 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random Hyperbolic Graphs: Degree Sequence and Clustering. In *ICALP*, 2012. doi:10.1007/978-3-642-31585-5_51.
- 20 Heinrich Jung. Ueber die kleinste Kugel, die eine räumliche Figur einschliesst. *Journal für die reine und angewandte Mathematik*, 1901. URL: <http://eudml.org/doc/149122>.
- 21 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 22 Maximilian Katzmann. *About the analysis of algorithms on networks with underlying hyperbolic geometry*. doctoralthesis, Universität Potsdam, 2023. doi:10.25932/publishup-58296.
- 23 Ralph Keusch. *Geometric Inhomogeneous Random Graphs and Graph Coloring Games*. PhD thesis, ETH Zurich, 2018. doi:10.3929/ethz-b-000269658.
- 24 Marcos Kiwi and Dieter Mitsche. A bound for the diameter of random hyperbolic graphs. In *ANALCO*, 2015.
- 25 Marcos Kiwi and Dieter Mitsche. Spectral gap of random hyperbolic graphs and related parameters. *The Annals of Applied Probability*, 2018. doi:10.1214/17-aap1323.
- 26 Marcos Kiwi and Dieter Mitsche. On the second largest component of random hyperbolic graphs. *SIAM Journal on Discrete Mathematics*, 2019. doi:10.1137/18m121201x.
- 27 Marcos Kiwi, Markus Schepers, and John Sylvester. Cover and hitting times of hyperbolic random graphs. *Random Structures & Algorithms*, 2024. doi:10.1002/rsa.21249.
- 28 Júlia Komjáthy and Bas Lodewijks. Explosion in weighted hyperbolic random graphs and geometric inhomogeneous random graphs. *Stochastic Processes and their Applications*, 2020. doi:10.1016/j.spa.2019.04.014.
- 29 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 2010. doi:10.1103/PhysRevE.82.036106.
- 30 Anton Krohmer. *Structures & algorithms in hyperbolic random graphs*. doctoralthesis, Universität Potsdam, 2016. URL: <https://publishup.uni-potsdam.de/frontdoor/index/index/docId/39597>.
- 31 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 1983. doi:10.1145/2402.322385.
- 32 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi:10.1017/CB09780511813603.
- 33 Tobias Müller and Merlijn Staps. The Diameter of KPKVB Random Graphs. *Advances in Applied Probability*, 2019. doi:10.1017/apr.2019.23.
- 34 Fragkiskos Papadopoulos, Dmitri Krioukov, Marian Boguna, and Amin Vahdat. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *INFOCOM*, 2010. doi:10.1109/infcom.2010.5462131.
- 35 Jose L. Walteros and Austin Buchanan. Why is maximum clique often easy in practice? *Operations Research*, 2020. doi:10.1287/opre.2019.1970.
- 36 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 2007. doi:10.4086/toc.2007.v003a006.

Multivariate Exploration of Metric Dilation

Aritra Banik ✉ 


National Institute of Science, Education and Research,
An OCC of Homi Bhabha National Institute, Bhubaneswar, India

Fedor V. Fomin ✉ 

University of Bergen, Norway

Petr A. Golovach ✉ 

University of Bergen, Norway

Tanmay Inamdar ✉ 

Indian Institute of Technology Jodhpur, India

Satyabrata Jana ✉ 

University of Warwick, UK

Saket Saurabh ✉ 

The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway

Abstract

Let G be a weighted graph embedded in a metric space (M, d_M) . The vertices of G correspond to the points in M , with the weight of each edge uv being the distance $d_M(u, v)$ between their respective points in M . The dilation (or stretch) of G is defined as the minimum factor t such that, for any pair of vertices u, v , the distance between u and v – represented by the weight of a shortest u, v -path – is at most $t \cdot d_M(u, v)$. We study DILATION t -AUGMENTATION, where the objective is, given a metric M , a graph G , and numerical values k and t , to determine whether G can be transformed into a graph with dilation t by adding at most k edges.

Our primary focus is on the scenario where the metric M is the shortest path metric of an unweighted graph Γ . Even in this specific case, DILATION t -AUGMENTATION remains computationally challenging. In particular, the problem is W[2]-hard parameterized by k when Γ is a complete graph, already for $t = 2$. Our main contribution lies in providing new insights into the impact of combinations of various parameters on the computational complexity of the problem. We establish the following.

- The parameterized dichotomy of the problem with respect to dilation t , when the graph G is sparse: Parameterized by k , the problem is FPT for graphs excluding a biclique $K_{d,d}$ as a subgraph for $t \leq 2$ and the problem is W[1]-hard for $t \geq 3$ even if G is a forest consisting of disjoint stars.
- The problem is FPT parameterized by the combined parameter $k + t + \Delta$, where Δ is the maximum degree of the graph G or Γ .

2012 ACM Subject Classification Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Metric dilation, geometric spanner, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.14

Related Version *Full Version*: <https://arxiv.org/abs/2501.04555> [2]

Funding *Fedor V. Fomin*: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).

Petr A. Golovach: Supported by the Research Council of Norway via the project BWCA (grant no. 314528).

Tanmay Inamdar: Supported by IITJ Research Initiation Grant (grant number I/RIG/T-NI/20240072).

Satyabrata Jana: Supported by the Engineering and Physical Sciences Research Council (EPSRC) via the project MULTIPROCESS (grant no. EP/V044621/1).



© Aritra Banik, Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Satyabrata Jana, and Saket Saurabh;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 14; pp. 14:1–14:17



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



Saket Saurabh: The author is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416); and he also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

Consider a finite metric space $\mathcal{M} = (V, d_M)$, and let G be a sparse weighted graph with vertices corresponding to the points in V . The weights assigned to the edges of G represent the distances in \mathcal{M} between their end-points. That is, the graph $G = (V, E)$ is embedded in a metric space $\mathcal{M} = (V, d_M)$. The graph G is called a t -spanner if, for every pair of vertices $u, v \in V$, the distance between them in G is at most $t \cdot d_M(u, v)$. The concept of spanners, introduced by Peleg and Schäffer [19], has evolved into a fundamental tool in various domains, including algorithms, distributed computing, networking, data structures and metric geometry, as highlighted in [18]. The minimum number t for which G is a t -spanner defines the *stretch* or *dilation* of G .

In their book [18, p.474, Problem 9], Narasimhan and Smid presented the problem of enhancing the dilation of a spanner by adding at most k edges: Develop an efficient algorithm to identify the $k > 1$ edges that minimize (or approximately minimize) the stretch factor of the resulting geometric graph. Formally, the problem is defined as follows.

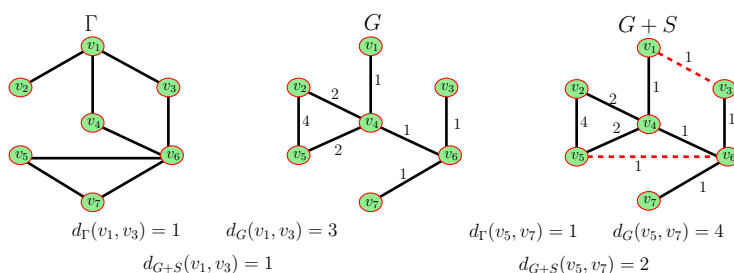
DILATION t -AUGMENTATION

Input: A graph $G = (V, E)$ embedded in a metric space $\mathcal{M} = (V, d_M)$ and an integer k .

Question: Does there exist a set of k edges $S \subseteq V \times V$ such that the dilation of $G' = (V, E \cup S)$ is at most t ?

The case where $k = 1$, was investigated by Farshi, Giannopoulos, Gudmundsson [8], Luo, Wulff-Nilsen [17], and Wulff-Nilsen [20]. However, for the more general scenario where $k > 1$, the problem becomes significantly more challenging and poorly understood. Giannopoulos et al. [12] and Gudmundsson and Smid [13] demonstrated that obtaining the best dilation spanner by adding k edges to an empty graph is already NP-hard. Gudmundsson and Wong [14] proposed an algorithm that in $\mathcal{O}(n^3 \log n)$ time, identifies k edges whose addition to G results in a graph with a stretch factor within $\mathcal{O}(k)$ of the minimum stretch factor. Related problems have also been studied in geometric settings, e.g., Aronov et al. [1], who showed that, given a set S of n points in \mathbb{R}^2 , and an integer $0 \leq k < n$, we can construct a *geometric graph* with vertex set S , at most $n - 1 + k$ edges, maximum degree five, and dilation $\mathcal{O}(n/(k + 1))$ in time $\mathcal{O}(n \log n)$.

We approach the DILATION t -AUGMENTATION problem from the perspective of Multivariate (Parameterized) Complexity – another popular paradigm to deal with intractable problems [4, 6, 10]. The two most natural parameters associated with the problem are the size of the solution k (the size of the augmented set) and the stretch factor t . We are looking for an algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$ or $g(k, t) \cdot n^{\mathcal{O}(1)}$. Here, $n = |V(G)|$. These algorithms are called fixed-parameter tractable (FPT) algorithms. There is also an accompanying theory of W -hardness that allows us to show that problems do not admit FPT algorithms (see [4, 6, 10] for further details). The problem of finding a t -spanner for a given graph is considered from the perspective of parameterized complexity in [15, 16]. Observe that, DILATION t -AUGMENTATION admits an algorithm with running time $n^{\mathcal{O}(k)}$ – try all possible subsets of size k of $V \times V$ as S . Thus, this naturally leads to the following.



■ **Figure 1** An instance of DILATION 2-AUGMENTATION with $k = 2$. The edges of the solution S are shown in dashed red. The edge weights in G are derived from the corresponding shortest path in Γ .

Does DILATION t -AUGMENTATION admit an FPT algorithm?

A simple result shows that, in its full generality, the problem is $W[2]$ -hard. In fact, consider $\mathcal{M} = (V, d_M)$ derived from an unweighted clique K on n vertices. That is, the vertices of the metric correspond to the vertices of K , and $d_M(u, v)$ is the length of shortest path between u and v in K , which is 1. Then, for $t = 2$, DILATION 2-AUGMENTATION corresponds to adding edges to G so that the diameter of the augmented graph becomes 2 (see Figure 1 for an illustration). This is the well-known DIAMETER 2-AUGMENTATION problem which is known to be $W[2]$ -hard [11]. Thus, we do not expect that DILATION t -AUGMENTATION to admit an algorithm with running time $g(k) \cdot n^{\mathcal{O}(t)}$. This simple hardness result motivates the following set of questions:

- For which metrics \mathcal{M} , does DILATION t -AUGMENTATION admit an FPT algorithm?
- For which family of input graphs \mathcal{G} , does DILATION t -AUGMENTATION admit an FPT algorithm?
- For which pairs of family of input graphs and metric, $(\mathcal{G}, \mathcal{M})$, does DILATION t -AUGMENTATION admit an FPT algorithm?

In this article, we specifically concentrate on the fundamental and non-trivial scenario where the metric \mathcal{M} corresponds to the shortest-path metric of an unweighted graph. Our contribution represents a substantial addition to the limited body of literature that addresses the challenging problem of DILATION t -AUGMENTATION.

1.1 Our Model, Results, and Methods

Throughout this paper, we deal with the shortest-path metric derived from an unweighted graph, unless otherwise mentioned.¹ For an edge weighted graph H , let $d_H(u, v)$ denote the *weighted shortest path distance* in the graph H between two vertices u and v . Additionally, we use $\text{hop}_H(u, v)$ to denote the length of the shortest path between u and v when we forget the edge weights. In other words, this denotes the length of the shortest path in H when all edges receive weight 1. We will use d_H and hop_H *only when H is edge weighted*; otherwise, both d_H and hop_H represent the same thing. Throughout the article, the metric \mathcal{M} will be derived from an undirected unweighted graph $\Gamma = (V, E)$. That is, the vertices of the metric correspond to the vertices of Γ , and $d_M(u, v)$ is assigned the shortest path distance in Γ , between u and v . Observe that $d_M(u, v) = d_{\Gamma}(u, v) = \text{hop}_{\Gamma}(u, v)$. We will use (G, Γ, k) as an

¹ Any metric can be derived from an edge weighted graph. Thus, our assumption represents a simplification.

instance of DILATION t -AUGMENTATION problem. Since \mathcal{M} is derived from Γ , for ease of notation, we use $d_\Gamma(u, v)$ instead of $d_M(u, v)$. Furthermore, recall that G is embedded in the metric space \mathcal{M} derived from Γ . That is, G is an edge-weighted graph, where the weights assigned to the edges of G represent the distances in Γ between their end-points. That is, $w(u, v) = d_\Gamma(u, v)$ and $d_G(u, v)$ denote the *weighted shortest path distance* in G .

The starting point of our research is the result of Peleg and Schäffer [19] that DILATION t -AUGMENTATION is NP-hard for $t = 2$, where G is an empty graph and the metric \mathcal{M} is derived from an undirected graph Γ . So, a natural question is whether this special subcase is FPT. Starting with this question, we trace the boundaries of the DILATION t -AUGMENTATION problem by instantiating different graph families to which Γ can belong or by instantiating different graph families to which G can belong. Our results consist of the following.

1. Our main algorithmic result is an FPT algorithm for DILATION 2-AUGMENTATION when G belongs to the family of $\mathcal{K}_{d,d}$ -free graphs. Recall that a graph G is $\mathcal{K}_{d,d}$ -free if it does not contain a complete bipartite graph with d vertices, each on both sides of the bipartition, as a subgraph. However, there is no restriction on Γ .

We complement this result by showing that this result cannot be extended for $t = 3$. Indeed, we show that when G is a disjoint union of a star and an independent set, and Γ is an arbitrary graph, then DILATION 3-AUGMENTATION is W[1]-hard. In a slight converse we also show that DILATION 3-AUGMENTATION is W[2]-hard when Γ is a star, and G is an arbitrary graph. On the other hand, in a generalization of the latter setting when Γ is a tree, we show that DILATION 2-AUGMENTATION is polynomial-time solvable.

2. Observe that for the families of stars we cannot bound the maximum degree of each graph uniformly. We show that if G (resp. Γ) belongs to the family of graphs with a maximum degree at most d and Γ (resp. G) is an arbitrary graph, then DILATION t -AUGMENTATION admits an algorithm FPT with running time $f(k, t, d) \cdot n^{\mathcal{O}(1)}$.

We complement this result by showing that this result cannot be extended for the weighted metric. That is, when G belongs to the family of graphs of maximum degree 3 and the graph Γ is edge-weighted, then the problem becomes intractable. In fact, the edges of Γ get weights from the two-size set $\{1, w\}$. We show that DILATION $(2 + \epsilon)$ -AUGMENTATION is W[1]-hard in this case.

It is important to note that the family of $\mathcal{K}_{d,d}$ -free graphs includes trees, planar graphs, graphs that exclude a fixed graph H as a minor, graphs of bounded expansion, nowhere dense graphs, and graphs of bounded degeneracy.² We refer to Table 1 for a quick reference of all the results shown in this article. Our algorithmic results are based on the structural interaction between Γ and G . We start by defining the notion of conflict pairs (a pair (u, v) in G such that $d_G(u, v) > t \cdot d_\Gamma(u, v)$) and show that to resolve these pairs it suffices to focus on those pairs that are adjacent in Γ (that is, $(u, v) \in E(\Gamma)$). We call them adjacent conflicts. Resolving adjacent conflict pairs is the key to our results, both algorithms and hardness.

Our main result regarding DILATION 2-AUGMENTATION when G is $\mathcal{K}_{d,d}$ -free is given in Section 3. Our algorithm can be broadly divided into three separate phases. First, in Section 3.1 we define the notion of a *conflict graph* and bound the size of the minimum vertex cover in this graph using the restrictions imposed on the solution for DILATION t -AUGMENTATION in the special case of $t = 2$. Then, in Section 3.2, we analyze the interplay

² For a formal definition of many of these graph classes, we refer to standard texts on graph theory, e.g., Diestel [5].

■ **Table 1** Complexity of the DILATION t -AUGMENTATION for different G , metric, parameters and t .

G	Metric (Γ)	Parameter(s)	t	Complexity
$\mathcal{K}_{d,d}$ -free	General	$k + d$	2	FPT (Section 3)
Star forest	General	k	3	W[1]-hard (Section 5.1)
General	Star	k	3	W[1]-hard (Section 5.2)
General	Tree	k	2	P (♠)
General	Bounded degree	$k + t + \Delta$	t	FPT (Section 4.1)
Bounded degree	General	$k + t + \Delta$	t	FPT (Section 4.2)
Subcubic	$(1, w)$ -weighted	k	$2 + \epsilon$	W[1]-hard (♠)
Edgeless	General	k	2	NP-hard (♠)
General	Clique	k	2	W[2]-hard (♠)

between the bounded-size vertex cover, along with the structural properties of G due to the fact that it is $\mathcal{K}_{d,d}$ -free, and use it to design a recursive algorithm that tries to guess a certain kind of edges that must belong to any solution in a YES-instance. At the leaves of this recursive procedure, we can upper bound the total number of vertices involved in conflicts. Nevertheless, we cannot get rid of all other vertices since they may be crucial for certain shortest paths. Finally, in Section 3.3 we bound the size of the instance by carefully eliminating a subset of conflict-free vertices, at which point we can enumerate all possible solutions. For the other two algorithmic results, we obtain a small set of V that contains all the end-points of edges of the solution. The hardness results are shown via parameter preserving reductions from known W-hard problems, using classical gadget constructions. The details of the results/statements marked with ♠ can be found in the full version of the paper [2].

Notations. Let $[n]$ be the set of integers $\{1, \dots, n\}$. For any graph G , we denote the set of vertices of G by $V(G)$ and the set of edges by $E(G)$. We denote the set of non-edges by $\overline{E}(G)$, that is, $\overline{E}(G) = \binom{V(G)}{2} \setminus E(G)$, where $\binom{V(G)}{2}$ is the set of all unordered pairs of distinct vertices in $V(G)$. For notational convenience, even though our edges/non-edges are undirected, we use the notation (a, b) instead of $\{a, b\}$ – note that due to this convention, $(a, b) = (b, a)$. A graph $G = (V, E)$ is said to be a *star*, if there exists a vertex $c \in V(G)$ such that $E = \{(u, c) : u \in V(G) \setminus \{c\}\}$. In this case, we say that c is the *center* of the star. Vertices of $V(G) \setminus \{c\}$ (if any) are said to be the *leaves* of the star. If every connected component of a graph G is a star, then we say that G is a *star forest*.

For a graph $G = (V, E)$ and a subset S of edges in $\overline{E}(G)$, we use the notation $G + S$ to mean the graph $G' = (V, E \cup S)$. For a graph G , a path is a sequence of distinct vertices $v_1 \dots v_\ell$ such that $(v_i, v_{i+1}) \in E(G)$ for $i \in [\ell - 1]$ and this path is denoted by $\langle v_1, \dots, v_\ell \rangle$. For an undirected unweighted graph H , we use $\text{hop}_H(u, v)$ to denote the length of the shortest path between u and v . For a graph H , a vertex v and an integer ℓ , $N_H^\ell(v)$ denotes all vertices w such that $\text{hop}_H(v, w) \leq \ell$. Similarly, for a graph H , a vertex subset W and an integer ℓ , $N_H^\ell(W) = \cup_{w \in W} N_H^\ell(w)$.

2 Conflict versus Adjacent Conflicts

Let (G, Γ, k) be an instance of DILATION t -AUGMENTATION problem. Two vertices u and v in G are in t -conflict if $d_G(u, v) > t \cdot d_\Gamma(u, v)$. Furthermore, two vertices are said to be in *adjacent t -conflict* with each other, if u and v are in conflict and u and v are adjacent in Γ . We say that a graph G is (*adjacent*) t -conflict-free if there is no pair of vertices (u, v) such that u and v are in (*adjacent*) t -conflict. Throughout the discussion, the value of t will be clear from the context, so we use the terms *conflict/conflict-free* instead of *t -conflict/ t -conflict-free*. The following lemma shows the relationship between conflict-free and adjacent conflict-free.

► **Lemma 1** (♠). *For any set of edges S , $G + S$ is conflict-free iff $G + S$ is adjacent conflict-free.*

For our problem, we add edges to G to make it conflict-free. Lemma 1 shows that for this purpose it is sufficient to focus *only* on adjacent conflicts. Thus, from now on, we only focus on adjacent conflicts. Unless explicitly mentioned otherwise, by *conflict*, we mean adjacent conflict. Nevertheless, we may use *adjacent conflict* at some places to emphasize the adjacent-ness of the conflict.

The next result formalizes the fact that the distances in G are lower bounded by the distances in Γ . The proof follows from definition, and hence omitted.

► **Observation 2.** *Let $G = (V, E)$ be a graph embedded in a metric space derived from the undirected graph Γ . Then for any pair of vertices x and y we have $d_\Gamma(x, y) \leq d_G(x, y)$.*

Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION and S be an (unknown) minimal solution. The set S is also called dilation t -augmentation set. We use V_S to denote the set of end-points of the edges in S . Let V_c denote the set of all vertices in G that are in conflict with some other vertex. In the next two sections, V_c and V_S will be used repeatedly.

3 Dilation 2-Augmentation for $\mathcal{K}_{d,d}$ -free Graphs

Let (G, Γ, k) be an instance of DILATION 2-AUGMENTATION. In this section, we consider the case where G belongs to the family of $\mathcal{K}_{d,d}$ -free graphs. Recall that a graph G is $\mathcal{K}_{d,d}$ -free if it does not contain a complete bipartite graph with d vertices each on both sides of the bipartition. However, there is no restriction on Γ .

3.1 Conflict Graph and Vertex Cover

We first define a *conflict graph* \mathbb{C} on the set of vertices $V(G)$, which captures all adjacent conflicts. We place an edge between two vertices u and v in \mathbb{C} if and only if u and v are in adjacent conflict with each other. Note that V_c (the set of vertices present in adjacent conflicts) is exactly the set of vertices with degree at least one in \mathbb{C} . The next result shows that every edge in $E(\mathbb{C})$ intersects some edge of a dilation 2-augmentation set S .

► **Lemma 3** (♠). *Let (G, Γ, k) be a YES-instance of DILATION 2-AUGMENTATION and let S be a minimal solution to the given instance. In addition, let \mathbb{C} be the corresponding conflict graph. Then, for any edge $(u, v) \in E(\mathbb{C})$, $|\{u, v\} \cap V_S| \geq 1$.*

Lemma 3 allows us to bound the size of a minimum vertex cover (set of vertices that intersects all the edges) of \mathbb{C} . To do so, we propose the next reduction rule and prove its correctness (or safety).

► **Reduction Rule 1.** *If the size of a maximum matching in \mathbb{C} is more than $2k$, return NO.*

► **Lemma 4** (♠). *Reduction Rule 1 is safe.*

First, we use a polynomial-time algorithm (e.g., [7]) to find the maximum matching M in \mathbb{C} and apply Reduction Rule 1. Notice that if the reduction rule 1 is not applicable, then $|M| \leq 2k$, which implies that it has a vertex cover of size at most $4k$ – in particular, we can take the set of end-points of the edges of in M to be such a vertex cover, call it R .

Let $I = V \setminus R$. Note that I induces an independent set in \mathbb{C} , but not necessarily in G . Next, we bound the degree of each vertex in R in \mathbb{C} . This will imply that the number of conflict pairs is bounded.

The set R is at most of size $4k$. For our algorithm at this stage, we “guess the edges of S that have both end-points in R ”. This results in an equivalent annotated instance, which is now formalized. We first define an “annotated” instance of the problem. An example of the annotated version of the problem is given by the tuple (G', Γ, k', R') , where we also provide a vertex cover R' of the corresponding conflict graph, which is used mainly for book keeping purposes. The task is still to find a dilation 2-augmentation set S of size at most k' . Here, we are not allowed to add an edge to a solution that does not have an end-point outside R .

Let \overline{E}_R be the set of non-edges in $G[R]$. For every subset $\emptyset \subseteq E_j \subseteq \overline{E}_R$ of size at most k , we create an instance \mathcal{I}_j of the annotated problem, as follows: $\mathcal{I}_j := (G_j, \Gamma, k_j, R)$ where $G_j = G + E_j$ and $k_j = k - |E_j|$. The following lemma is immediate.

► **Lemma 5.** *(G, Γ, k) is a YES-instance if and only if one of the instances in $\{(G_j, \Gamma, k_j, R) : \forall \emptyset \subseteq E_j \subseteq \overline{E}_R \text{ s.t. } k_j \leq k\}$ is a YES-instance.*

From now on, we assume that we have an annotated instance of DILATION 2-AUGMENTATION. That is, (G, Γ, k, R) is an instance of DILATION 2-AUGMENTATION and we are seeking a dilation 2-augmentation set S such that for any edge (x, y) in S , $|\{x, y\} \cap R| \leq 1$. That is, every edge added must contain at least one end-point that does not belong to R .

3.2 Bounding the Size of the V_c in Annotated Instances

We now describe how to solve the annotated version of the problem. As a first step, we give a recursive algorithm such that at the end we obtain instances such that the size of V_c gets upper-bounded by a function of k and d alone. To this end, and to make it easier to understand, we adapt the following technical definition of [3] in the context of our problem.

► **Definition 6** ([3]). *A decreasing FPT-Turing-reduction is an FPT algorithm which, given an annotated instance (G, Γ, k, R) , produces $\ell = g(k, d)$ annotated instances $(G_1, \Gamma, k_1, R_1), \dots, (G_\ell, \Gamma, k_\ell, R_\ell)$, such that: (1) (G, Γ, k, R) is a YES-instance iff one of the annotated instances in $\{(G_i, \Gamma, k_j, R_i) : 1 \leq j \leq \ell\}$ is a YES-instance, and (2) $k_j \leq k - 1$ for every $j \in [\ell]$.*

We now give a sequence of FPT Turing reductions to solve the problem. At a high level, we will iterate over the vertices in R that have high degrees in \mathbb{C} , and at each step we will add at least one vertex from I to R , and at least one new edge to G . Thus, in each step, the budget k reduces by at least one. At least one of the resulting instances will correspond to a “correct” set of choices. Note that initially the size of R is at most $4k$; however during the sequence of Turing FPT reductions, the size of R may increase. Nevertheless, we will maintain the invariant that $|R|$ remains bounded by $5k$.

14:8 Multivariate Exploration of Metric Dilation

Let us define an auxiliary function f that is useful for defining some parameters in the upcoming steps. This function is defined as follows.

$$f(i) = \begin{cases} d & \text{if } i = d \\ d \cdot k + k^2 + k & \text{if } i = d - 1 \\ d \cdot k^{d-i} + k^{d-i+1} + \left(2 \cdot \sum_{j=2}^{d-i} k^j\right) + k & \text{if } 0 \leq i \leq d - 2 \end{cases}$$

It is easy to verify that f satisfies the following property.

► **Proposition 7.** *For any $1 \leq i \leq d$, $f(i-1) = (f(i) + k) \cdot k + k$.*

Thus, if each vertex of R has at most $f(0)$ many neighbours in I in \mathbb{C} , then eventually $|V_{\mathbb{C}}|$ gets bounded by $5k \cdot f(0)$. We then move to the final step described below after Corollary 16.

Now we are in the situation where there is a vertex in R with at least $f(0) + 1$ neighbors in I in \mathbb{C} . Let v be such a vertex in R . Let $U = \{u_1, u_2, \dots, u_{\kappa}\} \subseteq I \cap N_{\mathbb{C}}(v)$ be the set of neighbours of v in I , where $\kappa > f(0)$.

► **Reduction Rule 2.** *If there is no vertex $w \in I$ such that $|U \cap N_G(w)| > f(1)$ then we return NO.*

► **Lemma 8.** *Reduction Rule 2 is safe.*

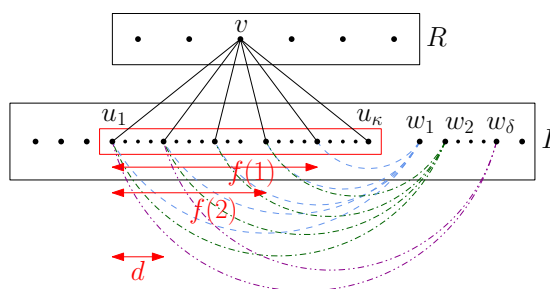
Proof. Assume that for every vertex $w \in I$, we have $|U \cap N_G(w)| \leq f(1)$. Then, we will show that we cannot resolve all conflicts involving v . Suppose (G, Γ, k, R) is a YES-instance of DILATION 2-AUGMENTATION and S be a minimal solution of size at most k . Since each (v, u_i) is an adjacent conflict, we have $d_{G+S}(v, u_i) \leq 2$. This implies $\text{hop}_{G+S}(v, u_i) \leq 2$. Therefore, the edge of S that resolves the conflict (v, u_i) must be adjacent to v or u_i or both. Fix an edge $(v, w) \in S$. Since S is a solution to the annotated instance of the vertex $w \in I$, the number of paths of hop 2 in $G + S$ starting at v , using (v, w) and ending at a vertex in U is upper-bounded by $|U \cap N_G(w)| + |S| \leq f(1) + k$. Therefore, the total number of conflicts that can be resolved by edges of the form (v, w) is at most $(f(1) + k)|Z|$. Here, $Z \subseteq S$ contains edges of the form (v, w) . Let us now focus on the conflicts that are resolved by edges of the form (w, u_i) , $u_i \in U$. This implies that we have paths of the form $\langle v, w, u_i \rangle$ such that $(v, w) \in E(G)$ and $(w, u_i) \in S$. The number of paths of this kind is upper bounded by $|S| - |Z|$. This implies that the total number of conflicts in which v participates, and can be resolved by S , is bounded by $(f(1) + k)|Z| + |S| - |Z| \leq f(1) \cdot k + k^2 + k = f(0)$. ◀

Now, if Reduction Rule 2 is not applicable, then there must be a vertex $w \in I$ such that w has at least $f(1)$ neighbors in U . We use w to identify a small set of edges that must intersect every solution S of size at most k .

► **Lemma 9.** *There exists a polynomial-time algorithm to find a set $W_v = \{w_1, w_2, \dots, w_{\delta}\}$ such that (1) $\delta < d$, and (2) If we have a YES-instance, then any solution S of size at most k satisfies that $S \cap \{(v, w_i) : i \in [\delta]\} \neq \emptyset$.*

Proof. By our definition of \mathbb{C} , the vertex v is in adjacent conflict with each vertex in U . First, by Reduction Rule 2 and Lemma 8, we know that for a YES-instance there exists a vertex $w \in I$ such that $|U \cap N_G(w)| > d \cdot k^d$. We let this vertex to be w_1 . Now either $(v, w_1) \in S$, and then we are done. Else we are in the case when $(v, w_1) \notin S$. Let us define $W_1 := \{w_1\}$.

Suppose we have inductively found a set $W_{i-1} = \{w_1, \dots, w_{i-1}\}$ for some $i \geq 2$. Further, inductively assume that $U_{i-1} := \bigcap_{j \in [i-1]} U \cap N_G(w_j)$ is such that $|U_{i-1}| > f(i-1)$. See Figure 3.2 for an illustration. We prove the following crucial claim which lets us obtain the next vertex in the sequence. Its proof is along the same lines as Lemma 8.



■ **Figure 2** Illustration of arguments used in Lemma 9.

▷ **Claim 10.** Suppose (G, Γ, k, R) is a YES-instance of DILATION 2-AUGMENTATION and S be a solution of size at most k . If $S \cap \{(v, w_j) : j \in [i - 1]\} = \emptyset$. Then there exists a vertex $w_i \in I \setminus W_{i-1}$ such that w_i has more than $f(i)$ neighbours in G , from U_{i-1} .

Proof. Assume that for every vertex $w \in I \setminus W_{i-1}$, we have $|U_{i-1} \cap N_G(w)| \leq f(i)$. Then, we will show that we cannot resolve all conflicts involving v with the vertices from U_{i-1} . Suppose (G, Γ, k, R) is a YES-instance of DILATION 2-AUGMENTATION and S is a solution of size at most k such that $S \cap \{(v, w_j) : j \in [i - 1]\} = \emptyset$. Now we have that for each $u \in U_{i-1}$, (v, u) is an adjacent conflict but $d_{G+S}(v, u) \leq 2$. This implies $\text{hop}_{G+S}(v, u) \leq 2$. Therefore, the edge of S that resolves the conflict (v, u) must be adjacent to v or u or both. Fix an edge $(v, w) \in S$. Since S is a solution to the annotated instance of the vertex $w \in I$, and $w \notin W_{i-1}$ (by assumption), the number of paths of hop 2 in $G + S$ starting at v , using (v, w) and ending at a vertex in U_{i-1} is upper-bounded by $|U_{i-1} \cap N_G(w)| + |S| \leq f(i) + k$. Therefore, the total number of conflicts (v, u) where $u \in U_{i-1}$ that can be resolved by edges of the form (v, w) where $w \notin W_{i-1}$ is at most $(f(i) + k) \cdot |Z| \leq (f(i) + k) \cdot k$. Here, $Z \subseteq S$ contains edges of the form (v, w) . Let us now focus on the conflicts that are resolved by edges of the form (w, u) , $u \in U_{i-1}$. This implies that we have paths of the form $\langle v, w, u \rangle$ such that $(v, w) \in E(G)$ and $(w, u) \in S$. The number of paths of this kind is upper bounded by $|S| - |Z| \leq k$. This implies that the total number of conflicts (v, u) where $u \in U_{i-1}$, and can be resolved by S , is bounded by $(f(i) + k) \cdot k + k = f(i - 1)$, by Proposition 7. Recall that, by induction, $|U_{i-1}| > f(i - 1)$. This contradicts that S is a solution, and the claim follows. ◁

Thus, if we find a vertex w_i satisfying the requirements as mentioned in Claim 10, we define $W_i = W_{i-1} \cup \{w_i\}$ and proceed to the next iteration. Otherwise, if we cannot find w_i , then we stop and output the current set W_{i-1} as the set W_v as required by the lemma. In this case, by Claim 10, we know that, if we have a YES-instance, then $S \cap \{(v, w_j) : j \in [i - 1]\} \neq \emptyset$.

Suppose the iterative procedure goes on for δ iterations, i.e., $W_v = W_\delta = \{w_1, w_2, \dots, w_\delta\}$. Observe that if $\delta \geq d$ then the vertex set $\{w_1, w_2, \dots, w_d\}$ and $\bigcap_{j \in [\delta]} (U \cap N_G(w_j))$ induce a $\mathcal{K}_{d,d}$ in G which contradicts the fact that G is $\mathcal{K}_{d,d}$ -free. Thus, $\delta < d$, and (1) holds. ◀

We use the algorithm in Lemma 9 to find the set W_v of size $\delta < d$ as claimed. We know that, if we have a YES-instance, then any minimal solution S contains at least one edge of the form (v, w_i) , $i \in [\delta]$. The following reduction rule essentially tries to guess the set $W' \subseteq W_v$ for which such edges belong to the solution, and add those vertices to the vertex cover R . Further, due to our invariant, when we add W' to R , we also need to guess the edges between W' and the vertices already in R . The following reduction rule is a Turing reduction from the annotated version of the problem to itself – we prove later that it is a valid decreasing Turing FPT reduction step.

14:10 Multivariate Exploration of Metric Dilation

► **Reduction Rule 3.** Let $v \in R$ be any vertex such that $\deg_{\mathbb{C}}(v) \geq f(0) + 1$. For every non-empty subset $W' \subseteq W_v$, and for any subset

$$\overline{E}_x \subseteq \left\{ (p, q) \in \overline{E}(G) : p \in W', q \in W' \cup (R \setminus \{v\}) \right\}$$

of non-edges in G , we create the following instances. (G_j, Γ, k_j, R') where $G_j = G + (\{(v, w_a) | w_a \in W'\} \cup \overline{E}_x)$, $k_j = k - |W'| - |\overline{E}_x|$ and $R' = R \cup W'$.

► **Lemma 11.** Reduction Rule 3 is a valid decreasing FPT-Turing-reduction.

Proof. First, note that $|W_v| \leq d$, which implies that the number of subsets $W' \subseteq W$ is at most 2^d . Then, for each fixed subset W' of size δ , there are at most $2^{\mathcal{O}(k \log d)}$ many subset \overline{E}_x . So in total the number of instances we created is bounded by $2^{\mathcal{O}(d+k \log d)}$. Now we show the safeness of the Reduction Rule 3.

▷ **Claim 12.** (G, Γ, k, R) is a YES-instance if and only if one of the instances in $\{(G_j, \Gamma, k_j, R')\}$ is a YES-instance.

Proof. The backward direction is trivial. In the forward direction, assume that (G, Γ, k, R) is a YES-instance and S is a minimal solution. Due to the Lemma 9 the solution S must satisfy the condition that $S \cap \{(v, w_i) : i \in [\delta]\} \neq \emptyset$. Let $S' := S \cap \{(v, w_i) : i \in [\delta]\}$, $W'' := V_{S'} \setminus \{v\}$, and $S'' := S \cap \{(w, r) : w \in W'', r \in R \cup W''\}$. Clearly $W'' \subseteq W_v$ and $S'' \subseteq \{(p, q) \in \overline{E}(G) : \text{either } p, q \in W' \text{ or } p \in W', q \in R \setminus \{v\}\}$. Now considering $S'' = \overline{E}_x$, $G_j = G + S' + S''$, $k_j = k - |S'| - |S''|$, and $R' = R \cup W''$, we have that (G_j, Γ, k_j, R') is a YES-instance. ◁

As W' is a non-empty subset of W_v so $k_j \leq k - |W'| \leq k - 1$. So in each created instance, the value of the parameter decreases by at least one. Hence the lemma follows. ◀

Next we formalize an observation that follows from the definition of FPT-Turing-reduction.

► **Observation 13.** Starting from an annotated instance (G, Γ, k, R) , let (G_j, Γ, k_j, R_j) be one of the instances produced by Reduction Rule 3. Then, $|R_j| \leq |R| + (k - k_j)$. In particular, this implies that the size of $|R'|$ in any of the instances produced by a sequence of decreasing Turing FPT reductions, remains bounded by $5k$.

We keep applying Reduction Rule 3 until each vertex in R has at most $f(0)$ neighbors in I in the graph \mathbb{C} . In each step, we maintain the invariant that we have included all possible solution edges within R in G and reduced k appropriately. Observe that when Reduction Rule 3 is not applicable, the total number of vertices in the conflict is at most $\mathcal{O}(k \cdot f(0))$.

Let (G, Γ, k, R) be an annotated instance of DILATION 2-AUGMENTATION. Find a dilation 2-augmentation set S such that $|\{x, y\} \cap R| \leq 1$. That is, every edge added must contain at least one end-point that does not belong to R . In addition, $|R| \leq 5k$ and $|V_c| = \mathcal{O}(k \cdot f(0))$, where $f(0) = d \cdot k^d + k^{d+1} + \left(2 \cdot \sum_{j=2}^d k^j\right) + k$.

3.3 Solving Annotated Instances with bounded V_c

In the rest of the section, we describe how to solve an annotated instance $\mathcal{I} = (G, \Gamma, k, R)$ when the size of V_c is bounded by some function of k and d . Note that although $|V_c| \leq h(k, d)$, we cannot completely forget about the vertices outside V_c – since some conflicts may be resolved by using edges incident to vertices outside V_c . We will argue that we do not need to keep all such vertices, but it suffices to keep one representative from each “equivalence class”, which we formalize below. Note that in this last step, we do not need the vertex cover R .

Let $O = V(G) \setminus V_c$ denote the vertices *outside* V_c . For each $A, B \subseteq V_c$ with $A \cap B = \emptyset$, let $O(A, B)$ denote the set of vertices $v \in O$ satisfying the following two properties:

1. Set of vertices $u \in V_c$ such that $d_G(u, v) = 1$ is *exactly* equal to A , and
 2. Set of vertices $w \in V_c$ such that $(v, w) \notin E(G)$, but $d_\Gamma(v, w) = 1$ is *exactly* equal to B .
- We have the following observation.

- **Observation 14.** 1. $\mathcal{P} = \{O(A, B) : A, B \subseteq V_c, A \cap B = \emptyset\}$ forms a partition of O .
2. $|\mathcal{P}|$ is bounded by $3^{|V_c|} \leq g(k, d)$ for some computable function g .

For each $O(A, B) \in \mathcal{P}$ such that $O(A, B) \neq \emptyset$, we mark an arbitrary vertex $v(A, B) \in O(A, B)$. Let $U \subseteq O$ denote the set of unmarked vertices. In the following reduction rule, we eliminate all the vertices of U from G and Γ , and then prove that it is correct.

- **Reduction Rule 4.** Given the annotated instance $\mathcal{I}_1 = (G, \Gamma, k, R)$, produce the annotated instance $\mathcal{I}_2 = (G', \Gamma', k, R)$, where $\Gamma' = \Gamma - U$ and $G' = G - U$.

- **Lemma 15.** Reduction Rule 4 is safe.

Proof. We argue that \mathcal{I}_1 is a YES-instance if and only if \mathcal{I}_2 is a YES-instance. The reverse direction is trivial, since any solution for \mathcal{I}_s is also a solution for \mathcal{I}_1 (recall that the vertices of U are not involved in any conflict). So we show the forward direction.

Let $S \subseteq \overline{E}(G)$ be a minimal solution of size k with the set of end-points being V_S . From this solution, we define another solution S' as follows. Consider any $u \in U \cap V_S$, and let $S(u) = \{(u, w) \in S\}$, and let $N(u) = \{w : (u, w) \in S(u)\}$. First, note that $N(u) \subseteq V_c$ – suppose not, i.e., there exists some $w \in N(u)$ but $w \notin V_c$, then (u, w) cannot be part of a path of hop length 2 between two vertices in V_c , which contradicts the minimality of S . Now, suppose u belongs to the set $O(A, B) \in \mathcal{O}$, then $O(A, B) \neq \emptyset$, which implies that some $v(A, B) \in V(G) \setminus U = V(\Gamma) \setminus U$. We define another set $S' := S \setminus S(u) \cup \{(v(A, B), w) : w \in S(u)\}$. Note that $|S| = |S'|$. We prove that S' is also a solution.

Consider any path $\langle x, u, y \rangle$ in $G + S$, such that $d_\Gamma(x, u) = d_\Gamma(u, y) = 1$. There are three cases: (i) $(x, u) \in E(G)$ and $(u, y) \in S$. In this case, note that $(x, v(A, B)) \in E(G')$ with $d_{G'}(x, v(A, B)) = d_G(x, v(A, B)) = 1$, and $d_\Gamma(u, y) = d_\Gamma(v(A, B), y) = 1$. Further, $(v(A, B), y) \in S'$. (ii) $(x, u) \in S$ and $(u, y) \in E(G)$ is symmetric. (iii) $(x, y), (y, u) \in S$. In this case, $d_\Gamma(v(A, B), x) = d_\Gamma(u, x) = 1$, and $d_\Gamma(v(A, B), y) = d_\Gamma(u, y) = 1$. Thus, in all three cases, $\langle x, v(A, B), y \rangle$ path exists in $G + S'$, or in other words, the replaced edges incident to $v(A, B)$ can resolve the same set of conflicts as the edges of $S(u)$. By iterating over the vertices of $V_S \cap U$ and modifying the solution in this way, we obtain a solution S'' containing entirely the edges within $E(\Gamma')$. This completes the forward direction. ◀

- **Corollary 16.** The number of vertices in the reduced annotated instance (G', Γ', k, R) is bounded by some $g(k, d)$.

Thus, there are at most $\binom{g(k, d)}{2k} = \binom{2^{\mathcal{O}(h(k, d))}}{2k} = 2^{\mathcal{O}(k \cdot h(k, d))}$ possibilities for the end-points of the solution edges. For each such subset, we can guess the actual set of at most k edges in time $k^{\mathcal{O}(k)}$. Thus, the total number of possibilities is bounded by $f(k, d)$ for some computable function f . We finish the discussion with the following theorem, and its immediate corollary.

- **Theorem 17.** DILATION 2-AUGMENTATION can be solved in time $f(k, d) \cdot n^{\mathcal{O}(1)}$ when G is a $\mathcal{K}_{d,d}$ -free graph for any $d \in \mathbb{N}$.

- **Corollary 18.** DILATION 2-AUGMENTATION can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ when G is a forest, planar graph, bounded-treewidth graph, an H -minor free graph for some fixed H , a nowhere dense graph, or bounded degeneracy graph.

4 DILATION t -AUGMENTATION for Bounded Degree Graphs is FPT

In this section either G is of bounded degree or Γ is of bounded degree. Observe that a $\mathcal{K}_{d,d}$ -free graph generalizes bounded-degree graphs, and thus the result when G is of bounded degree may appear to be subsumed by those presented in Section 3. However, the result presented here works for any value of t , but the result in Section 3 only worked for $t = 2$.

In either case, we will use the following easy lemma to bound the number of vertices in a ball of radius ℓ around a vertex subset of small size.

► **Observation 19.** *Let H be a graph with maximum degree Δ . For any subset of vertices $U \subseteq V(H)$, $|N_H^\ell(U)| \leq |U| \cdot \sum_{i=0}^{\ell} \Delta^i \leq |U| \cdot \Delta^{\ell+1}$.*

4.1 Γ is of Bounded Degree

Let (G, Γ, k) be an instance of DILATION t -AUGMENTATION. In this subsection, we consider the case where Γ belongs to the family of graphs of maximum degree at most Δ . However, there is no restriction on G . That is, G is an arbitrary undirected graph. The idea of the proof is to identify a subset of vertices of Γ of size at most $f(k, \Delta, t)$ such that V_S belongs to balls of radius t around them. Once the set is identified, the algorithm tries all potential solutions of size at most k and returns YES, if either leads to the desired solution. The hypothetical solution S is of size at most k , and hence from Observation 19 we have $|N_\Gamma^t(V_S)| \leq 2k \cdot \Delta^{t+1}$. First, we show that the vertices of V_S are in distance t from the vertices of V_c in Γ .

► **Lemma 20 (♠).** *Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION. Then, $V_S \subseteq N_\Gamma^t(V_c)$.*

Observe that we cannot upper bound the size of $N_\Gamma^t(V_c)$, as the size of V_c may not be bounded by any function of t, k and Δ . Next, we show a kind of converse of Lemma 20 showing that, in fact, V_c is contained inside the balls of radius t around V_S in Γ . The proof is identical to Lemma 20 and is presented separately for clarity.

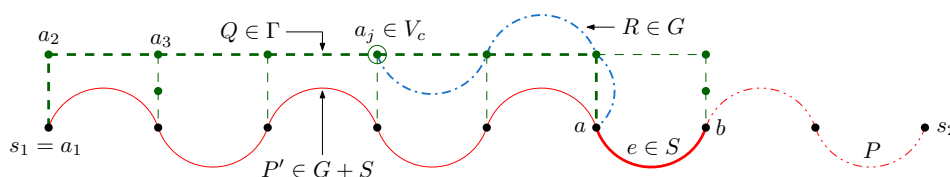
► **Lemma 21 (♠).** *Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION. Then, $V_c \subseteq N_\Gamma^t(V_S)$.*

Lemma 21, along with Observation 19 implies that in a YES-instance, the size of V_c is bounded by $2k \cdot \Delta^{t+1}$. Thus, if $|V_c| > 2k \cdot \Delta^{t+1}$, we say NO. Now assume that $|V_c| \leq 2k \cdot \Delta^{t+1}$. Next, via Lemma 20, we have that $V_S \subseteq N_\Gamma^t(V_c) =: Q$, say. Again, by Observation 19, $|Q| \leq |V_c| \cdot \Delta^{t+1} \leq 2k \cdot \Delta^{2t+2}$. We know we have to add at most k edges between the vertices in Q to resolve all the conflicts. Since the size of Q is bounded by a function of k, Δ , and t , this leads to the following theorem.

► **Theorem 22.** *DILATION t -AUGMENTATION can be solved in time $2^{\mathcal{O}(k \log k)} \cdot \Delta^{\mathcal{O}(kt)} \cdot n^{\mathcal{O}(1)}$, where Δ denotes the maximum degree of the graph Γ .*

4.2 G is of Bounded Degree

Let (G, Γ, k) be an instance of DILATION t -AUGMENTATION. In this subsection, we consider the case where G belongs to the family of graphs of maximum degree at most Δ . However, there is no restriction on Γ . The proof strategy in this section is similar to that used for Theorem 22. As before, the idea is to identify a subset of vertices of G of maximum size $f(k, \Delta, t)$ such that V_S belongs to balls of radius t^2 around them.



■ **Figure 3** Path in $G + S$ is shown in red (solid lines), path in Γ is shown in green (dashed) and path in G is shown in blue (dashed-dotted).

► **Lemma 23** (♠). *Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION. Then, $V_c \subseteq N_G^t(V_S)$.*

As $|V_S| \leq 2k$, the next observation follows from Observation 19.

► **Observation 24.** $|N_G^t(V_S)| \leq 2k \cdot \Delta^{t+1}$.

Lemma 23 and Observation 24 together yield the following reduction rule that yields an upper bound on the size of the conflict set V_c .

► **Reduction Rule 5.** *If $|V_c| > 2k \cdot \Delta^{t+1}$, return NO.*

To identify the vertices in V_S , we show that they lie in the balls of radius t^2 around V_c in G .

► **Lemma 25.** *Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION and S be a hypothetical minimal solution. Then, $V_S \subseteq N_G^{t^2}(V_c)$.*

Proof. Let $a \in V_S$ and (a, b) be an edge in S containing a . Due to the minimality of S , for every edge $e = (a, b)$ in S , there exist at least two vertices s_1 and s_2 such that (i) s_1 and s_2 are in conflict (again, adjacent conflict) with each other in G , and (ii) $e = (a, b)$ appears in a shortest path P of length at most t between s_1 and s_2 in $G + S$. Note that a and b may not be in conflict. We show that there exists a vertex $w \in V_c$ such that w is at most t^2 hop distance from the vertex a in G (that is, $\text{hop}_G(w, a) \leq t^2$). Let P' be the subpath of length $\tau \leq t$ between s_1 and a (see Figure 3). Since $G + S$ is also a graph embedded in a metric space derived from the undirected graph Γ , by Observation 2 we have $d_\Gamma(s_1, a) \leq d_{G+S}(s_1, a) = \tau \leq t$.

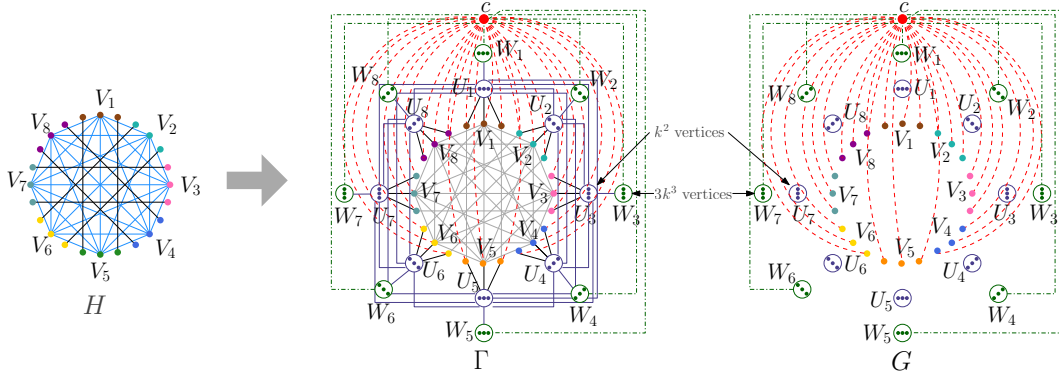
Let $Q = \langle s_1 = a_1, a_2, \dots, a_{\tau+1} = a \rangle$ be a path between s_1 and a in Γ of length $d_\Gamma(s_1, a) \leq t$. Let $j \in [\tau + 1]$ be the largest index such that $a_j \in V_c$. Since $s_1 \in V_c$, an index j always exists. Observe that for all $j \leq i \leq \tau$, a_i and a_{i+1} are *not in conflict* and a_i and a_{i+1} are *adjacent* in Γ . Therefore, there exists a path R_i between a_i and a_{i+1} , $j \leq i \leq \tau$, in G of maximum length t . Thus, by concatenating the paths $R_j \bullet R_{j+1} \bullet \dots \bullet R_\tau$, we get a walk with a weighted length at most t^2 between a_j and a in G . This implies that there is a path between a_j and a in G of length t^2 . That is, $d_G(a_j, a) \leq t^2$. Since all the edge weights in G are at least 1, we have $\text{hop}_G(a_j, a) \leq t^2$. This implies $a \in N_G^{t^2}(a_j) \subseteq N_G^{t^2}(V_c)$. ◀

From Lemma 25 and Reduction Rule 5, and Observation 19, we get the following.

► **Lemma 26.** *Let (G, Γ, k) be a YES-instance of DILATION t -AUGMENTATION. Then, $|N_G^{t^2}(V_c)| \leq 2k \cdot \Delta^{t+1} \cdot \Delta^{t^2+1}$.*

We have to add at most k edges between the vertices in $N_G^{t^2}(V_c)$ to resolve all the conflicts. Since the size of $N_G^{t^2}(V_c)$ is bounded by $f(k, t, \Delta)$, this leads to the following theorem.

► **Theorem 27.** *DILATION t -AUGMENTATION can be solved in time $2^{\mathcal{O}(k \log k)} \cdot \Delta^{\mathcal{O}(kt^2)} \cdot n^{\mathcal{O}(1)}$, where Δ denotes the maximum degree of the graph G .*



■ **Figure 4** Hardness for $t = 3$ when G is a disjoint union of a star, and a set of isolated vertices.

5 Dilation 3-Augmentation For Forest and Stars

In this section, we focus on DILATION 3-AUGMENTATION when G is a forest (Section 5.1), or when Γ is a star (Section 5.2). We show that DILATION 3-AUGMENTATION is W -hard in both cases. In contrast to this, we know that DILATION 2-AUGMENTATION is FPT when G is a forest, due to Theorem 17.

5.1 $W[1]$ -hardness of DILATION 3-AUGMENTATION when G is Forest

We sketch a polynomial-time parameter preserving reduction from the MULTICOLORED CLIQUE problem – which is known to be $W[1]$ -hard [9] – to an instance of DILATION 3-AUGMENTATION when G is a disjoint union of a star and an independent set. The input of MULTICOLORED CLIQUE consists of a graph H , an integer k , and a partition $\mathcal{V} = (V_1, V_2, \dots, V_k)$ of the vertices of H ; our aim is to decide if there is a clique of size k containing exactly one vertex from each set $V_i, i \in [k]$. We denote this instance by (H, k, \mathcal{V}) . We can assume that for each $i \in [k]$, V_i is an independent set. From an instance (H, k, \mathcal{V}) of MULTICOLORED CLIQUE for $k \geq 2$, we construct an instance (G, Γ, k') of DILATION 3-AUGMENTATION with the following way (see Figure 4).

- For every set $V_i, i \in [k]$, we introduce a set U_i of k^2 vertices and a set W_i of $3k^3$ vertices.
- Construction of the graph Γ :
 1. The vertex set in Γ consists of vertex set $H_G \cup \{U_1 \cup \dots \cup U_k\} \cup \{W_1 \cup \dots \cup W_k\} \cup \{c\}$ where $H_G = V(H)$.
 2. The edge set of Γ is defined by $E(\Gamma) = E_H \cup E_U \cup E_W \cup E_c$; here $E_H = E(H)$, $E_U = \{(v, u) | v \in V_i, u \in U_i, i \in [k]\} \cup \{(v, u) | v \in U_i, u \in U_j, i, j \in [k], i \neq j\}$, $E_W = \{(u, w) | u \in U_i, w \in W_i, i \in [k]\}$ and $E_c = \{(c, v) | v \in H_G \cup W_1 \cup \dots \cup W_k\}$.
- The graph G consists of the vertex set $V(\Gamma)$ and the edge set E_c .
- We set $k' = \binom{k}{2} + k^3$.

It is easy to see that in the constructed instance (G, Γ, k') the graph G is indeed a disjoint union of a star and independent set. Also, the construction can be performed in time polynomial in $|V(H)|$ and k . Towards the correctness of our reduction, we prove the following.

► **Lemma 28.** *(H, k, \mathcal{V}) is a YES-instance of MULTICOLORED CLIQUE if and only if (G, Γ, k') is a YES-instance of DILATION 3-AUGMENTATION.*

Proof. (\Rightarrow) In the forward direction, suppose that there is a multicolored clique Q in H . Let the set of vertices in Q be $V_Q = \{v_1, \dots, v_k\}$ where $v_i \in V_i$ and the edgeset of Q be E_Q . Consider the set of edges $E^{VU} = \{(v_i, u) | i \in [k], u \in U_i\}$. Observe that $|E_Q| = \binom{k}{2}$ and $|E^{VU}| = k \cdot k^2$ as $|U_i| = k^2$ for each $i \in [k]$. Next, we show that the dilation of $G + (E_Q \cup E^{VU})$ is at most 3. Notice that all the edges incident to c in Γ are also present in G . Hence c is not involved in any adjacent conflicts. Now we consider all other pairs of adjacent vertices in Γ , say a and b , and show that there exists a path between them of length at most 3 in $G + (E_Q \cup E^{VU})$. This suffices our proof due to Lemma 1. We argue with the following cases.

Case (i): $a \in V_i$ and $b \in V_j$. There is a length 2 path between them in G itself which is precisely $\langle a, c, b \rangle$.

Case (ii): $a \in V_i$ and $b \in U_i$. If $a = v_i$ then a and b are adjacent in $G + E^{VU}$. Otherwise, there is a length 3 path between them in $G + E^{VU}$ which is precisely $\langle a, c, v_i, b \rangle$.

Case (iii): $a \in U_i$ and $b \in U_j$. If $i \neq j$, we use two edges from $E_Q \cup E^{VU}$. The vertices a and b are connected by a length 3 path $\langle a, v_i, v_j, b \rangle$ in $G + (E_Q \cup E^{VU})$. Suppose that $i = j$. Then $\langle a, v_i, b \rangle$ is a path in $G + (E_Q \cup E^{VU})$ of length 2.

Case (iv): $a \in U_i$ and $b \in W_i$. In this case, we use an edge from E^{VU} . The vertices a and b are connected by a length 3 path $\langle a, v_i, c, b \rangle$ in $G + E^{VU}$.

(\Leftarrow) In the backward direction, let E_X be a solution to the instance (G, Γ, k') of DILATION 3-AUGMENTATION, that means dilation of $G + E_X$ is at most 3. We start with the following claim.

\triangleright **Claim 29.** For each $i \in [k]$ and every $u \in U_i$, the set E_X contains an edge (u, v) for $v \notin \bigcup_{j=1}^k U_j$.

Proof. The proof is by contradiction. Suppose that there are $i \in [k]$ and $u \in U_i$ such that for every $(u, v) \in E_X$, $v \in \bigcup_{j=1}^k U_j$. Because $|E_X| \leq k' = \binom{k}{2} + k^3$ and $|W_i| = 3k^3$, we have that $|W_i| - 2|E_X| \geq 1$ and, therefore, there is $w \in W_i$ such that w is not incident to any edge of E_X . Thus, any shortest (u, w) -path in $G + E_X$ contains the edge (w, c) and an edge (u, v) for some $v \in \bigcup_{j=1}^k U_j$. However, the distance between c and v in Γ is 2. This implies that the length of any (u, w) -path in $G + E_X$ is at least 4. Because u and w are adjacent in Γ , this contradicts that the dilation of $G + E_X$ is at most 3 and proves the claim. \triangleleft

Due to the above claim, summing over all $i \in [k]$, we have that the set E_X contains at least $k \cdot k^2 = k^3$ edges incident to the vertices of $\bigcup_{i=1}^k U_i$ whose second end-points are outside $\bigcup_{i=1}^k U_i$. Because $|E_X| \leq \binom{k}{2} + k^3$, we have that for each $i \in [k]$, there are at least $k^2 - 2\binom{k}{2} \geq k \geq 2$ vertices of U_i that are adjacent to exactly one edge of E_X . For $i \in [k]$, we denote by $U'_i \subseteq U_i$ the set of vertices incident to unique edges of E_X . We make the following observation.

\triangleright **Claim 30.** For each $i \in [k]$, there is $u \in U'_i$ such that u is incident to a single edge $(u, v) \in E_X$ such that $(u, v) \in E(\Gamma)$.

Proof. Consider arbitrary $u \in U'_i$. This vertex has a unique neighbor v in $G + E_X$ where $v \notin \bigcup_{j=1}^k U_j$ by Claim 29. If $(u, v) \in E(\Gamma)$, the claim holds. Suppose that $(u, v) \notin E(\Gamma)$. Recall that $|U'_i| \geq 2$. Thus, there is $u' \in U'_i$ distinct from u . In the same way as above, u' has a unique neighbor v' in $G + E_X$ where $v' \notin \bigcup_{j=1}^k U_j$. Then any (u, u') -path in $H + E_X$ contains the edges (u, v) and (u', v') . Since $(u, u') \in E(\Gamma)$ and the dilation of $G + E_X$ is at most 3, we have that the length of (u', v) is one. Therefore, $(u', v) \in E(\Gamma)$. This concludes the proof. \triangleleft

14:16 Multivariate Exploration of Metric Dilation

Using Claim 30, for every $i \in [k]$, we denote by u_i the vertex of U'_i that is incident to a single edge of $E_X \cap E(\Gamma)$ and we use v_i to denote the second end-point of the edge.

▷ **Claim 31.** $\{v_1, \dots, v_k\}$ is a clique of H .

Proof. First, we show that $v_i \in V_i$ for each $i \in [k]$. Consider $i \in [k]$. Note that by Claims 29 and 30 and the construction of Γ , either $v_i \in W_i$ or $v_i \in V_i$. Suppose that $v_i \in W_i$ and consider u_j for $j \in [k]$ distinct from i . We have that $(u_i, u_j) \in E(\Gamma)$. Therefore, $G + E_X$ should have a (u_i, u_j) -path of length at most 3. Any (u_i, u_j) -path contains the edges (u_i, v_i) and (u_j, v_j) . Thus, a shortest (u_i, u_j) -path should contain $(v_i, v_j) \in E(\Gamma)$. However, since $v_i \in W_i$ and $v_j \in V_j \cup W_j$, we have no such an edge by the construction of Γ . This contradiction proves that $v_i \in V_i$.

Finally, we show that for all distinct $i, j \in [k]$, v_i and v_j are adjacent in H . For this, we again observe that $G + E_X$ should have a (u_i, u_j) -path of length at most 3 that includes (u_i, v_i) and (u_j, v_j) . This implies that $(v_i, v_j) \in E(\Gamma)$. Because $v_i \in V_i$ and $v_j \in V_j$, we have that (v_i, v_j) is in E_X and is an edge of H . This proves the claim. ◁

Claim 31 completes the proof of the lemma. ◀

Hence, we have the following theorem.

► **Theorem 32.** DILATION 3-AUGMENTATION is $W[1]$ -hard parameterized by k when G is star forest.

5.2 $W[1]$ -hardness of DILATION 3-AUGMENTATION when Γ is Star

We present a parameter preserving reduction from the DOMINATING SET problem.

Let (H, k) be an instance of DOMINATING SET. We create the instance (G, Γ, k) of the DILATION 3-AUGMENTATION problem as follows. Graphs Γ and G are defined over the vertex sets $V(H) \cup \{c\}$, where c is a new vertex distinct from the vertices of $V(H)$. Γ is a star with center c and leaves $V(H)$. That is, $E(\Gamma) = \{(c, v) : v \in V(H)\}$. Note that in the shortest path metric defined by Γ , $d_\Gamma(u, v) = 2$ for any $u, v \in V(H)$. We let $E(G) = E(H)$, which implies that G is a disjoint union of H and the singleton vertex c . Note that the weight of each edge in G is 2. The proof of equivalence can be found in the full version.

► **Observation 33 (♠).** (H, k) is an YES-instance of DOMINATING SET if and only if (G, Γ, k) is an YES-instance of DILATION 3-AUGMENTATION.

Hence we have the following theorem.

► **Theorem 34.** DILATION 3-AUGMENTATION is $W[2]$ -hard parameterized by k even when Γ is star.

6 Conclusion

In this article, we introduced DILATION t -AUGMENTATION and explored its multivariate complexity of the problem by restricting either the graph class to which Γ could belong or the graph class to which G could belong. Other special graph classes that one could consider include geometric intersection graphs such as interval graphs, unit-disk graphs, disk-graphs, or, more generally, string graphs. In an alternate direction, we can consider the problem in its full generality. However, as we saw, the problem in its full generality cannot admit FPT algorithm. So, a next natural question that stems from our work is exploring these problems from the perspective of FPT-approximation.

References

- 1 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.
- 2 Aritra Banik, Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Satyabrata Jana, and Saket Saurabh. Multivariate exploration of metric dilation, 2025. arXiv:2501.04555.
- 3 Édouard Bonnet, Nicolas Bousquet, Stéphan Thomassé, and Rémi Watrigant. When maximum stable set can be solved in FPT time. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019, December 8-11, 2019, Shanghai University of Finance and Economics, Shanghai, China*, volume 149 of *LIPICs*, pages 49:1–49:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ISAAC.2019.49.
- 4 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.
- 5 Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.
- 6 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.
- 7 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.
- 8 Mohammad Farshi, Panos Giannopoulos, and Joachim Gudmundsson. Improving the stretch factor of a geometric network by edge augmentation. *SIAM J. Comput.*, 38(1):226–240, 2008. doi:10.1137/050635675.
- 9 Michael R Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the fixed-parameter intractability and tractability of multiple-interval graph properties. *Theoretical Computer Science. v410*, pages 53–61, 2007.
- 10 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.
- 11 Yong Gao, Donovan R. Hare, and James Nastos. The parametric complexity of graph diameter augmentation. *Discret. Appl. Math.*, 161(10-11):1626–1631, 2013. doi:10.1016/J.DAM.2013.01.016.
- 12 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.
- 13 Joachim Gudmundsson and Michiel H. M. Smid. On spanners of geometric graphs. *Int. J. Found. Comput. Sci.*, 20(1):135–149, 2009. doi:10.1142/S0129054109006486.
- 14 Joachim Gudmundsson and Sampson Wong. Improving the dilation of a metric graph by adding edges. *ACM Trans. Algorithms*, 18(3):20:1–20:20, 2022. doi:10.1145/3517807.
- 15 Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theor. Comput. Sci.*, 746:88–97, 2018. doi:10.1016/J.TCS.2018.06.031.
- 16 Yusuke Kobayashi. An FPT algorithm for minimum additive spanner problem. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPICs*, pages 11:1–11:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.STACS.2020.11.
- 17 Jun Luo and Christian Wulff-Nilsen. Computing best and worst shortcuts of graphs embedded in metric spaces. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, volume 5369 of *Lecture Notes in Computer Science*, pages 764–775. Springer, 2008. doi:10.1007/978-3-540-92182-0_67.
- 18 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.
- 19 David Peleg and Alejandro A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. doi:10.1002/JGT.3190130114.
- 20 Christian Wulff-Nilsen. Computing the dilation of edge-augmented graphs in metric spaces. *Comput. Geom.*, 43(2):68–72, 2010. doi:10.1016/J.COMGEO.2009.03.008.

Structure-Guided Automated Reasoning

Max Bannach  

European Space Agency, Advanced Concepts Team, Noordwijk, The Netherlands

Markus Hecher  

Univ. Artois, CNRS UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), 6230, France
Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology,
Cambridge, MA, USA

Abstract

Algorithmic meta-theorems state that problems definable in a fixed logic can be solved efficiently on structures with certain properties. An example is Courcelle’s Theorem, which states that all problems expressible in monadic second-order logic can be solved efficiently on structures of small treewidth. Such theorems are usually proven by algorithms for the model-checking problem of the logic, which is often complex and rarely leads to highly efficient solutions. Alternatively, we can solve the model-checking problem by grounding the given logic to propositional logic, for which dedicated solvers are available. Such encodings will, however, usually not preserve the input’s treewidth.

This paper investigates whether all problems definable in monadic second-order logic can efficiently be encoded into SAT such that the input’s treewidth bounds the treewidth of the resulting formula. We answer this in the affirmative and, hence, provide an alternative proof of Courcelle’s Theorem. Our technique can naturally be extended: There are treewidth-aware reductions from the optimization version of Courcelle’s Theorem to MAXSAT and from the counting version of the theorem to #SAT. By using encodings to SAT, we obtain, ignoring polynomial factors, the same running time for the model-checking problem as we would with dedicated algorithms. Another immediate consequence is a treewidth-preserving reduction from the model-checking problem of monadic second-order logic to integer linear programming (ILP). We complement our upper bounds with new lower bounds based on ETH; and we show that the block size of the input’s formula and the treewidth of the input’s structure are tightly linked.

Finally, we present various side results needed to prove the main theorems: A treewidth-preserving cardinality constraints, treewidth-preserving encodings from CNFs into DNFs, and a treewidth-aware quantifier elimination scheme for QBF implying a treewidth-preserving reduction from QSAT to SAT. We also present a reduction from projected model counting to #SAT that increases the treewidth by at most a factor of $2^{k+3.59}$, yielding a algorithm for projected model counting that beats the currently best running time of $2^{2^{k+4}} \cdot \text{poly}(|\psi|)$.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Theory of computation → Finite Model Theory; Theory of computation → Fixed parameter tractability

Keywords and phrases automated reasoning, treewidth, satisfiability, max-sat, sharp-sat, monadic second-order logic, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.15

Related Version *Technical Report*: <https://arxiv.org/abs/2312.14620>

Funding This research was funded by the Austrian Science Fund (FWF), grants J 4656 and P 32830, the Society for Research Funding in Lower Austria (GFF, Gesellschaft für Forschungsförderung NÖ) grant ExzF-0004, as well as the Vienna Science and Technology Fund (WWTF) grant ICT19-065.

Acknowledgements Part of the research was carried out while Hecher was visiting the Simons Institute for the Theory of Computing at UC Berkeley. Author names are stated alphabetically.



© Max Bannach and Markus Hecher;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 15; pp. 15:1–15:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

Many tools from the automated reasoning quiver can be implemented efficiently if a graphical representation of the given formula with good structural properties is given. The textbook example is the *satisfiability problem* (SAT), which can be solved in time $O(2^k \text{poly}(|\psi|))$ on formulas ψ whose *primal graph* G_ψ has *treewidth* k . (The primal graph contains a vertex for every variable of the formula and connects them if they appear together in a clause. Its treewidth intuitively measures how close it is to being a tree.) The result extends to the *maximum satisfiability problem* (MAXSAT), in which the clauses of the formula have weights and the goal is to minimize the weights of falsified clauses, and to the *model counting problem* (#SAT), in which the goal is to compute the *number* of satisfying assignments. In this article, we will use the notation $\text{tower}(h, t)$ to describe a tower of twos of height h with t at the top, and $\text{tower}^*(h, t)$ as shorthand to hide polynomial factors, e.g., $O(2^k \text{poly}(|\psi|)) = \text{tower}^*(1, k)$:

► **Fact 1** (folklore, see for instance [1, 4, 5, 14, 24, 29, 30]). *One can solve SAT, MAXSAT, and #SAT in time $\text{tower}^*(1, k)$ if a width- k tree decomposition is given.*

It is worth to take some time to inspect the details of Fact 1. The hidden polynomial factor is not the subject of this paper (as indicated by the notation), but can be made as small as $O(|\varphi|)$ [10, 26]. Our focus will be the value on top of the tower, which in Fact 1 is simply “ k ”. Under the *exponential-time hypothesis* (ETH), this is best possible.

The natural extension of the satisfiability problem to higher logic is the validity problem of fully *quantified Boolean formulas* (QSAT). While it is well-known that QSAT is fixed-parameter tractable (i.e., it is in FPT) with respect to treewidth [11], the dependencies on the treewidth is less sharp than in Fact 1. The height of the tower depends on the *quantifier alternation* $\text{qa}(\psi)$ of the formula, while the top value has the form $O(k + \log k + \log \log k + \dots)$ due to the management of nested tables in the involved dynamic program.

► **Fact 2** ([11, 10]). *One can solve QSAT in time $\text{tower}^*(\text{qa}(\psi) + 1, O(k))$ if a width- k tree decomposition is given.*

In contrast to Fact 1, there is a big-oh on top of the tower in Fact 2. The higher order version of the model counting problem is the *projected model counting problem* (PMC), in which we need to count the number of models that are not identical on a given set of variables.

► **Fact 3** ([15]). *One can solve PMC in time $\text{tower}^*(2, k + 4)$ if a width- k tree decomposition is given.*

The fine art of automated reasoning is *descriptive complexity*, which studies the complexity of problems in terms of the complexity of a description of these problems; independent of any abstract machine model [21, 25]. A prominent example is *Courcelle’s Theorem* that states that the problems that can be expressed in *monadic second-order logic* can be solved efficiently on instances of bounded treewidth [12]. Differently phrased, the theorem states that the *model-checking problem* for MSO logic (MC(MSO)) is fixed-parameter tractable (the parameter is the size of the formula and the treewidth of the structure):

► **Fact 4** ([12]). *One can solve MC(MSO) in time $\text{tower}^*(\text{qa}(\varphi) + 1, O(k + |\varphi|))$ if a width- k tree decomposition is given.*

For instance, the 3-coloring problem (Can we color the vertices of a graph with three colors such that adjacent vertices obtain different colors?) can be described by the sentence:

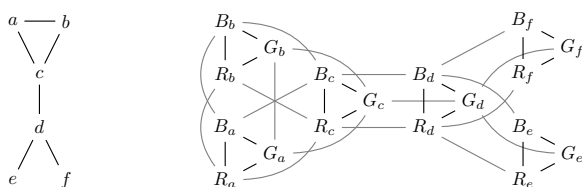
$$\begin{aligned} \varphi_{\text{3col}} = & \exists R \exists G \exists B \forall x \forall y. (Rx \vee Gx \vee Bx) \\ & \wedge Exy \rightarrow \neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By)). \end{aligned}$$

The sentence can be read aloud as: There are three colors red, blue, and green ($\exists R \exists G \exists B$) such that for all vertices x and y ($\forall x \forall y$) we have that (i) each vertex has at least one color ($Rx \vee Gx \vee Bx$), and (ii), if x and y are connected by an edge (Exy) then they do not have the same color ($\neg((Rx \wedge Ry) \vee (Gx \wedge Gy) \vee (Bx \wedge By))$). The model-checking problem $\text{MC}(\text{MSO})$ obtains as input a relational structure \mathcal{S} (say a graph like \mathfrak{G}_1 or \mathfrak{G}_2) and an MSO sentence φ (as the one from above) and asks whether \mathcal{S} is a model of φ , denoted by $\mathcal{S} \models \varphi$. In our example we have $\mathfrak{G}_1 \models \varphi_{3\text{col}}$ and $\mathfrak{G}_2 \not\models \varphi_{3\text{col}}$. Using Fact 4, we can conclude from $\varphi_{3\text{col}}$ that the 3-coloring problem parameterized by the treewidth lies in FPT.

Instead of utilizing Fact 4, another reasonable approach is to *ground* the MSO sentence to a propositional formula and to then apply Fact 1. Formally, this means to reduce the model checking problem $\text{MC}(\text{MSO})$ to SAT, i.e., given a relational structure \mathcal{S} and an MSO sentence φ , we need to produce, in polynomial time, a propositional formula ψ such that $\mathcal{S} \models \varphi$ iff $\psi \in \text{SAT}$. The naïve way of doing so is by generating an indicator variable X_u for every set variable X and every element u in the universe of \mathcal{S} . Then we replace every first-order \exists -quantifier by a “big-or” and \forall -quantifier by a “big-and”:

$$\psi_{3\text{col}} = \bigwedge_{u \in V(G)} \bigwedge_{v \in V(G)} \overbrace{\left(\underbrace{Rx \vee Gx \vee Bx}_{\text{propositional variables}} \right) \wedge \bigwedge_{\{u,v\} \in E(G)} \overbrace{\left(\underbrace{\neg((R_u \wedge R_v) \vee (G_u \wedge G_v) \vee (B_u \wedge B_v))}_{\text{propositional variables}} \right)}^{Exy \rightarrow}}$$

The emerging question now is whether an automated translation such as the one we just sketched preserves treewidth in the following sense: Given a relational structure \mathcal{S} of treewidth $\text{tw}(\mathcal{S})$ and an MSO sentence φ , can we mechanically derive a propositional formula ψ with $\mathcal{S} \models \varphi$ iff $\psi \in \text{SAT}$ and $\text{tw}(\psi) \leq f(\text{tw}(\mathcal{S}))$ for some function computable $f: \mathbb{N} \rightarrow \mathbb{N}$? Consider for instance the following graph shown on the left (it is “almost a tree” and has treewidth 2) and the *primal graph* of $\psi_{3\text{col}}$ obtained using the just sketched transformation on the right. In this example, the tree-like structure is preserved, as the treewidth gets increased by a factor of 3 and is at most 6:

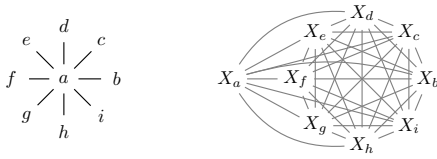


We recap this finding as the following observation: The automated grounding process from $\text{MC}(\text{MSO})$ to SAT *implies* a reduction from the 3-coloring problem parameterized by the input’s treewidth to SAT. We can, thus, derive that the 3-coloring problem can be solved in time $\text{tower}^*(1, 3k)$ using Fact 1 – without actually utilizing Courcelle’s Theorem!

For a second example consider the optimization and counting version of the dominating set problem: Given a graph G , the task is either to find a minimum-size set $S \subseteq V(G)$ of vertices such that every vertex is in S or adjacent to vertex in S , or to count the number of such sets. Optimization and counting problems can be modeled in descriptive complexity by “moving” an existential second-order quantifier (“guessing” the solution) out of the sentence and making it a free variable. The task is either to find a set of minimum size such that the given structure together with this set is a model of the formula, or to count the number of such sets. For instance, the following formula describes that X is a dominating set:

$$\varphi_{\text{ds}}(X) = \forall x \exists y \cdot Xx \vee (Exy \wedge Xy).$$

We will also say that the formula *Fagin-defines* the property that X is a dominating set. The problem $\#FD(MSO)$ asks, given a relational structure \mathcal{S} and an MSO formula with a free-set variable X , how many subsets S of the universe of \mathcal{S} satisfy $\mathcal{S} \models \varphi(S)$. The optimization problem $FD(MSO)$ gets as additional input an integer t and asks whether there is such a S with $|S| \leq t$. The reduction from $MC(MSO)$ to SAT can be extended to a reduction from $FD(MSO)$ to MAXSAT and from $\#FD(MSO)$ to PMC. In order to ground $FD(MSO)$, we add new indicator variables X_u for the free-variable X and every element u of \mathcal{S} (as we did for the second-order quantifiers). For $FD(MSO)$, we additionally add a soft clause $(\neg X_u)$ for each of these variables – implying that we seek a model that minimizes $|X|$. We may now again ask: If we mechanically ground $\varphi_{ds}(X)$ on a structure of bounded treewidth to a propositional formula ψ_{ds} , what can we say about the treewidth of ψ_{ds} ? Unfortunately, not so much. Even if the input has treewidth 1, the primal graph of ψ_{ds} may become a clique (of treewidth n):



It follows that we *cannot* derive an fpt-algorithm for the dominating set problem or its counting version by reasoning about ψ_{ds} , while we can conclude the fact from φ_{ds} using appropriate versions of Courcelle’s Theorem. To summarize, we can naturally describe model-checking, optimization, and counting problems using monadic second-order logic. Using Courcelle’s Theorem, we can solve all of these problems in fpt-time on structures of bounded treewidth. Alternatively, we may ground the MSO formulas to propositional logic and solve the problems using Fact 1. The produced encodings sometimes preserve the input’s structure (as for 3-coloring) and, thus, themselves serve as proof that the problems lie in FPT. However, the input’s structure can also get eradicated, as we observed for the dominating set problem. The present paper is concerned with the question whether there is a unifying grounding procedure that maps Fagin-defined MSO properties to propositional logic while preserving the input’s treewidth.

Contribution I: Faster Structure-guided Reasoning. Before we develop a unifying, structure-aware grounding process from the model-checking problem of monadic second order logic to propositional logic, we first improve both of the underlying results. In particular, we remove the logarithmic dependencies on k in top of the tower of Fact 2 and, thus, provide the first major improvement on QBF upper bounds with respect to treewidth since 20 years:

► **Theorem 1** (QBF Theorem). *One can solve QSAT in time $\text{tower}^*(\text{qa}(\psi) + 1, k + 3.92)$ if a width- k tree decomposition is given.*

This bound matches the ETH lower bound for QSAT:

► **Fact 5** ([16]). *Unless ETH fails, QSAT cannot be solved in time $\text{tower}^*(\text{qa}(\psi) + 1, o(\text{tw}(\psi)))$.*

We will prove Theorem 1 fully in the spirit of an automated reasoning paper by an encoding into SAT. In particular, we will not need any pre-requirements other than Fact 1. With a similar encoding scheme, we will also slightly improve on Fact 3:

► **Theorem 2** (PMC Theorem). *One can solve PMC in time $\text{tower}^*(2, k + 3.59)$ if a width- k tree decomposition is given.*

► **Fact 6** ([15]). *Unless ETH fails, PMC cannot be solved in time $\text{tower}^*(2, o(\text{tw}(\psi)))$.*

Contribution II: A SAT Version of Courcelle’s Theorem. We answer the main question of the introduction in the affirmative and provide a unifying, structure-aware encoding scheme from properties Fagin-defined with monadic second-order logic to variants of SAT:

► **Theorem 3** (A SAT Version of Courcelle’s Theorem). *Assuming that the MSO formulas on the left side are in prenex normal form and that a width- k tree decomposition is given, there are encodings from ...*

1. $\text{MC}(\text{MSO})$ to SAT of size $\text{tower}^*(\text{qa}(\varphi), (k+9)|\varphi| + 3.92)$;
2. $\text{FD}(\text{MSO})$ to MAXSAT of size $\text{tower}^*(\text{qa}(\varphi) + 1, (k+9)|\varphi| + 3.92)$;
3. $\#\text{FD}(\text{MSO})$ to $\#\text{SAT}$ of size $\text{tower}^*(\text{qa}(\varphi) + 1, (k+9)|\varphi| + 3.92)$.

All encodings of size $\text{tower}^(s, t)$ have a treewidth of $\text{tower}(s, t)$ and can be computed in linear time with respect to their size.*

In conjunction with Fact 1, the theorem implies Courcelle’s Theorem with sharp bounds on the values on top of the tower:

► **Corollary 4.** *One can solve $\text{MC}(\text{MSO})$ in time $\text{tower}^*(\text{qa}(\varphi) + 1, (k+9)|\varphi| + 3.92)$, and $\text{FD}(\text{MSO})$ and $\#\text{FD}(\text{MSO})$ in time $\text{tower}^*(\text{qa}(\varphi) + 2, (k+9)|\varphi| + 3.92)$ if a width- k tree decomposition is given.*

Since the reduction [27] from SAT to integer linear programming (ILP) is treewidth-preserving and results in an instance of bounded domain, another consequence of Theorem 3 is an “ILP Version of Courcelle’s Theorem” via the dynamic program for ILP [22].

Contribution III: ETH Lower Bounds for the Encoding Size. Given that we can encode MSO definable properties into SAT while preserving the input’s treewidth, we may ask next whether we can improve on the *size* of the encodings. While it is well-known that incarnations of Courcelle’s Theorem have to depend on the input’s treewidth and the formula’s size in a non-elementary way [3] (and hence, the encodings have to be huge at some point as well), these insights do not give us precise bounds on achievable encoding sizes.

► **Theorem 5** (ETH Lower Bound). *Under ETH, there is no SAT encoding for $\text{MC}(\text{MSO})$ of size $\text{tower}^*(\text{qa}(\varphi) - 2, o(\text{tw}(\mathcal{S})))$ that can be computed in this time.*

We can make the lower bound a bit more precise in the following sense: The value at the top of the tower actually does not just depend on the treewidth $\text{tw}(\mathcal{S})$, but on the product of the treewidth and the *block size* $\text{bs}(\varphi)$ of the sentence φ . The block size of a formula is the maximum number of consecutive quantifiers of the same type.

► **Theorem 6** (Trade-off Theorem). *Under ETH, there is no SAT encoding for $\text{MC}(\text{MSO})$ of size $\text{tower}^*(\text{qa}(\varphi) - 2, o(\text{tw}(\mathcal{S}) \text{bs}(\varphi)))$ that can be computed within this time.*

1.1 Related Work

The concept of treewidth was discovered multiple times. The name was coined in the work by Robertson and Seymour [28], while the concept was studied by Arnborg and Proskurowski [2] under the name partial k -trees simultaneously. However, treewidth was discovered even earlier by Bertelè and Brioschi [6], and independently by Halin [19]. Courcelle’s Theorem was proven in a series of articles by Bruno Courcelle [12], see also the textbook by Courcelle and Engelfriet for a detailed introduction [13]. The expressive power of monadic second-order logic was studied before, prominently by Büchi who showed that MSO over strings characterizes the regular languages [9]. Related to our treewidth-aware reduction from $\text{MC}(\text{MSO})$ to SAT is the work by Gottlob, Pichler, and Wei, who solve $\text{MC}(\text{MSO})$ using monadic Datalog [18]; and the work of Bliem, Pichler, and Woltran, who solve it using ASP [8].

1.2 Structure of this Article

We provide preliminaries in the next section, prove Theorem 1 and 2 in Section 3, and establish a SAT version of Courcelle’s Theorem in Section 4. The technical details of the latter can be found in the technical report version of this article. We extend the result to Fagin-definable properties in Section 5 and provide corresponding ETH lower bounds in Section 6. We conclude and provide pointers for further research in the last section, which also contains an overview table of this article’s results. Due to lack of space, most proofs are only available in the technical report and are replaced by a proof sketch within the main text. The corresponding positions are clearly marked with a “▼”.

2 Preliminaries: Background in Logic and Structural Graph Theory

We use the notation of Knuth [23] and consider *propositional formulas* in conjunctive normal form (CNFs) like $\psi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \wedge (x_2) \wedge (x_6)$ as *set of sets* $\{\{x_1, \neg x_2, \neg x_3\}, \{\neg x_1, x_4, \neg x_5\}, \{x_2\}, \{x_6\}\}$. We denote the sets of variables, literals, and clauses of ψ as $\text{vars}(\psi)$, $\text{lits}(\psi)$, and $\text{clauses}(\psi)$. A (*partial*) *assignment* is a subset $\beta \subseteq \text{lits}(\psi)$ such that $|\{x, \neg x\} \cap \beta| \leq 1$ for all $x \in \text{vars}(\psi)$, that is, a set of literals that does not contain both polarities of any variable. We use $\beta \sqsubseteq \text{vars}(\psi)$ to denote partial assignments. The formula *conditioned under a partial assignment* β is denoted by $\psi|\beta$ and obtained by removing all clauses from ψ that contain a literal $l \in \beta$ and by removing all literals l' with $\neg l' \in \beta$ from the remaining clauses. An assignment is *satisfying* for a CNF ψ if $\psi|\beta = \emptyset$, and it is *contradicting* if $\emptyset \in \psi|\beta$. A DNF is a disjunction of conjunctions, i.e., a set of terms. We use the same notations as for CNFs, however, in $\psi|\beta$ we delete terms that contain a literal that appears negated in β and remove the literals in β from the remaining terms. Hence, β is *satisfying* if $\emptyset \in \psi|\beta$, and *contradicting* if $\psi|\beta = \emptyset$.

The *model counting problem* asks to compute the number of satisfying assignments of a CNF and is denoted by #SAT. In *projected model counting* (PMC) we count the number of models that are not identical on a given set of variables. In the *maximum satisfiability problem* (MAXSAT) we partition the clauses of ψ into a set $\text{hard}(\psi)$ of *hard clauses* and a set $\text{soft}(\psi)$ of weighted *soft clauses*, i.e., every clause $C \in \text{soft}(\psi)$ comes with a *weight* $w(C) \in \mathbb{Q}$. The formula is then called a WCNF and the goal is to find under all assignments $\beta \sqsubseteq \text{vars}(\psi)$ with $\text{hard}(\psi)|\beta = \emptyset$ the one that maximizes $\sum_{C \in \text{soft}(\psi), \{C\}|\beta = \emptyset} w(C)$. In a fully *quantified Boolean formula* (a QBF, also called a *second-order propositional sentence*) all variables are bounded by existential or universal quantifiers. Throughout the paper we assume that QBFs are in prenex normal form, meaning that all quantifiers appear in the front of a quantifier-free formula called the *matrix*. As is customary, we assume that the matrix is a CNF if the last (i.e., most inner) quantifier is existential, and a DNF otherwise. A QBF is *valid* if it evaluates to true (see Chapter 29–31 in [7]). Define QSAT to be the problem of deciding whether a given QBF is valid.

2.1 Descriptive Complexity

A *vocabulary* is a finite set $\tau = \{R_1^{a_1}, R_2^{a_2}, \dots, R_\ell^{a_\ell}\}$ of relational symbols R_i of arity a_i . A (finite, relational) τ -*structure* \mathcal{S} is a tuple $(U(\mathcal{S}), R_1^{\mathcal{S}}, R_2^{\mathcal{S}}, \dots, R_\ell^{\mathcal{S}})$ with *universe* $U(\mathcal{S})$ and *interpretations* $R_i^{\mathcal{S}} \subseteq U(\mathcal{S})^{a_i}$. The *size* of \mathcal{S} is $|\mathcal{S}| = |U(\mathcal{S})| + \sum_{i=1}^{\ell} a_i \cdot |R_i^{\mathcal{S}}|$. We denote the *set of all τ -structures* by $\text{STRUC}[\tau]$ – e.g., $\text{STRUC}[\{E^2\}]$ is the set of directed graphs.

Let τ be a vocabulary and x_0, x_1, x_2, \dots be an infinite repertoire of first-order variables. The *first-order language* $\mathcal{L}(\tau)$ is inductively defined, where the *atomic formulas* are the strings $x_i = x_j$ and $R_i(x_1, \dots, x_{a_i})$ for relational symbols $R_i \in \tau$. If $\alpha, \beta \in \mathcal{L}(\tau)$ then so are

$\neg(\alpha)$, $(\alpha \wedge \beta)$, and $\exists x_i(\alpha)$. A variable that appears next to \exists is called *quantified* and *free* otherwise. We denote a formula $\varphi \in \mathcal{L}(\tau)$ with $\varphi(x_{i_1}, \dots, x_{i_q})$ if x_{i_1}, \dots, x_{i_q} are precisely the free variables in φ . A formula without free variables is called a *sentence*. As customary, we extend the language of first-order logic by the usual abbreviations, e.g., $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$ and $\forall x_i(\alpha) \equiv \neg\exists x_i(\neg\alpha)$. To increase readability, we will use other lowercase Latin letters for variables and drop unnecessary braces by using the usual operator precedence instead. Furthermore, we use the *dot notation* in which we place a “.” instead of an opening brace and silently close it at the latest syntactically correct position. A τ -structure \mathcal{S} is a *model* of a sentence $\varphi \in \mathcal{L}(\tau)$, denoted by $\mathcal{S} \models \varphi$, if it evaluates to true under the semantics of quantified propositional logic while interpreting equality and relational symbols as specified by the structure. For instance, $\varphi_{\text{undir}} = \forall x \forall y \cdot Exy \rightarrow Eyx$ over $\tau = \{E^2\}$ describes the set of undirected graphs, and we have $\text{undir} \models \varphi_{\text{undir}}$ and $\text{dir} \not\models \varphi_{\text{undir}}$.

We obtain the language of *second-order logic* by allowing quantification over relational variables of arbitrary arity, which we will denote by uppercase Latin letters. A relational variable is said to be *monadic* if its arity is one. A *monadic second-order* formula is one in which all quantified relational variables are monadic. The set of all such formulas is denoted by MSO. The *model checking problem* for a vocabulary τ is the set $\text{MC}_\tau(\text{MSO})$ that contains all pairs (\mathcal{S}, φ) of τ -structures \mathcal{S} and MSO sentences φ with $\mathcal{S} \models \varphi$. Whenever τ is not relevant (meaning that a result holds for all fixed τ), we will refer to the problem as $\text{MC}(\text{MSO})$. We note that in the literature there is often a distinction between MSO_1 - and MSO_2 -logic, which describes the way the input is encoded [20]. Since we allow arbitrary relations, we do not have to make this distinction.

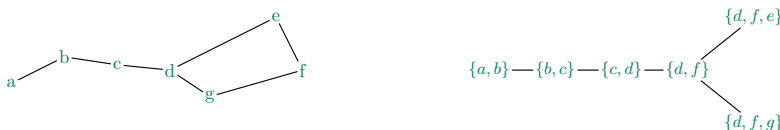
2.2 Treewidth and Tree Decompositions

While we consider graphs G as relational structures \mathcal{G} as discussed in the previous section, we also use common graph-theoretic terminology and denote with $V(G) = U(\mathcal{G})$ and $E(G) = E^{\mathcal{G}}$ the vertex and edge set of G . Unless stated otherwise, graphs in this paper are *undirected* and we use the natural set notations and write, for instance, $\{v, w\} \in E(G)$. The *degree* of a vertex is the number of its neighbors. A *tree decomposition* of G is a pair (T, χ) in which T is a tree (a connected graph without cycles) and $\chi: V(T) \rightarrow 2^{V(G)}$ a function with the following two properties:

1. for every $v \in V(G)$ the set $\{x \mid v \in \chi(x)\}$ is non-empty and connected in T ;
2. for every $\{u, v\} \in E(G)$ there is at least one node $x \in V(T)$ with $\{u, v\} \subseteq \chi(x)$.

The *width* of a tree decomposition is the maximum size of its bags minus one, i.e., $\text{width}(T, \chi) = \max_{x \in V(T)} |\chi(x)| - 1$. The *treewidth* $\text{tw}(G)$ of a graph G is the minimum width any tree decomposition of G must have. We do not require additional properties of tree decompositions, but we assume that T is rooted at a $\text{root}(T) \in V(T)$ and, thus, that nodes $t \in V(T)$ may have a $\text{parent}(t) \in V(T)$ and $\text{children}(t) \subseteq V(T)$. Without loss of generality, we may also assume $|\text{children}(t)| \leq 2$.

► **Example 7.** The treewidth of the Big Dipper constellation (as graph shown on the left) is at most two, as proven by the tree decomposition on the right:



2.3 Treewidth of Propositional Formulas and Relational Structures

The definition of treewidth can be lifted to other objects by associating a graph to them. The most common graph for CNFs (or DNFs) ψ is the *primal graph* G_ψ , which is the graph on vertex set $V(G_\psi) = \text{vars}(\psi)$ that connects two vertices by an edge if the corresponding variables appear together in a clause. We then define $\text{tw}(\psi) := \text{tw}(G_\psi)$ and refer to a tree decomposition of G_ψ as one of ψ . Note that other graphical representations lead to other definitions of the treewidth of propositional formulas. A comprehensive listing can be found in the *Handbook of Satisfiability* [7, Chapter 17]. A *labeled tree decomposition* (T, χ, λ) extends a tree decomposition with a mapping $\lambda: V(T) \rightarrow 2^\psi$ (i.e., a mapping from the nodes of T to a subset of the clauses (or terms) of ψ) such that for every clause (or term) C there is exactly one $t \in V(T)$ with $C \in \lambda(t)$ that contains all variables appearing in C . It is easy to transform a tree decomposition (T, χ) into a labeled one (T, χ, λ) by traversing the tree once's and by duplicating some bags. Hence, we will assume throughout this article that all tree decompositions are labeled.

A similar approach can be used to define tree decompositions of arbitrary structures: The *primal graph* $G_\mathcal{S}$ of a structure \mathcal{S} , in this context also called the *Gaifman graph*, has as vertex set the universe of \mathcal{S} , i.e., $V(G_\mathcal{S}) = U(\mathcal{S})$, and contains an edge $\{u, v\} \in E(G_\mathcal{S})$ iff u and v appear together in some tuple of \mathcal{S} . As before, we define $\text{tw}(\mathcal{S}) := \text{tw}(G_\mathcal{S})$. One can alternatively define the concept of tree decompositions directly over relational structures, which leads to the same definition [17].

3 New Upper Bounds for Second-Order Propositional Logic

Central to our reductions are treewidth-preserving encodings from QSAT to SAT and from PMC to #SAT. These encoding establishes new proofs of Chen's Theorem [11] and the theorem by Fichte et al. [15], and improve the dependencies on k in the tower of Fact 2 and 3.

3.1 Treewidth-Aware Encodings from QSAT to SAT

We use a quantifier elimination scheme that eliminates the most-inner quantifier block at the cost of introducing $O(2^k|\varphi|)$ new variables while increasing the treewidth by a factor of $12 \cdot 2^k$. Let first $\varphi = Q_1 S_1 \dots \forall_\ell S_\ell \cdot \psi$ be the given QBF, in which ψ is a DNF. Let further (T, χ, λ) be the given labeled width- k tree decomposition of φ . We describe an encoding into a QBF, in which the last quantifier block $Q_\ell S_\ell$ gets replaced by new variables in $S_{\ell-1}$.

We have to encode the fact that for an assignment on $\bigcup_{i=1}^{\ell-1} S_i$ all assignments to S_ℓ satisfy ψ , i.e., at least *one* term in ψ . For that end, we introduce auxiliary variables for every term $d \in \text{terms}(\psi)$ and any partial assignment α of the variables in S_ℓ that also appear in the bag that contains d . More precisely, let $\lambda^{-1}(d)$ be the node in $V(T)$ with $d \in \lambda(t)$ and let $\alpha \sqsubseteq \chi(\lambda^{-1}(d)) \cap S_\ell$ be an assignment of the variables of the bag that are quantified by Q_ℓ . We introduce the variable sat_d^α that indicates that this assignment satisfies d :

$$\bigwedge_{d \in \text{terms}(\psi)} \bigwedge_{\alpha \sqsubseteq \chi(\lambda^{-1}(d)) \cap S_\ell} \left[\text{sat}_d^\alpha \leftrightarrow \bigwedge_{x \in \text{lits}(\{d\}|\alpha)} x \right], \quad // \alpha \text{ may satisfy } d \quad (1)$$

$$\bigwedge_{d \in \text{terms}(\psi)} \bigwedge_{\alpha \sqsubseteq \chi(\lambda^{-1}(d)) \cap S_\ell} \left[\neg \text{sat}_d^\alpha \right]. \quad // \alpha \text{ falsifies } d \quad (2)$$

We have to track whether ψ can be satisfied by a local assignment α . For every $t \in V(T)$ and every $\alpha \sqsubseteq \chi(t) \cap S_\ell$ we introduce a variable $\text{sat}_{\leq t}^\alpha$ that indicates that α can be extended to a satisfying assignment for the subtree rooted at t . Furthermore, we create variables

$sat_{<t,t'}^\alpha$ for $t' \in \text{children}(t)$ that propagate the information about satisfiability along the tree decomposition. That is, $sat_{<t,t'}^\alpha$ is set to true if there is an assignment $\beta \sqsubseteq \chi(t') \cap S_\ell$ that can be extended to a satisfying assignment and that is compatible with α :

// Either there is a term satisfying the bag or we can propagate:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \chi(t) \cap S_\ell} \left[sat_{\leq t}^\alpha \leftrightarrow \bigvee_{d \in \lambda(t)} sat_d^\alpha \vee \bigvee_{t' \in \text{children}(t)} sat_{<t,t'}^\alpha \right], \quad (3)$$

// Propagate satisfiability:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \chi(t) \cap S_\ell} \bigwedge_{t' \in \text{children}(t)} \left[sat_{<t,t'}^\alpha \leftrightarrow \bigwedge_{\substack{\beta \sqsubseteq \chi(t') \cap S_\ell \\ \beta \cap \text{lits}(\chi(t)) = \alpha \cap \text{lits}(\chi(t'))}} sat_{\leq t'}^\beta \right]. \quad (4)$$

Finally, since $Q_\ell = \forall$, we need to ensure that for all possible assignments of S_ℓ there is at least one term that gets satisfied. Since satisfiability gets propagated to the root of the tree decomposition by the aforementioned constraint, we can enforce this property with:

$$\bigwedge_{\alpha \sqsubseteq \chi(\text{root}(T)) \cap S_\ell} sat_{\leq \text{root}(T)}^\alpha. \quad (5)$$

The following lemma observes the correctness of the construction, and the subsequent lemma handles the case $Q_\ell = \exists$.

► **Lemma 8** (▼). *There is an algorithm that, given a QBF $\varphi = Q_1 S_1 \dots \exists_{\ell-1} S_{\ell-1} \forall_\ell S_\ell \cdot \psi$ and a width- k tree decomposition of G_φ , outputs in time $O^*(2^k)$ a QBF $\varphi' = Q_1 S_1 \dots \exists_{\ell-1} S'_{\ell-1} \cdot \psi'$ and a width- $(12 \cdot 2^k)$ tree decomposition of $G_{\varphi'}$ such that φ is valid iff φ' is valid.*

► **Lemma 9** (▼). *There is an algorithm that, given a QBF $\varphi = Q_1 S_1 \dots \forall_{\ell-1} S_{\ell-1} \exists_\ell S_\ell \cdot \psi$ and a width- k tree decomposition of G_φ , outputs in time $O^*(2^k)$ a QBF $\varphi' = Q_1 S_1 \dots \forall_{\ell-1} S'_{\ell-1} \cdot \psi'$ and a width- $(12 \cdot 2^k)$ tree decomposition of $G_{\varphi'}$ such that φ is valid iff φ' is valid.*

Sketch of Proof. The case $Q_\ell = \exists$ (in which ψ is a CNF) works similarly: The result follows by negating the inverse, where the roles of CNF and DNF are switched, and universal and existential quantification are switched as well. ◀

Proof of Theorem 1. The theorem follows by exhaustively applying Lemma 8 and Lemma 9 until a CNF is reached. The price for removing one alternation are $O(2^k |\varphi|)$ new variables and an increase of the treewidth by a factor of $12 \cdot 2^k$. Hence, after removing one quantifier block we have a treewidth of $12 \cdot 2^k \leq 2^{k+\log 12}$, after two we have $12 \cdot 2^{k+\log 12} \leq 2^{k+\log 12+\log 12}$, after three we then have $2^{2^{k+\log 12+\log 12+\log 12}}$; and so on. We can bound all the intermediate “+ log 12” by adding a “+1” on top of the tower, leading to a bound on the treewidth of $\text{tower}(\text{qa}(\varphi), k + \log 12 + 1) \leq \text{tower}(\text{qa}(\varphi), k + 4.59)$. In fact, we can bound the top of the tower even tighter by observing $\log 12 \leq 3.59$ and guessing 3.92 as a fix point. Inserting yields $3.59 + 2^{3.59+k} \leq 2^{3.92+k}$ and $2^{3.92+2^{3.59+k}} \leq 2^{3.92+k}$. Consequently, we can bound the treewidth of the encoding by $\text{tower}(\text{qa}(\varphi), k+3.92)$ and the size by $\text{tower}^*(\text{qa}(\varphi), k+3.92)$. ◀

3.2 Treewidth-Aware Encodings from PMC to #SAT

Recall that the input for PMC is a CNF ψ and a set $X \subseteq \text{vars}(\psi)$. The task is to count the assignments $\alpha \sqsubseteq X$ that can be extended to models $\alpha^* \sqsubseteq \text{vars}(\psi)$ of ψ . We can also think of a formula $\psi(X) = \exists Y \cdot \psi'(X, Y)$ with free variables X and existential quantified variables Y

(ψ' is quantifier-free), for which we want to count the assignments to X that make the formula satisfiable. The idea is to rewrite $\psi(X) = \exists Y. \psi'(X, Y) \equiv \exists X \exists Y. \psi'(X, Y)$, and to use a similar encoding as in the proof of Lemma 9 to remove the second quantifier.

In detail, we add a variable sat_c^α for every clause $c \in \text{clauses}(\psi)$ and every assignment of the corresponding bag $\alpha \sqsubseteq \chi(\lambda^{-1}(c)) \cap Y$. The semantic of this variable is that the clause c is satisfiable under the partial assignment α . We further add the propagation variables $sat_{\leq t}^\alpha$ and $sat_{< t, t'}$ for all $t \in V(T)$, $t' \in \text{children}(t)$, and $\alpha \sqsubseteq \chi(\lambda^{-1}(c)) \cap Y$. The former indicates that the assignment α can be extended to a satisfying assignment of the subtree rooted at t ; the later propagates partial solutions from children to parents within the tree decomposition:

// α may satisfy c :

$$\bigwedge_{c \in \text{clauses}(\psi)} \bigwedge_{\alpha \sqsubseteq \chi(\lambda^{-1}(c)) \cap Y} \left[sat_c^\alpha \leftrightarrow \bigvee_{\ell \in \text{lits}(\{c\}|\alpha)} \ell \right], \quad (1)$$

// α satisfies c :

$$\bigwedge_{c \in \text{clauses}(\psi)} \bigwedge_{\alpha \sqsubseteq \chi(\lambda^{-1}(c)) \cap Y} \left[sat_c^\alpha \right]. \quad (2)$$

// Either there is a clause satisfying the bag or we can propagate:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \chi(t) \cap Y} \left[sat_{\leq t}^\alpha \leftrightarrow \bigwedge_{c \in \lambda(t)} sat_c^\alpha \wedge \bigwedge_{t' \in \text{children}(t)} sat_{< t, t'}^\alpha \right], \quad (3)$$

// Propagate satisfiability:

$$\bigwedge_{t \in V(T)} \bigwedge_{\alpha \sqsubseteq \chi(t) \cap Y} \bigwedge_{t' \in \text{children}(t)} \left[sat_{< t, t'}^\alpha \leftrightarrow \bigwedge_{\substack{\beta \sqsubseteq \chi(t') \cap Y \\ \beta \cap \text{lits}(\chi(t)) = \alpha \cap \text{lits}(\chi(t'))}} sat_{\leq t'}^\beta \right]. \quad (4)$$

Observe that the constraints (1)–(4) contain no variable from Y (we removed them by locally speaking about α) and, furthermore, constraints (1), (3), and (4) are pure propagations, which leave no degree of freedom on the auxiliary variables. Hence, models of these constraint only have freedom in the variables in X within constraint (2). We are left with the task to count only models that actually satisfy the input formula, which we achieve with:

$$\bigvee_{\alpha \sqsubseteq \chi(\text{root}(T)) \cap Y} sat_{\leq \text{root}(T)}^\alpha. \quad (5)$$

► **Lemma 10 (▼).** *There is an algorithm that, given a CNF ψ , a set $X \subseteq \text{vars}(\psi)$, and a width- k tree decomposition of G_ψ , outputs in time $O^*(2^k)$ a CNF ψ' and a width- $(12 \cdot 2^k)$ tree decomposition of $G_{\psi'}$ such that the projected model count of ψ on X equals $\#(\psi')$.*

Proof of Theorem 2. By applying Fact 1 to the formula generated by Lemma 10 we obtain an algorithm for PMC with running time $\text{tower}^*(2, k + 3.59)$. ◀

4 A SAT Version of Courcelle's Theorem

We demonstrate the power of treewidth-aware encodings by providing an alternative proof of Courcelle's theorem. We prove the main part of Theorem 3 in the following form:

► **Lemma 11.** *There is an algorithm that, given a relational structure \mathcal{S} , a width- k tree decomposition of \mathcal{S} , and an MSO sentence φ in prenex normal form, produces in time $\text{tower}^*(\text{qa}(\varphi), (k+9)|\varphi| + 3.92)$ a propositional formula ψ and tree decomposition of G_ψ of width $\text{tower}(\text{qa}(\varphi), (k+9)|\varphi| + 3.92)$ such that $\mathcal{S} \models \varphi \Leftrightarrow \psi \in \text{SAT}$.*

The lemma assumes that the sentence is in prenex normal form with a quantifier-free part ψ in CNF, i.e., $\varphi \equiv Q_1 S_1 \dots Q_{q-1} S_{q-1} Q_q s_q \dots Q_\ell s_\ell \cdot \bigwedge_{i=1}^p \psi_i$ with $Q_i \in \{\exists, \forall\}$ and S_i (s_i) being second-order (first-order) variables. The requirement that the second-order quantifiers appear before the first-order ones is for sake of presentation, the encoding works as is if the quantifiers are mixed. The main part of the proof is a treewidth-aware encoding from MC(MSO) into QSAT; which is then translated to SAT using Theorem 1.

4.1 Auxiliary Encodings

Let ψ be a propositional formula and $X \subseteq \text{lits}(\psi)$ be an arbitrary set of literals. A *cardinality constraint* $\text{card}_{\bowtie c}(X)$ with $\bowtie \in \{\leq, =, \geq\}$ ensures that $\{ \text{at most, exactly, at least} \} c$ literals of X get assigned to true. Classic encodings of cardinality constraints increase the treewidth of ψ by quite a lot. For instance, the naive encoding for $\text{card}_{\leq 1}(X) \equiv \bigwedge_{u,v \in X; u \neq v} (\neg u \vee \neg v)$ completes X into a clique. We encode a cardinality constraint without increasing the treewidth by distributing a *sequential unary counter*:

► **Lemma 12** (▼). *For every $c \geq 0$ we can, given a CNF ψ , a set $X \subseteq \text{lits}(\psi)$, and a width- k tree decomposition of ψ , encode $\text{card}_{\bowtie c}(X)$ such that $\text{tw}(\psi \wedge \text{card}_{\bowtie c}(X)) \leq k + 3c + 3$.*

Sketch of Proof. We add $c + 1$ variables to every bag t of the tree decomposition, which count the number of literals set to true in the subtree rooted at t . The semantics of the sequential counter encoding [31] is then implemented along the edges of the decomposition. To cover the new constraints, we can add the auxiliary variables of the (at most two) children of t to the bag of t as well, resulting in an overall increase of the treewidth by $3c + 3$. ◀

The second auxiliary encoding is a treewidth-preserving conversion from CNFs to DNFs.

► **Lemma 13.** *There is a polynomial-time algorithm that, given a CNF ψ and a width- k tree decomposition of G_ψ , produce a DNF ψ' and a width- $(k+4)$ tree decomposition of $G_{\psi'}$ such that for any $\alpha \sqsubseteq \text{vars}(\psi)$, $\psi|\alpha = \emptyset$ iff $\psi'|\alpha$ is a tautology ($\neg(\psi'|\alpha)$ is unsatisfiable).*

Sketch of Proof. For every clause C we add a variable f_C that is true iff C is satisfied. Satisfiability is encoded along the tree by variables $f_{\leq t}$ indicating that ψ is satisfied in the subtree rooted at t via $\bigvee_{t \in V(T)} \neg \left[f_{\leq t} \leftrightarrow \bigwedge_{C \in \lambda(t)} f_C \wedge \bigwedge_{t' \in \text{children}(t)} f_{\leq t'} \right]$. ◀

4.2 Indicator Variables for the Quantifiers

To prove Lemma 11 we construct a QBF for a given MSO sentence φ , structure \mathcal{S} , and tree decomposition of \mathcal{S} . We first define the primary variables of ψ , i.e., the prefix of ψ (primary here refers to the fact that we will also need some auxiliary variables later). For every second-order quantifier $\exists X$ or $\forall X$ we introduce, as we did in the introduction, an indicator variable X_u for every element $u \in U(\mathcal{S})$ with the semantic that X_u is true iff $u \in X$. These variables are either existentially or universally quantified, depending on the second-order quantifier. If there are multiple quantifiers (say $\exists X \forall Y$), the order in which the variables are quantified is the same as the order of the second-order quantifiers. For first-order quantifiers $\exists x$ or $\forall x$ we do the same construction, i.e., we add variables x_u for all $u \in U(\mathcal{S})$ with the semantics that x_u is true iff x was assigned to u . Of course, of these variables we have to set *exactly one* to true, which we enforce by adding $\text{card}_{=1}(\{x_u \mid u \in U(\mathcal{S})\})$ using Lemma 12.

4.3 Evaluation of Atoms

The last ingredient of our QBF encoding is the evaluation of the atoms in the MSO sentence φ . An atom is Rx_1, \dots, x_a for a relational symbol R from the vocabulary of arity a , containment in a second-order variable Xu , equality $x = y$, and the negation of the aforementioned. For every atom ι that appears in φ we introduce variables p_t^ι and $p_{\leq t}^\iota$ for all $t \in V(T)$ that indicate that ι is true in bag t or somewhere in the subtree rooted at t , respectively. Note that the same atom can occur multiple times in φ , for instance in

$$\forall x \forall y \exists z . (x = y \rightarrow x = z) \vee (x = y \rightarrow y = z)$$

there are two atoms $x = y$. However, since φ is in prenex normal form (and, thus, variables cannot be rebound), these always evaluate in exactly the same way. Hence, it is sufficient to consider the *set of atoms*, which we denote by $\text{atoms}(\varphi)$. We can propagate information about the atoms along the tree decomposition with:

$$\bigwedge_{t \in V(T)} \bigwedge_{\iota \in \text{atoms}(\varphi)} \left[p_{\leq t}^\iota \leftrightarrow (p_t^\iota \vee \bigvee_{t' \in \text{children}(t)} p_{\leq t'}^\iota) \right].$$

This encoding introduces two variables per atom ι per bag t (namely p_t^ι and $p_{\leq t}^\iota$), which increases the treewidth by at most $2 \cdot |\text{atoms}(\varphi)|$. To synchronize with the two children t' and t'' , we add $p_{\leq t'}^\iota$ and $p_{\leq t''}^\iota$ to $\chi(t)$, yielding a total treewidth of at most $4 \cdot |\text{atoms}(\varphi)|$.

An easy atom to evaluate is $x = y$, since if x and y are equal (i.e., they both got assigned to the same element $u \in U(\mathcal{S})$), we can conclude this fact within a bag that contains u :

$$\bigwedge_{t \in V(T)} \left[p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \right].$$

For every $u \in U(\mathcal{S})$ and every quantifier $\exists x$ (or $\forall x$), we add the propositional variable x_u to all bags containing u . We increase the treewidth by at most the quantifier rank and, in return, cover constraints as the above trivially. Similarly, if there is a second-order variable X and a first-order variable x , the atom Xx can be evaluated locally in every bag:

$$\bigwedge_{t \in V(T)} \left[p_t^{Xx} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \right].$$

We have to evaluate atoms corresponding to relational symbols R of the vocabulary. For each such symbol of arity a we encode:

$$\bigwedge_{t \in V(T)} \left[p_t^{R(x_1, x_2, \dots, x_a)} \leftrightarrow \bigvee_{\substack{u_1, \dots, u_a \in \chi(t) \\ (u_1, \dots, u_a) \in R^{\mathcal{S}}}} ((x_1)_{u_1} \wedge (x_2)_{u_2} \wedge \dots \wedge (x_a)_{u_a}) \right].$$

Here “ $R(x_1, x_2, \dots, x_a)$ ” is an atom in which R is a relational symbol and x_1, x_2, \dots, x_a are quantified first-order variables. In the inner “big-or” we consider all u_1, \dots, u_a in $\chi(t)$, i.e., elements $u_1, \dots, u_a \in U(\mathcal{S})$ that are in the relation $(u_1, \dots, u_a) \in R^{\mathcal{S}}$. Then “ $(x_i)_{u_i}$ ” is a variable that describes that x_i gets assigned to u_i . Note that all tuples in $R^{\mathcal{S}}$ appear together in at least one bag of the tree decomposition and, hence, there is at least one bag t for which $p_t^{R(x_1, x_2, \dots, x_a)}$ can be evaluated to true. The propagation ensures that, for every $\iota \in \text{atoms}(\varphi)$, the variable $p_{\leq \text{root}(T)}^\iota$ will be true iff ι is true. Since the quantifier-free part of φ is a CNF $\bigwedge_{j=1}^p \psi_j$, we can encode it by replacing every occurrence of ι in ψ_j with $p_{\leq \text{root}(T)}^\iota$.

4.4 The Full Encoding in one Figure

For the readers convenience, we compiled the encoding into Figure 1. Combining the insights of the last sections proves Lemma 11, but if the inner-most quantifier is universal, existentially projecting the encoding variables would produce a QBF with one more block. This can, however, be circumvent using Lemma 13. We formally prove that “combining the insights” indeed leads to a sound proof of Lemma 11 in the technical report.

Cardinality Propagation

$$c_{\leq t}^x \leftrightarrow \bigvee_{u \in \chi(t) \setminus \chi(\text{parent}(t))} x_u \vee \bigvee_{t' \in \text{children}(t)} c_{\leq t'}^x \quad \text{for every } t \text{ in } T, x \in \{s_q, \dots, s_\ell\} \quad (1)$$

At-Least-One Constraint

$$c_{\leq \text{root}(T)}^x \quad \text{for every } x \in \{s_q, \dots, s_\ell\} \quad (2)$$

At-Most-One Constraint

$$\neg x_u \vee \neg x_{u'} \quad \text{for every } t \text{ in } T, u, u' \in \chi(t), u \neq u', x \in \{s_q, \dots, s_\ell\} \quad (3)$$

$$\neg x_u \vee \neg c_{\leq t'}^x \quad \text{for every } t \text{ in } T, t' \in \text{children}(t), u \in \chi(t) \setminus \chi(\text{parent}(t)), x \in \{s_q, \dots, s_\ell\} \quad (4)$$

$$\neg c_{\leq t'}^x \vee \neg c_{\leq t''}^x \quad \text{for every } t \text{ in } T, t', t'' \in \text{children}(t), t' \neq t'', x \in \{s_q, \dots, s_\ell\} \quad (5)$$

Proofs of MSO Atoms

$$p_t^{x=y} \leftrightarrow \bigvee_{u \in \chi(t)} (x_u \wedge y_u) \quad \text{for every } t \text{ in } T, x, y \in \{s_q, \dots, s_\ell\}, (x=y) \in \text{atoms}(\varphi) \quad (6)$$

$$p_t^{X(x)} \leftrightarrow \bigvee_{u \in \chi(t)} (X_u \wedge x_u) \quad \text{for every } t \text{ in } T, X \in \{S_1, \dots, S_{q-1}\}, x \in \{s_q, \dots, s_\ell\}, X(x) \in \text{atoms}(\varphi) \quad (7)$$

$$p_t^{R(x_1, \dots, x_a)} \leftrightarrow \bigvee_{\substack{u_1, \dots, u_a \in \chi(t) \\ (u_1, \dots, u_a) \in R^S}} ((x_1)_{u_1} \wedge \dots \wedge (x_a)_{u_a}) \quad \text{for every } t \text{ in } T, \{x_1, \dots, x_a\} \subseteq \{s_q, \dots, s_\ell\}, \\ R \in \mathcal{S}, R(x_1, \dots, x_r) \in \text{atoms}(\varphi) \quad (8)$$

$$p_{\leq t}^\iota \leftrightarrow p_t^\iota \vee \bigvee_{t' \in \text{children}(t)} p_{\leq t'}^\iota \quad \text{for every } t \text{ in } T, \iota \in \text{atoms}(\varphi) \quad (9)$$

Deriving MSO Atoms requires Proof

$$\iota \leftrightarrow p_{\leq \text{root}(T)}^\iota \quad \text{for every } t \text{ in } T, \iota \in \text{atoms}(\varphi) \quad (10)$$

Verify MSO Formula

$$\psi \quad (11)$$

■ **Figure 1** The reduction $\mathcal{R}_{\text{MSO} \rightarrow \text{QSAT}}(\varphi, \mathcal{S}, \mathcal{T})$ that takes as input an MSO formula in prenex normal form $\varphi = Q_1 S_1 \dots Q_{q-1} S_{q-1} Q_q s_q \dots Q_\ell s_\ell \cdot \psi$ and a structure \mathcal{S} with a TD $\mathcal{T} = (T, \chi)$ of \mathcal{S} of width k . It obtains a QBF $\varphi' = Q_1 S'_1 \dots Q_\ell S'_\ell \exists E' \cdot \psi'$, where ψ' is the conjunction of Equations (1)–(11), $S'_i = \{(S_i)_u \mid u \in U(\mathcal{S})\}$ and $E' = \text{vars}(\psi') \setminus (\bigcup_{i=1}^\ell S'_i)$. Formula ψ' can be easily converted into CNF of width linear in k (for constant-size MSO formulas φ).

5 Fagin Definability via Automated Reasoning

In this section we prove the remaining two items of Theorem 3, i.e., a treewidth-aware encoding of the optimization version of Courcelle’s Theorem to MAXSAT; and a #SAT encoding of the counting version of the theorem. The general approach is as follows: We obtain a MSO formula

$\varphi(X)$ with a free set variable X as input (rather than a MSO sentence as in Lemma 11). The objective of the model-checking problems adds requirements to this variable (for $\text{FD}(\text{MSO})$ we seek a $S \subseteq U(\mathcal{S})$ of minimum size such that $\mathcal{S} \models \varphi(S)$; for $\#\text{FD}(\text{MSO})$ we want to count the number of sets $S \subseteq U(\mathcal{S})$ with $\mathcal{S} \models \varphi(S)$). The “trick” is to rewrite $\varphi(X) = \xi$ as $\varphi' = \exists X \xi$ and apply Lemma 11 to φ' in order to obtain a propositional formula ψ . Observe that the quantifier alternation of φ' may be one larger than the one of φ .

► **Lemma 14** (▼). *There is an algorithm that, given a structure \mathcal{S} with weights $w_i: U(\mathcal{S}) \rightarrow \mathbb{Q}$ for $i \in \{1, \dots, \ell\}$, a width- k tree decomposition of \mathcal{S} , and an MSO formula $\varphi(X_1, \dots, X_\ell)$ in prenex normal form, produces in time $\text{tower}^*(\text{qa}(\varphi) + 1, (k + 9)|\varphi| + 3.92)$ a WCNF ψ and a tree decomposition of width $\text{tower}(\text{qa}(\varphi) + 1, (k + 9)|\varphi| + 3.92)$ of G_ψ such that the maximum weight of any model of ψ equals the maximum value of $\sum_{i=1}^{\ell} \sum_{s \in S_i} w_i(s)$ under $S_1, \dots, S_\ell \subseteq U(\mathcal{S})$ with $\mathcal{S} \models \varphi(S_1, \dots, S_\ell)$.*

Sketch of Proof. Consider $\psi \wedge \bigwedge_{u \in U(\mathcal{S})} (\neg X_u)$ such that the clauses in ψ are *hard* and the added clauses are *soft*. A model maximizing the soft clauses will minimize the number of X_u variables set to true, i.e., corresponds to a minimum-size set S with $\mathcal{S} \models \varphi(S)$. ◀

► **Lemma 15** (▼). *There is an algorithm that, given a relational structure \mathcal{S} , a width- k tree decomposition of \mathcal{S} , and an MSO formula $\varphi(X_1, \dots, X_\ell)$ in prenex normal form, produces in time $\text{tower}^*(\text{qa}(\varphi) + 1, (k + 9)|\varphi| + 3.92)$ a CNF ψ and a tree decomposition of width $\text{tower}(\text{qa}(\varphi) + 1, (k + 9)|\varphi| + 3.92)$ of G_ψ such that the number of models of ψ equals the number of sets $S_1, \dots, S_\ell \subseteq U(\mathcal{S})$ with $\mathcal{S} \models \varphi(S_1, \dots, S_\ell)$.*

Sketch of Proof. We need to compute the number of models of ψ projected to the X_u variables. In other words, it is sufficient to solve the *projected model counting problem* on the instance generated with Lemma 11 using Lemma 10. ◀

6 Lower Bounds for the Encoding Size of Model Checking Problems

We companion our SAT encodings for $\text{MC}(\text{MSO})$ with lower bounds on the achievable encoding size under ETH. The first lower bound (Theorem 5) is obtained by an encoding from QSAT into $\text{MC}(\text{MSO})$ that implies that SAT encodings of $\text{MC}(\text{MSO})$ lead to faster QSAT algorithms.

► **Lemma 16** (▼). *There is a polynomial-time algorithm that, given a QSAT sentence ψ , outputs a structure \mathcal{S} and an MSO sentence φ with $\text{tw}(\mathcal{S}) \leq \text{tw}(\psi) + 1$ and $\text{qa}(\varphi) \leq \text{qa}(\psi) + 2$ such that $\mathcal{S} \models \varphi$ iff ψ evaluates to true.*

Sketch of Proof. The structure \mathcal{S} is the *incidence graph* of ψ (the graph containing a node for every variable and every clause that connects variables to the clauses containing them) with some additional labels. The sentence φ uses $\text{qa}(\psi)$ second-order quantifier to guess the assignment of ψ , and one additional $\forall x \exists y$ -block to evaluate it. ◀

Proof of Theorem 5. Combine Lemma 16 with Fact 5. ◀

6.1 An Encoding for Compressing Treewidth

For QSAT one can “move” complexity from the quantifier rank of the formula to its treewidth and *vice versa* [16]. By Lemma 16, this means that any reduction from QSAT to $\text{MC}(\text{MSO})$ may produce an instance with small treewidth or quantifier alternation while increasing the other. We show that one can also decrease the treewidth by increasing the *block size*.

► **Lemma 17 (▼).** *For every $c > 0$ there is a polynomial-time algorithm that, on input of a CNF ψ and a width- k tree decomposition of G_ψ , outputs a constant-size MSO sentence φ with $\text{qa}(\varphi) = 2$ and $\text{bs}(\varphi) = c$, and a structure \mathcal{S} with $\text{tw}(\mathcal{S}) \leq \lceil \frac{k+1}{c} \rceil$ such that $\psi \in \text{SAT} \Leftrightarrow \mathcal{S} \models \varphi$.*

Sketch of Proof. The idea of the proof is to (i) encode the input’s formula as an incidence graph over which we reason with an MSO sentence; we then (ii) replace this structure by its tree decomposition with additional “sync edges”; and finally we (iii) contract vertices in the tree decomposition to lower the treewidth, while we encode statements like $x \in S$ (for a set variable S) by defining new set variables S_1, \dots, S_c and by interpreting $y \in S_i$ as “the i th vertex contracted to y is in S ”. ◀

Proof of Theorem 6. We obtain the Trade-off Theorem by combining the proof strategy of Lemma 17 with the reduction from QSAT to MC(MSO) of Lemma 16. The result is a polynomial-time algorithm for every $c > 0$ that, on input of a QBF ψ and a width- k tree decomposition of G_ψ , outputs a constant-size MSO sentence φ with $\text{qa}(\varphi) \leq \text{qa}(\psi) + 2$ and $\text{bs}(\varphi) = c$, and a structure \mathcal{S} with $\text{tw}(\mathcal{S}) \leq \lceil \frac{k+1}{c} \rceil$ such that ψ is valid iff $\mathcal{S} \models \varphi$. ◀

It is out of the scope of this article, but worth mentioning, that the proofs of Lemma 17 and Theorem 6 can be generalized to the following finite-model theoretic result:

► **Proposition 18.** *For every $c > 0$ there is a polynomial-time algorithm that, given a relational structure \mathcal{S} , a width- k tree decomposition of \mathcal{S} , and an MSO sentence φ , outputs a structure \mathcal{S}' and a sentence φ' such that:*

1. $\mathcal{S} \models \varphi \iff \mathcal{S}' \models \varphi'$;
2. $\text{tw}(\mathcal{S}') \leq \lceil \frac{k+1}{c} \rceil$;
3. $\text{bs}(\varphi') \leq c \cdot \text{bs}(\varphi)$.

7 Conclusion and Further Research

We studied *structure-guided automated reasoning*, where we utilize the input’s structure in propositional encodings. The scientific question we asked was whether we can encode every MSO definable problem on structures of bounded treewidth into SAT formulas of bounded treewidth. We proved this in the affirmative, implying an alternative proof of Courcelle’s Theorem. The most valuable aspects are, in our opinion, the simplicity of the proof (it is “just” an encoding into propositional logic) and the potential advantages in practice for formulas of small quantifier alternation (SAT solvers are known to perform well on instances of small treewidth, even if they do not actively apply techniques such as dynamic programming). Another advantage is the surprisingly simple generalization to the optimization and counting version of Courcelle’s Theorem – we can directly “plug in” MAXSAT or #SAT and obtain the corresponding results. As a byproduct, we also obtain new proofs showing (purely as encodings into propositional logic) that QSAT parameterized by the input’s treewidth plus quantifier alternation is fixed-parameter tractable (improving a complex dynamic program with nested tables) and that PMC parameterized by treewidth is fixed-parameter tractable (improving a multi-pass dynamic program). Table 1 provides an overview of the encodings presented within this article.

Our encodings are exponentially smaller than the best known running time for MC(MSO), i.e., when we solve the instances using Fact 1, we obtain the same runtime. We complemented this finding with new ETH-based lower bounds. Further research will be concerned with closing the remaining gap in the height of the tower between the lower and upper bounds. We show in an upcoming paper that the terms “ $\text{qa}(\varphi)$ ” in Theorem 3 and “ $\text{qa}(\varphi) - 2$ ” in

■ **Table 1** We summarize the encodings presented within this article. An encoding maps *from* one problem *to* another. The third and fourth columns define the treewidth and size of the encoding, whereby we assume that $c > 0$ is a constant, a width- k tree decomposition is given, ψ is a propositional formula, and φ is a fixed MSO formula.

<i>Encoding...</i>				
<i>From</i>	<i>To</i>	<i>Treewidth</i>	<i>Size</i>	<i>Reference</i>
QSAT	SAT	$\text{tower}(\text{qa}(\psi), k + 3.92)$	$\text{tower}^*(\text{qa}(\psi), k + 3.92)$	Theorem 1
PMC	#SAT	$\text{tower}(1, k + 3.59)$	$\text{tower}^*(1, k + 3.59)$	Theorem 2
$\text{card}_{\geq c}(X)$	SAT	$k + 3c + 3$	$O(c \psi)$	Lemma 12
CNF	DNF	$k + 4$	$O(\psi)$	Lemma 13
MC(MSO)	SAT	$\text{tower}(\text{qa}(\varphi), (9k + 9) \varphi + 3.92)$	$\text{tower}^*(\text{qa}(\varphi), (9k + 9) \varphi + 3.92)$	Lemma 11
FD(MSO)	MAXSAT	$\text{tower}(\text{qa}(\varphi) + 1, (9k + 9) \varphi + 3.92)$	$\text{tower}^*(\text{qa}(\varphi) + 1, (9k + 9) \varphi + 3.92)$	Lemma 14
#FD(MSO)	#SAT	$\text{tower}(\text{qa}(\varphi) + 1, (9k + 9) \varphi + 3.92)$	$\text{tower}^*(\text{qa}(\varphi) + 1, (9k + 9) \varphi + 3.92)$	Lemma 15
SAT	MC(MSO)	$\lceil \frac{k+1}{c} \rceil$	$O(k \psi)$	Lemma 17
QSAT	MC(MSO)	$\lceil \frac{k+1}{c} \rceil$	$O(k \psi)$	Theorem 6

Theorem 5 can be replaced by “ $\text{qa}_2(\varphi)$ ” on *guarded* formulas, i.e., formulas in which there are only two first-order quantifiers that are only allowed to quantify edges. Here, $\text{qa}_2(\varphi)$ refers to the quantifier alternation of the second-order quantifiers only. Hence, on such guarded formulas (e.g., on all examples in the introduction), the bounds are tight. Another task that remains for further research is to evaluate the encodings in practice. This would also be interesting for the auxiliary encodings, e.g., can a treewidth-aware cardinality constraint compete with classical cardinality constraint?

References

- 1 Michael Alekhovich and Alexander A. Razborov. Satisfiability, Branch-Width and Tseitin Tautologies. *Comput. Complex.*, 20(4):649–678, 2011. doi:10.1007/S00037-011-0033-1.
- 2 Stefan Arnborg and Andrzej Proskurowski. Problems on Graphs with Bounded Decomposability. *Bull. EATCS*, 25:7–10, 1985.
- 3 Albert Atserias and Sergi Oliva. Bounded-width QBF is PSPACE-complete. *Journal of Computer and System Sciences*, 80(7):1415–1429, 2014. doi:10.1016/j.jcss.2014.04.014.
- 4 Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and Complexity Results for #SAT and Bayesian Inference. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 340–351, 2003. doi:10.1109/SFCS.2003.1238208.
- 5 Max Bannach, Malte Skambath, and Till Tantau. On the Parallel Parameterized Complexity of MaxSAT Variants. In *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, pages 19:1–19:19, 2022. doi:10.4230/LIPIcs.SAT.2022.19.
- 6 Umberto Bertelè and Francesco Brioschi. On Non-serial Dynamic Programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973. doi:10.1016/0097-3165(73)90016-2.
- 7 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability, Second Edition*. IOS Press, 2021. doi:10.3233/FAIA336.
- 8 Bernhard Bliem, Reinhard Pichler, and Stefan Woltran. Declarative Dynamic Programming as an Alternative Realization of Courcelle’s Theorem. In *Parameterized and Exact Computation – 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, pages 28–40, 2013. doi:10.1007/978-3-319-03898-8_4.
- 9 J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960.

- 10 Florent Capelli and Stefan Mengel. Tractable QBF by Knowledge Compilation. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, pages 18:1–18:16, 2019. doi:10.4230/LIPICS.STACS.2019.18.
- 11 Hubie Chen. Quantified Constraint Satisfaction and Bounded Treewidth. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 161–165, 2004.
- 12 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.
- 13 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.
- 14 Adnan Darwiche. Decomposable Negation Normal Form. *J. ACM*, 48(4):608–647, 2001. doi:10.1145/502090.502091.
- 15 Johannes Klaus Fichte, Markus Hecher, Michael Morak, Patrick Thier, and Stefan Woltran. Solving Projected Model Counting by Utilizing Treewidth and its Limits. *Artif. Intell.*, 314:103810, 2023. doi:10.1016/j.artint.2022.103810.
- 16 Johannes Klaus Fichte, Markus Hecher, and Andreas Pfandler. Lower Bounds for QBFs of Bounded Treewidth. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 410–424, 2020. doi:10.1145/3373718.3394756.
- 17 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.
- 18 Georg Gottlob, Reinhard Pichler, and Fang Wei. Monadic Datalog Over Finite Structures of Bounded Treewidth. *ACM Trans. Comput. Log.*, 12(1):3:1–3:48, 2010. doi:10.1145/1838552.1838555.
- 19 Rudolf Halin. S-functions For Graphs. *Journal of geometry*, 8(1):171–186, 1976.
- 20 Petr Hliněný, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width Parameters Beyond Treewidth and their Applications. *Comput. J.*, 51(3):326–362, 2008. doi:10.1093/comjnl/bxm052.
- 21 Neil Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.
- 22 Bart M. P. Jansen and Stefan Kratsch. A Structural Approach to Kernels for ILPs: Treewidth and Total Unimodularity. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 779–791, 2015. doi:10.1007/978-3-662-48350-3_65.
- 23 Donald E. Knuth. *The Art of Computer Programming*, volume 4, Fascicle 6. Addison-Wesley, 2016.
- 24 Tuukka Korhonen and Matti Järvisalo. Integrating Tree Decompositions into Decision Heuristics of Propositional Model Counters (Short Paper). In *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, pages 8:1–8:11, 2021. doi:10.4230/LIPICS.CP.2021.8.
- 25 Stephan Kreutzer. Algorithmic meta-theorems. In *Finite and Algorithmic Model Theory*, pages 177–270. Cambridge University Press, 2011.
- 26 Michael Lampis, Stefan Mengel, and Valia Mitsou. QBF as an Alternative to Courcelle’s Theorem. In *Theory and Applications of Satisfiability Testing – SAT 2018 – 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, pages 235–252, 2018. doi:10.1007/978-3-319-94144-8_15.
- 27 Ruiming Li, Dian Zhou, and Donglei Du. Satisfiability and Integer Programming as Complementary Tools. In *Proceedings of the 2004 Conference on Asia South Pacific Design Automation: Electronic Design and Solution Fair 2004, Yokohama, Japan, January 27-30, 2004*, pages 879–882, 2004. doi:10.1109/ASPAC.2004.178.

15:18 Structure-Guided Automated Reasoning

- 28 Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-width. *Journal of Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.
- 29 Sigve Hortemo Sæther, Jan Arne Telle, and Martin Vatshelle. Solving #SAT and MAXSAT by Dynamic Programming. *J. Artif. Intell. Res.*, 54:59–82, 2015. doi:10.1613/jair.4831.
- 30 Marko Samer and Stefan Szeider. Algorithms for Propositional Model Counting. *J. Discrete Algorithms*, 8(1):50–64, 2010. doi:10.1016/j.jda.2009.06.002.
- 31 Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming – CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 827–831, 2005. doi:10.1007/11564751_73.

Listing Spanning Trees of Outerplanar Graphs by Pivot-Exchanges

Nastaran Behrooznia ✉

Department of Computer Science, University of Warwick, Coventry, UK

Torsten Mütze ✉ 

Institut für Mathematik, Universität Kassel, Germany

Department of Theoretical Computer Science and Mathematical Logic,

Charles University, Prague, Czech Republic

Abstract

We prove that the spanning trees of any outerplanar triangulation G can be listed so that any two consecutive spanning trees differ in an exchange of two edges that share an end vertex. For outerplanar graphs G with faces of arbitrary lengths (not necessarily 3) we establish a similar result, with the condition that the two exchanged edges share an end vertex or lie on a common face. These listings of spanning trees are obtained from a simple greedy algorithm that can be implemented efficiently, i.e., in time $\mathcal{O}(n \log n)$ per generated spanning tree, where n is the number of vertices of G . Furthermore, the listings correspond to Hamilton paths on the 0/1-polytope that is obtained as the convex hull of the characteristic vectors of all spanning trees of G .

2012 ACM Subject Classification Mathematics of computing → Trees

Keywords and phrases Spanning tree, generation, edge exchange, Hamilton path, Gray code

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.16

Funding This work was supported by Czech Science Foundation grant GA 22-15272S. Both authors participated in the workshop “Combinatorics, Algorithms and Geometry” in March 2024, which was funded by German Science Foundation grant 522790373.

Acknowledgements We thank both reviewers of this paper for a number of very helpful suggestions that improved the writing.

1 Introduction

For a given graph G , let $\mathcal{T}(G)$ denote the set of all spanning trees of G . Two spanning trees of G differ in an *edge exchange* if the symmetric difference of their edge sets is a 2-element set, i.e., each of the two spanning trees is obtained from the other one by removing one edge and adding another. The *flip graph* $\mathcal{F}(G)$ has the set $\mathcal{T}(G)$ as vertex set, and an edge between any two spanning trees that differ in an edge exchange; see Figure 1. It is well-known that $\mathcal{F}(G)$ is the skeleton of the 0/1-polytope that is obtained as the convex hull of all characteristic vectors $\chi(T)$ of spanning trees $T \in \mathcal{T}(G)$ (see [25, Thm. 40.6]). Specifically, if the edge set of G is $\{1, \dots, m\}$, then for all $e \in \{1, \dots, m\}$ the characteristic vector $\chi(T)$ has a 1-bit at position e if the edge e belongs to T , and a 0-bit at position e otherwise.

We are interested in computing Hamilton paths or cycles in the flip graph $\mathcal{F}(G)$, i.e., we aim to list all spanning trees of G such that any two consecutive spanning trees differ in an edge exchange. Such a listing of combinatorial objects subject to some small-change condition is generally referred to as a *Gray code* [24, 19]. A Gray code is called *cyclic* if the first and last object also differ in the small-change condition, i.e., this corresponds to a Hamilton cycle in the flip graph.

Cummins [8] first proved that $\mathcal{F}(G)$ admits a Hamilton cycle for any graph G . He showed more generally that for any prescribed edge of $\mathcal{F}(G)$, there is a Hamilton cycle containing this edge. Similar results were obtained by Shank [26], Kamae [11], and Kishi and Kajitani [13].



© Nastaran Behrooznia and Torsten Mütze;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

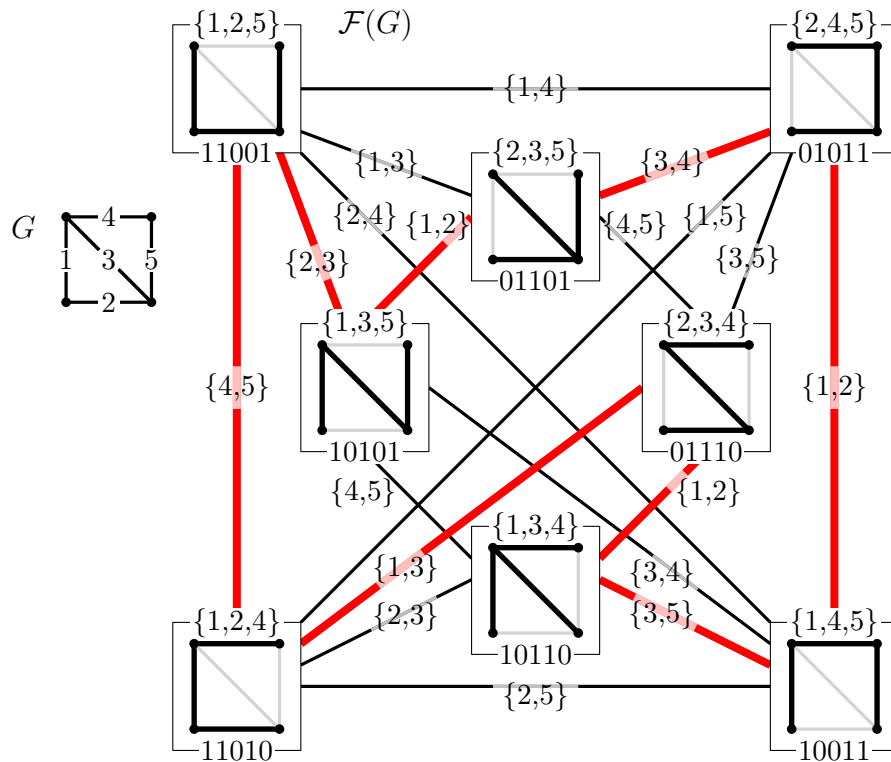
Article No. 16; pp. 16:1–16:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** The spanning tree flip graph $\mathcal{F}(G)$ for the “diamond” graph G on the left, with a Hamilton cycle highlighted. For each spanning tree, the set of edges is shown above, and the characteristic vector is shown below. Edges of $\mathcal{F}(G)$ are labelled by the two edges of G being exchanged.

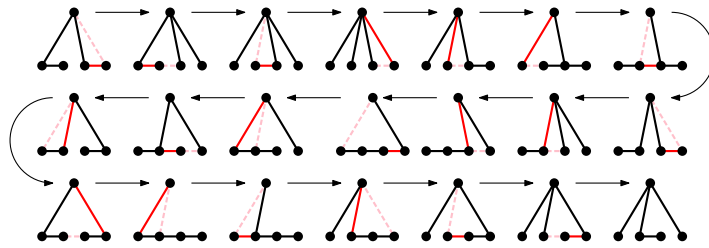
Harary and Holzmam [10] proved more generally that the base exchange flip graph of any matroid has a Hamilton cycle. They showed this in the stronger sense that any edge of this flip graph can be prescribed to be contained in the cycle, and to be avoided by another cycle. Furthermore, Naddef and Pulleyblank [20, 21] proved that the skeleton of any 0/1-polytope either admits a Hamilton path between any two prescribed end vertices, or it is a hypercube, in which case it admits a Hamilton path between any two end vertices of opposite parity.

Algorithmically, a Hamilton path in $\mathcal{F}(G)$ can be computed in time $\mathcal{O}(1)$ on average per generated tree using Smith’s algorithm [30] (this is Algorithm S in [14, Sec. 7.2.1.6]); see Figure 2 (a). Given these strong Hamiltonicity properties of the graph $\mathcal{F}(G)$, there has recently been interest to strengthen them by restricting the allowed edge exchanges.

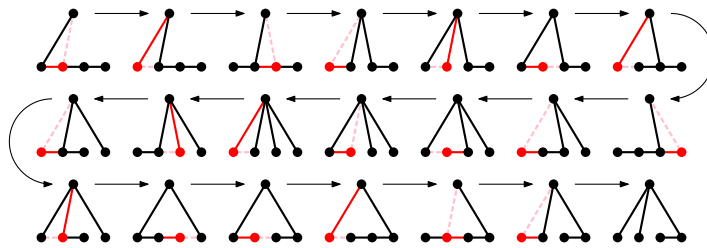
1.1 The pivot-exchange property

We introduce some more notation. For a graph G , we write $V(G)$ and $E(G)$ for set of vertices and edges of G , respectively. For a subgraph $H \subseteq G$ and edges $e \in E(H)$ and $f \in E(G) \setminus E(H)$, we write $H - e$ and $H + f$ for the graphs obtained from H by removing and adding the edges e and f , respectively. We will think of a subgraph H as a subset of edges from $E(G)$, so the operations $H - e$ and $H + f$ remove and add an element from the set, respectively. Formally, an *edge exchange* for a spanning tree $T \in \mathcal{T}(G)$ is a pair $\{e, f\}$ of edges such that $e \in E(T)$ and $f \in E(G) \setminus E(T)$ with $T - e + f = T \Delta \{e, f\} \in \mathcal{T}(G)$.

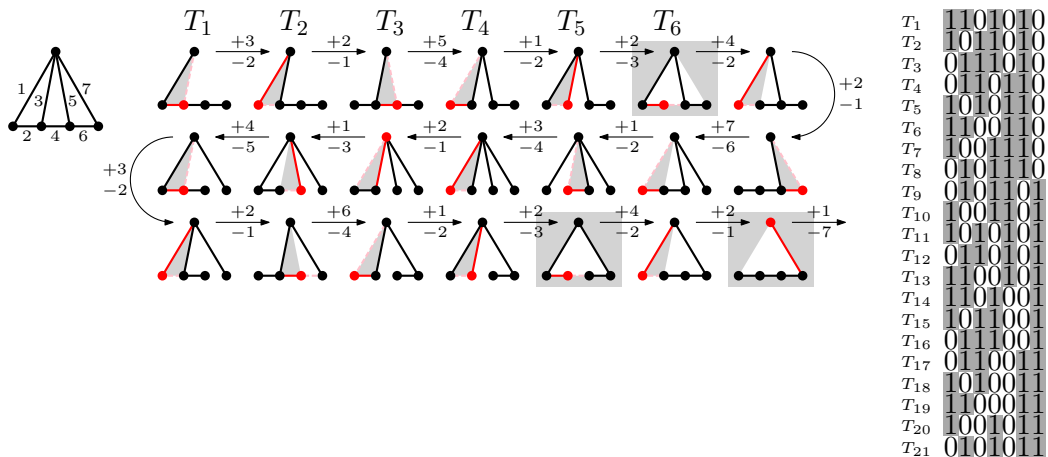
(a) Smith [Smi97]



(b) Cameron, Grubb, Sawada [CGS24]



(c) Merino, Mütze, Williams [MMW22]



■ **Figure 2** Three different edge exchange Gray codes for listing the 21 spanning trees of the fan graph F_5 . In each spanning tree, the edge removed to reach the next tree is highlighted (prefixed by $-$ in (c)), and the non-edge being added is dashed (prefixed by $+$ in (c)). In (b) and (c), the common end vertex of each pivot-exchange operation is highlighted, and in (c), the common face of each face-exchange operation is highlighted. The right-hand side of (c) shows the characteristic vectors of each spanning tree.

Cameron, Grubb, and Sawada [3] introduced the stronger notion of a *pivot-exchange*, which is an exchange $\{e, f\}$ with the additional property that e and f have a common end vertex. For example, in the graph G shown in Figure 1, all exchanges except $\{1, 5\}$ and $\{2, 4\}$ are pivot-exchanges. They raised the following problem.

► **Problem 1.** *Does every graph G admit a pivot-exchange Gray code of its spanning trees?*

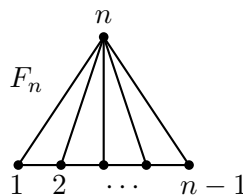
16:4 Listing Spanning Trees of Outerplanar Graphs

Additionally, they asked if such a listing can be computed using a greedy strategy, and possibly by an efficient algorithm. Problem 1 is a special case of a more general question raised by Knuth in Vol. 4A of his seminal series “The Art of Computer Programming” [14], stated as problem 102 in Section 7.2.1.6 with a difficulty rating of 46/50: ¹

► **Problem 2.** *Does every directed graph admit an edge exchange Gray code of its oriented spanning trees, also known as arborescences, i.e., spanning trees in which all arcs are oriented away from a fixed root vertex r ?*

Note that for any exchange $\{e, f\}$ in this directed setting, in order to preserve the arborescence property, the arcs e and f must point to the same vertex, i.e., it must be a pivot-exchange. Furthermore, a positive answer to Problem 2 would imply an affirmative answer to Problem 1: Indeed, given an undirected graph G , construct the directed graph G^{\leftrightarrow} by replacing each undirected edge of G by two oppositely oriented arcs, and pick an arbitrary vertex as root r . Listing the oriented spanning trees of G^{\leftrightarrow} by arc exchanges produces every spanning tree of G exactly once, namely in the orientation forced by the choice of r .

1.2 Fan graphs



■ **Figure 3** The fan graphs F_n .

As a first step towards Problem 1, Cameron, Grubb, and Sawada [3] provided a pivot-exchange Gray code for listing the spanning trees of *fan graphs* F_n , which are obtained by joining an extra vertex to all vertices of a path on $n - 1$ vertices; see Figure 3.

► **Theorem 3** ([3, Thm. 4]). *For any $n \geq 3$, there is a pivot-exchange Gray code for the spanning trees of F_n .*

The output of their algorithm for the case $n = 5$ is shown in Figure 2 (b). The algorithm uses a greedy strategy that prioritizes exchanges based on vertex labels.

1.3 A simple greedy algorithm

Merino, Mütze and Williams [17] discovered a simple greedy algorithm for listing the spanning trees of a graph G by (arbitrary) edge exchanges. The algorithm operates based on a total ordering of the edges of G , which is captured by labeling the edges by integers. Specifically, if m denotes the number of edges of G , then an *edge labeling* is a bijection $\ell : E(G) \rightarrow \{1, \dots, m\}$. For an edge $e \in E(G)$, we refer to $\ell(e)$ as the *label* of the edge e . In the following examples, we will often identify edges by their labels. In particular, edge exchanges $\{e, f\}$ will be denoted by the pairs of labels $\{\ell(e), \ell(f)\}$. We will also use the abbreviation $[m] := \{1, \dots, m\}$.

The output of Algorithm G when applied to the fan graph F_5 is shown in Figure 2 (c), using the edge labeling displayed on the left. To illustrate the greedy rule in step G2, when reaching the sixth spanning tree $T_6 = \{1, 2, 5, 6\}$, there are seven possible edge exchanges to

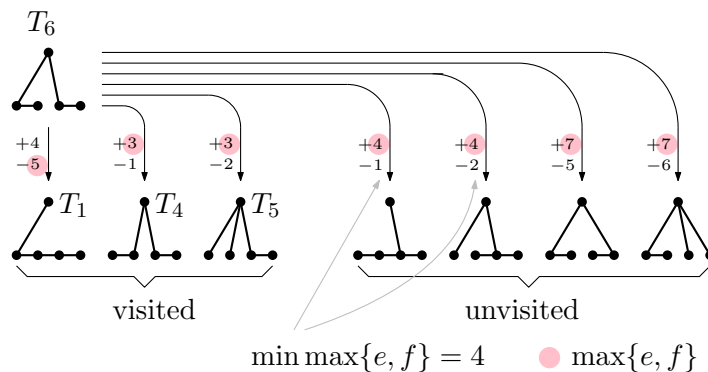
¹ A flawed attempt at settling this problem was published in [5] (see [22]).

■ **Algorithm G** Greedy edge exchanges.

This algorithm greedily generates the spanning trees of a graph G with m edges via edge exchanges, using an edge labeling $\ell : E(G) \rightarrow [m]$ and an initial spanning tree $\tilde{T} \in \mathcal{T}(G)$.

- G1. [Initialize] Visit the initial spanning tree \tilde{T} .
- G2. [Exchange] Perform an edge exchange in the current spanning tree that minimizes the larger of the two edge labels in the exchange and that yields an unvisited spanning tree from $\mathcal{T}(G)$. If no such exchange exists, then terminate. Otherwise visit this new spanning tree and repeat G2.

obtain another spanning tree in $\mathcal{T}(F_5)$, namely $\{1, 3\}, \{2, 3\}, \{1, 4\}, \{2, 4\}, \{4, 5\}, \{5, 7\}$ and $\{6, 7\}$; see Figure 4. Only $\{1, 4\}, \{2, 4\}, \{5, 7\}$ and $\{6, 7\}$ give an unvisited spanning tree, and among those, $\{1, 4\}$ and $\{2, 4\}$, minimize the larger label, which is 4. In this case, the exchange $\{2, 4\}$ is applied, but $\{1, 4\}$ would be a valid alternative for the algorithm.



■ **Figure 4** Illustration of the sixth iteration of Algorithm G in the run shown in Figure 2 (c).

In general, in step G2 there may be several edge exchanges

$$\{e_1, f\}, \{e_2, f\}, \dots, \{e_t, f\} \text{ with } \ell(e_1) < \ell(e_2) < \dots < \ell(e_t) < \ell(f), \tag{1}$$

applicable to the current spanning tree to give an unvisited spanning tree, which all have $\ell(f)$ as the larger label, differing only in the smaller label $\ell(e_i), i \in [t]$. We refer to such a situation as a *tie*. A *tie-breaking rule* is a procedure that determines which exchange to apply in case of a tie.

By definition of the step G2, Algorithm G only selects edge exchanges that result in a previously unvisited spanning tree of G , i.e., the produced listing of spanning trees will not contain repetitions. However, it could be that the algorithm terminates before having visited the entire set $\mathcal{T}(G)$, a situation that is ruled out by the next theorem.

► **Theorem 4** ([17, Thm. 10]). *For any graph G , for any edge labeling of G , for any initial spanning tree $\tilde{T} \in \mathcal{T}(G)$, and for any tie-breaking rule, Algorithm G yields a genlex listing of all spanning trees of G .*

A listing of bitstrings is called *genlex* if all strings with the same suffix appear consecutively. In particular, all strings ending with 0 appear before all strings ending with 1, or vice versa. A genlex listing of bitstrings is also sometimes referred to as suffix-partitioned in the literature. Note that genlex order generalizes colexicographic order. In the context of spanning trees, the *genlex* property refers to the corresponding characteristic vectors $\chi(T)$ of spanning

trees $T \in \mathcal{T}(G)$; see the right-hand side of Figure 2 (c). In other words, all spanning trees not containing the highest-labeled edge appear before all spanning trees containing this edge, or vice versa, and this property is true recursively within the blocks.

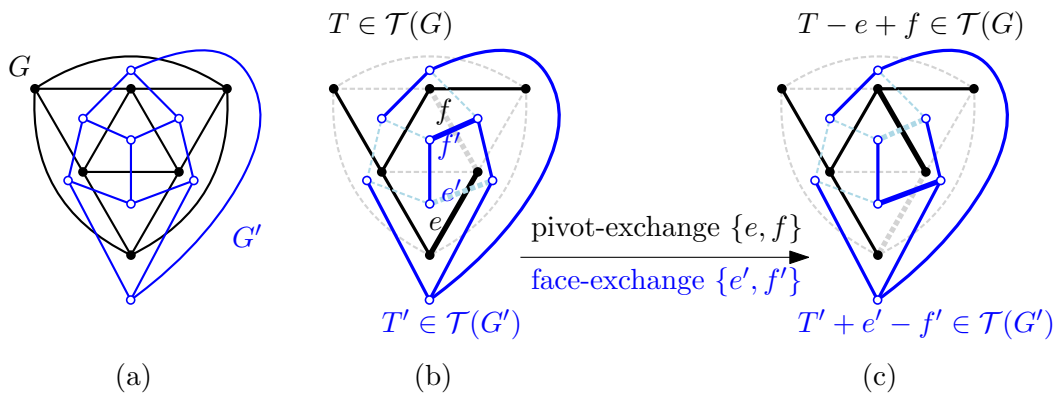
As evidenced by this theorem, the greedy Algorithm G is very powerful and versatile. In fact, the algorithm can be generalized for listing the bases of any matroid by base exchanges, and even more generally, for traversing a Hamilton path on the skeleton of any 0/1-polytope [16].

1.4 The face-exchange property

The paper [17] also introduced another closeness condition for edge exchanges, which is well-defined only for plane graphs. Specifically, a *face-exchange* between two spanning trees of a plane graph is an exchange of two edges that lie on a common face. If an edge exchange is both a pivot-exchange *and* face-exchange, then we refer to it as a *pivot \wedge face-exchange*. A weaker requirement is that it is a pivot-exchange *or* a face-exchange, and then we refer to it as a *pivot \vee face-exchange*. These notions give rise to the following question.

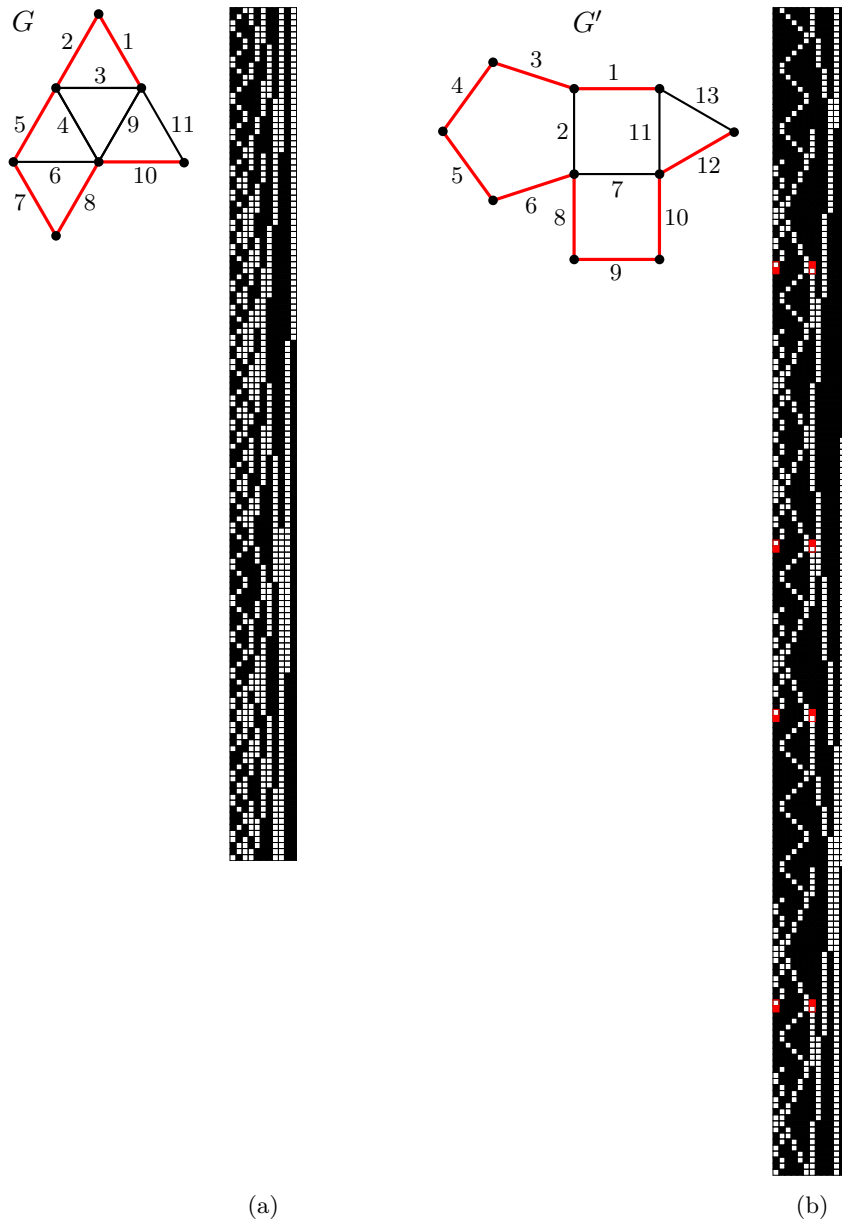
► **Problem 5.** *Does every plane graph G admit a pivot \wedge face-exchange or a pivot \vee face-exchange Gray code of its spanning trees?*

Pivot- and face-exchanges are connected through the well-known concept of the dual graph; see Figure 5 (a). For a plane graph G , we write $F(G)$ for the set of faces of G . The *dual graph* G' is the plane graph obtained from G as follows: For every face $\alpha \in F(G)$, the dual graph G' has a vertex $\alpha' \in V(G')$, and for every edge e of G between faces α and β of G , the dual graph G' has the edge $e' = (\alpha', \beta')$. For every vertex $v \in V(G)$, we write $v' \in F(G')$ for the corresponding face of G' dual to it. For a spanning tree $T \in \mathcal{T}(G)$, the *dual spanning tree* $T' \in \mathcal{T}(G')$ has the dual edge $e' \in E(T')$ for every non-edge $e \in E(G) \setminus E(T)$ and a dual non-edge $e' \notin E(T')$ for every edge $e \in E(T)$; see Figure 5 (b). Observe that a pivot-exchange $\{e, f\}$ in T is a face-exchange $\{e', f'\}$ in the dual spanning tree T' ; see Figure 5 (c). Symmetrically, a face-exchange $\{e, f\}$ in T is a pivot-exchange $\{e', f'\}$ in T' .



■ **Figure 5** Connection between pivot- and face-exchanges via the dual spanning tree.

Merino, Mütze and Williams [17] strengthened Theorem 3, by showing that for a suitable edge labeling of the fan graph F_n and a suitable tie-breaking rule, Algorithm G yields a pivot \wedge face-exchange Gray code. Specifically, the edges of F_n are labeled from *left-to-right*, as shown in Figure 2 (c), and ties are broken according to the *closest* tie-breaking rule, which in case of a tie as in (1) selects the exchange $\{e_t, f\}$, i.e., the exchange that maximizes the smaller of the edge labels $\ell(e_t)$ (equivalently, the one for which the smaller label is closest to the larger label $\ell(f)$).



■ **Figure 6** (a) Pivot-exchange Gray code for the spanning trees of the outerplane triangulation G . (b) Pivot \vee face-exchange Gray code for the spanning trees of the outerplane graph G' , which has the four marked face-exchanges $\{1, 7\}$, and the rest pivot-exchanges. Both listings were computed by Algorithm G, and the spanning trees are represented by their characteristic vectors, with 1-bits and 0-bits drawn as black and white squares, respectively. The initial spanning tree is highlighted in both graphs.

► **Theorem 6.** *For any $n \geq 3$, for the left-to-right labeling of the edges of F_n , for any initial spanning tree $\tilde{T} \in \mathcal{T}(F_n)$, and for the closest tie-breaking rule, Algorithm G yields a genlex pivot \wedge face-exchange Gray code for the spanning trees of F_n .*

In fact, the Gray code from Theorem 3 also has the stronger pivot \wedge face-exchange property.

1.5 Our results

In this work, we solve Problems 1 and 5 for certain families of plane graphs that generalize fan graphs, thus generalizing Theorems 3 and 6.

An *outerplane* graph is a plane graph in which all vertices are incident to the outer face. An outerplane graph is a *triangulation* if all of its faces, except possibly the outer face, are triangles. Note that fan graphs are a very special case of outerplane triangulations.

► **Theorem 7.** *For any outerplane triangulation G , there is an edge labeling of G , so that for any initial spanning tree $\tilde{T} \in \mathcal{T}(G)$, there is a tie-breaking rule for which Algorithm G yields a genlex pivot-exchange Gray code for the spanning trees of G .*

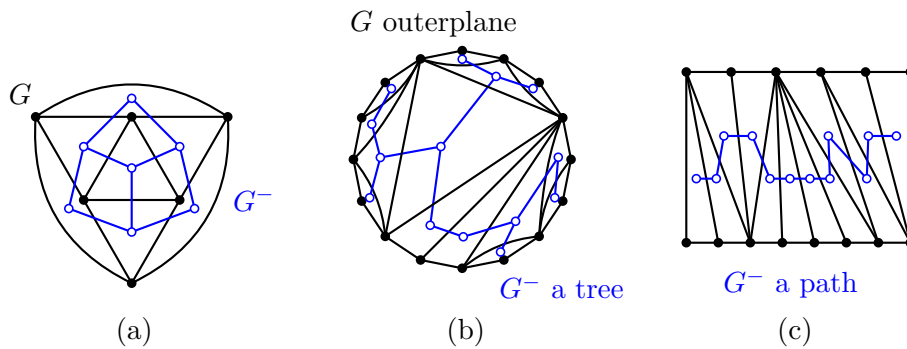
► **Theorem 8.** *For any outerplane graph G , there is an edge labeling of G , so that for any initial spanning tree $\tilde{T} \in \mathcal{T}(G)$, there is a tie-breaking rule for which Algorithm G yields a genlex pivot\face-exchange Gray code for the spanning trees of G .*

These theorems directly yield efficient algorithms. Specifically, using the techniques described in [16, Sec. 7.2+Cor. 32], Algorithm G can be implemented to output each spanning tree in time $\mathcal{O}(n \log n)$, where n is the number of vertices of G . The required space for the algorithm is $\mathcal{O}(n)$. In particular, this implementation is *history-free* in the sense that no previously computed spanning trees apart from the current spanning tree need to be stored, plus some simple data structures for bookkeeping.

Two examples of Gray code listings of spanning trees obtained from these theorems are shown in Figure 6.

► **Corollary 9.** *Any outerplane triangulation admits a genlex pivot-exchange Gray code of its spanning trees. Any outerplane graph admits a genlex pivot\face-exchange Gray code of its spanning trees.*

The *weak dual* graph of a plane graph G , denoted G^- is the graph obtained from the dual graph G' by removing the vertex corresponding to the outer face of G ; see Figure 7 (a).



■ **Figure 7** Illustration of (a) weak dual graph; (b) 2-connected outerplane triangulation and its weak dual tree; (c) 2-connected outerplane triangulation whose weak dual tree is a path.

Note that G is a 2-connected outerplane graph if and only if the weak dual G^- is a tree; see Figure 7 (b). Also note that G^- is a path if and only if G is 2-connected and all but at most two sides of every inner face are incident with the outer face; see Figure 7 (c). In particular, for a triangulation G , we have that G^- is a path if and only if G is 2-connected and at least one side of every triangle touches the outer face. This is true in particular for fan graphs F_n .

We show that fan graphs, and more generally outerplane triangulations for which the weak dual is a path, are exactly the outerplane graphs that have the maximum number of spanning trees for a fixed number of edges. Moreover, the counts are the Fibonacci numbers, defined as $f_0 := 0$, $f_1 := 1$ and

$$f_{m+1} := f_m + f_{m-1} \tag{2}$$

for all $m \geq 1$. We write $t(G) := |\mathcal{T}(G)|$ for the number of spanning trees of G .

► **Theorem 10.** *For any outerplane graph G with m edges we have $t(G) \leq f_{m+1}$, with equality if and only if G is a triangulation such that G^- is a path.*

The identity $t(G) = f_{m+1}$ when G is a triangulation for which G^- is a path was already noticed by Slater [29, Prop. 1].

► **Remark 11.** We remark that Problems 1, 2 and 5 are perfectly valid also for multigraphs, i.e., graphs that may have parallel edges and/or loops (though loops are irrelevant in the context of spanning trees). In fact, Theorems 4, 7 and 8, and hence Corollary 9, also hold in this more general setting. An “outerplane triangulation” in this case has all inner faces of lengths at most 3, instead of exactly 3. The time and space bounds stated after Theorem 8 change to $\mathcal{O}(m \log n)$ and $\mathcal{O}(m)$, respectively, where m is the number of edges of the multigraph. However, for simplicity we do our proofs in the setting of simple graphs, where no parallel edges nor loops are allowed. Nonetheless, our proof of Theorem 10 will actually use multigraphs.

► **Remark 12.** The listings of spanning trees produced by Algorithm G are in general not Hamilton cycles in $\mathcal{F}(G)$, but only Hamilton paths, i.e., the first and last spanning tree in general do not differ in an edge exchange. This remark applies to all the Gray codes mentioned in Theorems 4, 6, 7, and 8, and in Corollary 9. For some graphs, some edge labelings, and some initial spanning trees, however, the Gray codes are cyclic, such as the one mentioned in Theorem 6 for the L-shaped initial spanning tree shown in Figure 2 (c), which ends with the mirrored L.

1.6 Related work

There has been an extensive amount of work [23, 9, 15, 12, 27, 28] on efficiently generating the set $\mathcal{T}(G)$ of all spanning trees of G , *without* the requirement that any two consecutive trees differ in an edge exchange, i.e., the computed listings are *not* Hamilton paths or cycles in the flip graph $\mathcal{F}(G)$. The algorithms in the last three of the aforementioned papers achieve this in time $\mathcal{O}(1)$ on average per generated tree. See the survey [4] for a comparison of the different algorithms.

If instead of listing all spanning trees of G , we want to count them, then this can be achieved by Kirchhoff’s Matrix-Tree Theorem, which reduces the problem to computing a determinant, which can be done efficiently. This problem is closely connected to finding the so-called *most vital edge* of G , which is the edge contained in the most spanning trees from $\mathcal{T}(G)$. Random sampling [1, 2] and ranking/unranking [6, 7] of spanning trees have also been considered.

1.7 Outline of this paper

In Section 2 we prove Theorems 7 and 8. In Section 3 we prove Theorem 10. We conclude with some open questions in Section 4, and there we also report on some experimental evidence.

2 Proof of Theorems 7 and 8

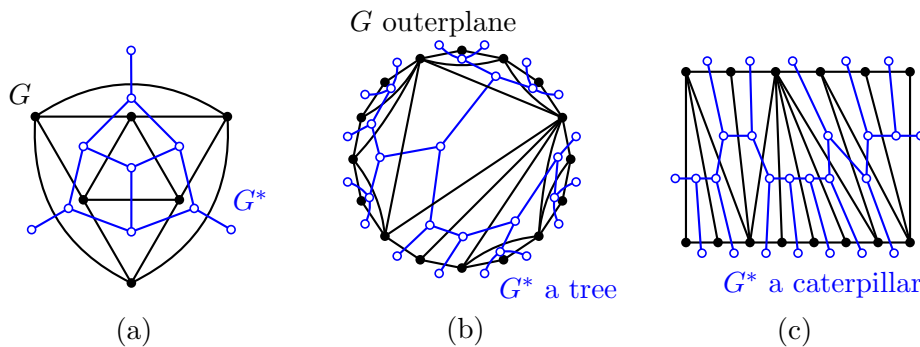
For the proofs of Theorems 7 and 8, we will assume w.l.o.g. that the outerplane graph G is 2-connected. If G is not 2-connected, then all the arguments presented in the following apply to each of its blocks, and the spanning trees of G are obtained by combining the spanning trees in each block in all possible ways, and pivot- or face-exchanges remain valid within the blocks.

We first describe the labeling of edges of a 2-connected outerplane graph G that we use in order to run Algorithm G on the graph G . We then establish two important properties of these labelings (Lemmas 13 and 14) that are crucial to show that whenever an arbitrary edge exchange becomes applicable in step G2 of Algorithm G, then ties can be broken to instead use a pivot- or face-exchange (Lemma 15).

2.1 The edge labeling

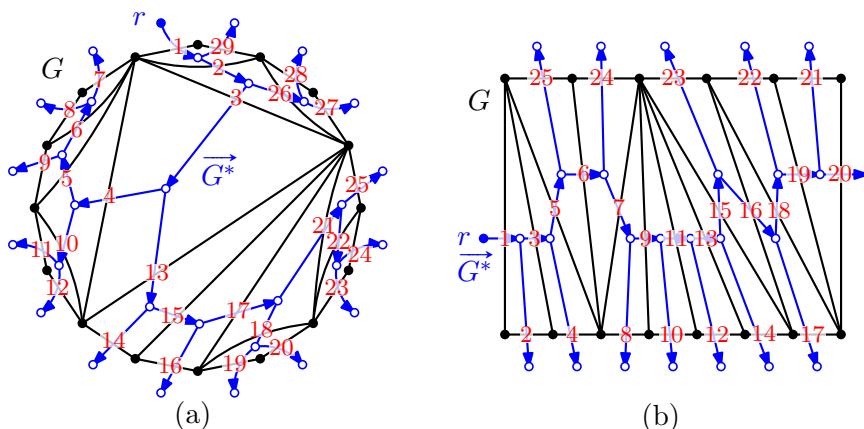
To define the edge labeling, we need another definition. Let v be the vertex of G' corresponding to the outer face of G , and let d be the degree of v . The *split dual* graph, denoted G^* , is obtained from G' by splitting v into d many degree-1 vertices, one adjacent to each neighbor of v ; see Figure 8 (a).

Note that the weak dual graph G^- is a tree if and only if the split dual graph G^* is a tree; see Figures 7 (b) and 8 (b). Furthermore, G^- is a path if and only if G^* is a caterpillar, i.e., a tree in which all vertices are in distance ≤ 1 from a central path; see Figures 7 (c) and 8 (c).



■ **Figure 8** Illustration of (a) split dual graph; (b) 2-connected outerplane triangulation and its split dual tree; (c) 2-connected outerplane triangulation whose split dual tree is a caterpillar (cf. Figure 7).

Let G be a 2-connected outerplane graph with m edges. We consider the split dual tree G^* , we pick a leaf r of this tree as root, and we orient all of its edges away from r , giving an oriented tree \vec{G}^* , referred to as *oriented split dual*. We label the edges of \vec{G}^* in a depth-first-search manner with integers $1, \dots, m$, starting at the root r and processing subtrees in counterclockwise (ccw) order; see Figure 9. The labeling of the edges of \vec{G}^* induces a labeling of the corresponding dual edges of G with integers from $[m]$. We refer to this labeling of the edges of G as *dual-tree labeling*. Note that there is freedom in the choice of the root in the tree G^* , and different choices yield different oriented split dual trees \vec{G}^* and hence different edge labelings of the same graph G . Note that the left-to-right labeling of the fan graph $G = F_n$ shown in Figure 2 (c) is one particular dual-tree labeling.

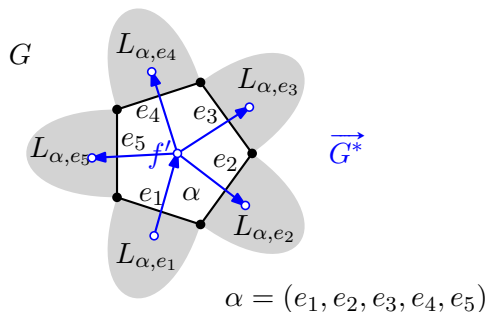


■ **Figure 9** Illustration of the dual-tree labeling procedure.

2.2 Proofs of theorems

Throughout this section, we let G be a 2-connected outerplane graph with m edges, \vec{G}^* an oriented split dual tree, and $\ell : E(G) \rightarrow [m]$ the corresponding dual-tree labeling of the edges of G . Before proving Theorems 7 and 8, we first derive two properties of the edge labelings defined in the previous section, stated in Lemmas 13 and 14 below.

The following definitions are illustrated in Figure 10. We denote a face α of G of length t by the sequence of edges (e_1, \dots, e_t) bounding this face in ccw order, starting with the edge e_1 whose dual edge in \vec{G}^* is oriented towards α' (all other edges of \vec{G}^* are oriented away from α'). For a face $\alpha = (e_1, \dots, e_t)$ and an index $i \in [t]$, the (α, e_i) -lobe L_{α, e_i} is the set of edges of G that are dual to the edges in the maximal subtree of G^* that contains the edge e'_i , but none of the other edges e'_j , $j \in [t] \setminus \{i\}$.



■ **Figure 10** Illustration of lobes.

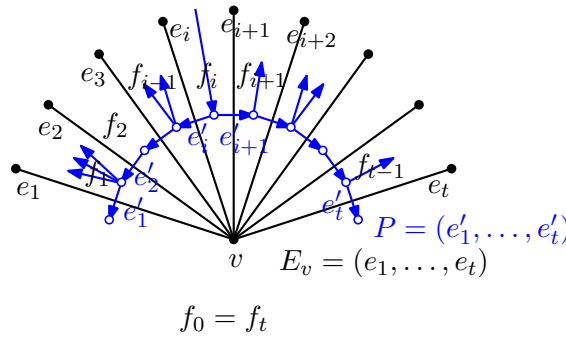
► **Lemma 13.** *For any face $\alpha = (e_1, \dots, e_t)$ of G , we have $\ell(e_1) < \ell(e_2) < \dots < \ell(e_t)$. Furthermore, for any $i \in \{2, \dots, t\}$ and $f \in L_{\alpha, e_i} \setminus \{e_i\}$ we have $\ell(e_i) < \ell(f)$, and $\ell(f) < \ell(e_{i+1})$ if $i < t$.*

Proof. This is an immediate consequence of the labeling procedure which processes subtrees in the oriented split dual in ccw order and in a depth-first-search manner. Specifically, for all $i \in \{2, \dots, t\}$, all edges of $L_{\alpha, e_i} \setminus \{e_i\}$ are labeled directly after e_i , and directly before e_{i+1} if $i < t$. ◀

16:12 Listing Spanning Trees of Outerplanar Graphs

For a vertex v of G , the *incidence list* E_v is the sequence of edges incident to v in clockwise order around v , starting and ending with the two edges that bound the outer face; see Figure 11. We say that an edge e incident with v is a *cw-edge* or *ccw-edge*, respectively, according to the clockwise or counterclockwise orientation of the dual edge e' in $\overrightarrow{G^*}$ around v .

► **Lemma 14.** *For any vertex v , let $E_v =: (e_1, \dots, e_t)$ be its incidence list. Then there is an index $i \in \{0, \dots, t\}$ such that e_1, \dots, e_i are ccw-edges, e_{i+1}, \dots, e_t are cw-edges, and we have $\ell(e_i) < \ell(e_{i-1}) < \dots < \ell(e_1) < \ell(e_{i+1}) < \ell(e_{i+2}) < \dots < \ell(e_t)$. In particular, for any cw-edge e_j and any $i \in [j - 1]$ we have $\ell(e_i) < \ell(e_j)$.*



■ **Figure 11** Illustration of Lemma 14.

Proof. For any sequence of edges $P = (a_1, \dots, a_t)$ that form a path in the oriented tree $\overrightarrow{G^*}$, there is an index $i \in \{0, \dots, t\}$ such that a_1, \dots, a_i are oriented towards the start vertex of P and a_{i+1}, \dots, a_t are oriented towards the end vertex of P . The special cases $i = 0$ and $i = t$ mean that the path is oriented conformly, towards the end or start vertex, respectively.

The sequence of dual edges $P := (e'_1, \dots, e'_t)$ is a path in $\overrightarrow{G^*}$ to which the aforementioned observation applies. For $j \in \{1, \dots, t - 1\}$, let f_j be the face of G bounded by e_j and e_{j+1} . Furthermore, we denote the outer face of G by $f_0 = f_t$. Note that e'_j is oriented towards f'_{j-1} in $\overrightarrow{G^*}$ for $j = 1, \dots, i$, i.e., these are all ccw-edges w.r.t. v . Furthermore, e'_j is oriented towards f'_j in $\overrightarrow{G^*}$ for $j = i + 1, \dots, t$, i.e., these are all cw-edges w.r.t. v .

Applying Lemma 13 to the faces f_j , $j = 1, \dots, i - 1$ and $j = i + 1, \dots, t - 1$, yields $\ell(e_i) < \ell(e_{i-1}) < \dots < \ell(e_1)$ and $\ell(e_{i+1}) < \ell(e_{i+2}) < \dots < \ell(e_t)$, respectively. Furthermore, applying Lemma 13 to L_{f_i, e_i} and $L_{f_i, e_{i+1}}$ proves that $\ell(e_1) < \ell(e_{i+1})$. Combining these inequalities proves the lemma. ◀

► **Lemma 15.** *Let $T \in \mathcal{T}(G)$ and let $\{e, f\}$ with $\ell(e) < \ell(f)$ be an edge exchange for T . Then there is a pivot\face-exchange $\{d, f\}$ with $\ell(d) < \ell(f)$. Furthermore, if G is a triangulation, then $\{d, f\}$ is a pivot-exchange.*

Proof. We let $\alpha = (e_1, \dots, e_t)$ be the face incident with f in G for which the dual edge f' in $\overrightarrow{G^*}$ is oriented away from α' . As $\ell(f) > \ell(e) \geq 1$ we have $\ell(f) > 1$, and consequently α is not the outer face, but an inner face. We have $f = e_i$ for some $i \in \{2, \dots, t\}$. The graph $T \cup \{e, f\}$ contains exactly one cycle C , which contains both edges e and f . As $\ell(e) < \ell(f)$, Lemma 13 implies that $e \notin L_{\alpha, f}$, and as $e \in C$, we obtain that C contains edges from every lobe L_{α, e_i} for all $i \in [t]$, and furthermore $e \in L_{\alpha, e_j}$ for some $j \in \{1, \dots, i - 1\}$.

We now distinguish the cases whether $f \in T$ or $f \notin T$, i.e., whether the exchange $\{e, f\}$ adds the edge e and removes f , or removes e and adds f , respectively. These two cases are illustrated in Figure 12 (a) and (b), respectively.

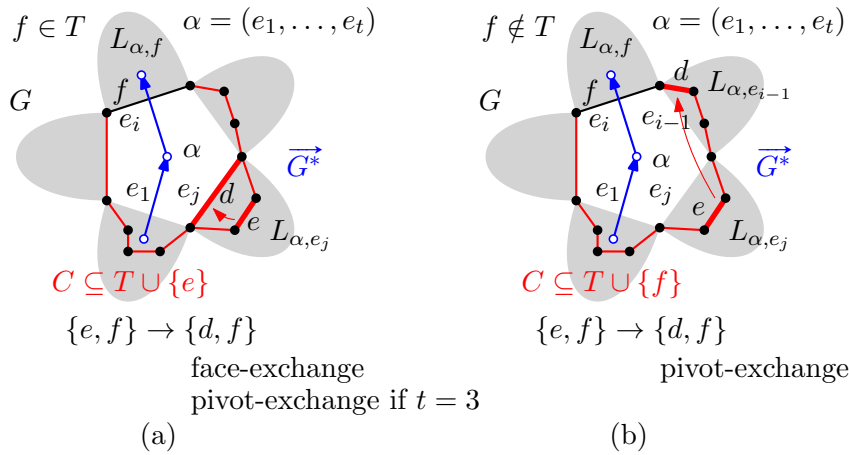


Figure 12 Illustration of the two cases in the proof of Lemma 15.

Case (a). $f \in T$. Note that $\{d, f\}$ with $d := e_j$ is a valid face-exchange for T , and we have $\ell(d) < \ell(f)$ by Lemma 13. Furthermore, if G is a triangulation, then we have $t = 3$ and consequently $f = e_i$ and $d = e_j$ share an end vertex, so the exchange $\{d, f\}$ is a pivot-exchange.

Case (b). $f \notin T$. Let d be the edge of C incident with f in the lobe $L_{\alpha, e_{i-1}}$. Note that $\{d, f\}$ is a valid pivot-exchange for T . If $i > 2$, then we have $\ell(d) < \ell(f)$ by Lemma 13. If $i = 2$, then let v be the common end vertex of $e_1 = e_j$ and $e_2 = e_i$, and consider the incidence list E_v of v . The edge e_1 is a cw-edge and d comes before it (or is equal to e_1) on the list E_v , and so Lemma 14 yields $\ell(d) \leq \ell(e_1)$. By Lemma 13 we have $\ell(e_1) < \ell(f)$ and therefore $\ell(d) < \ell(f)$. ◀

We are now in position to prove Theorems 7 and 8.

Proof of Theorems 7 and 8. By Lemma 15, whenever Algorithm G considers an edge exchange $\{e, f\}$ with $\ell(e) < \ell(f)$, there is a pivot-face-exchange $\{d, f\}$ with $\ell(d) < \ell(f)$. By Theorem 4, the listing of spanning trees produced by Algorithm G has the genlex property, which implies that if $T \triangle \{e, f\}$ is unvisited, then $T \triangle \{d, f\}$ is also unvisited. It follows that there is a tie-breaking rule for Algorithm G to only ever use pivot-face-exchanges.

Furthermore, if G is a triangulation, then by Lemma 15 the alternative exchange $\{d, f\}$ is a pivot-exchange, so there is a tie-breaking rule for Algorithm G to only ever use pivot-exchanges. ◀

3 Proof of Theorem 10

We prove Theorem 10 in the more general setting of multigraphs (recall Remark 11), i.e., from now on we allow multiple edges between pairs of vertices. Two edges between the same pair of vertices are sometimes referred to as *parallel* edges. Loops may be present, but are never contained in a spanning tree and hence irrelevant for us. However, the distinction of parallel edges is important. For example, the plane graph formed by two vertices connected by m parallel edges has m different spanning trees, each containing exactly one of the m edges.

16:14 Listing Spanning Trees of Outerplanar Graphs

All notions introduced in Section 1 are valid in the more general context of multigraphs. The only exception is the definition of an *outerplane triangulation*, which we change from “face length 3” to “face length ≤ 3 ”. A length-2 face comes from two parallel edges, and is called a *digon*.

In addition to edge removal and edge addition, we now also introduce the operation of *edge contraction*. Given a graph G and an edge $e \in E(G)$, we write G/e for the graph obtained from G by contracting the edge e . Note that even if G is a simple graph, G/e may still contain parallel edges, and if G has parallel edges, then G/e may contain loops (which can be ignored when counting spanning trees). The number of spanning trees $t(G)$ of a graph G obeys the well-known recursive relation

$$t(G) = t(G - e) + t(G/e), \quad (3)$$

valid for any (non-loop) edge $e \in E(G)$ (see [18]).

We need the following auxiliary lemma about Fibonacci numbers.

► **Lemma 16.** *For any two integers $i, j \geq 1$, we have $f_i \cdot f_j \leq f_{i+j-1}$, with equality if and only if $i = 1$ or $j = 1$.*

Proof. If $i = 1$ or $j = 1$, then one can check directly that the claimed equality holds.

Now assume that $i, j \geq 2$. We prove that $f_i \cdot f_j < f_{i+j-1}$ by induction on $i + j$. In the base case $i + j = 4$ we have $i = j = 2$ and therefore $f_i = f_j = f_2 = 1$ and $f_{i+j-1} = f_3 = 2$, so the claim is true. For the induction step we distinguish the cases $j \in \{2, 3\}$ and $j > 3$. If $j = 2$ we have $f_i \cdot f_j = f_i < f_{i+1} = f_{i+j-1}$ (recall that $i \geq 2$). If $j = 3$ we have $f_i \cdot f_j = 2f_i < f_i + f_{i+1} = f_{i+2} = f_{i+j-1}$. It remains to consider the case $j > 3$. We have

$$f_i \cdot f_j = f_i \cdot (f_{j-2} + f_{j-1}) = f_i \cdot f_{j-2} + f_i \cdot f_{j-1} < f_{i+j-3} + f_{i+j-2} = f_{i+j-1},$$

where the strict inequality in the third step holds by induction, using that $j - 2 > 1$. ◀

Theorem 10 is an immediate consequence of the following more general statement for multigraphs.

► **Theorem 17.** *For any outerplane (multi)graph G with m edges we have $t(G) \leq f_{m+1}$, with equality if and only if G is a triangulation such that G^- is a path and all digons are incident with the outer face.*

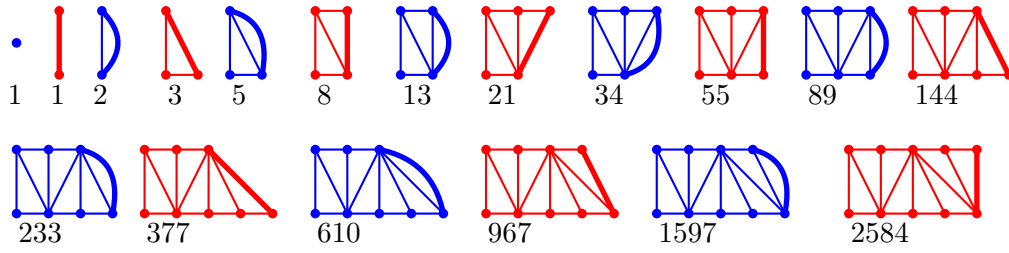
Note that digons incident with the outer face correspond to degree 1 vertices in the dual graph G^- , so if G^- is a path, then there are at most two such digons corresponding to end vertices of the path.

Theorem 17 is illustrated in Figure 13.

Proof. We argue by induction on m . The induction basis $m = 0$ is trivial. For the induction step we assume that $m \geq 1$.

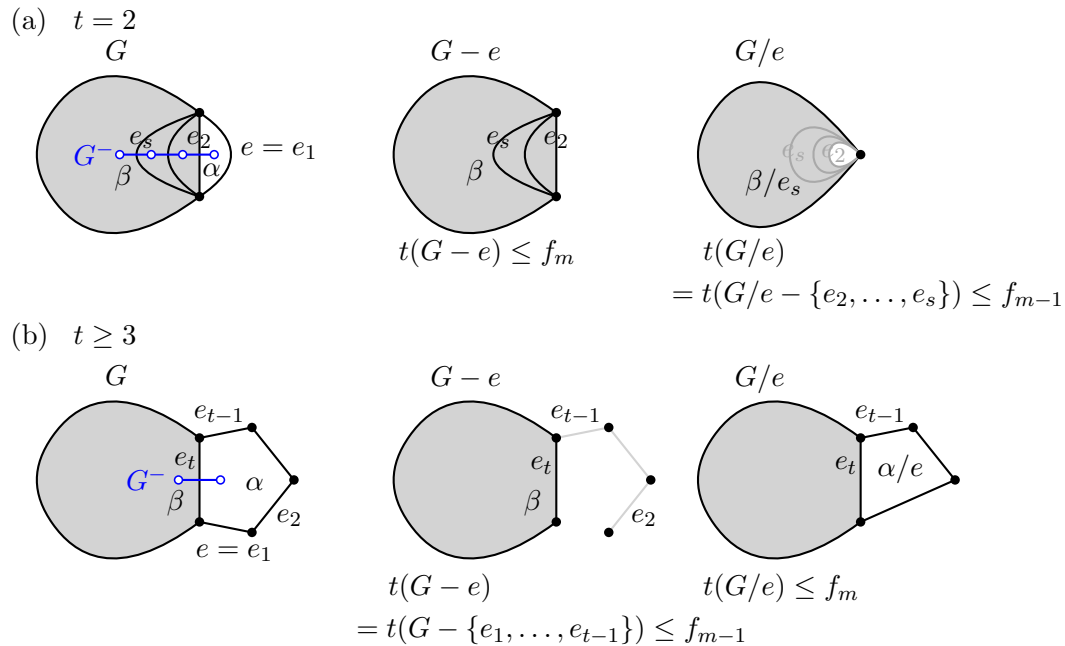
If G is not 2-connected, then it has at least two blocks A and B with $a \geq 1$ and $b \geq 1$ edges, respectively, where $a + b = m$. We have $t(G) = t(A) \cdot t(B)$ and therefore $t(G) \leq f_{a+1} \cdot f_{b+1} < f_{a+b+1} = f_{m+1}$, where the strict inequality follows from Lemma 16.

It remains to consider the case that G is 2-connected. If G is a single edge, then $t(G) = 1 = f_2$. If G has only one inner face, then it is a cycle and we have $t(G) = m \leq f_{m+1}$, and this inequality is tight for $m \in \{2, 3\}$ and strict for $m \geq 4$. The same estimates hold if all inner faces of G are digons, i.e., G has 2 vertices and all m edges are parallel to each other. For the rest of the proof we assume that G has at least two faces, not all of which



■ **Figure 13** Sequence of outerplane graphs that maximize the number of spanning trees, with the corresponding counts, the Fibonacci numbers. They are all triangulations for which G^- is a path, and they either have no digons (red) or one digon incident with the outer face (blue). For each graph, deleting or contracting the bold edge yields the two preceding graphs.

are digons. This implies in particular that $m \geq 4$. We consider a face α of G for which the dual vertex α' has degree at most 1 in G^- . We denote the sequence of edges bounding α by e_1, \dots, e_t in clockwise order, ending with the edge e_t that has a dual edge in G^- . We distinguish two cases, namely whether α is a digon ($t = 2$) or not ($t \geq 3$), and we bound the number of spanning trees using (3) with respect to the edge $e := e_1$, i.e., the first edge of the face α . These two cases are illustrated in Figure 14 (a) and (b), respectively.



■ **Figure 14** Illustration of the proof of Theorem 17.

Case (a). $t = 2$. $G - e$ has $m - 1$ edges, and therefore

$$t(G - e) \leq f_m \tag{4}$$

by induction. Let $s \geq 2$ be the number of edges that are parallel to e_1 (including e_1 and e_2), and denote them by e_1, e_2, \dots, e_s in ccw order around the corresponding common end vertex; see Figure 14 (a). Furthermore, we let β be the face incident with e_s but not e_{s-1} . In the graph G/e , the edges e_2, \dots, e_s are loops, and we can therefore remove all of them, so we have

16:16 Listing Spanning Trees of Outerplanar Graphs

$$t(G/e) = t(G/e - \{e_2, \dots, e_s\}) \leq f_{m-1} \quad (5)$$

by induction. Combining (2), (3), (4), and (5) proves that $t(G) \leq f_{m+1}$, as claimed. Furthermore, this inequality is tight if and only if (4) and (5) are tight, which happens if and only if the conditions stated in the theorem hold for $G - e$ and G/e , which happens if and only if $s = 2$ and β is a triangle that is incident with the outer face, if and only if the conditions stated in the theorem hold for G .

Case (b). $t \geq 3$. Let β be the face incident with e_t but not e_1 . G/e has $m - 1$ edges, and therefore

$$t(G/e) \leq f_m \quad (6)$$

by induction. In the graph $G - e$, the edges e_2, \dots, e_{t-1} are present in every spanning tree, so we have

$$t(G - e) = t(G - \{e_1, \dots, e_{t-1}\}) \leq f_{m-1} \quad (7)$$

by induction. Combining (2), (3), (6) and (7) proves that $t(G) \leq f_{m+1}$, as claimed. Furthermore, this inequality is tight if and only if (6) and (7) are tight, which happens if and only if the conditions stated in the theorem hold for G/e and $G - e$, which happens if and only if $t = 3$ and β is a triangle that is incident with the outer face, if and only if the conditions stated in the theorem hold for G . ◀

4 Open problems

Problems 1 and 5 remain open for general graphs, i.e., for graphs not covered by Theorems 7 and 8 (recall Remark 11). Also, Knuth's more general Problem 2 for directed graphs remains very much open.

Concerning Problem 1, we checked experimentally that all 2-connected graphs on up to 6 vertices admit a pivot-exchange Gray code for their spanning trees, and these Gray codes are all cyclic. Regarding Problem 2, we checked that all directed graphs on up to 5 vertices (oppositely oriented arcs are allowed) whose underlying undirected graph is 2-connected admit an edge exchange Gray code for their arborescences, for any choice of fixed root. In some cases those flip graphs admit no Hamilton cycle, but only a Hamilton path (see [22]), so not all of these Gray codes are cyclic. Regarding Problem 5, we checked that all outerplane graphs on up to 6 vertices admit a cyclic pivot ^ face-exchange Gray code for their spanning trees.

Furthermore, Knuth [14] asked in Exercise 101 in Section 7.2.1.6 whether the complete graph K_n admits a “nice” Gray code listing of its spanning trees. One interpretation of this question, in the spirit of Problem 1, could be: Is there a pivot-exchange Gray code for the spanning trees of K_n ? Alternatively, is there a Gray code listing that provides a more fine-grained explanation of Cayley's formula n^{n-2} for the number of spanning trees? Similar questions can be asked for the complete bipartite graph $K_{n,n}$.

Does Theorem 10 hold without the outerplane requirement for m sufficiently large? The only complete graphs K_n that violate the bound $t(K_n) \leq f_{\binom{n}{2}+1}$ arise for $n = 4, 5, 6$.

References

- 1 D. J. Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.*, 3(4):450–465, 1990. doi:10.1137/0403039.
- 2 A. Z. Broder. Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 442–447. IEEE Computer Society, 1989. doi:10.1109/SFCS.1989.63516.
- 3 B. Cameron, A. Grubb, and J. Sawada. Pivot Gray codes for the spanning trees of a graph ft. the fan. *Graphs Combin.*, 40(4):Paper No. 78, 26 pp., 2024. doi:10.1007/s00373-024-02808-2.
- 4 M. Chakraborty, S. Chowdhury, J. Chakraborty, R. Mehera, and R. K. Pal. Algorithms for generating all possible spanning trees of a simple undirected connected graph: an extensive review. *Complex & Intelligent Systems*, 5:265–281, 2019.
- 5 W.-K. Chen. Hamilton circuits in directed-tree graphs. *IEEE Trans. Circuit Theory*, CT-14:231–233, 1967.
- 6 C. J. Colbourn, R. P. J. Day, and L. D. Nel. Unranking and ranking spanning trees of a graph. *J. Algorithms*, 10(2):271–286, 1989. doi:10.1016/0196-6774(89)90016-3.
- 7 C. J. Colbourn, W. J. Myrvold, and E. Neufeld. Two algorithms for unranking arborescences. *J. Algorithms*, 20(2):268–281, 1996. doi:10.1006/jagm.1996.0014.
- 8 R. L. Cummins. Hamilton circuits in tree graphs. *IEEE Trans. Circuit Theory*, CT-13:82–90, 1966.
- 9 H. N. Gabow and E. W. Myers. Finding all spanning trees of directed and undirected graphs. *SIAM J. Comput.*, 7(3):280–287, 1978. doi:10.1137/0207024.
- 10 C. A. Holzmann and F. Harary. On the tree graph of a matroid. *SIAM J. Appl. Math.*, 22:187–193, 1972. doi:10.1137/0122021.
- 11 T. Kamae. The existence of a Hamilton circuit in a tree graph. *IEEE Trans. Circuit Theory*, CT-14:279–283, 1967.
- 12 S. Kapoor and H. Ramesh. Algorithms for generating all spanning trees of undirected, directed and weighted graphs. In *Algorithms and data structures (Ottawa, ON, 1991)*, volume 519 of *Lecture Notes in Comput. Sci.*, pages 461–472. Springer, Berlin, 1991. doi:10.1007/BFb0028284.
- 13 G. Kishi and Y. Kajitani. On Hamilton circuits in tree graphs. *IEEE Trans. Circuit Theory*, CT-15(1):42–50, 1968.
- 14 D. E. Knuth. *The Art of Computer Programming. Vol. 4A. Combinatorial Algorithms. Part 1*. Addison-Wesley, Upper Saddle River, NJ, 2011.
- 15 T. Matsui. A flexible algorithm for generating all the spanning trees in undirected graphs. *Algorithmica*, 18(4):530–543, 1997. doi:10.1007/PL00009171.
- 16 A. Merino and T. Mütze. Traversing combinatorial 0/1-polytopes via optimization. *SIAM J. Comput.*, 53(5):1257–1292, 2024. doi:10.1137/23M1612019.
- 17 A. Merino, T. Mütze, and A. Williams. All your bases are belong to us: listing all bases of a matroid by greedy exchanges. In *11th International Conference on Fun with Algorithms*, volume 226 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Paper No. 22, 28. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/lipics.fun.2022.22.
- 18 G. Minty. A simple algorithm for listing all the trees of a graph. *IEEE Trans. Circuit Theory*, 12(1):120, 1965.
- 19 T. Mütze. Combinatorial Gray codes—an updated survey. *Electron. J. Combin.*, DS26(Dynamic Surveys):99 pp., 2023. doi:10.37236/11023.
- 20 D. J. Naddef and W. R. Pulleyblank. Hamiltonicity and combinatorial polyhedra. *J. Combin. Theory Ser. B*, 31(3):297–312, 1981. doi:10.1016/0095-8956(81)90032-0.
- 21 D. J. Naddef and W. R. Pulleyblank. Hamiltonicity in (0-1)-polyhedra. *J. Combin. Theory Ser. B*, 37(1):41–52, 1984. doi:10.1016/0095-8956(84)90043-1.
- 22 V. Rao and N. Raju. On tree graphs of directed graphs. *IEEE Trans. Circuit Theory*, 19(3):282–283, 1972.


16:18 Listing Spanning Trees of Outerplanar Graphs

- 23 R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975. doi:10.1002/net.1975.5.3.237.
- 24 C. D. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.
- 25 A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Matroids, trees, stable sets, Chapters 39–69.
- 26 H. Shank. A note on Hamilton circuits in tree graphs. *IEEE Trans. Circuit Theory*, 15(1):86, 1968. doi:10.1109/TCT.1968.1082765.
- 27 A. Shioura and A. Tamura. Efficiently scanning all spanning trees of an undirected graph. *J. Oper. Res. Soc. Japan*, 38(3):331–344, 1995. doi:10.15807/jorsj.38.331.
- 28 A. Shioura, A. Tamura, and T. Uno. An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM J. Comput.*, 26(3):678–692, 1997. doi:10.1137/S0097539794270881.
- 29 P. J. Slater. Fibonacci numbers in the count of spanning trees. *Fibonacci Quart.*, 15(1):11–14, 1977.
- 30 M. J. Smith. Generating spanning trees. Master’s thesis, University of Victoria, 1997.

Tight Approximation and Kernelization Bounds for Vertex-Disjoint Shortest Paths

Matthias Bentert ✉

University of Bergen, Norway

Fedor V. Fomin ✉ 

University of Bergen, Norway

Petr A. Golovach ✉ 

University of Bergen, Norway

Abstract

We examine the possibility of approximating MAXIMUM VERTEX-DISJOINT SHORTEST PATHS. In this problem, the input is an edge-weighted (directed or undirected) n -vertex graph G along with k terminal pairs $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$. The task is to connect as many terminal pairs as possible by pairwise vertex-disjoint paths such that each path is a shortest path between the respective terminals. Our work is anchored in the recent breakthrough by Locht [SODA '21], which demonstrates the polynomial-time solvability of the problem for a fixed value of k .

Lochet's result implies the existence of a polynomial-time ck -approximation for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS, where $c \leq 1$ is a constant. (One can guess $1/c$ terminal pairs to connect in $k^{O(1/c)}$ time and then utilize Locht's algorithm to compute the solution in $n^{f(1/c)}$ time.) Our first result suggests that this approximation algorithm is, in a sense, the best we can hope for. More precisely, assuming the gap-ETH, we exclude the existence of an $o(k)$ -approximation within $f(k) \cdot \text{poly}(n)$ time for any function f that only depends on k .

Our second result demonstrates the infeasibility of achieving an approximation ratio of $m^{1/2-\epsilon}$ in polynomial time, unless $P = NP$. It is not difficult to show that a greedy algorithm selecting a path with the minimum number of arcs results in a $\lceil \sqrt{\ell} \rceil$ -approximation, where ℓ is the number of edges in all the paths of an optimal solution. Since $\ell \leq n$, this underscores the tightness of the $m^{1/2-\epsilon}$ -inapproximability bound.

Additionally, we establish that MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is fixed-parameter tractable when parameterized by ℓ but does not admit a polynomial kernel. Our hardness results hold for undirected graphs with unit weights, while our positive results extend to scenarios where the input graph is directed and features arbitrary (non-negative) edge weights.

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis; Theory of computation \rightarrow Shortest paths; Theory of computation \rightarrow Problems, reductions and completeness; Theory of computation \rightarrow Fixed parameter tractability

Keywords and phrases Inapproximability, Fixed-parameter tractability, Parameterized approximation

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.17

1 Introduction

We study a variant of the well-known problem VERTEX-DISJOINT PATHS. In the latter, the input comprises a (directed or undirected) graph G and k terminal pairs. The task is to identify whether pairwise vertex-disjoint paths can connect all terminals. VERTEX-DISJOINT PATHS has long been established as NP-complete [21] and has played a pivotal role in the graph-minor project by Robertson and Seymour [29].

Eilam-Tzoref [14] introduced a variant of VERTEX-DISJOINT PATHS where all paths in the solution must be *shortest* paths between the respective terminals. The parameterized complexity of this variant, known as VERTEX-DISJOINT SHORTEST PATHS, was recently



resolved [25] and subsequently the running time improved [3]: The problem, parameterized by k , is W[1]-hard and in XP (that is, polynomial-time solvable for constant k) for undirected graphs. On directed graphs, the problem is NP-hard already for $k = 2$ if zero-weight edges are allowed [16]. The problem is solvable in polynomial time for $k = 2$ for positive edge weights [4]. It is NP-hard when k is part of the input and the complexity for constant $k > 2$ remains open.

The optimization variant of VERTEX-DISJOINT SHORTEST PATHS, where not necessarily all terminal pairs need to be connected, but at least p of them, is referred to as MAXIMUM VERTEX-DISJOINT SHORTEST PATHS.

MAXIMUM VERTEX-DISJOINT SHORTEST PATHS

Input: A graph $G = (V, E)$, an edge-length function $w: E \rightarrow \mathbb{Q}_{\geq 0}$, terminal pairs $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ where $s_i \neq t_i$ for $i \in [k]$, and an integer p .

Question: Is there a set $S \subseteq [k]$ with $|S| \geq p$ such that there is a collection $\mathcal{C} = \{P_i\}_{i \in S}$ of pairwise vertex-disjoint paths such that for each $i \in S$, path P_i is a shortest path from s_i to t_i ?

A few remarks are in order. In the literature concerning VERTEX-DISJOINT PATHS and its variants, one usually distinguishes between vertex-disjoint and internally vertex-disjoint paths. In the latter, two paths in a solution might share common endpoints while in the former, all paths must be completely vertex disjoint – including the two ends. We focus on the variant where paths must be completely vertex-disjoint, but most of our results also hold for internally vertex-disjoint paths.

Note that VERTEX-DISJOINT SHORTEST PATHS is a special case of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS with $p = k$. For the maximization version, we are not given p as input but are instead asked to find a set S that is as large as possible. Slightly abusing notation, we do not distinguish between these two variants and refer to both as MAXIMUM VERTEX-DISJOINT SHORTEST PATHS.

While parameterization by k yields strong hardness bounds (both in terms of parameterized complexity and, as we will show later, approximation), another natural parameterization would be the sum of path lengths in a solution. We initiate the study of a related parameter ℓ , the minimum number of edges in an optimal solution (assuming the instance is a yes-instance, otherwise, we define $\ell = n$). If we confine all edge weights to be positive integers, then ℓ serves as a lower bound for the sum of path lengths. Since our hardness results apply to unweighted graphs, studying ℓ instead of the sum of path lengths does not weaken the negative results and ℓ proves to be very useful for approximation and parameterized algorithms. Note that the sum of path lengths is not a suitable parameter as dividing all edge weights by $m \cdot w_{\max}$ (where w_{\max} is the maximum weight of any edge in the input) yields an equivalent instance where the sum of path lengths in the solution is at most one.

For the parameterized complexity of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS, we note that the results for VERTEX-DISJOINT SHORTEST PATHS [3, 25] for the parameterization by k directly translate for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS parameterized by p . The problem is W[1]-hard as a generalization of VERTEX-DISJOINT SHORTEST PATHS, and to obtain an XP algorithm, it is sufficient to observe that in $n^{O(p)}$ time we can guess a set $S \subseteq [k]$ of size p and apply the XP algorithm for VERTEX-DISJOINT SHORTEST PATHS for the selected set of terminal pairs.

Before the recent work of Chitnis, Thomas, and Wirth [8], little was known about approximation algorithms for the MAXIMUM VERTEX-DISJOINT SHORTEST PATHS problem. Chitnis, Thomas, and Wirth demonstrated that no $(2 - \varepsilon)$ -approximation could be achieved in time $f(k) \cdot n^{o(k)}$ assuming the gap-ETH.

■ **Table 1** Overview of our results. New results are bold. All hardness results hold for unweighted and undirected graphs, while all new algorithmic results hold even for directed graphs with arbitrary non-negative edge weights.

Parameter	Exact	Approximation
no	NP-complete	no $m^{1/2-\varepsilon}$-approximation in poly(n) time
k	XP and W[1]-hard	no $o(k)$-approximation in $f(k) \cdot \text{poly}(n)$ time
ℓ	FPT and no poly kernel	$\lceil \sqrt{\ell} \rceil$ -approximation

For the related MAXIMUM VERTEX-DISJOINT PATHS, where the task is to connect the maximum number of terminal pairs by disjoint but not necessarily shortest paths, $O(\sqrt{n})$ -approximation algorithms are due to Kleinberg [23] and Kolliopoulos and Stein [24]. The best known lower bounds for this variant are $2^{\Omega(\sqrt{\log n})}$ and $n^{\Omega(1/(\log \log n)^2)}$. The first lower bound holds even if the input graph is an unweighted planar graph, while the second holds even if the input graph is an unweighted grid graph [9, 10]. For these two special cases, there are approximation algorithms achieving ratios $\tilde{O}(n^{9/19})$ and $\tilde{O}(n^{1/4})$, respectively [9, 10].

When requiring the solution paths to be edge-disjoint rather than vertex-disjoint, it is known that even computing a $m^{1/2-\varepsilon}$ -approximation is NP-hard in the directed setting [18]. There have also been some studies on relaxing the notion so that each edge appears in at most $c > 1$ of the solution paths. The integer c is called the congestion and the currently best known approximation algorithm achieves a $\text{poly}(\log n)$ -approximation with $c = 2$ [11].

Our results. We show that computing a $m^{1/2-\varepsilon}$ -approximation for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is NP-hard for any $\varepsilon > 0$ (Theorem 3). Moreover in terms of FPT-approximations, we demonstrate in Theorem 1 that any $k^{o(1)}$ -approximation in time $f(k) \cdot \text{poly}(n)$ implies $\text{FPT} = \text{W}[1]$ and that it is impossible to achieve an $o(k)$ -approximation in time $f(k) \cdot \text{poly}(n)$ unless the gap-ETH fails. This significantly improves the current state of approximation results for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS in two ways. First, we use the weaker assumption $\text{FPT} \neq \text{W}[1]$ instead of the gap-ETH. Second, our theorem excludes approximation factors polynomial in the input size, rather than only constant factors larger than 2 as shown by Chitnis et al. [8].

We complement the first lower bound by showing that a simple greedy strategy for MAXIMUM VERTEX-DISJOINT PATHS achieves a $\lceil \sqrt{\ell} \rceil$ -approximation also for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS (Theorem 4). In Theorems 5 and 7, we show that MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is fixed-parameter tractable when parameterized by ℓ , but it does not admit a polynomial kernel. We mention that our hardness results hold for undirected graphs with unit weights, and all our positive results hold even for directed and edge-weighted input graphs. We summarize our results in Table 1.

2 Preliminaries

For a positive integer x , we denote by $[x] = \{1, 2, \dots, x\}$ the set of all positive integers at most x . We denote by $G = (V, E)$ a graph and by n and m the number of vertices and edges in G , respectively. A graph G is said to be k -partite if V can be partitioned into k disjoint sets V_1, V_2, \dots, V_k such that each set V_i induces an independent set, that is, there is no edge $\{u, v\} \in E$ with $\{u, v\} \subseteq V_i$ for some $i \in [k]$. The *degree* of a vertex v is the number of edges in E that contain v as an endpoint and the *maximum degree* of a graph is the highest degree of any vertex in the graph.

A *path* in a graph G is a sequence $(v_0, v_1, \dots, v_\ell)$ of distinct vertices such that each pair (v_{i-1}, v_i) is connected by an edge in G . The first and last vertex v_0 and v_ℓ are called the *ends* of P . We also say that P is a path *from* v_0 *to* v_ℓ or a v_0 - v_ℓ -path. The length of a path is the sum of its edge lengths or simply the number ℓ of edges if the graph is unweighted. For two vertices v, w , we denote the length of a shortest v - w -path in G by $\text{dist}_G(v, w)$ or $\text{dist}(v, w)$ if the graph G is clear from the context.

We assume the reader to be familiar with the big-O notation and basic concepts in computational complexity like NP-completeness and reductions. We refer to the textbook by Garey and Johnson [17] for an introduction.

For a detailed introduction to parameterized complexity and kernelization, we refer the reader to the text books by Cygan et al. [12] and Fomin et al. [15]. A *parameterized problem* P is a language containing pairs (I, ρ) where I is an instance of an (unparameterized) problem and ρ is an integer called the *parameter*. In this paper, the parameter will usually be either the number k of terminal pairs or the minimum number ℓ of edges in a solution (a maximum collection of vertex-disjoint shortest paths between terminal pairs). A parameterized problem P is *fixed-parameter tractable* if there exists an algorithm solving any instance (I, ρ) of P in $f(\rho) \cdot \text{poly}(|I|)$ time, where f is some computable function only depending on ρ . To show that a problem is presumably not fixed-parameter tractable, one usually shows that the problem is hard for a complexity class known as $W[1]$. The class XP contains all parameterized problems which can be solved in $|I|^{f(\rho)}$ time, that is, in polynomial time if ρ is constant. A parameterized problem is said to admit a *polynomial kernel*, if there is a polynomial-time algorithm that given an instance (I, ρ) computes an equivalent instance (I', ρ') (called the *kernel*) such that $|I'| + \rho'$ are upper-bounded by a polynomial in ρ . It is known that any parameterized problem admitting a polynomial kernel is fixed-parameter tractable and each fixed-parameter tractable problem is contained in XP .

An α -approximation algorithm for a maximization problem is a polynomial-time algorithm that for any input returns a solution of size at least OPT/α where OPT is the size of an optimal solution. A parameterized α -approximation algorithm also returns a solution of size at least OPT/α , but its running time is allowed to be $f(\rho) \cdot \text{poly}(n)$, where ρ is the parameter and f is some computable function only depending on ρ . In this work, we always consider (unparameterized) approximation algorithms unless we specifically state a parameterized running time.

To exclude an α -approximation for an optimization problem, one can use the framework of *approximation-preserving reductions*. A strict approximation-preserving reduction is a pair of algorithms – called the *reduction algorithm* and the *solution-lifting algorithm* – that both run in polynomial time and satisfy the following. The reduction algorithm takes as input an instance I of a problem L and produces an instance I' of a problem L' . The solution-lifting algorithm takes any solution S' of I' and transforms it into a solution S of I such that if S' is an α -approximation for I' for some $\alpha \geq 1$, then S is an α -approximation for I . If a strict approximation-preserving reduction from L to L' exists and L is hard to approximate within some value β , then L' is also hard to approximate within β .

The *exponential time hypothesis (ETH)* introduced by Impagliazzo and Paturi [20] states that there is some $\varepsilon > 0$ such that each (unparameterized) algorithm solving 3-SAT takes at least $2^{\varepsilon n + o(n)}$ time, where n is the number of variables in the input instance. A stronger conjecture called the *gap-ETH* was independently introduced by Dinur [13] and Manurangsi and Raghavendra [27]. It states that there exist $\varepsilon, \delta > 0$ such that any $(1 - \varepsilon)$ -approximation algorithm for MAX 3-SAT¹ takes at least $2^{\delta n + o(n)}$ time.

¹ MAX 3-SAT is a generalization of 3-SAT where the question is not whether the input formula is satisfiable

3 Approximation

In this section, we show that MAXIMUM VERTEX-DISJOINT SHORTEST PATHS admits no $o(k)$ -approximation in $f(k) \cdot \text{poly}(n)$ time unless the gap-ETH fails and no $m^{1/2-\varepsilon}$ -approximation in polynomial time unless $P = NP$. We complement the latter result by developing a $\lceil \sqrt{\ell} \rceil$ -approximation algorithm that runs in polynomial time. We start with a reduction based on a previous reduction by Bentert et al. [3] and make it approximation-preserving.² Moreover, our result is tight in the sense that a k -approximation can be computed in polynomial time by simply connecting any terminal pair by a shortest path. A ck -approximation for any constant $c \leq 1$ can also be computed in polynomial time by guessing $\frac{1}{c}$ terminal pairs to connect and then using the XP-time algorithm by Bentert et al. [3] to find a solution. Note that since c is a constant, the XP-time algorithm for $\frac{1}{c}$ terminal pairs runs in polynomial time.

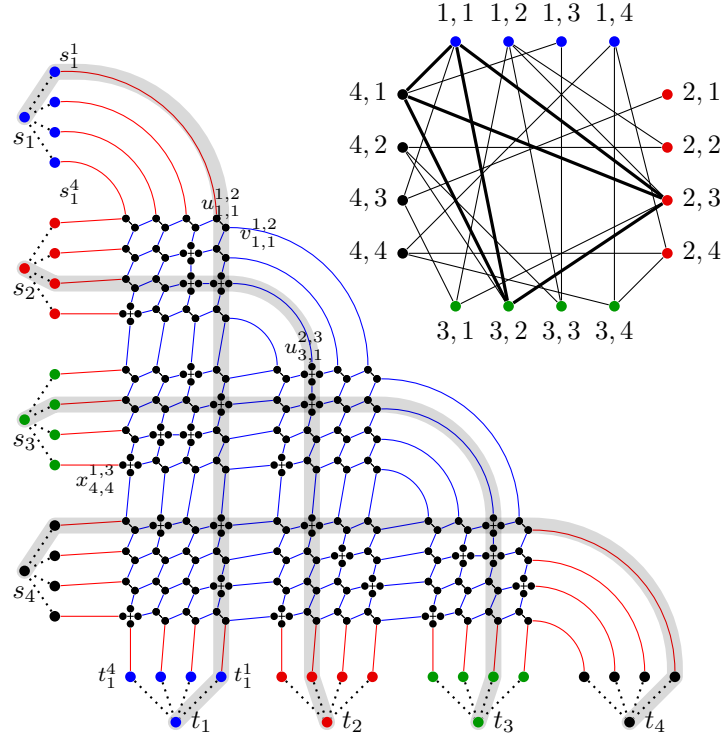
► **Theorem 1.** *Unless $FPT = W[1]$, MAXIMUM VERTEX-DISJOINT SHORTEST PATHS cannot be $k^{o(1)}$ -approximated in $f(k) \cdot \text{poly}(n)$ time, and assuming the gap-ETH, it cannot be $o(k)$ -approximated in $f(k) \cdot \text{poly}(n)$ time. All of these results hold even for subcubic graphs with terminals of degree one.*

Proof. We present a strict approximation-preserving reduction from MULTICOLORED CLIQUE to MAXIMUM VERTEX-DISJOINT SHORTEST PATHS such that the maximum degree is three and each terminal vertex has degree one. Moreover, the maximum number OPT of vertex-disjoint shortest paths between terminal pairs will be equal to the largest clique in the original instance. The theorem then follows from the fact that a $f(k) \cdot \text{poly}(n)$ -time $k^{o(1)}$ -approximation for CLIQUE would imply that $FPT = W[1]$ [7, 22], a $f(k) \cdot \text{poly}(n)$ -time $o(k)$ -approximation for CLIQUE refutes the gap-ETH [6], and that the textbook reduction from CLIQUE to MULTICOLORED CLIQUE only increases the number of vertices by a quadratic factor and does not change the size of a largest clique in the graph [12].

The reduction is depicted in Figure 1 and works as follows. Let $G = (V, E)$ be a k -partite graph (or equivalently a graph colored with k colors where all vertices of any color form an independent set) with ν vertices of each color. Let $V_i = \{v_1^i, v_2^i, \dots, v_\nu^i\}$ be the set of vertices of color $i \in [k]$ in G . We start with a terminal pair (s_i, t_i) for each color i and a pair of (non-terminal) vertices (s_j^i, t_j^i) for each vertex $v_j^i \in V_i$. Next for each color i , we add a binary tree of height $\lceil \log(\nu) \rceil$ where the vertices s_j^i are the leaves for all $v_j^i \in V_i$. We make s_i adjacent to the root of the binary tree. Analogously, we add a binary tree of the same height with leaves t_j^i and make t_i adjacent to the root. Next, we add a crossing gadget for each pair of vertices (v_j^i, v_b^a) with $i < a$. If $\{v_j^i, v_b^a\} \in E$, then the gadget consists of four vertices $u_{j,b}^{i,a}, v_{j,b}^{i,a}, x_{j,b}^{i,a}$, and $y_{j,b}^{i,a}$ and edges $\{u_{j,b}^{i,a}, v_{j,b}^{i,a}\}$ and $\{x_{j,b}^{i,a}, y_{j,b}^{i,a}\}$. If $\{v_j^i, v_b^a\} \notin E$, then the gadget consists of only two vertices $u_{j,b}^{i,a}$ and $v_{j,b}^{i,a}$ and the edge $\{u_{j,b}^{i,a}, v_{j,b}^{i,a}\}$. For the sake of notational convenience, we will in the latter case also denote $u_{j,b}^{i,a}$ by $x_{j,b}^{i,a}$ and $v_{j,b}^{i,a}$ by $y_{j,b}^{i,a}$. To complete the construction, we connect the different gadgets as follows. First, we connect via paths of length two $v_{j,b}^{i,a}$ and $u_{j,b+1}^{i,a}$ for all $b < \nu$ and $y_{j,b}^{i,a}$ and $x_{j-1,b}^{i,a}$ for all $j > 1$. Second, we connect via paths of length two the vertices $v_{j,\nu}^{i,a}$ to $u_{j,1}^{i,a+1}$ for all $j \in [\nu]$ and all $a < k$

but rather how many clauses can be satisfied simultaneously.

² We mention in passing that essentially the same modification to the reduction by Bentert et al. has been found by Akmal et al. [1] in independent research. While we use the modification to show stronger inapproximability bounds, they use it to show stronger fine-grained hardness results with respect to the minimum degree of a polynomial-time algorithm for constant k .



■ **Figure 1** An illustration of the reduction from MULTICOLORED CLIQUE to MAXIMUM VERTEX-DISJOINT SHORTEST PATHS.

Top right: Example instance for MULTICOLORED CLIQUE with $k = 4$ colors and $n = 4$ vertices per color. A multicolored clique is highlighted (by thick edges).

Bottom left: The constructed instance with the four shortest paths corresponding to the vertices of the clique highlighted. Note that these paths are pairwise disjoint. The dotted edges (incident to s_i and t_i vertices) indicate binary trees (where all leaves have distance $\lceil \log(\nu) \rceil$ from the root). Red edges indicate paths of length 2ν and blue edges indicate paths of length 2.

and $y_{1,b}^{i,a}$ to $x_{\nu,b}^{i+1,a}$ for all $b \in [\nu]$ and all $i < a - 1$. Third, we connect also via paths of length two $y_{1,b}^{i,i+1}$ to $u_{b,1}^{i+1,i+2}$ for all $i < k - 1$ and all $b \in [\nu]$. Next, we connect via paths of length 2ν each vertex s_j^i to $x_{\nu,j}^{1,i}$ for each $i > 1$ and $j \in [\nu]$. Similarly, $v_{j,\nu}^{i,k}$ is connected to t_j^i via paths of length 2ν . Finally, we connect s_j^1 with $u_{j,1}^{1,2}$ for all $j \in [\nu]$ and $y_{1,j}^{k-2,k-1}$ with t_j^k for all $j \in [\nu]$. This concludes the construction.

We next prove that all shortest s_i - t_i -paths are of the form

$$s_i - s_i^j - x_{\nu,j}^{1,i} - y_{1,j}^{i-1,i} - u_{j,1}^{i,i+1} - v_{j,\nu}^{i,k} - t_i^j - t_i \quad (1)$$

for some $j \in [\nu]$ and where the s_1 - t_1 -paths go directly from s_1^j to $u_{j,1}^{1,2}$ and the s_k - t_k -paths go directly from $y_{1,j}^{k-1,k}$ to t_j^k . We say that the respective path is the j^{th} canonical path for color i .

To show the above claim, first note that the distance from s_i to any vertex s_i^j is the same value $x = \lceil \log(\nu) \rceil + 1$ for all pairs of indices i and j . Moreover, the same holds for t_i and t_i^j , each s_i - t_i -path contains at least one vertex s_i^j and one vertex $t_i^{j'}$ for some $j, j' \in [\nu]$, and all paths of the form in Equation (1) are of length $y = 2x + 4\nu + 3(k-1)\nu - 2$. We first show that each s_i - t_i -path of length at most y contains an edge of the form $y_{1,b}^{i,i+1}$ to $u_{b,1}^{i+1,i+2}$. Consider the graph where all of these edges are removed. Note that due to the

grid-like structure, the distance between s_i and $x_{j,b}^{i',a}$ for any values $i' \leq i \leq a, j$, and b is at least $x + 2\nu + 3(i' - 1)\nu + 3(\nu - j)$ if $i = a$ and at least $x + 2\nu + 3(i' - 1)\nu + 3(a - i)\nu + 3(\nu - j) + 3b$ if $i < a$.³ Hence, all shortest s_i - t_i -paths use an edge of the form $y_{1,b}^{i,i+1}$ to $u_{b,1}^{i+1,i+2}$ and the shortest path from s_i^j to some vertex $y_{1,b}^{i,i+1}$ is to the vertex $y_{1,j}^{i,i+1}$. Note that the other endpoint of the specified edge is $u_{j,1}^{i,i+1}$ and the shortest path to t_i now goes via t_i^j for analogous reasons. Thus, all shortest s_i - t_i -paths have the form (1).

We next prove that any set of p disjoint shortest paths between terminal pairs (s_i, t_i) in the constructed graph has a one-to-one correspondence to a multicolored clique of size p for any p . For the first direction, assume that there is a set P of disjoint shortest paths between p terminal pairs. Let $S \subseteq [k]$ be the set of indices such that the paths in P connect s_i and t_i for each $i \in S$. Moreover, let j_i be the index such that the shortest s_i - t_i -path in P is the j_i^{th} canonical path for i for each $i \in S$. Now consider the set $K = \{v_{j_i}^i \mid i \in S\}$ of vertices in G . Clearly K contains at most one vertex of each color and is of size p as S is of size p . It remains to show that K induces a clique in G . To this end, consider any two vertices $v_{j_i}^i, v_{j_{i'}}^{i'} \in K$. We assume without loss of generality that $i < i'$. By assumption, the j_i^{th} canonical path for i and the $j_{i'}^{\text{th}}$ canonical path for i' are disjoint. This implies that $u_{j_i, j_{i'}}^{i, i'} \neq x_{j_i, j_{i'}}^{i, i'}$ as the j_i^{th} canonical path for i contains the former and the $j_{i'}^{\text{th}}$ canonical path for i' contains the latter. By construction, this means that $\{v_{j_i}^i, v_{j_{i'}}^{i'}\} \in E$. Since the two vertices were chosen arbitrarily, it follows that all vertices in K are pairwise adjacent, that is, K induces a multicolored clique of size p .

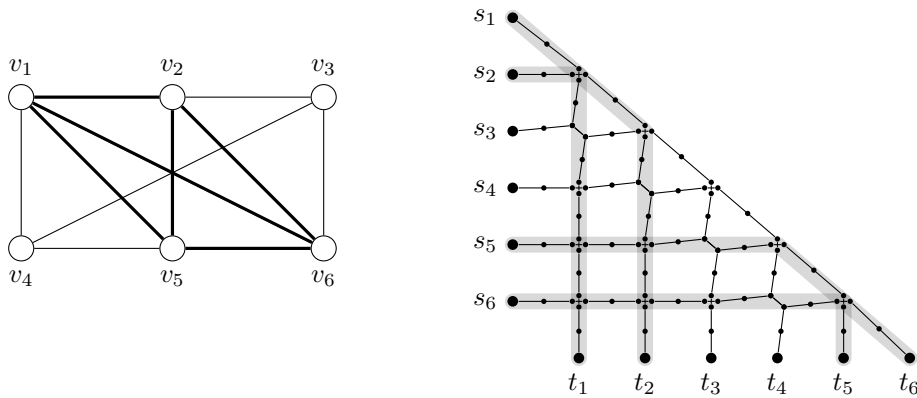
For the other direction assume that there is a multicolored clique $C = \{v_{j_1}^{i_1}, v_{j_2}^{i_2}, \dots, v_{j_p}^{i_p}\}$ of size p in G . We will show that the j_q^{th} canonical path for i_q is vertex-disjoint from the j_r^{th} canonical path for i_r for all $q \neq r \in [p]$. Let q, r be two arbitrary distinct indices in $[p]$ and let without loss of generality be $q < r$. Note that the two mentioned paths can only overlap in vertices $u_{j_q, j_r}^{i_q, i_r}, v_{j_q, j_r}^{i_q, i_r}, x_{j_q, j_r}^{i_q, i_r}$, or $y_{j_q, j_r}^{i_q, i_r}$ and that the j_q^{th} canonical path for i_q only contains vertices $u_{j_q, j_r}^{i_q, i_r}$ and $v_{j_q, j_r}^{i_q, i_r}$ and the j_r^{th} canonical path for i_r only contains $x_{j_q, j_r}^{i_q, i_r}$ and $y_{j_q, j_r}^{i_q, i_r}$. Moreover, since by assumption $v_{j_q}^{i_q}$ and $v_{j_r}^{i_r}$ are adjacent, it holds by construction that $u_{j_q, j_r}^{i_q, i_r}, v_{j_q, j_r}^{i_q, i_r}, x_{j_q, j_r}^{i_q, i_r}$, and $y_{j_q, j_r}^{i_q, i_r}$ are four distinct vertices. Thus, we found vertex disjoint paths between p distinct terminal pairs. This concludes the proof of correctness.

To finish the proof, observe that the constructed instance has maximum degree three, all terminal vertices have degree one, and the construction can be computed in polynomial time. \blacktriangleleft

We mention in passing that in graphs of maximum degree three with terminal vertices of degree one, two paths are vertex disjoint if and only if they are edge disjoint. Hence, Theorem 1 also holds for MAXIMUM EDGE-DISJOINT SHORTEST PATHS, the edge-disjoint version of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS.

► **Corollary 2.** *Unless $FPT = W[1]$, MAXIMUM EDGE-DISJOINT SHORTEST PATHS cannot be $k^{o(1)}$ -approximated in $f(k) \cdot \text{poly}(n)$ time, and assuming the gap-ETH, it cannot be $o(k)$ -approximated in $f(k) \cdot \text{poly}(n)$ time. All of these results hold even for subcubic graphs with terminals of degree one.*

³ We mention that there are some pairs of vertices $x_{j_1, b_1}^{i_1, a_1}$ and $x_{j_2, b_2}^{i_2, a_2}$, where the distance between the two is less than $3(|i_1 - i_2| + |a_1 - a_2|)\nu + 3(|j_1 - j_2| + |b_1 - b_2|)$. An example is the pair $(x_{1,1}^{1,2} = u_{1,1}^{1,2}, x_{2,2}^{1,2})$ in Figure 1 with a distance of 4. However, in all examples it holds that $i_1\nu - j_1 \neq i_2\nu - j_2$ and $a_1\nu + b_1 \neq a_2\nu + b_2$ such that the left side is either smaller in both inequalities or larger in both inequalities. Hence, these pairs cannot be used as shortcuts as they move “down and left” instead of towards “down and right” in Figure 1.



■ **Figure 2** An illustration of the reduction from CLIQUE to MAXIMUM VERTEX-DISJOINT SHORTEST PATHS.

Left side: Example instance for CLIQUE with a highlighted solution (by thick edges).

Right side: The constructed instance with the four shortest paths corresponding to the solution on the left side highlighted. Note that each shortest s_i - t_i -path uses exactly two of the diagonal edges.

We continue with an unparameterized lower bound by establishing that computing a $m^{\frac{1}{2}-\varepsilon}$ -approximation is NP-hard. We mention that the reduction is quite similar to the reduction in the proof for Theorem 1.

► **Theorem 3.** *Computing a $m^{1/2-\varepsilon}$ -approximation for any $\varepsilon > 0$ for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is NP-hard.*

Proof. It is known that computing a $n^{1-\varepsilon}$ -approximation for CLIQUE is NP-hard [19, 30]. We present an approximation-preserving reduction from CLIQUE to MAXIMUM VERTEX-DISJOINT SHORTEST PATHS based on Theorem 1. We use basically the same reduction as in Theorem 1 but we start from an instance of CLIQUE and have a separate terminal pair for each vertex in the graph. Moreover, we do not require the binary trees pending from the terminal vertices and neither do we require long induced paths (red edges in Figure 1). These are instead paths with one internal vertex. An illustration of the modified reduction is given in Figure 2. Note that the number of vertices and edges in the graph is at most $3N^2$, where N is the number of vertices in the instance of CLIQUE. Moreover, for each terminal pair (s_i, t_i) , there is exactly one shortest s_i - t_i -path (the path that moves horizontally in Figure 2 until it reaches the main diagonal, then uses exactly two edges on the diagonal, and finally moves vertically to t_i).

We next prove that for any p , there is a one-to-one correspondence between a set of p disjoint shortest paths between terminal pairs (s_i, t_i) in the constructed graph and a clique of size p in the input graph. For the first direction, assume that there is a set P of disjoint shortest paths between p terminal pairs. Let $S \subseteq [k]$ be the set of indices such that the paths in P connect s_i and t_i for each $i \in S$. Now consider the set $K = \{v_i \mid i \in S\}$ of vertices in G . Clearly K contains p vertices. It remains to show that K induces a clique in G . To this end, consider any two vertices $v_i, v_j \in K$. We assume without loss of generality that $i < j$. By assumption, the unique shortest s_i - t_i -path and the unique shortest s_j - t_j -path are vertex-disjoint. By the description of the shortest paths between terminal pairs and the fact that s_i is higher than s_j and t_i is to the left of t_j , it holds that the two considered paths both visit the region that is to the right of s_j and above t_i . This implies that two edges must be crossing at this position, that is, there are four vertices in the described region and not

only two. By construction, this means that $\{v_i, v_j\} \in E$. Since the two vertices were chosen arbitrarily, it follows that all vertices in K are pairwise adjacent, that is, K induces a clique of size p in the input graph.

For the other direction assume that there is a clique $C = \{v_{i_1}, v_{i_2}, \dots, v_{i_p}\}$ of size p in the input graph. We will show that the unique shortest s_{i_q} - t_{i_q} -path is vertex-disjoint from the unique shortest s_{i_r} - t_{i_r} -path for all $q \neq r \in [p]$. Let q, r be two arbitrary distinct indices in $[p]$ and let without loss of generality be $q < r$. Note that the two mentioned paths can only overlap in the region that is to the right of s_{i_r} and above t_{i_q} . Moreover, since by assumption v_{i_q} and v_{i_r} are adjacent, it holds by construction that there are four distinct vertices in the described region and the two described paths are indeed vertex-disjoint. Thus, we found vertex disjoint paths between p distinct terminal pairs.

We conclude by analyzing the approximation ratio. Note that we technically did not prove a strict reduction for the factor $m^{1-\varepsilon}$ as the number of vertices in the original instance and the number of edges in the constructed instance are not identical. Still, the number m of edges in the constructed instance is at most $3N^2$, where N is the number of vertices in the original instance of CLIQUE. Hence, any $m^{1/2-\varepsilon}$ -approximation for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS corresponds to a $(3N^2)^{1/2-\varepsilon} = N^{1-\varepsilon'}$ -approximation for CLIQUE for some $0 < \varepsilon' < 2\varepsilon$ and therefore computing a $m^{1/2-\varepsilon}$ -approximation for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is NP-hard. ◀

Note that the maximum degree of the constructed instance is again three and all terminal vertices are of degree one. Thus, Theorem 3 also holds for the edge disjoint version of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS. However in this case, a very similar result was already known before. Guruswami et al. [18] showed that computing a $m^{1/2-\varepsilon}$ -approximation is NP-hard for a related problem called LENGTH BOUNDED EDGE-DISJOINT PATHS. Their reduction immediately implies the same hardness for MAXIMUM EDGE-DISJOINT SHORTEST PATHS. To the best of our knowledge, the best known unparameterized approximation lower bound for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS was the $2 - \varepsilon$ lower bound due to Chitnis et al. [8] and we are not aware of any lower bound for MAXIMUM VERTEX-DISJOINT PATHS.

We next show that this result is tight, that is, we show how to compute a \sqrt{n} -approximation for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS in polynomial time. We also show that the same algorithm achieves a $\lceil \sqrt{\ell} \rceil$ -approximation. Note that we can always assume that $\ell \leq n$ as a set of vertex-disjoint paths is a forest and the number of edges in a forest is less than its number of vertices. We mention that this algorithm is basically identical to the best known (unparameterized) approximation algorithm for MAXIMUM DISJOINT PATHS [23, 24].

► **Theorem 4.** *There is a polynomial-time algorithm for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS on directed and weighted graphs that achieves an approximation factor of $\min\{\sqrt{n}, \lceil \sqrt{\ell} \rceil\}$.*

Proof. Let OPT be a maximum subset of terminal pairs that can be connected by shortest pairwise vertex-disjoint paths and let j be the index of a terminal pair (s_j, t_j) such that a shortest (s_j, t_j) -path contains a minimum number of arcs. We can compute the index j as well as a shortest s_j - t_j -path with a minimum number of arcs by running a folklore modification of Dijkstra's algorithm from each terminal vertex s_i .⁴ Let ℓ_j be the number of arcs in the

⁴ The standard Dijkstra's algorithm is modified by assigning to each vertex a pair of labels: the distance

found path. Our algorithm iteratively picks the shortest s_j - t_j -path using ℓ_j arcs, removes all involved vertices from the graph, recomputes the distance between all terminal pairs, removes all terminal pairs whose distance increased, updates the index j , and recomputes ℓ_j . We distinguish whether $\ell_j + 1 \leq \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ or not.

While $\ell_j + 1 \leq \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$, note that we removed at most $\ell_j + 1$ terminal pairs in OPT. Hence, if $\ell_j + 1 \leq \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ holds at every stage, then we connected at least $|\text{OPT}| / \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ terminal pairs, that is, we found a $\min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ -approximation.

So assume that at some point $\ell_j > \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ and let x be the number of terminal pairs that we already connected by disjoint shortest paths. By the argument above, we have removed at most $x \cdot \min(\sqrt{n}, \lceil \sqrt{\ell} \rceil)$ terminal pairs from OPT thus far. We now make a case distinction whether or not $\sqrt{n} \leq \lceil \sqrt{\ell} \rceil$. If $\ell_j + 1 > \lceil \sqrt{\ell} \rceil \geq \sqrt{n}$, then we note that all remaining paths in OPT contain at least \sqrt{n} vertices each and since the paths are vertex-disjoint, there can be at most \sqrt{n} paths left in OPT. Hence, we can infer that $|\text{OPT}| \leq (x + 1) \cdot \sqrt{n}$. Consequently, even though we might remove all remaining terminal pairs in OPT by connecting s_j and t_j , this is still a \sqrt{n} -approximation (and a $\lceil \sqrt{\ell} \rceil$ -approximation as we assumed $\lceil \sqrt{\ell} \rceil \geq \sqrt{n}$).

If $\ell_j + 1 > \sqrt{n} \geq \lceil \sqrt{\ell} \rceil$, then we note that all remaining paths in OPT contain at least $\ell_j > \lceil \sqrt{\ell} \rceil - 1$ edges each. Moreover, since ℓ_j and $\lceil \sqrt{\ell} \rceil$ are integers, each path contains at least $\lceil \sqrt{\ell} \rceil$ edges each. Since all paths in OPT contain by definition at most ℓ edges combined, the number of paths in OPT is at most $\ell / \lceil \sqrt{\ell} \rceil \leq \lceil \sqrt{\ell} \rceil$. Hence, we can infer in that case that $|\text{OPT}| \leq (x + 1) \cdot \lceil \sqrt{\ell} \rceil$. Again, even if we remove all remaining terminal pairs in OPT by connecting s_j and t_j , this is still a $\lceil \sqrt{\ell} \rceil$ -approximation (and a \sqrt{n} -approximation as we assumed $\sqrt{n} \geq \lceil \sqrt{\ell} \rceil$). This concludes the proof. ◀

4 Exact Algorithms

In this section, we show that MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is fixed-parameter tractable when parameterized by ℓ , but it does not admit a polynomial kernel. The proof for the first result uses the technique of *color coding* of Alon, Yuster, and Zwick [2]. Imagine we are searching for some structure of size k in a graph. The idea of color coding is to color the vertices (or edges) of the input graph with a set of k colors and then only search for colorful solutions, that is, structures in which all vertices have distinct colors. Of course, this might not yield an optimal solution, but by trying enough different random colorings, one can often get a constant error probability in $f(k) \cdot \text{poly}(n)$ time. Using the standard tool of (n, k) -perfect hash families, these types of algorithms can be derandomized without significant overhead in the running time.

► **Theorem 5.** *MAXIMUM VERTEX-DISJOINT SHORTEST PATHS on weighted and directed graphs can be solved in $2^{O(\ell)} \text{poly}(n)$ time.*

Proof. Let $(G, w, (s_1, t_1), \dots, (s_k, t_k), p)$ be an instance of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS. First, we guess the value of ℓ by starting with $\ell = p$ and increasing the value of ℓ by one whenever we cannot find a solution with at least p shortest paths and at most ℓ edges. We start with $\ell = p$ as any set of p disjoint paths contains at least ℓ edges. Notice that the total number of vertices in a (potential) solution with p paths is at most $\ell + p$. We use the color-coding technique of Alon, Yuster, and Zwick [2]. We color the

from the terminal and the number of arcs in the corresponding path; then the pairs of labels are compared lexicographically.

vertices of G uniformly at random using $p + \ell$ colors (the set of colors is $[\ell + p]$) and observe that the probability that all the vertices in the paths in a solution have distinct colors is at least $\frac{(p+\ell)!}{(p+\ell)^{(p+\ell)}} \geq e^{-(p+\ell)}$. We say that a solution to the considered instance is *colorful* if distinct paths in the solution have no vertices of the same color. Note that we do not require that the vertices within a path in the solution are colored by distinct/equal colors. The crucial observations are that any colorful solution is a solution and the probability of the existence of a colorful solution for a yes-instance of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is at least $e^{-(p+\ell)}$ as any solution in which all vertices receive distinct colors is a colorful solution.

We use dynamic programming over subsets of colors to find a colorful solution. More precisely, we find the minimum number of arcs in a collection $\mathcal{C} = \{P_i\}_{i \in S}$ of p pairwise vertex-disjoint paths for some $S \subseteq [k]$ satisfying the conditions: (i) for each $i \in S$, the path P_i is a shortest path from s_i to t_i and (ii) there are no vertices of distinct paths of the same color.

For a subset $X \subseteq [p + \ell]$ of colors and a positive integer $r \leq p$, we denote by $f[X, r]$ the minimum total number of arcs in r shortest paths connecting distinct terminal pairs such that the paths contains only vertices of colors in X and there are no vertices of distinct paths of the same color. We set $f[X, r] = \infty$ if such a collection of r paths does not exist.

To compute f , if $r = 1$, then let $W \subseteq V$ be the subset of vertices colored by the colors in X . We use Dijkstra's algorithm to find the set $I \subseteq [k]$ of all indices $i \in [k]$ such that the lengths of the shortest s_i - t_i -paths in G and $G[W]$ are the same. If $I = \emptyset$, then we set $f[X, 1] = \infty$. Assume that this is not the case. Then, we use the variant of Dijkstra's algorithm mentioned in Theorem 4 to find the index $i \in I$ and a shortest s_i - t_i -path P in $G[W]$ with a minimum number of arcs. Finally, we set $f[X, 1]$ to be equal to the number of arcs in P .

For $r \geq 2$, we compute $f[X, r]$ for each $X \subseteq [p + \ell]$ using the recurrence relation

$$f[X, r] = \min_{Y \subset X} \{f[X \setminus Y, r - 1] + f[Y, 1]\}. \quad (2)$$

The correctness of computing the values of $f[X, 1]$ follows from the description and the correctness of recurrence (2) follows from the condition that distinct paths should not have vertices of the same color.

We compute the values $f[X, r]$ in order of increasing $r \in [p]$. Since computing $f[Y, 1]$ for a given set Y of colors can be done in polynomial time, we can compute all values in overall $3^{p+\ell} \text{poly}(n)$ time. Once all values $f[X, r]$ are computed, we observe that a colorful solution exists if and only if $f[S, p] \leq \ell$.

If there is a colorful solution, then we conclude that $(G, w, (s_1, t_1), \dots, (s_k, t_k), p)$ is a yes-instance of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS. Otherwise, we discard the considered coloring and try another random coloring and iterate. If we fail to find a solution after executing $N = \lceil e^{p+\ell} \rceil$ iterations, we obtain that the probability that $(G, w, (s_1, t_1), \dots, (s_k, t_k), p)$ is a yes-instance is at most $(1 - \frac{1}{e^{p+\ell}})^{e^{p+\ell}} \leq e^{-1}$. Thus, we return that $(G, w, (s_1, t_1), \dots, (s_k, t_k), p)$ is a no-instance with the error probability upper bounded by $e^{-1} < 1$. Since the running time in each iteration is $3^{p+\ell} \text{poly}(n)$ and $p \leq \ell$, the total running time is in $2^{O(\ell)} \text{poly}(n)$. Note that we do the color coding and dynamic programming for each value between p and the actual value ℓ . However, this only adds an additional factor of $\ell \leq n$ which disappears in the $\text{poly}(n)$.

The above algorithm can be derandomized using the results of Naor, Schulman, and Srinivasan [28] by replacing random colorings by perfect hash families. We refer to the textbook by Cygan et al. [12] for details on this common technique. ◀

17:12 Tight Approximation and Kernelization Bounds for Vertex-Disjoint Shortest Paths

The FPT result of Theorem 5 immediately raises the question about the existence of a polynomial kernel. To show that a parameterized problem P does presumably not admit a polynomial kernel, one can use the framework of *cross-compositions*. Given an NP-hard problem L , a polynomial equivalence relation R on the instances of L is an equivalence relation such that (i) one can decide for any two instances in polynomial time whether they belong to the same equivalence class, and (ii) for any finite set S of instances, R partitions the set into at most $\max_{I \in S} \text{poly}(|I|)$ equivalence classes. Given an NP-hard problem L , a parameterized problem P , and a polynomial equivalence relation R on the instances of L , an OR-cross-composition of L into P (with respect to R) is an algorithm that takes q instances I_1, I_2, \dots, I_q of L belonging to the same equivalence class of R and constructs in $\text{poly}(\sum_{i=1}^q |I_i|)$ time an instance (I, ρ) of P such that (i) ρ is polynomially upper-bounded by $\max_{i \in [q]} |I_i| + \log(q)$, and (ii) (I, ρ) is a yes-instance of P if and only if at least one of the instances I_i is a yes-instance of L . If a parameterized problem admits an OR-cross-composition, then it does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP}/\text{poly}$ [5].

In order to exclude a polynomial kernel, we first show that a special case of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS remains NP-hard. We call this special case LAYERED VERTEX-DISJOINT SHORTEST PATHS and it is the special case of VERTEX-DISJOINT SHORTEST PATHS where all edges have weight 1 and the input graph is layered, that is, there is a partition of the vertices into (disjoint) sets $V_1, V_2, \dots, V_\lambda$ such that all edges $\{u, v\}$ are between two consecutive layers, that is $u \in V_i$ and $v \in V_{i+1}$ or $u \in V_{i+1}$ and $v \in V_i$ for some $i \in [\lambda - 1]$. Moreover, each terminal pair (s_i, t_i) satisfies that $s_i \in V_1$, $t_i \in V_\lambda$, and each shortest path between the two terminals is *monotone*, that is, it contains exactly one vertex of each layer. LAYERED VERTEX-DISJOINT SHORTEST PATHS is formally defined as follows.

LAYERED VERTEX-DISJOINT SHORTEST PATHS

Input: A λ -layered graph $G = (V, E)$ with a λ -partition $\{V_1, V_2, \dots, V_\lambda\}$ of the vertex set, terminal pairs $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ with $s_i \in V_1$, $t_i \in V_\lambda$, and $\text{dist}(s_i, t_i) = \lambda - 1$ for all $i \in [k]$.

Question: Is there a collection $\mathcal{C} = \{P_i\}_{i \in [k]}$ of pairwise vertex-disjoint paths such that P_i is an s_i - t_i -path of length $\lambda - 1$ for all $i \in [k]$?

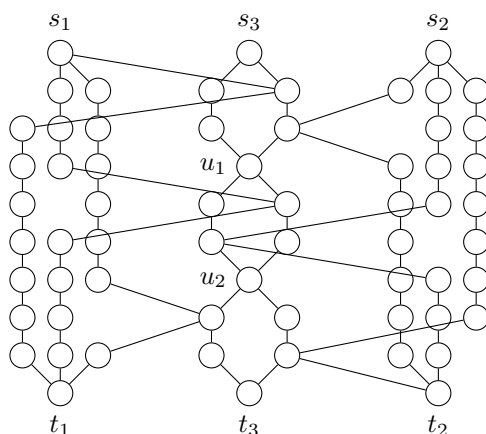
It is quite straight-forward to prove that LAYERED VERTEX-DISJOINT SHORTEST PATHS is NP-complete.

► **Proposition 6.** *LAYERED VERTEX-DISJOINT SHORTEST PATHS is NP-complete.*

Proof. We focus on the NP-hardness as LAYERED VERTEX-DISJOINT SHORTEST PATHS is a special case of VERTEX-DISJOINT SHORTEST PATHS and therefore clearly in NP. We reduce from 3-SAT. The main part of the reduction is a selection gadget. The gadget consists of a set U of $n + 1$ vertices u_0, u_1, \dots, u_n and between each pair of consecutive vertices u_{i-1}, u_i , there are two paths with m internal vertices each. Let the set of vertices be $V_i = \{v_1^i, v_2^i, \dots, v_m^i\}$ and $W_i = \{w_1^i, w_2^i, \dots, w_m^i\}$. The set of edges in the selection gadget is

$$E = \{\{u_{i-1}, v_1^i\}, \{u_{i-1}, w_1^i\}, \{v_m^i, u_i\}, \{w_m^i, u_i\} \mid i \in [n]\} \\ \cup \{\{v_j^i, v_{j+1}^i\}, \{w_j^i, w_{j+1}^i\} \mid i \in [n] \wedge j \in [m-1]\}.$$

The constructed instance will have $m + 1$ terminal pairs and is depicted in Figure 3. We set $s_{m+1} = u_0$ and $t_{m+1} = u_n$ and we will ensure that any shortest s_{m+1} - t_{m+1} -path contains all vertices in U and for each $i \in [n]$ either all vertices in V_i or all vertices in W_i . These

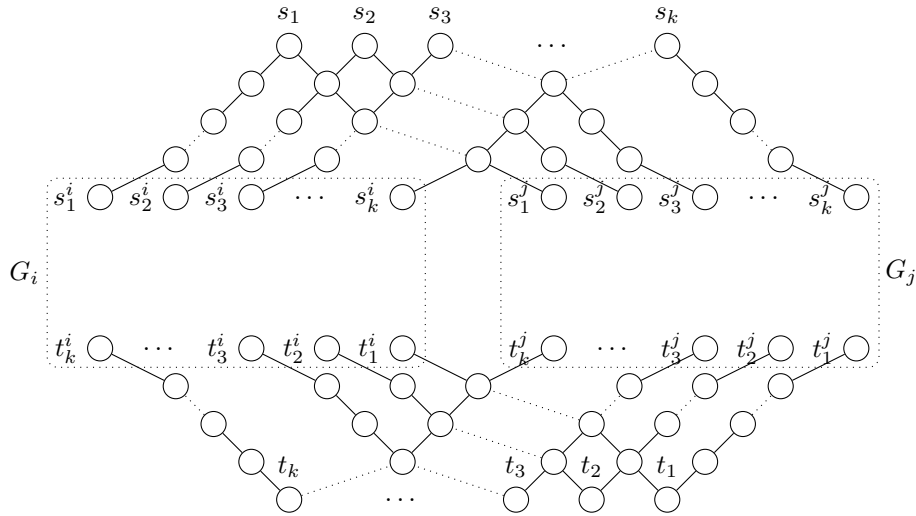


■ **Figure 3** An example of the construction in the proof of Proposition 6 for the input formula $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$.

choices will correspond to setting the i^{th} variable to either true or false. Additionally, we have a terminal pair (s_j, t_j) for each clause C_j . There are (up to) three disjoint paths between s_j and t_j , each of which is of length $n \cdot (m + 1)$. These paths correspond to which literal in the clause satisfies it. For each of these paths, let x_i be the variable corresponding to the path. If x_i appears positively in C_j , then we identify the $(i - 1)(m + 1) + j + 1^{\text{st}}$ vertex in the path with w_j^i and if x_i appears negatively, then we identify the vertex with v_j^i . Note that the constructed instance is $(m + 1)n$ -layered and that once any monotone path starting in s_{m+1} leaves the selection gadget, it cannot end in t_{m+1} as any vertex outside the selection gadget has degree at most two and at the end of these paths are only terminals t_1, t_2, \dots, t_m .

Since the construction runs in polynomial time, we focus on the proof of correctness. If the input formula is satisfiable, then we connect all terminal pairs as follows. Let β be a satisfying assignment. The terminal pair (s_{m+1}, t_{m+1}) is connected by a path containing all vertices in U and for each $i \in [n]$, if β assigns the i^{th} variable to true, then the path contains all vertices in V_i and otherwise all vertices in W_i . For each clause C_j , let x_{i_j} be variable in C_j which β uses to satisfy C_j (if multiple such variables exist, we choose an arbitrary one). By construction, there is a path associated with x_{i_j} that connects s_j and t_j and only uses one vertex in W_i if x_{i_j} appears positively in C_j and a vertex in V_i , otherwise. Since each vertex in V_i and W_i is only associated with at most one such path, we can connect all terminal pairs. For the other direction assume that all $m + 1$ terminal pairs can be connected by disjoint shortest paths. As argued above, the s_{m+1} - t_{m+1} -path stays in the selection gadget. We define a truth assignment by assigning the i^{th} variable to true if and only if the s_{m+1} - t_{m+1} -path contains the vertices in V_i . For each clause C_j , we look at the neighbor of s_j in the solution. This vertex belongs to a path of degree-two vertices that at some point joins the selection gadget. By construction, the vertex where this happens is not used by the s_{m+1} - t_{m+1} -path, which guarantees that C_j is satisfied by the corresponding variable. Since all clauses are satisfied by the same assignment, the formula is satisfiable and this concludes the proof. ◀

With the NP-hardness of LAYERED VERTEX-DISJOINT SHORTEST PATHS at hand, we can now show that it does not admit a polynomial kernel when parameterized by ℓ by providing an OR-cross-composition from its unparameterized version to the version parameterized by ℓ .



■ **Figure 4** The construction to merge two instances of LAYERED VERTEX-DISJOINT SHORTEST PATHS into one equivalent instance. The dotted edges can be read as regular edges for $k = 4$ and indicate where additional vertices and edges have to be added for more terminal pairs. Note that the height of a vertex in the drawing does not indicate its layer as dotted edges distort the picture.

► **Theorem 7.** *LAYERED VERTEX-DISJOINT SHORTEST PATHS* parameterized by $\ell = k \cdot (\lambda - 1)$ does not admit a polynomial kernel unless $NP \subseteq coNP/poly$.

Proof. We present an OR-cross-composition from LAYERED VERTEX-DISJOINT SHORTEST PATHS into LAYERED VERTEX-DISJOINT SHORTEST PATHS parameterized by ℓ . To this end, assume we are given t instances of LAYERED VERTEX-DISJOINT SHORTEST PATHS all of which have the same number λ of layers and the same number k of terminal pairs. Moreover, we assume that t is some power of two. Note that we can pad the instance with at most t trivial no-instances to reach an equivalent instance in which the number of instances is a power of two and the size of all instances combined has at most doubled.

The main ingredient for our proof is a construction to merge two instances into one. The construction is depicted in Figure 4. We first prove that the constructed instance is a yes-instance if and only if at least one of the original instances was a yes-instance. Afterwards, we will show how to use this construction to get an OR-cross-composition for all t instances.

To show that the construction works correctly, first assume that one of the two original instances is a yes-instance. Since both cases are completely symmetrical, assume that there are shortest disjoint paths between all terminal pairs (s_a^i, t_a^i) for all $a \in [k]$ in G_i . Then, we can connect all terminal pairs (s_b, t_b) by using the unique shortest paths between s_b and s_b^i and between t_b^i and t_b for all $b \in [k]$ together with the solution paths inside G_i . Now assume that there is a solution in the constructed instance, that is, there are pairwise vertex-disjoint shortest paths between all terminal pairs (s_b, t_b) for all $b \in [k]$. First assume that the s_1 - t_1 -path passes through G_i . Then, this path uses the unique shortest path from t_1^i to t_1 . Note that this path blocks all paths between t_b^j and vertices in G_j for all $b \neq 1$. Thus, all paths have to pass through the graph G_i . Note that the only possible way to route vertex-disjoint paths from all s -terminals to all s^i terminals and from all t^i -terminals to all t -terminals is to connect s_a to s_a^i and t_a^i to t_a for all $a \in [k]$. This implies that there is a solution that contains vertex-disjoint shortest paths between s_a^i and t_a^i in G_i for all $a \in [k]$, that is, at least one of the two original instances is a yes-instance. The case where the s_1 - t_1 -path passes

through G_j is analogous since the only monotone path from s_1 to a vertex in G_j is the unique shortest s_1 - s_1^j -path and this path blocks all monotone paths from s_a to vertices in G_i for all $a \neq 1$.

Note that the constructed graph is layered and that the number of layers is $\lambda + 2k$. Moreover, the size of the new instance is in $O(|G_i| + |G_j| + k^2)$. To complete the reduction, we iteratively half the number of instances by partitioning all instances into arbitrary pairs and merge the two instances in a pair into one instance. After $\log t$ iterations, we are left with a single instance which is a yes-instance if and only if at least one of the t original instances is a yes-instance. The size of the instance is in $O(\sum_{i \in [t]} |G_i| + t \cdot k^2)$ which is clearly polynomial in $\sum_{i \in [t]} |G_i|$ as each instance contains at least k vertices. Moreover, the parameter ℓ in the constructed instance is $k \cdot (\lambda - 1) + 2k \log t$, which is polynomial in $|G_i| + \log t$ for each graph G_i as G_i contains at least one vertex in each of the λ layers and at least k terminal vertices. Thus, all requirements of an OR-cross-composition are met and this concludes the proof. \blacktriangleleft

Note that since LAYERED VERTEX-DISJOINT SHORTEST PATHS is a special case of VERTEX-DISJOINT SHORTEST PATHS (and therefore of MAXIMUM VERTEX-DISJOINT SHORTEST PATHS), Theorem 7 also excludes polynomial kernels for these problems parameterized by ℓ .

► **Corollary 8.** *VERTEX-DISJOINT SHORTEST PATHS and MAXIMUM VERTEX-DISJOINT SHORTEST PATHS parameterized by ℓ do not admit polynomial kernels unless $NP \subseteq coNP/poly$.*

5 Conclusion

In this paper, we studied MAXIMUM VERTEX-DISJOINT SHORTEST PATHS. We show that there is no $m^{1/2-\epsilon}$ -approximation in polynomial time unless $P = NP$. Moreover, if $FPT \neq W[1]$ or assuming the stronger gap-ETH, we show that there are no non-trivial approximations for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS in $f(k) \cdot \text{poly}(n)$ time. When parameterized by ℓ , there is a simple $\lceil \sqrt{\ell} \rceil$ -approximation in polynomial time that matches the $m^{1/2-\epsilon}$ lower bound as $\ell \leq \min(n, m)$. Finally, we showed that MAXIMUM VERTEX-DISJOINT SHORTEST PATHS is fixed-parameter tractable when parameterized by ℓ , but it does not admit a polynomial kernel.

A way to combine approximation algorithms and the theory of (polynomial) kernels are *lossy kernels* [26]. Since the exact definition is quite technical and not relevant for this work, we only give an intuitive description. An α -approximate kernel or lossy kernel for an optimization problem is a pair of algorithms that run in polynomial time which are called *pre-processing algorithm* and *solution-lifting algorithm*. The pre-processing algorithm takes as input an instance (I, ρ) of a parameterized problem P and outputs an instance (I', ρ') of P such that $|I'| + \rho' \leq g(\rho)$ for some computable function g . The solution-lifting algorithm takes any solution S of (I', ρ') and transforms it into a solution S^* of (I, ρ) such that if S is a γ -approximation for (I', ρ') for some $\gamma \geq 1$, then S^* is an $\gamma \cdot \alpha$ -approximation for (I, ρ) . If the size of the kernel is $g(\rho)$ and if g is constant or a polynomial, then we call it a constant-size or polynomial-size α -approximate kernel, respectively. It is known that a (decidable) parameterized problem admits a constant-size approximate α -kernel if and only if the unparameterized problem associated with P can be α -approximated (in polynomial time) [26]. Moreover, any (decidable) parameterized problem admits an α -approximate kernel (of arbitrary size) if and only if the problem can be α -approximated in $f(\rho) \cdot \text{poly}(|I|)$ time.

In terms of lossy kernelization, our results imply that there are no non-trivial lossy kernels for the parameter k . For the parameter ℓ , Theorem 4 implies a constant-size lossy kernel for $\alpha \in \Omega(\sqrt{\ell})$ and Theorem 5 implies an $f(\ell)$ -size lossy kernels for any $\alpha \geq 1$. This leaves the following gap which we pose as an open problems.



► **Open Problem 1.** *Are there any poly(ℓ)-size lossy kernels for MAXIMUM VERTEX-DISJOINT SHORTEST PATHS with $\alpha \in o(\sqrt{\ell})$ (or even constant α)?*

References

- 1 Shyan Akmal, Virginia Vassilevska Williams, and Nicole Wein. Detecting disjoint shortest paths in linear time and more. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming, ICALP*, pages 9:1–9:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.9.
- 2 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
- 3 Matthias Bentert, André Nichterlein, Malte Renken, and Philipp Zschoche. Using a geometric lens to find k -disjoint shortest paths. *SIAM Journal on Discrete Mathematics*, 37(3):1674–1703, 2023.
- 4 Kristof Berczi and Yusuke Kobayashi. The directed disjoint shortest paths problem. In *Proceedings of the 25th Annual European Symposium on Algorithms (ESA)*, pages 13:1–13:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ESA.2017.13.
- 5 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.
- 6 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM Journal on Computing*, 49(4):772–810, 2020. doi:10.1137/18M1166869.
- 7 Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. Simple combinatorial construction of the $k^{o(1)}$ -lower bound for approximating the parameterized k -clique. *CoRR*, abs/2304.07516, 2023. doi:10.48550/arXiv.2304.07516.
- 8 Rajesh Chitnis, Samuel Thomas, and Anthony Wirth. Lower bounds for approximate (& exact) k -disjoint-shortest-paths. In *Proceedings of the 22nd International Workshop on Approximation and Online Algorithms (WAOA)*, 2024. Accepted for Publication.
- 9 Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Almost polynomial hardness of node-disjoint paths in grids. *Theory of Computing*, 17:1–57, 2021. doi:10.4086/TOC.2021.V017A006.
- 10 Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. New hardness results for routing on disjoint paths. *SIAM Journal on Computing*, 51(2):17–189, 2022. doi:10.1137/17M1146580.
- 11 Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. *Journal of the ACM*, 63(5):45:1–45:51, 2016. doi:10.1145/2893472.
- 12 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.
- 13 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity*, 2016.
- 14 Tali Eilam-Tzoref. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998. doi:10.1016/S0166-218X(97)00121-2.
- 15 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization*. Cambridge University Press, Cambridge, 2019.
- 16 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

- 17 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 18 Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, F. Bruce Shepherd, and Mihalis Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67(3):473–496, 2003. doi:10.1016/S0022-0000(03)00066-7.
- 19 Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 627–636. IEEE Computer Society, 1996.
- 20 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k -sat. *Journal on Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/JCSS.2000.1727.
- 21 Richard M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. doi:10.1002/NET.1975.5.1.45.
- 22 Karthik C. S. and Subhash Khot. Almost polynomial factor inapproximability for parameterized k -clique. In *Proceedings of the 37th Computational Complexity Conference (CCC)*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CCC.2022.6.
- 23 Jon M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Massachusetts Institute of Technology, 1996.
- 24 Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004. doi:10.1007/S10107-002-0370-6.
- 25 William Lochet. A polynomial time algorithm for the k -disjoint shortest paths problem. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 169–178. SIAM, 2021. doi:10.1137/1.9781611976465.12.
- 26 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 224–237. ACM, 2017. doi:10.1145/3055399.3055456.
- 27 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense CSPs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ICALP.2017.78.
- 28 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 182–191. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492475.
- 29 Neil Robertson and Paul D. Seymour. Graph minors. XIII: The disjoint paths problem. *Journal of Combinatorial Theory. Series B*, 63(1):65–110, 1995. doi:10.1006/JCTB.1995.1006.
- 30 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007. doi:10.4086/TOC.2007.V003A006.

Online Disjoint Set Covers: Randomization Is Not Necessary

Marcin Bienkowski  

University of Wrocław, Poland

Jarosław Byrka  

University of Wrocław, Poland

Łukasz Jeż  

University of Wrocław, Poland

Abstract

In the online disjoint set covers problem, the edges of a hypergraph are revealed online, and the goal is to partition them into a maximum number of disjoint set covers. That is, n nodes of a hypergraph are given at the beginning, and then a sequence of hyperedges (subsets of $[n]$) is presented to an algorithm. For each hyperedge, an online algorithm must assign a color (an integer). Once an input terminates, the gain of the algorithm is the number of colors that correspond to valid set covers (i.e., the union of hyperedges that have that color contains all n nodes).

We present a deterministic online algorithm that is $O(\log^2 n)$ -competitive, exponentially improving on the previous bound of $O(n)$ and matching the performance of the best randomized algorithm by Emek et al. [ESA 2019].

For color selection, our algorithm uses a novel potential function, which can be seen as an online counterpart of the derandomization method of conditional probabilities and pessimistic estimators. There are only a few cases where derandomization has been successfully used in the field of online algorithms. In contrast to previous approaches, our result extends to the following new challenges: (i) the potential function derandomizes not only the Chernoff bound, but also the coupon collector's problem, (ii) the value of OPT of the maximization problem is not bounded a priori, and (iii) we do not produce a fractional solution first, but work directly on the input.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Disjoint Set Covers, Derandomization, pessimistic Estimator, potential Function, online Algorithms, competitive Analysis

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.18

Funding Supported by Polish National Science Centre grants 2022/45/B/ST6/00559, 2020/39/B/ST6/01641, and 2020/39/B/ST6/01679.

Acknowledgements The authors are grateful to the anonymous reviewers for their insightful comments.

1 Introduction

In this paper, we study online algorithms for maximizing the number of set covers of a set of nodes. We focus on a hypergraph (set system) $G = (V, E)$ that has $n = |V|$ nodes and where each hyperedge $S \in E$ is a subset of nodes from V . A subset $E' \subseteq E$ is called *set cover* if $\bigcup_{S \in E'} S = V$, i.e., every node is covered by at least one hyperedge of E' . In the *disjoint set covers* (DSC) problem [15, 12, 20], the goal is to partition the set of hyperedges E into *maximum number* of mutually disjoint subsets $E = E_1 \uplus E_2 \uplus \dots \uplus E_k$, where each E_j is a set cover. Note that E is a multi-set, i.e., it can contain multiple copies of the same hyperedge.

The problem has been studied in a theoretical setting, but as we discuss later, it also finds applications in sensor networks [12] or assignment tasks [20].



© Marcin Bienkowski, Jarosław Byrka, and Łukasz Jeż;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 18; pp. 18:1–18:16



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Coloring Perspective. When constructing a solution to the DSC problem, it is convenient to think in terms of coloring hyperedges.¹ Each color then corresponds to a subset of hyperedges that have that color, and the color is *fully used* if the hyperedges colored with it form a set cover. The problem then becomes equivalent to coloring hyperedges, so that the number of fully used colors is maximized.

Online Variant. In this paper, we focus on the *online* variant of the DSC problem, where the set V is given in advance, but the hyperedges of E arrive in an online fashion. Once a hyperedge $S \in E$ arrives, it must be colored immediately and irrevocably. Again, the goal is to maximize the number of fully used colors. The performance of an online algorithm is measured by the competitive ratio, i.e., the ratio of the number of fully used colors produced by the optimal *offline* algorithm OPT to that of an online algorithm ALG .

Our Contribution. We present a deterministic online $O(\log^2 n)$ -competitive algorithm DET for the DSC problem, which exponentially improves on the $O(n)$ -competitive algorithm by Emek et al. [14] and matches the performance of their randomized algorithm [14]. We discuss the challenges and technical contribution in more detail in Subsection 1.3.

1.1 Offline Scenario: Previous Results

The disjoint set covers problem is a fundamental NP-complete problem [12] that can be approximated within a factor of $(1 + o(1)) \cdot \ln |V|$ [20] and cannot be approximated within a factor of $(1 - \varepsilon) \cdot \ln |V|$ for $\varepsilon > 0$ unless $\text{NP} \subseteq \text{DTIME}(n^{\log \log n})$ [15].²

OPT vs. Min-degree. We use $\text{OPT}(E)$ to denote the maximum number of disjoint set covers of a hypergraph $G = (V, E)$. This value is also called *cover-decomposition number* [6]. We denote the minimum degree of the hypergraph $G = (V, E)$ as $\delta(E) \triangleq \min_{i \in V} |S \in E : i \in S|$. Note that trivially $\text{OPT}(E) \leq \delta(E)$. While this bound may not be tight (cf. Subsection 1.4), $\delta(E)$ serves as a natural benchmark for approximation and online algorithms.

Random Coloring and its Straightforward Derandomization. An $O(\log n)$ -approximation algorithm for the *offline* DSC problem [21] can be obtained by coloring each hyperedge with a color chosen uniformly at random from the palette of $\Theta(\delta(E)/\log n)$ colors. To analyze this approach, we focus on a single node $i \in V$. We say that node i *gathers* color r if i is contained in a hyperedge colored with r . Since node i is contained in at least $\delta(E)$ hyperedges, and there are $\Theta(\delta(E)/\log n)$ available colors, i gathers all palette colors with high probability. By the union bound, this property holds for all nodes, i.e., all $\Theta(\delta(E)/\log n)$ colors are fully used by an algorithm (also with high probability). Since $\text{OPT}(E) \leq \delta(E)$, the approximation ratio of $O(\log n)$ follows.

The hyperedges can be processed in a fixed order, and the random choice of a color can be replaced by the deterministic one by a simple application of the method of conditional probabilities [21, 3].

¹ The DSC problem should not be confused with the hyperedge coloring problem, which requires that the hyperedges of the same color be disjoint.

² The authors of [15] call this problem *set cover packing* or *one-sided domatic problem*.

1.2 Online Scenario: Previous Results

In the online case, an algorithm first learns the set of nodes V , and then the edges of E are revealed sequentially. For notational convenience, we use E to denote both the set and a sequence of hyperedges (the input). We use $\text{ALG}(E)$ to denote the number of fully used colors in the solution of an online algorithm ALG .

It is important to note that no parameter of the hypergraph other than the number of nodes is known a priori. In particular, an online algorithm does not know the value of $\delta(E)$ in advance.

Competitive Ratio. We say that ALG is (*non-strictly*) c -competitive if there exists $\beta \geq 0$ such that

$$\text{ALG}(E) \geq \text{OPT}(E) / c - \beta. \quad (1)$$

While β can be a function of n , it cannot depend on the input sequence E . If (1) holds with $\beta = 0$, the algorithm is *strictly* c -competitive.

For randomized algorithms, we replace $\text{ALG}(E)$ with its expected value taken over the random choices of the algorithm.

Randomized Algorithms. The current best randomized algorithm was given by Emek et al. [14]; it achieves the (strict) competitive ratio of $O(\log^2 n)$. The general idea of their algorithm is as follows. To color a hyperedge S , their algorithm first computes the minimum degree δ^* of a node from S . By a combinatorial argument, they show that they can temporarily assume that $\delta(E) = O(n \cdot \delta^*)$. Their algorithm then chooses ℓ uniformly at random from the set $\{\delta^*, 2\delta^*, 4\delta^*, \dots, 2^{\log n} \cdot \delta^*\}$; with probability $\Omega(1/\log n)$ such ℓ is a 2-approximation of $\delta(E)$. Finally, to color S , it chooses a color uniformly at random from the palette of $\Theta(\ell/\log^2 n)$ colors, using the arguments similar to those in the offline case.

We refrain from discussing it further here, as we present a variant of their algorithm, called RAND , in Subsection 2.1 (along with a description of the differences from their algorithm).

The best lower bound for the (strict and non-strict) competitive ratio of a randomized algorithm is $\Omega(\log n / \log \log n)$ [14]; it improves on an earlier bound of $\Omega(\sqrt{\log n})$ by Pananjady et al. [20].

Deterministic Algorithms. The deterministic case is well understood if we restrict our attention to *strict* competitive ratios. In such a case the lower bound is $\Omega(n)$ [20]. The asymptotically matching upper bound $O(n)$ is achieved by a simple greedy algorithm [14].

On the other hand, the current best lower bound for the non-strict deterministic competitive ratio is $\Omega(\log n / \log \log n)$ [14].³ The $O(n)$ upper bound of [14] clearly holds also in the non-strict setting, but no better algorithm has been known so far.

Lack of General Derandomization Tools. Unlike approximation algorithms, derandomization is extremely rare in online algorithms.⁴ To understand the difficulty, consider the standard (offline) method of conditional probabilities [3] applied to the offline DSC problem:

³ This discrepancy in achievable strict and non-strict competitive ratios is quite common for many maximization problems: the non-strict competitive ratio allows the algorithm to have zero gain on very short sequences, thus avoiding initial choices that would be bad in the long run.

⁴ Many online problems (e.g., caching [23, 1] or metrical task systems [7, 8, 9]) exhibit a provable exponential discrepancy between the performance of the best randomized and deterministic algorithms. For many other problems, the best known deterministic algorithms are quite different (and more complex) than randomized ones.

■ **Table 1** Previous and new bounds on the strict and non-strict competitive ratios for the online DSC problem, for randomized and deterministic algorithms.

	Upper bound	Lower bound
randomized strict and non-strict	$O(\log^2 n)$ [14]	$\Omega(\log n / \log \log n)$ [14]
deterministic strict	$O(n)$ [14]	$\Omega(n)$ [20]
deterministic non-strict	$O(n)$ [14] $O(\log^2 n)$ (Theorem 5)	$\Omega(\log n / \log \log n)$ [14]

the resulting algorithm considers all the random choices (color assignments) it could make for a given hyperedge S , and computes the probability that future random choices, *conditioned on the current one*, will lead to the desired solution. Choosing the color that maximizes this probability ensures that the probability of reaching the desired solution does not decrease as subsequent hyperedges are processed. In some cases, exact computations are not possible, but the algorithm can instead estimate this probability by computing a so-called *pessimistic estimator* [22]. This is not feasible in the online setting, since an algorithm does not know the future hyperedges, and thus cannot even estimate these probabilities.⁵

1.3 Our Technical Contribution

In this paper, we present a deterministic online polynomial-time algorithm DET that is $O(\log^2 n)$ -competitive, which exponentially improves the previous bound of $O(n)$. This resolves an open question posed by the authors of [14] who asked whether the method of conditional expectations could be used to derandomize their algorithm.

Our bound is obtained for the non-strict competitive ratio: as we discussed in the previous subsection, this is unavoidable for the DSC problem. Our result requires a relatively small ($1/4$) additive term β in the definition of the competitive ratio (1).

We begin by constructing a *randomized* solution RAND. It will be a variation of the approach by Emek et al. [14]; we present it in Subsection 2.1. As RAND is closely related to the previous randomized algorithm, it is quite plausible that it achieves the same competitive ratio of $O(\log^2 n)$. However, our analysis does not support this conjecture. Instead, we use RAND as a stepping stone to our deterministic algorithm in the following way.

- We define a particular random event, denoted \mathcal{T}_E , of RAND executed on instance E .
- We show that for each execution of RAND satisfying \mathcal{T}_E , it holds that $\text{RAND}(E) \geq \Omega(1/\log^2 n) \cdot \text{OPT}(E) - 1/4$.

We will provide the exact definition of the event \mathcal{T}_E in Subsection 2.2. For now, we just note that \mathcal{T}_E is a property that certifies that, *at each step t* , we can relate the number of colors gathered so far by each node i to its current degree.

While a relaxed version of the property \mathcal{T}_E (requiring that the relation between the gathered colors and the current degree *only at the end of the input*) follows quite easily with a constant probability, it seems that \mathcal{T}_E itself is quite strong and does not hold with a reasonable probability. However, this is not an obstacle to creating a deterministic solution, as we show in the following claim.

⁵ As observed by Pananjady et al. [20], however, these probabilities can be estimated if an online algorithm knows the final min-degree $\delta(E)$ in advance, which would lead to an $O(\log n)$ -competitive algorithm for this semi-offline scenario.

- It is possible to replace the random choices of RAND by deterministic ones (in an online manner), so that \mathcal{T}_E always holds.

This will immediately imply the competitive ratio of the resulting deterministic algorithm.

Our approach is based on a novel potential function that guides the choice of colors. As we discuss later in Subsection 2.2, our approach can be seen as an online-computable counterpart of the method of conditional probabilities that simultaneously controls the derandomization of the Chernoff bound and the coupon collector's problem.

1.4 Related Work

Applications of the DSC problem include allocating servers to users in file systems and users to tasks in crowd-sourcing platforms [20].

Another application arises in a sensor network, where each node corresponds to a monitoring target and each hyperedge corresponds to a sensor that can monitor a set of targets. At any given time, all targets must be monitored. A possible battery-saving strategy is to partition the sensors into disjoint groups (each group covering all targets) and activate only one group at a time, while the other sensors remain in a low-power mode [12].

When the assumption that each sensor can only participate in a single group is dropped, this leads to more general sensor coverage problems. The goal is then to maximize the lifetime of the network of sensors while maintaining the coverage of all targets. The offline variants of this problem have been studied both in general graphs [4, 13, 21] and in geometric settings [11, 16].

Another line of work studied the relationship between the minimum degree $\delta(E)$ and the cover-decomposition number $\text{OPT}(E)$. As pointed out in Subsection 1.1, $\delta(E)/\text{OPT}(E) \geq 1$. This ratio is constant if G is a graph [17], and it is at most $O(\log n)$ for every hypergraph; the latter bound is asymptotically tight [6]. Interestingly enough, all the known papers on the DSC problem (including ours) relate the number of disjoint set covers to $\delta(E)$. It is an open question whether our estimate of the competitive ratio is tight: it could possibly be improved by relating the gain of an algorithm directly to $\text{OPT}(E)$ instead of $\delta(E)$.

1.5 Preliminaries

An input to our problem is a gradually revealed hypergraph $G = (V, E)$. V consists of n nodes numbered from 1 to n , i.e., $V = [n]$. The set E of hyperedges is presented one by one: in a step t , an algorithm learns $S_t \in E$, where $S_t \subseteq [n]$, and has to color it. We say that node i *gathers* color r if i is contained in a hyperedge colored with r . We say that a color r is *fully used* if all nodes have gathered it. The objective of an algorithm is to maximize the number of fully used colors.

At any point in time, the *degree* of a node j , denoted $\text{deg}(j)$, is the current number of hyperedges containing j . Let $\delta(E) = \min_{i \in [n]} |\{S_t \in E : v_i \in S_t\}|$, i.e., $\delta(E)$ is the minimum degree of G at the end of the input E . Clearly, $\text{OPT}(E) \leq \delta(E)$. It is important to note that $\delta(E)$ is not known in advance to an online algorithm.

■ **Algorithm 1** Definition of RAND for a hyperedge S in step t .

```

1: ▷ Initialization
2: for each node  $i \in [n]$  do
3:    $p(i) \leftarrow 0$                                      ▷ all nodes start in phase 0
4:   for each integer  $k \geq 0$  do                           ▷ and they have no colors yet
5:      $C_{i,k} \leftarrow \emptyset$ 

6: ▷ Runtime
7: for each hyperedge  $S$  appearing in sequence  $E$  do
8:    $p_S \leftarrow \min_{i \in S} p(i)$ 
9:   Sample  $k^*$  from  $\{p_S, p_S + 1, \dots, p_S + h - 1\}$    ▷ uniform distribution
10:  Sample color  $r$  from  $R_{k^*}$                                ▷ uniform distribution
11:  for each node  $i \in S$  do
12:     $C_{i,k^*} \leftarrow C_{i,k^*} \cup \{r\}$ 
13:    if  $|C_{i,p(i)}| \geq q_{p(i)}$  then                       ▷ end node phase if it gathered  $q_{p(i)}$  colors
14:       $p(i) \leftarrow p(i) + 1$ 

```

2 Our Algorithm

2.1 Definition of RAND

We start with some notions, defined for each integer $k \geq 0$:

$$h \triangleq \lceil \log n \rceil, \quad q_k \triangleq \left\lceil \left(1 - \frac{1}{2n}\right) \cdot 2^k \right\rceil, \quad R_k \triangleq \{2^k, \dots, 2^{k+1} - 1\}.$$

Note that $|R_k| = 2^k$.

For each node i independently, RAND maintains its *phase* $p(i)$, initially set to 0. We use $C_{i,k}$ to denote the set of colors from the palette R_k that node i has gathered so far. A phase k for node i ends when it has gathered q_k colors from palette R_k . In such a case, node i increments its phase number $p(i)$ at the end of the step.

We now describe the behavior of RAND in a single step, when a hyperedge S appears. Let $p_S = \min_{i \in S} p(i)$, where $p(i)$ is the phase of node i before S appears. RAND first chooses a random integer k^* uniformly from the set $\{p_S, p_S + 1, \dots, p_S + h - 1\}$. Second, RAND chooses a color r uniformly at random from the set R_{k^*} and colors S with it; in effect all nodes in S gather color r .

The pseudocode of RAND is given in Algorithm 1.

Comparison with the Previous Randomized Algorithm. RAND is closely related to the randomized algorithm by Emek et al. [14]. The main difference is that the phases of the nodes in their algorithm are of fixed lengths, being powers of 2.⁶ They use probabilistic arguments to argue that with high probability each node gathers q_k colors in a phase of length $\Theta(2^k \cdot \log^2 n)$. Instead, we treat the number of colors gathered in a phase as a hard constraint (we require that each node gathers q_k of them), and instead the phase lengths of the nodes become random variables. As it turns out, this subtle difference allows us to derandomize the algorithm.

⁶ The pseudocode of their algorithm does not use phases, but with some fiddling with constants, it can be transformed into one that does.

Another small difference concerns the color selection. While RAND chooses the color uniformly at random from the set R_{k^*} , the algorithm of [14] would choose it from the set $\uplus_{j=0}^{k^*} R_j$. This difference affects only the constant factors of the analysis.

Gain of RAND. As mentioned earlier, the gain of RAND is directly ensured by the algorithm definition *provided* that each node has completed some number of phases.

Note that q_k is slightly less than $|R_k| = 2^k$; this gives a better bound on the the expected phase lengths, while still ensuring that if all nodes completed phase ℓ , they gathered at least $2^{\ell-1}$ common colors.

► **Lemma 1.** *If every node has completed phase ℓ , then $RAND(E) \geq 2^{\ell-1}$.*

Proof. In phase ℓ , each node gathers at least q_ℓ colors from the palette R_ℓ , i.e., all colors from R_ℓ except at most $|R_\ell| - q_\ell \leq 2^\ell / (2n)$ colors. Thus, all nodes share at least $|R_\ell| - n \cdot (2^\ell / (2n)) = 2^{\ell-1}$ common colors from R_ℓ , i.e., RAND fully uses at least $2^{\ell-1}$ colors. ◀

Note that we could sum the above bound over all phases completed by all nodes, but this would not change the asymptotic gain.

The above lemma points us to the goal: to show that for an input E with a sufficiently large $\delta(E)$, each node completes an appropriately large number of phases. In Subsection 2.3, we show that if $\delta(E) = \Omega(2^\ell \cdot \log^2 n)$, then each node completes ℓ phases, *provided* a certain random event \mathcal{T}_E occurs.

2.2 Constructing the Potential: Insights and Definitions

In the field of online algorithms, the derandomization has been successfully conveyed several times by replacing the method of conditional probabilities by an *online-computable potential function* that guides the choice of the deterministic algorithm [2, 5, 10, 18].

This method is based on the following framework. First, introduce ℓ random expressions $\{Z_i\}_{i=1}^\ell$ to be controlled. Second, define a potential function $\Phi = \sum_{i=1}^\ell \exp(Z_i)$. Third, show that the random actions of an algorithm at each step decrease $\exp(Z_i)$ (for each i) in expectation, which implies that Φ decreases as well. (By the probabilistic method, this implies the existence of a *deterministic action* of an algorithm in a step that does not increase Φ .) Finally, by the non-negativity of the exponential function, $\exp(Z_i)$ is bounded by the initial value Φ^0 of the potential (in each step), which shows that Z_i can always be bounded by $\ln(\Phi^0)$. This process can be seen as a derandomization of the Chernoff bound / high probability argument.

In order to apply this very general framework, we must overcome several technical difficulties, which we explain below. Except for the definition of Φ (and related definitions of w_{ik} , c_{ik} , d_k and Z_i), the discussion in this section is informal and will not be used in formal proofs later.

Variables to be Controlled. The first step is to identify the variables to control. Natural choices are the node degrees and the number of colors gathered so far by each node.

For this purpose, we define $c_{ik} = |C_{ik}|$ for each node i and each phase $k \geq 0$.

We also introduce counters w_{ik} , which are initially set to zero. Recall that whenever a new hyperedge S containing node i arrives, RAND computes $p_S = \min_{i \in S} p(i)$. For each node $i \in S$ such that $p(i) \leq p_S + h - 1$, we increment the counter $w_{i,p(i)}$. Note that these $w_{i,p(i)}$ variables are incremented exactly for nodes i for which there is a non-zero probability of increasing their set of colors $C_{i,p(i)}$ (as then a random integer k^* chosen by RAND has a non-zero chance of being equal to $p(i)$).

By the definition of w_{ik} , at each step $\sum_{k \geq 0} w_{ik} \leq \deg(i)$. While these quantities are not necessarily equal, we will treat $\sum_{k \geq 0} w_{ik}$ as a good proxy for $\deg(i)$ and deal with the discrepancy between these two quantities later.

Linking the Variables. Now we want to introduce an expression that links $\sum_{k \geq 0} w_{ik}$ (the proxy for degree) with $\sum_{k \geq 0} c_{ik}$ (the number of colors gathered) for a node i . A simple difference of these two terms does not make sense: the expected growth of c_{ik} varies over time, since it is easier to gather new colors when c_{ik} is small. This effect has been studied in the context of the coupon collector's problem [19]. To overcome this issue, we introduce the following function, defined for each integer $k \geq 0$:

$$d_k(m) \triangleq h \cdot \sum_{j=1}^m \frac{2^k}{2^k - j + 1} \quad (\text{defined for each } m \leq 2^k)$$

Note that in a process of choosing random colors from palette R_k of 2^k colors, the expected number of steps till m different colors are gathered is $d_k(m)/h$.

Now we focus on a single node i in phase $p(i)$. We consider a sequence of hyperedges S containing node i , neglecting those hyperedges S for which $p(i) \geq p_S + h$. That is, all considered hyperedges increment the counter $w_{i,p(i)}$. Then, the value of $d_k(c_{i,p(i)})$ corresponds to the expected number of such hyperedges needed to gather $c_{i,p(i)}$ colors from the palette $R_{p(i)}$. We can thus use the expression $\sum_{k \geq 0} (w_{ik} - 2 \cdot d_k(c_{ik}))$ to measure the progress of node i : small values of this expression indicate that it is gathering colors quite fast, while large values indicate that it is falling behind. Note that since $c_{ik} \leq q_k \leq 2^k$, the value of $d_k(c_{ik})$ is always well defined.

We note that the previous applications of the potential function method [2, 5, 10, 18] did not require such transformations of variables: in their case, the potential function was used to guide deterministic *rounding*: the function Φ directly compared the cost (or gain) of a deterministic algorithm with that of an online fractional solution. Instead, our solution operates directly on the input, without the need to generate a fractional solution first.

Scaling. Using the random choices of RAND, we can argue that in expectation the value of $Z_i^* \triangleq \sum_{k \geq 0} (w_{ik} - 2 \cdot d_k(c_{ik}))$ is decreasing in time. However, to argue that $\exp(Z_i^*)$ is also decreasing in expectation, we would have to ensure that Z_i^* is upper-bounded by a small constant (and use the fact that $\exp(x)$ can be approximated by a linear function for small x).

In the previous papers [2, 5, 10, 18], this property was achieved by scaling down Z^* by the value of OPT. An algorithm was then either given an upper bound on OPT (in the case of the throughput maximization of the virtual circuit routing [10]) or OPT was estimated by standard doubling techniques (in the case of the cost minimization for set cover variants [2, 5, 18]). In the latter case, the algorithm was restarted each time the estimate on OPT was doubled. Unfortunately, the DSC problem (which is an unbounded-gain maximization problem) does not fall into any of the above categories, and the doubling approach does not seem to work here.

Instead, we replace the scaling with a weighted average. That is, for each node i , we define

$$Z_i \triangleq \sum_{k \geq 0} \frac{w_{ik} - 2 \cdot d_k(c_{ik})}{4h \cdot 2^k}$$

and the potential as

$$\Phi \triangleq \sum_{i \in [n]} \exp(Z_i).$$

Note that all counters and variables defined above are random variables; they depend on particular random choices of RAND. We use $p^t(i)$, $\deg^t(i)$, w_{ik}^t , c_{ik}^t , Z_i^t and Φ^t to denote the values of the corresponding variables at the end of step t of the algorithm (after RAND has processed the hyperedge presented in step t). The value of $t = 0$ corresponds to the state of these variables at the beginning of the algorithm; note that $p^0(i) = \deg^0(i) = w_{ik}^0 = c_{ik}^0 = Z_i^0 = 0$ for all i and k . Therefore,

$$\Phi^0 = n. \tag{2}$$

Random event \mathcal{T}_E . For an input instance E consisting of T steps, we define a random event \mathcal{T}_E that occurs if $\Phi^t \leq n$ for each step $t \in \{0, \dots, T\}$ of RAND execution on input E .

2.3 Relating Potential to Algorithm Performance

We begin by presenting the usefulness of the event \mathcal{T}_E . We emphasize that the following lemma holds for all executions of RAND in which the event \mathcal{T}_E occurs. Its proof is deferred to Section 3.

► **Lemma 2.** *Fix a sequence E such that $\delta(E) > 24h \cdot \ln(4e \cdot n) \cdot 2^\ell$ for some integer $\ell \geq 0$. If \mathcal{T}_E occurs, then each node has completed its phase ℓ .*

Using the lemma above, we can relate the gain of RAND on an arbitrary input E to that of OPT, if only \mathcal{T}_E occurs.

► **Lemma 3.** *Fix a sequence E . If \mathcal{T}_E occurs, then $\text{RAND}(E) \geq \text{OPT}(E)/(96h \cdot \ln(4e \cdot n)) - 1/4$.*

Proof. Let $r \triangleq 24h \cdot \ln(4e \cdot n)$. We consider two cases.

- First, we assume that $\delta(E) > r$. Then we can find an integer $\ell \geq 0$ such that $r \cdot 2^\ell < \delta(E) \leq r \cdot 2^{\ell+1}$. By Lemma 2, each node then completes its phase ℓ , and so by Lemma 1, $\text{RAND}(E) \geq 2^{\ell-1} \geq \delta(E)/(4r)$.
- Second, we assume that $\delta(E) \leq r$. Trivially, $\text{RAND}(E) \geq 0 \geq (\delta(E) - r)/(4r)$.

In both cases, $\text{RAND}(E) \geq (\delta(E) - r)/(4r) \geq \text{OPT}(E)/(4r) - 1/4$. ◀

2.4 Derandomization of RAND

To analyze the evolution of Φ , we note that Φ^t (and also other variables w_{ik}^t , c_{ik}^t or Z_i^t) depends only on the random choices of RAND till step t (inclusively). Moreover, $\{\Phi^t\}_{t \geq 0}$ is a supermartingale with respect to the random choices of RAND in consecutive steps. Specifically, the following lemma holds; its proof is deferred to Section 4.

► **Lemma 4.** *Fix a step t and an outcome of random choices till step $t - 1$ inclusively. (In particular, this will fix the value of Φ^{t-1} .) Then, $\mathbb{E}[\Phi^t] \leq \Phi^{t-1}$, where the expectation is taken exclusively over random choices of RAND in step t .*

The above lemma states that the value of Φ is non-increasing in expectation. In fact, an inductive application of this lemma shows that $\mathbb{E}[\Phi^t] \leq \Phi^0 = n$. However, this does not imply that \mathcal{T}_E occurs with a reasonable probability, especially when the input length is large.⁷

⁷ By Markov's inequality, for a chosen step t , $\Phi^t \leq 2n$ holds with probability at least $1/2$. While such a relaxed bound on Φ^t would be sufficient for our needs, in our proof, we need such a bound to hold for all steps t simultaneously.

18:10 Online Disjoint Set Covers: Randomization Is Not Necessary

However, using Lemma 4, we can easily derandomize RAND using the method of conditional probabilities using potential Φ as an online-computable counterpart of a pessimistic estimator. To do this, we proceed iteratively, ensuring at each step t that $\Phi^t \leq n$. This is trivial at the beginning, since $\Phi^0 = n$ by (2).

Suppose we have already fixed the random choices of RAND till step $t - 1$ inclusively and have $\Phi^{t-1} \leq n$. Consider a hyperedge S presented in step t . Note that all other variables indexed by $t - 1$, such as $p^{t-1}(i)$, are also fixed. Then Lemma 4 states that $\mathbb{E}[\Phi^t] \leq \Phi^{t-1} \leq n$. That is, the random choice of a color at step t guarantees that $\mathbb{E}[\Phi^t] \leq n$. This choice is made from a finite and well-defined set of colors $\mathcal{R} \triangleq \biguplus_{p_S \leq k \leq p_S + h - 1} R_k$, where $p_S = \min_{i \in S} p^{t-1}(i)$.

By the probabilistic method, at each step t , there exists a *deterministic* choice of a color from \mathcal{R} that ensures that $\Phi^t \leq n$. The resulting algorithm is called DET. (If more than one color leads to $\Phi^t \leq n$, DET chooses any of them.) Since $|\mathcal{R}|$ is bounded by a polynomial of n and $|E|$, DET can be implemented in polynomial time by simply checking all possible colors of \mathcal{R} .

► **Theorem 5.** *DET is $O(\log^2 n)$ -competitive for the DSC problem.*

Proof. As described above, DET guarantees that $\Phi^t \leq n$ for each step $t \in \{0, \dots, T\}$, i.e., \mathcal{T}_E occurs. Note that Lemma 3 lower-bounds the gain of RAND in every execution conditioned on \mathcal{T}_E , and DET can be seen as such an execution. Therefore, the bound of Lemma 3 can be applied, yielding

$$\text{DET}(E) \geq \frac{\text{OPT}(E)}{96 \cdot h \cdot \ln(4e \cdot n)} - \frac{1}{4},$$

i.e., the competitive ratio of DET is at most $96 \cdot h \cdot \ln(4e \cdot n) = O(\log^2 n)$.

Note that, by the lower bounds of [20, 14], an additive term (in our case equal to $1/4$) is inevitable for obtaining a sub-linear competitive ratio. ◀

3 Bounding Number of Phases

In this section, we fix an input sequence E consisting of T steps. Our goal is to estimate the number of phases of nodes in the execution of RAND, conditioned on the random event \mathcal{T}_E , i.e., to prove Lemma 2. To this end, we consider a node i , assume that it has completed ℓ phases, and show an upper bound on the final degree of i as a function of ℓ .

Bounding Variables w Using Potential. Recall that in some steps where the degree of a node i grows, the counter $w_{i,p(i)}$ is incremented. Thus, our first goal is to upper-bound values of these counters at the end of the execution of RAND.

Below, $H(m)$ denotes the m -th harmonic number. The following technical claim is proved in Appendix A.

▷ **Claim 6.** For each $k \geq 0$ it holds that $H(2^k) - H(2^k - q_k) \leq \ln(4e \cdot n)$.

► **Lemma 7.** Fix a step $t \leq T$, a node i and a phase $k \geq 0$. Then, $d_k(c_{ik}^t) \leq h \cdot \ln(4e \cdot n) \cdot 2^k$.

Proof. Note that at all times, $c_{ik}^t \leq q_k$. Using the definition of d_k ,

$$\begin{aligned} d_k(c_{ik}^t) &\leq d_k(q_k) = h \cdot \sum_{j=1}^{q_k} \frac{2^k}{2^k - j + 1} \\ &= h \cdot (H(2^k) - H(2^k - q_k)) \cdot 2^k \\ &\leq h \cdot \ln(4e \cdot n) \cdot 2^k. \end{aligned} \quad \text{(by Claim 6)} \quad \blacktriangleleft$$

► **Lemma 8.** *Fix a node i and a phase $\ell \geq 0$. If the event \mathcal{T}_E occurs, then $\sum_{k=0}^{\ell} w_{ik}^T \leq 8h \cdot \ln(4e \cdot n) \cdot 2^\ell$.*

Proof. Fix the last step $t \leq T$ in which $\sum_{k=0}^{\ell} w_{ik}$ increases. By the choice of t , we have $w_{ik}^t = c_{ik}^t = 0$ for every phase $k > \ell$.

Since \mathcal{T}_E occurs, $n \geq \Phi^t = \sum_{j \in [n]} \exp(Z_j^t)$. Due to the non-negativity and monotonicity of the exponential function, $Z_i^t \leq \ln n$. Using the definition of Z_i^t , we get

$$\begin{aligned} \ln n \geq Z_i^t &= \sum_{k \geq 0} \frac{w_{ik}^t - 2 \cdot d_k(c_{ik}^t)}{4h \cdot 2^k} \\ &= \sum_{k=0}^{\ell} \frac{w_{ik}^t - 2 \cdot d_k(c_{ik}^t)}{4h \cdot 2^k} \\ &\geq \frac{1}{4h \cdot 2^\ell} \cdot \left(\sum_{k=0}^{\ell} w_{ik}^t - 2 \cdot \sum_{k=0}^{\ell} h \cdot \ln(4e \cdot n) \cdot 2^k \right). \quad (\text{by Lemma 7}) \end{aligned}$$

Hence, again by the choice of t ,

$$\sum_{k=0}^{\ell} w_{ik}^T = \sum_{k=0}^{\ell} w_{ik}^t \leq 4h \cdot 2^\ell \cdot \ln n + 4h \cdot \ln(4e \cdot n) \cdot 2^\ell < 8h \cdot \ln(4e \cdot n) \cdot 2^\ell. \quad \blacktriangleleft$$

Bounding Node Degrees. Recall that whenever a new hyperedge S containing node i arrives, $p_S = \min_{i \in S} p(i)$ is determined. Then, for each node $i \in S$, if $p(i) \leq p_S + h - 1$, the counter $w_{i,p(i)}$ is incremented. If $p(i) \geq p_S + h$, the degree of i grows, but $w_{i,p(i)}$ is not incremented. To estimate the degree of i , we therefore introduce the counters s_{ik} , which are incremented in the latter case. That is, for each node $i \in S$, we always have $\sum_{k \geq 0} (w_{ik} + s_{ik}) = \deg(i)$.

The growth of the variables s_{ik} is not controlled by the potential, but they grow only for nodes whose degree is very high compared to the current minimum degree. By constructing an appropriate charging argument, we can link their growth to the growth of other variables w_{ik} .

► **Lemma 9.** *Fix a step $t \leq T$, a node i , and a phase $\ell \geq 0$. Then, $s_{i\ell}^t \leq \sum_{j \in [n]} \sum_{r=0}^{\ell-h} w_{jr}^t$.*

Proof. We fix node i , phase $\ell \geq 0$, and show the lemma by induction on t . The inductive basis is trivial, as $s_{i\ell}^0 = 0 = \sum_{j \in [n]} \sum_{r=0}^{\ell-h} w_{jr}^0$.

Now suppose that the lemma statement holds for step $t - 1$. We look at how both sides of the inequality change as we increase the step superscripts from $t - 1$ to t , and argue that the increase of the right hand side is at least as large as the increase of the left hand side. If $s_{i\ell}$ does not change, the inductive claim follows trivially. Otherwise, $s_{i\ell}$ is incremented by 1. This happens only if $\ell = p(i)$, $i \in S$, and $\ell \geq p_S + h$. By the definition of p_S , this means that there exists at least one node $j^* \in S$ such that $p(j^*) = p_S$, and the corresponding counter w_{j^*,p_S} is also incremented. This means that the right hand side of the lemma inequality is incremented by at least $\sum_{j \in [n]} \sum_{r=0}^{\ell-h} (w_{jr}^t - w_{jr}^{t-1}) \geq w_{j^*,p_S}^t - w_{j^*,p_S}^{t-1} = 1$, and the inductive claim follows. \blacktriangleleft

► **Lemma 10.** *Fix a node i and a phase $\ell \geq 0$. If the event \mathcal{T}_E occurs, then $\sum_{k=0}^{\ell} s_{ik}^T \leq 16h \cdot \ln(4e \cdot n) \cdot 2^\ell$.*

18:12 Online Disjoint Set Covers: Randomization Is Not Necessary

Proof. By Lemma 9,

$$\begin{aligned} s_{ik}^T &\leq \sum_{j \in [n]} \sum_{r=0}^{k-h} w_{jr}^T \leq n \cdot 8h \cdot \ln(4e \cdot n) \cdot 2^{k-h} && \text{(by Lemma 8)} \\ &\leq 8h \cdot \ln(4e \cdot n) \cdot 2^k. && \text{(as } h = \lceil \log n \rceil) \end{aligned}$$

Hence, $\sum_{k=0}^{\ell} s_{ik}^T < 16h \cdot \ln(4e \cdot n) \cdot 2^{\ell}$. \blacktriangleleft

Finally, we can show Lemma 2, restated below.

► **Lemma 2.** *Fix a sequence E such that $\delta(E) > 24h \cdot \ln(4e \cdot n) \cdot 2^{\ell}$ for some integer $\ell \geq 0$. If \mathcal{T}_E occurs, then each node has completed its phase ℓ .*

Proof. Suppose for a contradiction that there exists a node i for which $p(i) \leq \ell$ at the end of the input. Then,

$$\delta(E) \leq \deg^T(i) = \sum_{k \geq 0} (w_{ik}^T + s_{ik}^T) = \sum_{k=0}^{\ell} (w_{ik}^T + s_{ik}^T) \leq 24h \cdot \ln(4e \cdot n) \cdot 2^{\ell},$$

where the last inequality follows by Lemma 8 and Lemma 10. This would contradict the assumption of the lemma. \blacktriangleleft

4 Controlling the Potential

In this section, we show that $\{\Phi^t\}_{t \geq 0}$ is a supermartingale with respect to the choices of RAND made in corresponding steps, i.e., we prove Lemma 4.

Throughout this section, we focus on a single step t , in which RAND processes a hyperedge S . Recall that RAND first chooses a random integer k^* uniformly from the set $\{p_S, p_S + 1, \dots, p_S + h - 1\}$. Second, conditioned on the first choice, it chooses a random color uniformly from the set R_{k^*} .

By the definition of our variables, for each node i and each integer $k \geq 0$, it holds that $w_{ik}^t - w_{ik}^{t-1} \in \{0, 1\}$ and $c_{ik}^t - c_{ik}^{t-1} \in \{0, 1\}$.

► **Lemma 11.** *Fix a node $i \in S$ and let $p = p^{t-1}(i)$. If $p \leq p_S + h - 1$, then $c_{ip}^t = c_{ip}^{t-1} + 1$ with probability $(2^p - c_{ip}^{t-1})/(h \cdot 2^p)$.*

Proof. By the definition of p_S , we have $p \geq p_S$. Combining this with the lemma assumption, we get $p \in \{p_S, \dots, p_S + h - 1\}$.

Note that $c_{ip}^t = c_{ip}^{t-1} + 1$ when node i gathers a new color from R_p , and $c_{ip}^t = c_{ip}^{t-1}$ otherwise. For a node i to gather a new color from R_p , first the integer k^* chosen randomly from the set $\{p_S, \dots, p_S + h - 1\}$ must be equal to p , which happens with probability $1/h$. Second, conditioned on the former event, a color chosen randomly from the palette R_p must be different from all c_{ip}^{t-1} colors from R_p gathered so far by node i , which happens with probability $(|R_p| - c_{ip}^{t-1})/|R_p| = (2^p - c_{ip}^{t-1})/2^p$. Hence, the probability of gathering a new color is $(2^p - c_{ip}^{t-1})/(h \cdot 2^p)$. \blacktriangleleft

We emphasize that the relations involving random variables in the following lemma (e.g., the statements such as $Z_i^t \leq Z_i^{t-1}$) hold for all random choices made by RAND.

► **Lemma 12.** Fix a node i . Let $p = p^{t-1}(i)$. Then, either $Z_i^t \leq Z_i^{t-1}$ or

$$Z_i^t \leq Z_i^{t-1} + \frac{1}{4h \cdot 2^p} + \begin{cases} -1/(2h \cdot 2^p \cdot \alpha) & \text{with probability } \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

where $\alpha = (2^p - c_{ip}^{t-1})/(h \cdot 2^p)$. The probability is computed exclusively with respect to random choices of RAND in step t .

Proof. For brevity, for an integer $k \geq 0$, we define

$$\begin{aligned} \Delta w_{ik} &= w_{ik}^t - w_{ik}^{t-1}, \\ \Delta d_k(c_{ik}) &= d_k(c_{ik}^t) - d_k(c_{ik}^{t-1}). \end{aligned}$$

As $c_{ik}^t \geq c_{ik}^{t-1}$ and d_k is a non-decreasing function, we have $\Delta d_k(c_{ik}) \geq 0$.

Let S be the hyperedge presented in step t . By the definition of the variables w_{ik} (cf. Subsection 2.2), we have

$$\Delta w_{ik} = \begin{cases} 1 & \text{if } i \in S \text{ and } k = p \text{ and } p \leq p_S + h - 1, \\ 0 & \text{otherwise.} \end{cases}$$

Now we consider two cases.

■ It holds that $i \notin S$ or $p \geq p_S + h$. Then,

$$Z_i^t - Z_i^{t-1} = \sum_{k \geq 0} \frac{\Delta w_{ik} - 2 \cdot \Delta d_k(c_{ik})}{4h \cdot 2^k} \leq \sum_{k \geq 0} \frac{\Delta w_{ik}}{4h \cdot 2^k} = 0.$$

■ It holds that $i \in S$ and $p \geq p_S + h - 1$. Then, $\Delta w_{ip} = 1$ and $\Delta w_{ik} = 0$ for $k \neq p$. Therefore,

$$\begin{aligned} Z_i^t - Z_i^{t-1} &= \sum_{k \geq 0} \frac{\Delta w_{ik} - 2 \cdot \Delta d_k(c_{ik})}{4h \cdot 2^k} \\ &= \frac{\Delta w_{ip} - 2 \cdot \Delta d_p(c_{ip})}{4h \cdot 2^p} + \sum_{k \neq p} \frac{\Delta w_{ik} - 2 \cdot \Delta d_k(c_{ik})}{4h \cdot 2^k} \\ &\leq \frac{1}{4h \cdot 2^p} - \frac{\Delta d_p(c_{ip})}{2h \cdot 2^p}. \end{aligned}$$

To complete the proof, it now suffices to argue that $\Delta d_p(c_{ip}^t) = 1/\alpha$ with probability α and 0 with the remaining probability. This follows immediately by Lemma 11: With probability α we have $c_{ip}^t = c_{ip}^{t-1} + 1$, and hence $\Delta d_p(c_{ip}) = d_p(c_{ip}^t) - d_p(c_{ip}^{t-1}) = h \cdot 2^p / (2^p - c_{ip}^t + 1) = h \cdot 2^p / (2^p - c_{ip}^{t-1}) = 1/\alpha$. With the remaining probability $c_{ip}^t = c_{ip}^{t-1}$ and thus $\Delta d_p(c_{ip}) = 0$. ◀

For the final lemma, we need the following technical bound (proven in Appendix A). This can be seen as a reverse Jensen's type inequality.

▷ **Claim 13.** Fix $\varepsilon \in [0, 1]$, $\alpha \in [\varepsilon, 1]$ and a real x . Let X be a random variable such that

$$X = \begin{cases} x - \varepsilon/\alpha & \text{with probability } \alpha, \\ x & \text{otherwise.} \end{cases}$$

Then, $\mathbb{E}[e^{-X}] \leq e^{x-\varepsilon/2}$.

Finally, we can prove Lemma 4, restated below.

► **Lemma 4.** *Fix a step t and an outcome of random choices till step $t - 1$ inclusively. (In particular, this will fix the value of Φ^{t-1} .) Then, $\mathbb{E}[\Phi^t] \leq \Phi^{t-1}$, where the expectation is taken exclusively over random choices of RAND in step t .*

Proof. Fix a node i . We will show that

$$\mathbb{E}[\exp(Z_i^t)] \leq \exp(Z_i^{t-1}). \quad (3)$$

The lemma then follows by summing the above inequality over all nodes.

If $Z_i^t \leq Z_i^{t-1}$, (3) follows trivially. Otherwise, Lemma 12 implies that

$$Z_i^t \leq Z_i^{t-1} + \frac{1}{4h \cdot 2^p} + \begin{cases} -1/(2h \cdot 2^p \cdot \alpha) & \text{with probability } \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

where $p = p^{t-1}(i)$ and $\alpha = (2^p - c_{ip}^{t-1})/(h \cdot 2^p)$. As the random choices are fixed until step $t - 1$, the variables Z_i^{t-1} , p and α are no longer random variables, but real numbers.

Note that $c_{ip}^{t-1} \leq q_p - 1 \leq 2^p - 1$ as otherwise phase p of node i would have ended in an earlier step. Hence, $\alpha = (2^p - c_{ip}^{t-1})/(h \cdot 2^p) \geq 1/(2h \cdot 2^p)$.

Thus, $x = Z_i^{t-1} + 1/(4h \cdot 2^p)$, $\varepsilon = 1/(2h \cdot 2^p)$, and α satisfy the conditions of Claim 13. (In particular, $\varepsilon \leq \alpha \leq 1$.) This claim now yields

$$\mathbb{E}[\exp(Z_i^t)] \leq \exp\left(Z_i^{t-1} + \frac{1}{4h \cdot 2^p} - \frac{1}{2} \cdot \frac{1}{2h \cdot 2^p}\right) = \exp(Z_i^{t-1}),$$

which concludes the proof of (3), and thus also the lemma. ◀

5 Conclusions

In this paper, we have constructed a deterministic $O(\log^2 n)$ -competitive algorithm for the disjoint set covers (DSC) problem. Closing the remaining logarithmic gap between the current upper and lower bounds is an interesting open problem that seems to require a new algorithm that goes beyond the phase-based approach.

We have developed new derandomization techniques that extend the existing potential function methods. We hope that these extensions will be useful for derandomizing other online randomized algorithms, and eventually for providing a coherent online derandomization toolbox.

References

- 1 Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An $O(\log k)$ -competitive algorithm for generalized caching. *ACM Transactions on Algorithms*, 15(1):6:1–6:18, 2019. doi:10.1145/3280826.
- 2 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009. doi:10.1137/060661946.
- 3 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Second Edition*. John Wiley, 2000. doi:10.1002/0471722154.
- 4 Piotr Berman, Gruia Călinescu, C. Shah, and Alexander Zelikovsky. Power efficient monitoring management in sensor networks. In *2004 IEEE Wireless Communications and Networking Conference*, pages 2329–2334. IEEE, 2004. doi:10.1109/WCNC.2004.1311452.

- 5 Marcin Bienkowski, Björn Feldkord, and Pawel Schmidt. A nearly optimal deterministic online algorithm for non-metric facility location. In *Proc. 38th Symp. on Theoretical Aspects of Computer Science (STACS)*, LIPIcs, pages 14:1–14:17, 2021. doi:10.4230/LIPIcs.STACS.2021.14.
- 6 Béla Bollobás, David Pritchard, Thomas Rothvoß, and Alex D. Scott. Cover-decomposition and polychromatic numbers. *SIAM Journal on Discrete Mathematics*, 27(1):240–256, 2013. doi:10.1137/110856332.
- 7 Alan Borodin, Nati Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992. doi:10.1145/146585.146588.
- 8 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In *Proc. 55th ACM Symp. on Theory of Computing (STOC)*, pages 581–594. ACM, 2023. doi:10.1145/3564246.3585132.
- 9 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. In *Proc. 30th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 89–97. SIAM, 2019. doi:10.1137/1.9781611975482.6.
- 10 Niv Buchbinder and Joseph (Seffi) Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009. doi:10.1287/MOOR.1080.0363.
- 11 Adam L. Buchsbaum, Alon Efrat, Shaili Jain, Suresh Venkatasubramanian, and Ke Yi. Restricted strip covering and the sensor cover problem. In *Proc. 18th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1056–1063. SIAM, 2007. URL: <http://dl.acm.org/citation.cfm?id=1283383.1283497>, doi:10.5555/1283383.1283497.
- 12 Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005. doi:10.1007/S11276-005-6615-6.
- 13 Mihaela Cardei, My T. Thai, Yingshu Li, and Weili Wu. Energy-efficient target coverage in wireless sensor networks. In *Proc. 2005 IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1976–1984. IEEE, 2005. doi:10.1109/INFOCOM.2005.1498475.
- 14 Yuval Emek, Adam Goldbraikh, and Erez Kantor. Online disjoint set cover without prior knowledge. In *Proc. 27th European Symp. on Algorithms (ESA)*, LIPIcs, pages 44:1–44:16, 2019. doi:10.4230/LIPICS.ESA.2019.44.
- 15 Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the domatic number. *SIAM Journal on Computing*, 32(1):172–195, 2002. doi:10.1137/S0097539700380754.
- 16 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.
- 17 Ram P. Gupta. On the chromatic index and the cover index of a multigraph. In *Theory and Applications of Graphs*, pages 204–215. Springer Berlin Heidelberg, 1978.
- 18 Ngoc Mai Le, Seeun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In *Proc. 2023 ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1594–1610. SIAM, 2023. doi:10.1137/1.9781611977554.CH59.
- 19 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.
- 20 Ashwin Pananjady, Vivek Kumar Bagaria, and Rahul Vaze. The online disjoint set cover problem and its applications. In *Proc. 2015 IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1221–1229. IEEE, 2015. doi:10.1109/INFOCOM.2015.7218497.
- 21 Ashwin Pananjady, Vivek Kumar Bagaria, and Rahul Vaze. Optimally approximating the coverage lifetime of wireless sensor networks. *IEEE/ACM Transactions on Networking*, 25(1):98–111, 2017. doi:10.1109/TNET.2016.2574563.

18:16 Online Disjoint Set Covers: Randomization Is Not Necessary

- 22 Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988. doi:10.1016/0022-0000(88)90003-7.
- 23 Neal E. Young. On-line file caching. *Algorithmica*, 33(3):371–383, 2002. doi:10.1007/S00453-001-0124-5.

A Proofs of Technical Claims

▷ Claim 6. For each $k \geq 0$ it holds that $H(2^k) - H(2^k - q_k) \leq \ln(4e \cdot n)$.

Proof. Assume first that $2^k < 4n$. As $H(m) \leq 1 + \ln m = \ln(e \cdot m)$ for each m , we get $H(2^k) - H(2^k - q_k) < H(4n) \leq \ln(4e \cdot n)$ and the lemma follows.

Hence, in the following, we assume that $2^k \geq 4n$, obtaining

$$\begin{aligned} 2^k - q_k &= 2^k - \lceil (1 - 1/(2n)) \cdot 2^k \rceil && \text{(by the definition of } q_k) \\ &\geq 2^k - (1 - 1/(2n)) \cdot 2^k - 1 \\ &= 2^k/(2n) - 1 \\ &\geq 2^k/(4n). && \text{(as } 2^k \geq 4n) \end{aligned}$$

As $q_k \leq 2^k$, we can use the relation $H(m) \geq \ln m$ holding for every $m \geq 0$, obtaining

$$H(2^k) - H(2^k - q_k) \leq \ln(e \cdot 2^k) - \ln(2^k - q_k) \leq \ln(e \cdot 2^k) - \ln(2^k/(4n)) = \ln(4e \cdot n). \quad \triangleleft$$

▷ Claim 13. Fix $\varepsilon \in [0, 1]$, $\alpha \in [\varepsilon, 1]$ and a real x . Let X be a random variable such that

$$X = \begin{cases} x - \varepsilon/\alpha & \text{with probability } \alpha, \\ x & \text{otherwise.} \end{cases}$$

Then, $\mathbb{E}[e^X] \leq e^{x - \varepsilon/2}$.

Proof. Using $\alpha \geq \varepsilon$, we obtain

$$(1 + \varepsilon/\alpha) \cdot (1 - \varepsilon/(2\alpha)) = 1 + \varepsilon/(2\alpha) - \varepsilon^2/(2\alpha^2) \geq 1. \quad (4)$$

Next, we use the relation $e^{-y} \leq (1 + y)^{-1}$ (holding for every $y > -1$) and (4) to argue that

$$\exp(-\varepsilon/\alpha) \leq (1 + \varepsilon/\alpha)^{-1} \leq 1 - \varepsilon/(2\alpha). \quad (5)$$

Finally, this gives

$$\begin{aligned} \mathbb{E}[e^X] &= \alpha \cdot \exp(x - \varepsilon/\alpha) + (1 - \alpha) \cdot \exp(x) \\ &= e^x \cdot (\alpha \cdot \exp(-\varepsilon/\alpha) + 1 - \alpha) \\ &\leq e^x \cdot (\alpha - \varepsilon/2 + 1 - \alpha) && \text{(by (5))} \\ &\leq e^{x - \varepsilon/2}. && \text{(as } 1 - \varepsilon/2 \leq e^{-\varepsilon/2} \text{ for each } \varepsilon) \quad \triangleleft \end{aligned}$$

The Complexity of Learning LTL, CTL and ATL Formulas

Benjamin Bordais 

TU Dortmund University, Center for Trustworthy Data Science and Security,
University Alliance Ruhr, Dortmund, Germany

Daniel Neider 

TU Dortmund University, Center for Trustworthy Data Science and Security,
University Alliance Ruhr, Dortmund, Germany

Rajarshi Roy 

Department of Computer Science, University of Oxford, UK

Abstract

We consider the problem of learning temporal logic formulas from examples of system behavior. Learning temporal properties has crystallized as an effective means to explain complex temporal behaviors. Several efficient algorithms have been designed for learning temporal formulas. However, the theoretical understanding of the complexity of the learning decision problems remains largely unexplored. To address this, we study the complexity of the passive learning problems of three prominent temporal logics, Linear Temporal Logic (LTL), Computation Tree Logic (CTL) and Alternating-time Temporal Logic (ATL) and several of their fragments. We show that learning formulas with unbounded occurrences of binary operators is NP-complete for all of these logics. On the other hand, when investigating the complexity of learning formulas with bounded occurrences of binary operators, we exhibit discrepancies between the complexity of learning LTL, CTL and ATL formulas (with a varying number of agents).

2012 ACM Subject Classification Theory of computation

Keywords and phrases Temporal logic, passive learning, complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.19

Related Version *Full Version:* <https://arxiv.org/abs/2408.04486> [8]

Funding Rajarshi Roy acknowledges partial funding by the ERC under the European Union's Horizon 2020 research and innovation programme (grant agreement No.834115, FUN2MODEL).

1 Introduction

Temporal logics are the de-facto standard for expressing temporal properties for software and cyber-physical systems. Originally introduced in the context of program verification [33, 15], temporal logics are now well-established in numerous areas, including reinforcement learning [40, 25, 10], motion planning [17, 12], process mining [13], and countless others. The popularity of temporal logics can be attributed to their unique blend of mathematical rigor and resemblance to natural language.

Until recently, formulating properties in temporal logics has been a manual task, requiring human intuition and expertise [6, 39]. To circumvent this step, in the past ten years, there have been numerous works to automatically learn (i.e., generate) properties in temporal logic. Among them, a substantial number of works [29, 11, 35, 26, 41] target Linear Temporal Logic (LTL) [33]. There is now a growing interest in learning formulas [16, 34, 9] in Computation Tree Logic (CTL) [15] and Alternating-time Temporal Logic (ATL) [1] due to their ability to express branching-time properties, including for multi-agent systems.



© Benjamin Bordais, Daniel Neider, and Rajarshi Roy;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 19; pp. 19:1–19:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Table 1** The complexity results for learning LTL, CTL and ATL formulas. The notation ATL^k refers to ATL-formulas with k agents. $\mathbf{U}^t \subseteq \{\neg, \mathbf{X}, \mathbf{F}, \mathbf{G}\}$ refers to the set of unary temporal operators.

	Unbounded use of binary operators	Bounded use of binary operators		
		$\mathbf{X} \in \mathbf{U}^t$	$\mathbf{X} \notin \mathbf{U}^t$	
			$\{\mathbf{F}, \mathbf{G}\} \subseteq \mathbf{U}^t$	$\mathbf{U}^t = \{\mathbf{F}\}, \{\mathbf{G}\}$
LTL	NP-c	L		
CTL		NP-c	NL-c	
ATL^2		NP-c		P-c
ATL^k		NP-c		

While existing approaches for learning temporal properties demonstrate impressive empirical performance, detailed comparisons of computational complexity across different temporal logics remain underexplored. Most related works focus on LTL, either in the verification domain [18, 27] or the database domain [19, 24]. These studies primarily report complexity results, often highlighting NP-completeness for learning LTL-formulas and their fragments. In contrast, the computational complexity of learning CTL- and ATL-formulas has not yet been thoroughly examined.

In this work, we extend the study of learning temporal properties to include CTL- and ATL-formulas. Additionally, we broaden existing results for LTL to cover a more comprehensive set of operators, specifically addressing all binary operators (temporal or not).

To elaborate on our contributions, let us precisely describe the problem that we consider, the *passive learning* problem for temporal logic [29, 11]. Its decision version asks the following question: given two sets \mathcal{P} , \mathcal{N} of positive and negative examples of a system’s behavior and a size bound B , does there exist a “separating” formula of size at most B , which is satisfied by the positive examples and violated by the negative ones.

Our instantiation of the above problem depends on the considered logic, following related literature [29, 34, 9]: LTL-formulas express linear-time properties, CTL-formulas express branching-time properties, and ATL-formulas express properties on multi-agent systems. Accordingly, the input examples for learning LTL, CTL and ATL are linear structures (or equivalently infinite words), Kripke structures and concurrent game structures, respectively. We refer to Section 2 for formal definitions and other prerequisites.

We summarize our contributions in Table 1. Our first result, illustrated in the left column, shows that allowing formulas with unrestricted use of at least one binary operator makes the corresponding learning decision problem NP-complete for all considered logics. Some of these NP-hardness results are (closely) inspired by [27], involving reductions from the hitting set problem – one of Karp’s 21 NP-complete problem; some others require novel proof techniques, e.g. one involves a reduction from an NP-complete modulo-2 calculus problem. We describe the outline of the proofs in Section 3.

All of the above NP-hardness proofs rely on separating formulas with linearly many (in the size of the input) occurrences of binary operators. Thus, in the search of expressive temporal logic fragments with lower complexities, we focus on formulas with a bounded occurrences of binary logical operators such as \wedge (and), \vee (or), etc. and no binary temporal operators such as \mathbf{U} (until), \mathbf{R} (release), etc. This choice of formulas is motivated by the fact that such formulas can still express interesting properties (e.g., GR(1) [32] formulas, mode-target formulas [4], etc.) and are used in several practical applications (see Section 4.1 for details). We explore several fragments with different unary temporal operators, \mathbf{X} (next),

F (eventually) and **G** (globally), and present the results in the rightmost column of Table 1. We notice that, in this case, the complexity of the learning problems varies considerably between different logics and unary operators. Importantly, we exhibit fragments where the learning problem is below NP. We prove the three NP-hardness results using a reduction from the hitting set problem; we give key insights on all of these results in Section 4.

All details can be found in the extended version [8].

Related Works. The most closely related works are [18] and [27], which operate within a similar framework to ours. Both works consider learning problems in several fragments of LTL, especially involving boolean operators such as \vee and \wedge , and temporal operators such as **X**, **F** and **G** and prove their NP-completeness. We extend part of their work by categorizing fragments based on the arity of the operators and studying which type of operators contribute to the hardness. Moreover, there are several differences in the parameters considered for the learning problem. The most important one is the following: the above works consider the size upper bound B to be in binary, while we assume B given in unary. Although, in complexity problems, integers are most often assumed to be written in binary, we believe that considering size bound in unary is justified since one may want to not only decide the existence of a formula but also effectively output one, which will require to explicitly write it. The other differences with the setting of the above works are mostly due to the fact that we do not only consider LTL learning, but CTL and ATL learning as well. A thorough discussion of these differences can be found in the extended version of this paper [8].

In the past, complexity analysis of passive learning has been studied for formalisms other than temporal logics. For instance, [21] and [2] proved NP-completeness of the passive learning problems of deterministic finite automata (DFAs) and regular expressions (REs).

When it comes to temporal logics, most related works focus on developing efficient algorithms for learning temporal logic formulas. Among these, the emphasis has predominantly been on learning LTL (or its significant fragments), which has been discussed in detail in a recent survey summarizing various learning techniques [30]. Broadly, the techniques can be categorized into three main types: constraint solving [29, 11, 37, 20, 22], enumerative search [35, 41], and neuro-symbolic techniques [26, 42].

For learning CTL, some approaches rely on handcrafted templates [14, 43] for simple enumerative search, while others employ constraint-solving methods to learn formulas with arbitrary structures [34]. The constraint-solving methods are extended to learn ATL-formulas as well [9]. There are also works on learning other logics such as Signal Temporal Logic [7, 28], Metric Temporal Logic [36], Past LTL [3], Property Specification Language [38], etc.

2 Preliminaries and Definitions

We let \mathbb{N} denote the set of all integers and \mathbb{N}_1 denote the set of all positive integers. For all $i \leq j \in \mathbb{N}$, we let $[i, \dots, j] \subseteq \mathbb{N}$ denote the set of integers $\{i, i+1, \dots, j\}$.

Given any non-empty set Q , we let Q^* , Q^+ and Q^ω denote the sets of finite, non-empty finite and infinite sequences of elements in Q , respectively. For all $\rho \in Q^+$, we denote by $|\rho| \in \mathbb{N}$ the number of elements of ρ . For all $\bullet \in \{+, \omega\}$, $\rho \in Q^\bullet$ and $i \in \mathbb{N}_1$, if ρ has at least i elements, we let: $\rho[i] \in Q$ denote the i -th element in ρ , in particular $\rho[1] \in Q$ is the first element of ρ ; $\rho[:i] \in Q^+$ denotes the non-empty finite sequence $\rho_1 \cdots \rho_i \in Q^+$; $\rho[i:] \in Q^\bullet$ denotes the non-empty sequence $\rho_i \cdot \rho_{i+1} \cdots \in Q^\bullet$, in particular we have $\rho[1:] = \rho$.

For the remainder of this section, we fix a non-empty set of propositions Prop .

2.1 Structures

Usually, ATL-formulas are interpreted on concurrent game structures, i.e. games where, at each state, the concurrent actions of several agents have an impact on the next state reached. A special kind of concurrent game structures are turn-based game structures, where each state belongs to a specific agent who decides what the next state is. Here, we introduce only this special kind of games mainly due to a lack of space, but also because all of our hardness results, presented in Table 1, hold even when only considering turn-based game structures.

► **Definition 1.** A turn-based game structure (TGS for short) $T = \langle Q, I, \text{Succ}, \text{Ag}, \alpha, \text{Prop}, \pi \rangle$ is a tuple where: Q is a finite set of states; $I \subseteq Q$ is the set of initial states; $\text{Succ} : Q \rightarrow 2^Q \setminus \emptyset$ maps each state to its set of successors; $\text{Ag} \subseteq \mathbb{N}$ denotes the set of agents; $\alpha : Q \rightarrow \text{Ag}$ maps each state to the agent owning it; and $\pi : Q \mapsto 2^{\text{Prop}}$ maps each state $q \in Q$ to the set of propositions that hold in q . A state q is said to be self-looping if $q \in \text{Succ}(q)$. A structure is self-looping if all of its states are self-looping.

For all coalitions of agents $A \subseteq \text{Ag}$, a strategy s_A for the coalition A is a function $s_A : Q^+ \rightarrow Q$ such that, for all $\rho = \rho_1 \cdots \rho_n \in Q^+$, if $\alpha(\rho_n) \in A$, then $s_A(\rho) \in \text{Succ}(\rho_n)$. We denote by S_A the set of strategies for the coalition A . Then, from any state $q \in Q$, we define the set $\text{Out}(q, s_A)$ of infinite paths compatible with the strategy s_A from q : $\text{Out}(q, s_A) := \{\rho \in Q \cdot Q^\omega \mid \forall i \in \mathbb{N}_1 : \alpha(\rho[i]) \in A \implies \rho[i+1] = s_A(\rho[:i])\}$.

Finally, the size $|T|$ of the turn-based structure T is equal to: $|T| = |Q| + |\text{Ag}| + |\text{Prop}|$.

Unless otherwise stated, a turn-based structure T will always refer to the tuple $T = \langle Q, I, \text{Succ}, A, \alpha, \text{Prop}, \pi \rangle$.

There are also special kinds of turn-based structures of interest for us, introduced below.

► **Definition 2.** A Kripke structure is a turn-based structure with only one agent. A linear structure is a Kripke structure such that: $|I| = 1$, and for all $q \in Q$, we have $|\text{Succ}(q)| = 1$. Finally, a turn-based structure is size-1 if $|Q| = 1$.

Unless otherwise stated, a Kripke structure K will always refer to a tuple $\langle Q, I, \text{Succ}, \text{Prop}, \pi \rangle^1$.

We have introduced the notion of linear structures as we are going to interpret LTL-formulas on them. In the literature, they are usually interpreted on ultimately periodic words. However, both models are equivalent and can be encoded into each other straightforwardly.

2.2 ATL, CTL and LTL formulas

The LTL, CTL and ATL-formulas that we consider throughout this paper use the following temporal operators: **X** (neXt), **F** (Future), **G** (Globally), **U** (Until), **R** (Release), **W** (Weak until), **M** (Mighty release). We group these operators into the sets of unary and binary operators: $\text{Op}_{\text{Un}} := \{\neg, \mathbf{X}, \mathbf{F}, \mathbf{G}\}$ and $\text{Op}_{\text{Bin}}^{\text{tp}} := \{\mathbf{U}, \mathbf{R}, \mathbf{W}, \mathbf{M}\}$. We also let $\text{Op}_{\text{Bin}}^{\text{lg}}$ be the set of all logical binary operators, i.e. classical logical operators, along with their negations: $\text{Op}_{\text{Bin}}^{\text{lg}} := \{\vee, \wedge, \implies, \Leftarrow, \Leftrightarrow, \neg\vee, \neg\wedge, \neg\implies, \neg\Leftarrow, \neg\Leftrightarrow\}$ (we have $|\text{Op}_{\text{Bin}}^{\text{lg}}| = 10$).

To define ATL-formulas, we consider two types of formulas: state formulas – where strategic operators occur, denoted with the Greek letter ϕ – and path formulas – where temporal operators occur, denoted with the Greek letter ψ . Consider some $\text{U}^t \subseteq \text{Op}_{\text{Un}}$, $\text{B}^t \subseteq \text{Op}_{\text{Bin}}^{\text{tp}}$, and $\text{B}^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$. For all $k \in \mathbb{N}_1$, we denote by $\text{ATL}^k(\text{Prop}, \text{U}^t, \text{B}^t, \text{B}^l)$ the set of ATL^k -state formulas defined by the grammar:

$$\phi ::= p \mid \neg\phi \mid \phi * \phi \mid \langle\langle A \rangle\rangle\psi \qquad \psi ::= *_1\phi \mid \phi *_2\phi$$

¹ In Kripke structures, there is only one agent, thus Ag and α are irrelevant.

where ϕ is a state-formula, ψ is a path formula, $p \in \text{Prop}$, $* \in \mathbf{B}^l$, $A \subseteq [1, \dots, k]$ is a subset of agents, $*_1 \in \mathbf{U}^t \setminus \{\neg\}$, and $*_2 \in \mathbf{B}^t$. We denote by ATL^k the set of all ATL^k -state formulas ϕ . Note that CTL-formulas are $\text{ATL}^1(\text{Prop}, \mathbf{U}^t, \mathbf{B}^t, \mathbf{B}^l)$ -formulas. Hence, there are only two possible strategic operator: $\langle\langle \emptyset \rangle\rangle$, usually denoted \forall , and $\langle\langle \{1\} \rangle\rangle$ usually denoted \exists . We define LTL-formulas as ATL^1 -formulas using only the quantifier \exists . Since LTL-formulas are interpreted on linear structures, where each state has exactly one successor, the strategic operators used have no impact on the satisfaction of the formula. For readability, we will depict LTL-formulas without the \exists quantifier.

The set of sub-formulas $\text{SubF}(\phi)$ of a formula ϕ is then defined inductively as follows: $\text{SubF}(\phi) := \{\phi\} \cup S$ where $S := \emptyset$ if $\phi = p \in \text{Prop}$, $S := \text{SubF}(\phi')$ if $\phi \in \{\neg\phi', \langle\langle A \rangle\rangle *_1 \phi'\}$ and $S := \text{SubF}(\phi_1) \cup \text{SubF}(\phi_2)$ if $\phi \in \{\phi_1 * \phi_2, \langle\langle A \rangle\rangle(\phi_1 *_2 \phi_2)\}$. The size $|\phi|$ of a formula is then defined as its number of sub-formulas: $|\phi| := |\text{SubF}(\phi)|$. We also denote by $|\phi|_{\text{bin}}$ the number of sub-formulas of ϕ using a binary operator, $|\phi|_{\text{bin}} := |\text{SubBin}(\phi)|$ with: $\text{SubBin}(\phi) := \{\phi_1 * \phi_2 \in \text{SubF}(\phi) \mid \phi_1, \phi_2 \in \text{SubF}(\phi), * \in \text{Op}_{\text{Bin}}^{\text{tp}} \cup \text{Op}_{\text{Bin}}^{\text{lg}}\}$.

We interpret ATL-formulas over TGS using the standard definitions [1]. That is, given a state q and a state formula ϕ , the fact q satisfies ϕ , denoted $q \models \phi$, is defined inductively:

$$\begin{aligned} q \models p & \quad \text{iff } p \in \pi(q) & q \models \phi_1 * \phi_2 & \quad \text{iff } (q \models \phi_1) * (q \models \phi_2) = \text{True} \\ q \models \neg\phi & \quad \text{iff } q \not\models \phi & q \models \langle\langle A \rangle\rangle\psi & \quad \text{iff } \exists s_A \in \mathbf{S}_A, \forall \pi \in \text{Out}(q, s), \pi \models \psi \end{aligned}$$

where $* \in \text{Op}_{\text{Bin}}^{\text{lg}}$ is a binary operator seen as a boolean function $* : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$ with $\mathbb{B} := \{\text{True}, \text{False}\}$. Furthermore, given a path $\pi \in Q^\omega$ and a path formula ψ , the fact that ψ holds for the path π , also denoted $\pi \models \psi$, is defined inductively as follows:

$$\begin{aligned} \pi \models \mathbf{X}\phi & \quad \text{iff } \pi[2:] \models \phi; & \pi \models \phi_1 \mathbf{U}\phi_2 & \quad \text{iff } \exists i \in \mathbb{N}_1, \pi[i:] \models \phi_2 \text{ and} \\ \pi \models \mathbf{F}\phi & \quad \text{iff } \exists i \in \mathbb{N}_1, \pi[i:] \models \phi; & & \quad \forall 1 \leq j \leq i-1, \pi[j:] \models \phi_1 \\ \pi \models \mathbf{G}\phi & \quad \text{iff } \forall i \in \mathbb{N}_1, \pi[i:] \models \phi & \pi \models \phi_1 \mathbf{R}\phi_2 & \quad \text{iff } \pi \models \neg(\neg\phi_1 \mathbf{U}\neg\phi_2) \\ \pi \models \phi_1 \mathbf{W}\phi_2 & \quad \text{iff } \pi \models (\phi_1 \mathbf{U}\phi_2) \vee \mathbf{G}\phi_1 & \pi \models \phi_1 \mathbf{M}\phi_2 & \quad \text{iff } \pi \models (\phi_1 \mathbf{R}\phi_2) \wedge \mathbf{F}\phi_1 \end{aligned}$$

An ATL-formula ϕ accepts a TGS T , denoted by $T \models \phi$, if $q \models \phi$ for all initial states $q \in I$, otherwise it rejects it. Given two formulas ϕ, ϕ' , we write $\phi \implies \phi'$ if, for all TGS T , if $T \models \phi$, then $T \models \phi'$. We write $\phi \equiv \phi'$ when $\phi \implies \phi'$ and $\phi' \implies \phi$.

2.3 Learning decision problem

We define the LTL, CTL and ATL learning problems below, where models for LTL, CTL, and ATL are linear structures, Kripke structures and turn-based game structures, respectively.

► **Definition 3.** Let $\text{TL} \in \{\text{LTL}, \text{CTL}, \text{ATL}^k \mid k \in \mathbb{N}_1\}$ and consider some sets of operators $\mathbf{U}^t \subseteq \text{Op}_{\text{Un}}$, $\mathbf{B}^t \subseteq \text{Op}_{\text{Bin}}^{\text{tp}}$ and $\mathbf{B}^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$. For all $n \in \mathbb{N} \cup \{\infty\}$, we denote by $\text{TL}_{\text{Learn}}(\mathbf{U}^t, \mathbf{B}^t, \mathbf{B}^l, n)$ the decision problem:

- *Input:* $(\text{Prop}, \mathcal{P}, \mathcal{N}, B)$ where Prop is a set of propositions, \mathcal{P}, \mathcal{N} are two finite sets of models for TL, and $B \in \mathbb{N}$.
- *Output:* yes if and only if there exists a TL-formula $\varphi \in \text{TL}(\text{Prop}, \mathbf{U}^t, \mathbf{B}^t, \mathbf{B}^l)$ such that $|\varphi| \leq B$, $|\varphi|_{\text{bin}} \leq n$, and φ is separating, i.e. such that : for all $X \in \mathcal{P}$ (resp. $X \in \mathcal{N}$), we have $X \models \varphi$ (resp. $X \not\models \varphi$).

The size of the input is equal to $|\text{Prop}| + |\mathcal{P}| + |\mathcal{N}| + B$ (i.e. B is written in unary).

As the model checking problems for LTL, CTL, ATL are in P [1], it follows that the learning problems for all these logics are in NP, with a straightforward guess-and-check subroutine.

► **Proposition 4.** For all $U^t \subseteq \text{Op}_{U^n}$, $B^t \subseteq \text{Op}_{\text{Bin}}^{\text{tp}}$, $B^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$, $n \in \mathbb{N} \cup \{\infty\}$, and $\text{TL} \in \{\text{LTL}, \text{CTL}, \text{ATL}^k \mid k \in \mathbb{N}_1\}$, the decision problem $\text{TLearn}(U^t, B^t, B^l, n)$ is in NP.

2.4 Hitting set problem

We recall below the NP-complete problem from which we will establish almost all of our (NP-hardness) reductions.

- **Definition 5 (Hitting set problem).** We denote by *Hit* the following decision problem:
- *Input:* a triple (l, C, k) where $l \in \mathbb{N}_1$, $C = C_1, \dots, C_n$ are non-empty subsets of $[1, \dots, l]$
 - *Output:* yes iff there is a subset $H \subseteq [1, \dots, l]$ of size at most k such that, we have $H \cap C_i \neq \emptyset$ for all $1 \leq i \leq n$. In such a case, the set H is called a hitting set.

In the following, if (l, C, k) is an instance of the hitting set problem, then C refers to C_1, \dots, C_n for some $n \in \mathbb{N}_1$.

3 Learning with unbounded use of binary operators

First, we consider the case of learning a formula with arbitrarily many occurrences of binary operators. The main result of this section is stated in Theorem 6 below.

► **Theorem 6.** Let $B^t \subseteq \text{Op}_{\text{Bin}}^{\text{tp}}$ and $B^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$ such that $B^t \cup B^l \neq \emptyset$. Then, for all $U^t \subseteq \text{Op}_{U^n}$, the decision problem $\text{LTL}_{\text{Learn}}(U^t, B^t, B^l, \infty)$ is NP-hard.

In the passive learning setting that we consider, the size of the formulas is crucial due to the upper bound B . Therefore, although it is possible to express e.g. disjunctions with conjunctions and negations, since doing so affects the size of the formulas involved, if we have proved that a learning problem is NP-hard with the operators \vee, \neg , it does not imply a priori that it is also NP-hard with the operator \wedge . Hence, for the sake of completeness, we consider all those fourteen binary operators (ten logical, four temporal), although it seems that some of these binary operators (like \vee or \wedge) make much more sense to consider than others (like \Leftrightarrow or $\neg \Leftrightarrow$). Since these operators behave differently, we cannot do a single reduction working for all these operators at once. However, we do partition these operators in different groups and exhibit a reduction per group of operators.

Most of the reductions use only size-1 structures, that are (almost) entirely defined by the subset of propositions labeling their only state. In addition, most of the reductions are done from the hitting set problem. In that case, how we extract a hitting set from a (small enough) separating formula relies only on the variables that need to occur in a formula separating the positive and negative structures, regardless of the operators involved.

We start with the operators $\vee, \Rightarrow, \Leftarrow$, i.e. we assume that $B^l \cap \{\vee, \Rightarrow, \Leftarrow\} \neq \emptyset$. The reduction for this case is actually a straightforward adaptation of the proof of [27, Theorem 2]. We describe it here. Given an instance (l, C, k) of the hitting set problem, we let $\text{Prop} := \{a_j, b_j \mid 1 \leq j \leq l\}$. Furthermore, for all subsets $T \subseteq [1, \dots, l]$, we let $\mathcal{L}(T)$ denote a size-1 (linear) structure whose only state is labeled by the set $\{a_j, b_{j'} \mid j \in T, j' \notin T\}$. Then, we let $\text{In}^{\vee, \Rightarrow, \Leftarrow} := (\text{Prop}, \mathcal{P}, \mathcal{N}, B)$ for $\mathcal{P} := \{\mathcal{L}(C_i) \mid 1 \leq i \leq n\}$, $\mathcal{N} := \{\mathcal{L}(\emptyset)\}$, and $B := 2k - 1$. Let us illustrate this reduction on a simple example. Assume that $l = 4$, $C = (\{1, 2, 3\}, \{2, 4\}, \{1, 4\})$, and $k = 2$. Then, the sets labeling the only state of the positive structures are $\{a_1, a_2, a_3, b_4\}$, $\{b_1, a_2, b_3, a_4\}$, and $\{a_1, b_2, b_3, a_4\}$ while the set labeling the only state of the negative structure is $\{b_1, b_2, b_3, b_4\}$. Furthermore, $B = 3$. Then, $H := \{1, 4\}$ is the a hitting set with $|H| \leq 2$, while $\varphi_{\vee} := a_1 \vee a_4$, $\varphi_{\Rightarrow} := b_1 \Rightarrow a_4$, and $\varphi_{\Leftarrow} := a_1 \Leftarrow b_4$ are all separating formulas with $|\varphi_{\vee}| = |\varphi_{\Rightarrow}| = |\varphi_{\Leftarrow}| \leq 3$.

We claim that (l, C, k) is a positive instance of **Hit** iff $\text{In}^{\vee, \Rightarrow, \Leftarrow}$ is a positive instance of $\text{LTL}_{\text{Learn}}(\mathbf{U}^t, \mathbf{B}^t, \mathbf{B}^l, \infty)$. Indeed, given a hitting set $H = \{i_1, \dots, i_r\}$ with $r \leq k$, one can check that the LTL-formula $\varphi_{\vee} := \bigvee_{1 \leq x \leq r} a_{i_x}$ of size $2r - 1 \leq B$ accepts \mathcal{P} and rejects \mathcal{N} . Note that, the LTL-formulas $\varphi_{\Rightarrow} := b_{j_1} \Rightarrow (b_{j_2} \Rightarrow (\dots \Rightarrow a_{j_r}))$ and $\varphi_{\Leftarrow} := ((a_{j_1} \Leftarrow b_{j_2}) \Leftarrow \dots) \Leftarrow b_{j_r}$ of size $2r + 1 \leq B$ also accept \mathcal{P} and reject \mathcal{N} . On the other hand, consider an LTL-formula φ of size at most B that accepts \mathcal{P} and rejects \mathcal{N} . We let $H := \{1 \leq j \leq l \mid a_j \text{ or } b_j \text{ occurs in } \varphi\}$. Since $|\varphi| \leq B$, we have $|H| \leq k$. Furthermore, consider any $1 \leq i \leq n$. Let us consider the set S_i (resp. S) labeling the only state of the structure $\mathcal{L}(C_i)$ (resp. $\mathcal{L}(\emptyset)$). We have $\Delta(S_i, S) := S_i \setminus S \cup S \setminus S_i = \{a_j, b_j \mid j \in C_i\}$. One can then show (rather straightforwardly, by induction on LTL-formulas) that, since φ accepts $\mathcal{L}(C_i)$ and rejects $\mathcal{L}(\emptyset)$, at least one variable in $\Delta(S_i, S)$ occurs in φ . That is, $C_i \cap H \neq \emptyset$ and H is a hitting set of size at most k .

In fact, the reduction for the operators $\wedge, \neg \Rightarrow, \neg \Leftarrow$ is obtained from the above one by reversing the positive and negative sets (the arguments are almost identical).

We then handle the operators $\neg \vee, \neg \wedge$. The above reductions cannot be used since, when the operator $\neg \wedge$ (or the operator $\neg \vee$) is used successively, the formula obtained is semantically equivalent to an alternation of conjunctions and disjunctions. For instance, consider six variables $r_1, r_2, r_3, r_4, x_1, x_2$ to use in a single LTL-formula using only the $\neg \wedge$ operator, e.g.: $\varphi := r_1 \neg \wedge (x_1 \neg \wedge (r_2 \neg \wedge (x_2 \neg \wedge (r_3 \neg \wedge r_4))))$. It is semantically equivalent to: $\varphi \equiv \neg r_1 \vee (x_1 \wedge (\neg r_2 \vee (x_2 \wedge (\neg r_3 \vee \neg r_4))))$. This is in sharp contrast with the above-formulas $\varphi_{\vee}, \varphi_{\Leftarrow}$ and φ_{\Rightarrow} . To circumvent this difficulty, we change the reduction by adding propositions labeling the only state of all the positive size-1 linear structures (but not the only state of all the negative ones). We can then place these propositions where x_2 and x_4 were in the above formula. That way, we semantically obtain a disjunction on relevant variables r_1, r_2, r_3, r_4 . The obtained reduction is slightly more subtle than the previous ones.

Before considering the last two logical operators $\Leftrightarrow, \neg \Leftrightarrow$, we handle the temporal operators **W, M**. The two previous reductions only use size-1 structures. On such structures, the temporal operators **W, M** are actually equivalent to \vee and \wedge respectively. Hence, the reductions for \vee and \wedge can also be used as is for the operators **W** and **M** respectively.

We then handle the final two logical operators $\Leftrightarrow, \neg \Leftrightarrow$. These operators are unlike the other operators. Let us give an intuition of how the learning problems with these operators behave. Consider an LTL-formula φ using only the operators \neg, \Rightarrow and $\neg \Leftrightarrow$ and a size-1 structure \mathcal{L} . Let S denote the set of propositions labeling the only state of \mathcal{L} . We let $\text{Neg}(\varphi)$ denote the number of occurrences of the operators $\neg, \neg \Leftrightarrow$ in φ . We also let $\text{NbOc}_{\bar{S}}(\varphi)$ denote the number of occurrences of the propositions not in S in φ . Then, one can realize that $\mathcal{L} \models \varphi$ if and only if $\text{Neg}(\varphi)$ and $\text{NbOc}_{\bar{S}}(\varphi)$ have the same parity. This simple observation suggests that the learning problem with the operators $\Leftrightarrow, \neg \Leftrightarrow$ is linked to modulo-2 calculus. The reduction for these operators is established from an NP-complete problem dealing with modulo-2 calculus, known as the Coset Weight problem [5].

Finally, we handle the temporal operators **U** and **R**. On size-1 structures, for all LTL-formulas φ_1, φ_2 , we have the following equivalences: $\varphi_1 \mathbf{U} \varphi_2 \equiv \varphi_1 \mathbf{R} \varphi_2 \equiv \varphi_2$. That is, contrary to the temporal operators **W** and **M**, on size-1 structures, **U** and **R** are equivalent to unary operators. Hence, the reduction that we consider does not involve only size-1 structures. It is once again established from the hitting set problem, though the construction and the correctness proof are more involved than for the above cases.

On top of that, for all sets of operators, ATL learning is at least as hard as CTL learning, which is itself at least as hard as LTL learning. Thus, from Theorem 6, we obtain that CTL and ATL learning with unbounded use of binary operators are NP-hard. This justifies the leftmost column of Table 1.

4 Learning with a bounded amount of binary operators

Since, with unbounded use of binary operators, all the learning problems are NP-hard, we focus on learning formulas where the number of occurrences of binary operators is bounded. Note that the bound n parameterizes the decision problem itself, and therefore is independent of the input. For simplicity, we restrict ourselves to formulas that do not use at all binary temporal operators. Before we dive into the details of our results as summarized in the rightmost column of Table 1, let us first argue why this fragment is interesting to focus on.

4.1 Expressivity

The passive learning problem that we consider in this paper bounds the size of the formulas considered. This is because we want a separating formula not to overfit the input (i.e. not to simply describe the positive and negative models). However, another benefit is that the smaller the formulas, the more understandable they are for users. Similarly, using too many binary operators could make the formulas hard to grasp, regardless of their size.

In addition, there are examples of interesting specifications that can be expressed with a bounded amount of binary operators. We give three examples below with LTL-formulas.

Consider first so-called “mode-target” formulas of the shape $\bigwedge_j (\mathbf{F G} M_j \Rightarrow \bigvee_i \mathbf{F G} T_{i,j})$, where all $M_j, T_{i,j}$ are propositions. These types of formulas were introduced in [4] and exhibit two interesting features: the corresponding LTL-synthesis problem is tractable, and these formulas express an interesting property, which can be summarized as follows: if a model eventually settles in a mode M_j , then it should eventually settle in one of the target $T_{i,j}$. Interestingly, when the number of different modes and targets that a system can have is fixed, then the number of binary operators sufficient to express such specification is also bounded.

Similarly, there are also interesting specifications related to “generalized reactivity” (from [32] for LTL-formulas). Such specifications are of the shape $\bigwedge_i \mathbf{G F} \psi_i \Rightarrow \bigwedge_i \mathbf{G F} \psi'_i$, where all formulas ψ_i and ψ'_i do not feature at all temporal operators. As such, up to introducing additional propositions, these could be expressible with few binary operators. These formulas can be read as an implication between assumptions and guarantees. As above, when the number of assumptions and guarantees is bounded, then the number of binary operators sufficient to express such formulas also is.

Finally, one of the popular LTL learning tools, Scarlet [35], relies on a fragment of LTL, directed LTL and its dual, which uses unary temporal operators and binary logical operators only. In these fragments, formulas of a fixed length (a search parameter they define) can use several of $\mathbf{F G}$ and \mathbf{X} operators while using only bounded occurrences of \wedge and \vee operators.

4.2 Abstract recipes

The six decision problems captured in the rightmost column of Table 1 are of two kinds: three are NP-complete, while three others are below NP. In fact, the proofs of all three results of the same kind will follow the same abstract recipes. We present them below.

Recipe for the membership-below-NP proofs. Let TL denote either LTL formulas, or CTL formulas without the operator \mathbf{X} , or ATL^2 formulas with only one unary operator \mathbf{F} or \mathbf{G} (i.e. one of the three logical fragment for which the corresponding decision problem is below NP). Then, we follow the two steps below:

- A) First, we show that given the set of propositions Prop and the bound B , there is a set of relevant TL-formulas $\text{RelForm}(\text{Prop}, B)$ such that: 1) For all TL(Prop)-formulas ϕ of size at most B , there is a formula $\phi' \in \text{RelForm}(\text{Prop}, B)$ such that $\phi \equiv \phi'$; and 2) the size of $\text{RelForm}(\text{Prop}, B)$ is polynomial in $|\text{Prop}|$ and B .

- B) Second, we show that for all TL-formulas $\phi \in \text{RelForm}(\text{Prop}, B)$, deciding if ϕ satisfies a TL-model M can be done, depending on $|\text{Prop}|, B, |M|$, within the resources allowed, i.e. logarithmic space for the LTL case, non-deterministic logarithmic space for the CTL case, and polynomial time for the ATL² case.

Due to a lack of space, in this paper we will only present these two steps in the context of formulas that do not use any binary operator. Since the occurrences of binary operators is bounded in any case, the arguments are essentially the same for the general case. For instance, for the first step, from the result established for formulas without binary operators, we can straightforwardly deduce the result for all formulas, by induction on the bound n . That way, we obtain that $|\text{RelForm}(\text{Prop}, B)|$ could be exponential in the bound n , but this does not have an impact complexity-wise, since n is fixed.

Recipe for the NP-hardness proofs. The formulas that we consider only use a bounded amount of binary operators. Thus, contrary to the NP-hardness reductions of Section 3, here, our NP-hardness proofs do not rely on binary operators. In fact, these binary operators make it harder to argue about how the permitted unary operators interact. For this reason, our proof of NP-hardness is decomposed into two steps. We first exhibit reductions for the learning problems without binary operators. Then, from these reductions, we devise reductions for the learning problems with bounded occurrences of binary operators. We present in details the former reductions in this paper and give intuition behind the later reductions below.

Let $n \in \mathbb{N}$ and $*$ $\in \text{Op}_{\text{Bin}}^{\text{lg}}$ be a binary (non-temporal) operator. We consider n propositions $\{p_1, \dots, p_n\}$ and we define multiple size-1 structures using the propositions $\{p_1, \dots, p_n\}$ forming two sets $\mathcal{A}_{n,*}$ and $\mathcal{B}_{n,*}$. The idea is that to distinguish these two sets, a separating formula will necessarily feature all the propositions $\{p_1, \dots, p_n\}$. In fact, from a positive and a negative sets of structures \mathcal{P} and \mathcal{N} on the set of proposition $\{p\}$ ² (which is the only proposition that unary formulas can use in our reductions), we can show the following: if a formula of size at most $B + 2n$, with at most n occurrences of binary operators, separates both \mathcal{P} and \mathcal{N} , and $\mathcal{A}_{n,*}$ and $\mathcal{B}_{n,*}$, then there is a unary formula of size at most B that separates \mathcal{P} and \mathcal{N} .³ That way, a reduction for the learning problem without binary operators can be translated (in logspace) into a reduction for the learning with bounded occurrences of binary operators. Note that the arguments presented in this paragraph are not straightforward to formally state and prove (this is handled in Theorem “Proving NP-hardness without binary operators is sufficient” in the extended version [8]).

Let us now consider how we handle the reduction without binary operators. From an instance (l, C, k) of the hitting problem, we proceed as follows. We define a sample of structures (and a bound B) such that all separating formulas have a specific shape, and there is a bijection between subsets $H \subseteq [1, \dots, l]$ and formulas $\varphi(l, H)$ of that specific shape. This correspondence allows us to extract a hitting set. More specifically, we follow the abstract recipe below:

- (a) We define the bound B and positive and negative structures that “eliminate” certain operators or pattern of operators from any potential separating formula. This way we ensure that any separating formula will be of the form $\varphi(l, H)$, for some $H \subseteq [1, \dots, l]$.
- (b) We define a negative structure satisfied by a formula $\varphi(l, H)$ if and only if $|H| \geq k + 1$.
- (c) For all $1 \leq i \leq n$, we define a positive structure that a formula $\varphi(l, H)$ accepts if and only if $H \cap C_i \neq \emptyset$.

² In fact, for technical reason, in [8], we use two propositions $\{p, \bar{p}\}$.

³ Actually, we can also show the converse (which is important for us to prove that the reduction is correct).

19:10 Learning LTL, CTL and ATL Formulas

By construction, the instance of the learning decision problem that we obtain is a positive instance if and only if the hitting set instance (l, C, k) also is. Furthermore, note that in all three cases, this reduction can be computed in logspace.

4.3 LTL learning

We start with LTL learning. We have the proposition below.

► **Proposition 7.** *For all sets of unary operators $U^t \subseteq \text{Op}_{U_n}$, sets of binary (non-temporal) operators $B^l \subseteq \text{Op}_{B_{in}}^{lg}$, and $n \in \mathbb{N}$, the decision problem $\text{LTL}_{\text{Learn}}(U^t, \emptyset, B^l, n)$ is in L.*

We present Steps A and B of Section 4.2 in the case $n = 0$. Toward Step A, we have the equivalences below (see e.g. [27, Prop. 8]), which imply the corollary that follows.

► **Observation 8.** *For all LTL-formulas φ and $k \in \mathbb{N}$, we have: 1) $\mathbf{F} \mathbf{X}^k \varphi \equiv \mathbf{X}^k \mathbf{F} \varphi$, $\mathbf{G} \mathbf{X}^k \varphi \equiv \mathbf{X}^k \mathbf{G} \varphi$; 2) $\mathbf{F} \mathbf{F} \varphi \equiv \mathbf{F} \varphi$, $\mathbf{G} \mathbf{G} \varphi \equiv \mathbf{G} \varphi$; 3) $\mathbf{F} \mathbf{G} \mathbf{F} \varphi \equiv \mathbf{G} \mathbf{F} \varphi$, $\mathbf{G} \mathbf{F} \mathbf{G} \varphi \equiv \mathbf{F} \mathbf{G} \varphi$.*

► **Corollary 9.** *Consider a set of propositions Prop . We let $\text{Lit}(\text{Prop}) := \{x, \neg x \mid x \in \text{Prop}\}$ and $\text{LTL}_{U_n}(\text{Prop}) := \{\mathbf{X}^k x, \mathbf{X}^k \mathbf{F} x, \mathbf{X}^k \mathbf{G} x, \mathbf{X}^k \mathbf{F} \mathbf{G} x, \mathbf{X}^k \mathbf{G} \mathbf{F} x \mid k \in \mathbb{N}, x \in \text{Lit}(\text{Prop})\}$.*

Then, for any LTL-formula $\varphi \in \text{LTL}(\text{Prop}, \text{Op}_{U_n}, \emptyset, B^l, 0)$, there is an LTL-formula $\varphi' \in \text{LTL}_{U_n}(\text{Prop}) \cap \text{LTL}(\text{Prop}, \text{Op}_{U_n}, \emptyset, B^l, 0)$ such that $\varphi \equiv \varphi'$ and $|\varphi'| \leq |\varphi|$.

Proof sketch. With the equivalences 1) from Observation 8, we can push the \mathbf{X} operators in φ at the beginning of the formula. The equivalences 2) and 3) from Observation 8 ensure that it is possible to have at most two nested \mathbf{F}, \mathbf{G} operators in the resulting formula φ' . ◀

The set of relevant formulas $\text{RelForm}(\text{Prop}, B)$ is then obtained directly from the set of formulas $\text{LTL}_{U_n}(\text{Prop})$. Note that however, how it is obtained depends on the exact operators in U^t . For instance, if $\mathbf{G} \notin U^t$ while $\neg, \mathbf{F} \in U^t$, we should replace the occurrences of \mathbf{G} in formulas in $\text{RelForm}(\text{Prop}, B)$ by $\neg \mathbf{F} \neg$. Nonetheless, in any case, we obtain a set $\text{RelForm}(\text{Prop}, B)$ of relevant formulas whose number of elements is linear in $|\text{Prop}| \cdot B$. This concludes the arguments for Step A. As for Step B, one can realize that since there are at most two nested \mathbf{F}, \mathbf{G} operators in formulas in $\text{RelForm}(\text{Prop}, B)$, then checking that they hold on a linear structure can be done in logarithmic space (because it suffices to have a constant number of pointers browsing the structure).

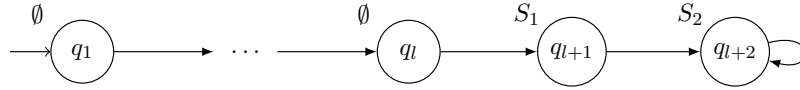
4.4 CTL learning

Consider now the more involved case of CTL learning. As can be seen in Table 1, we distinguish two cases: with and without the operator \mathbf{X} .

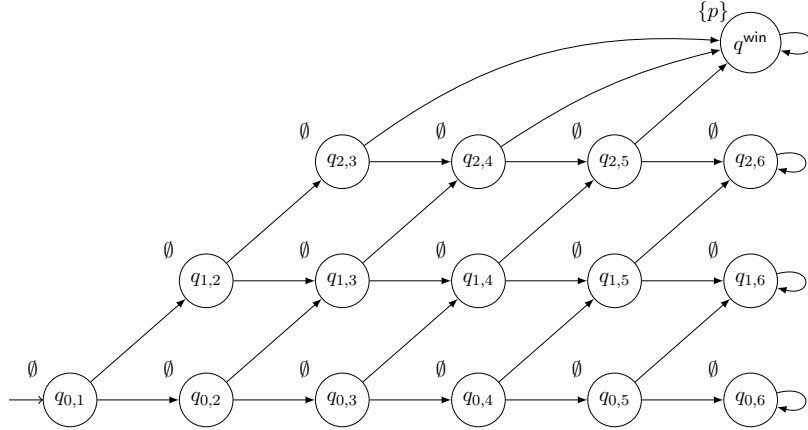
Assume that $\mathbf{X} \in U^t$. The goal is to show the theorem below.

► **Theorem 10.** *For all sets $U^t \subseteq \text{Op}_{U_n}$, $B^l \subseteq \text{Op}_{B_{in}}^{lg}$, and bound $n \in \mathbb{N}$, if $\mathbf{X} \in U^t$, then the decision problem $\text{CTL}_{\text{Learn}}(U^t, \emptyset, B^l, n)$ is NP-hard.*

As stated in Section 4.2, we argue the theorem in the case $n = 0$. Recall that in that case we consider a single proposition $\{p\}$. Consider an instance (l, C, k) of the hitting set problem Hit. We follow the three Steps a, b, and c. Toward Step a, we define Kripke structures that prevent the use of the operators $\mathbf{F}, \mathbf{G}, \neg$. To do so, we let $B := l + 1$ and for two sets $S_1, S_2 \subseteq \{p\}$, we consider the Kripke structure K_{l, S_1, S_2} that is depicted in Figure 1. These structures satisfy the lemma below.



■ **Figure 1** The structure K_{l,S_1,S_2} where $S_1 \subseteq \{p\}$ (resp. $S_2 \subseteq \{p\}$) labels q_{l+1} (resp. q_{l+2}).



■ **Figure 2** The Kripke structure $K_{\exists > 2}^5$.

► **Lemma 11.** *A formula $\phi \in \text{CTL}(\{p\}, \text{U}^t, \emptyset, \text{B}^l, 0)$ of size at most $l + 1$ accepting $K_{l,\{p\},\emptyset}$ and rejecting $K_{l,\emptyset,\{p\}}$ and $K_{l,\emptyset,\emptyset}$ cannot use the operators $\mathbf{F}, \mathbf{G}, \neg$.*

Proof sketch. Consider an equivalent CTL-formula ϕ' with $|\phi'| \leq |\phi|$ where negations, if any, occur right before the proposition p . Then, if ϕ' uses the operator \mathbf{G} , it cannot distinguish the structures $K_{l,\{p\},\emptyset}$ and $K_{l,\emptyset,\emptyset}$. Otherwise, if it uses the operator \mathbf{F} , it cannot distinguish the structures $K_{l,\{p\},\emptyset}$ and $K_{l,\emptyset,\{p\}}$. Otherwise, since $K_{l,\{p\},\emptyset}, K_{l,\emptyset,\{p\}}$, and $K_{l,\emptyset,\emptyset}$ coincide on the first l states, ϕ' has to use at least l operators \mathbf{X} . Since $|\phi'| \leq l + 1$, it cannot use a negation. Thus ϕ' does not use $\mathbf{F}, \mathbf{G}, \neg$, and neither does ϕ . ◀

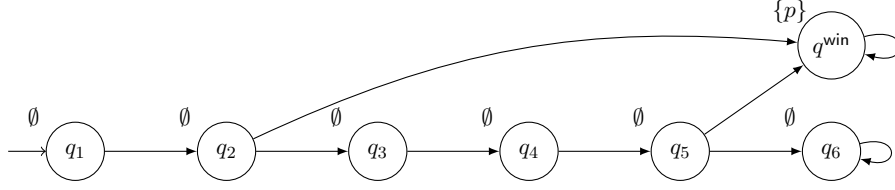
In fact, a CTL-formula $\phi \in \text{CTL}(\{p\}, \text{U}^t, \emptyset, \text{B}^l, 0)$ of size at most $l + 1$ accepting $K_{l,\{p\},\emptyset}$ and rejecting $K_{l,\emptyset,\{p\}}, K_{l,\emptyset,\emptyset}$ necessarily uses exactly l operators \mathbf{X} followed by the proposition p . Such a formula is therefore entirely defined by the \mathbf{X} operators before which it uses the \exists quantifier. This suggests the definition below of the CTL-formula $\phi(l, H)$ induced by a subset $H \subseteq [1, \dots, l]$.

► **Definition 12.** *For all $H \subseteq [1, \dots, l]$, we let $\phi(l, H) \in \text{CTL}(\{p\}, \text{U}^t, \emptyset, \text{B}^l, 0)$ denote the CTL-formula defined by $\phi(l, H) := Q_1 \mathbf{X} \dots Q_l \mathbf{X} p$ where, for all $1 \leq i \leq l$, we have $Q_i \in \{\exists, \forall\}$ and $Q_i = \exists$ if and only if $i \in H$.*

For $1 \leq i \leq l + 1$, we let $\phi_i(l, H) := Q_i \mathbf{X} \dots Q_l \mathbf{X} p$ (with $\phi_{l+1}(l, H) := p$).

Let us now turn toward Step b. We define a structure $K_{\exists > k}^l$ with $k + 2$ different levels, where: the single starting state $q_{0,1}$ is at the bottommost level; the proposition p only labels the state q^{win} at the topmost level; and every state of the bottom $k + 1$ levels has a successor at the same level and one level higher. That way, going from $q_{0,1}$ to q^{win} is equivalent to leveling up $k + 1$ times. Furthermore, the top most level can be reached in at most l . An example is depicted in Figure 2. This structure satisfies the lemma below.

► **Lemma 13.** *For all $H \subseteq [1, \dots, l]$, we have $K_{\exists > k}^l \models \phi(l, H)$ if and only if $|H| > k$.*



■ **Figure 3** The Kripke structure $K_{5, \{2,5\}}$.

Consider now Step c. For $C \subseteq [1, \dots, l]$, we define the Kripke structure $K(l, C)$ with $\{q_1, \dots, q_l, q_{l+1}, q^{\text{win}}\}$ as set of states where q^{win} is the only state labeled with p ; and for all $1 \leq j \leq l$, q_j branches to q_{j+1} and, if (and only if) $j \in C$, q_j also branches to q^{win} , as exemplified in Figure 3. Such structures satisfy the lemma below.

► **Lemma 14.** *For all $C, H \subseteq [1, \dots, l]$, we have $K_{(l,C)} \models \phi(l, H)$ if and only if $C \cap H \neq \emptyset$.*

Proof sketch. We can show by induction on $l + 1 \geq j \geq 1$ the property $\mathcal{P}(j)$: $q_j \models \phi_j(l, H)$ if and only if $H \cap [j, \dots, l] \cap C \neq \emptyset$. The lemma is then given by $\mathcal{P}(1)$. ◀

We can finally define the reduction that we consider. We let $\text{In}^{\text{CTL}} := (\{p\}, \mathcal{P}, \mathcal{N}, B)$, with $B := l + 1$, $\mathcal{P} := \{K_{l, \{p\}, \emptyset}, K_i \mid 1 \leq i \leq n\}$ and $\mathcal{N} := \{K_{l, \emptyset, \emptyset}, K_{l, \emptyset, \{p\}}, K_{\exists > k}^l\}$, be an input of the decision problem $\text{CTL}_{\text{Learn}}(\text{U}^t, \emptyset, \text{B}^l, 0)$. By Lemmas 11, 13, 14, In^{CTL} is a positive instance of the decision problem $\text{CTL}_{\text{Learn}}(\text{U}^t, \emptyset, \text{B}^l, 0)$ if and only if (l, C, k) is a positive instance of the decision problem Hit. Theorem 10 follows (in the case $n = 0$).

Assume that $X \notin \text{U}^t$. In that case, the CTL learning problem is now in NL.

► **Theorem 15.** *For all sets of operators $\text{U}^t \subseteq \{\mathbf{F}, \mathbf{G}, \neg\}$, $\text{B}^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$, and bounds $n \in \mathbb{N}$, the decision problem $\text{CTL}_{\text{Learn}}(\text{U}^t, \emptyset, \text{B}^l, n)$ is in NL.*

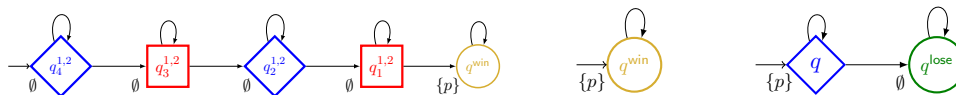
Toward Step A, a crucial observation is that using the operators \mathbf{F} or \mathbf{G} twice in a row is useless. This is stated in the lemma below in the context of ATL-formula because this lemma will be used again in the next subsection.

► **Lemma 16.** *Let $I \subseteq J \subseteq \mathbb{N}$, and ϕ be an ATL-formula. We have:*

$$\langle\langle J \rangle\rangle \mathbf{F} \phi \equiv \langle\langle I \rangle\rangle \mathbf{F} \langle\langle J \rangle\rangle \mathbf{F} \phi \equiv \langle\langle J \rangle\rangle \mathbf{F} \langle\langle I \rangle\rangle \mathbf{F} \phi \quad \langle\langle I \rangle\rangle \mathbf{G} \phi \equiv \langle\langle I \rangle\rangle \mathbf{G} \langle\langle J \rangle\rangle \mathbf{G} \phi \equiv \langle\langle J \rangle\rangle \mathbf{G} \langle\langle I \rangle\rangle \mathbf{G} \phi$$

Proof sketch. We argue the result for \mathbf{F} , the case of \mathbf{G} is dual. We have $\langle\langle J \rangle\rangle \mathbf{F} \phi \implies \langle\langle I \rangle\rangle \mathbf{F} \langle\langle J \rangle\rangle \mathbf{F} \phi$ by definition of \mathbf{F} . Furthermore, if a state q satisfies $\langle\langle I \rangle\rangle \mathbf{F} \langle\langle J \rangle\rangle \mathbf{F} \phi$ then there is a strategy s_I for the coalition I such that eventually a state satisfying $\langle\langle J \rangle\rangle \mathbf{F} \phi$ is surely reached. For all such states q , we consider a strategy s_J^q for the coalition J ensuring to eventually visit a state satisfying ϕ . Then, consider a strategy s'_J for the coalition J that mimics s_I (which is possible since $I \subseteq J$) until a state q satisfying $\langle\langle J \rangle\rangle \mathbf{F} \phi$ is reached, and then switches to the strategy s_J^q . That strategy ensures eventually reaching a state satisfying ϕ . Therefore, $\langle\langle I \rangle\rangle \mathbf{F} \langle\langle J \rangle\rangle \mathbf{F} \phi \implies \langle\langle J \rangle\rangle \mathbf{F} \phi$. This is similar for $\langle\langle J \rangle\rangle \mathbf{F} \langle\langle I \rangle\rangle \mathbf{F} \phi$. ◀

From this, we can actually deduce (this is not direct) that there is a bound $M \in \mathbb{N}$ such that, for any set of propositions Prop and for all $\text{U}^t \subseteq \{\mathbf{F}, \mathbf{G}, \neg\}$, given any $\text{CTL}(\text{Prop}, \text{U}^t, \emptyset, \text{B}^l, 0)$ -formula ϕ , there is an equivalent $\text{CTL}(\text{Prop}, \text{U}^t, \emptyset, \text{B}^l, 0)$ -formula ϕ' , with $|\phi'| \leq M$. Thus, the number of CTL-formulas to consider is linear in $|\text{Prop}|$. As for Step B, consider any such formula ϕ . Since the number of quantifiers it uses is bounded by M and $\text{NL} = \text{coNL}$, we deduce that checking that it satisfies a Kripke structure can be done in NL.



■ **Figure 4** The turn-based structure $T_{4:1,2}$. ■ **Figure 5** On the left T_p , on the right $T_{no 2 G}$.

Proof of NL-hardness. In Table 1, we do not state only that CTL learning without the operator \mathbf{X} is in NL, but also that it is NL-hard. Proving this result is actually straightforward. We exhibit a reduction from the problem of reachability in a graph (which is NL-complete [23]). Given an input (\mathcal{G}, s, t) of that problem, with \mathcal{G} a graph, s the source state and t the target state, we define a positive Kripke structure K that is obtained from \mathcal{G} by making s its only initial state, and t the only state labeled by the proposition p . Additionally, we consider $B := 2$ as the bound, and with an additional structure, we ensure that if there is a separating formula, then the formula $\phi := \exists \mathbf{F} p$ is separating.

4.5 ATL learning

We have seen that CTL learning with the operator \mathbf{X} is NP-hard, which implies that it is also the case for ATL learning. Here, we consider the case of ATL learning without the operator \mathbf{X} . First, let us informally explain why the NP-hardness reduction that we have described above for CTL cannot possibly work without the operator \mathbf{X} . A central aspect of the proof of Lemma 14 is to be able to associate a specific operator in a prospective formula with a specific state in a Kripke structure. That is intrinsically not possible with the operator \mathbf{F} since this operator looks at arbitrarily distant horizons. At least, this is true with CTL-formulas interpreted on Kripke structures. However, with ATL-formulas interpreted on turn-based structures, it is possible to “block the horizon” of \mathbf{F} operators. Indeed, consider the structure of Figure 4, where blue lozenge-shaped states are Agent-1 states, and red square-shaped states are Agent-2’s. Here, one can see that $q_2^{1,2} \not\models \langle\langle 1 \rangle\rangle \mathbf{F} p$ because Agent 2 can enforce to loop on the Agent-2 state $q_1^{1,2}$ and not see the state q^{win} , labeled by p .

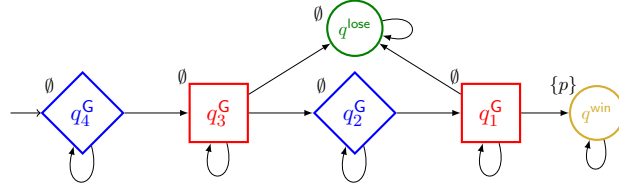
These kinds of turn-based games will be extensively used in the following. In all generality, there are defined as follows: given a pair of agents $i \neq j$ and $l \in \mathbb{N}$, in the turn-based structure $T_{l:i,j}$, there are $l + 1$ self-looping states, alternatively belonging to Agents i and j , that can get closer and closer to the self-looping sink q^{win} , the only state labeled by p . In fact, such structures are linked to alternating-formulas, defined below.

► **Definition 17.** An ATL-formula is positive if it does not use any negation. For a pair of agents $i \neq j$ and $l \in \mathbb{N}$, a positive ATL-formula ϕ is (i, j) -free if it does not use an operator $\langle\langle A \rangle\rangle \mathbf{F}$ with $i, j \in A$. It is (i, j, l) -alternating if it is (i, j) -free and if there are at least l alternating occurrences of operators $\langle\langle A_i \rangle\rangle \mathbf{F}$ with $i \in A_i$ and $\langle\langle A_j \rangle\rangle \mathbf{F}$ with $j \in A_j$.

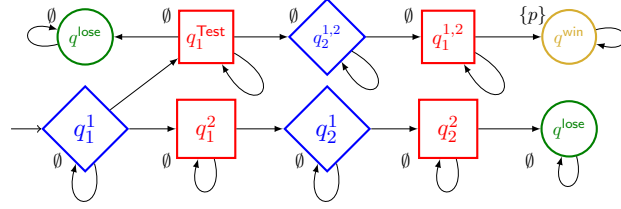
► **Lemma 18.** Consider two agents $i \neq j$, $l \in \mathbb{N}$, and a positive ATL-formula ϕ that is (i, j) -free. The formula ϕ accepts the structure $T_{l:i,j}$ if and only if it is (i, j, l) -alternating.

ATL² learning with $\{\mathbf{F}, \mathbf{G}\} \subseteq \mathbf{U}^t$. Here, all the turn-based structures that we consider use the set of agents $\text{Ag} = \{1, 2\}$. The goal is to show the theorem below.

► **Theorem 19.** For all sets $\mathbf{U}^t \subseteq \text{Op}_{\mathbf{U}^t}$, $\mathbf{B}^l \subseteq \text{Op}_{\mathbf{B}^l}^{\text{lg}}$, and bound $n \in \mathbb{N}$, if $\{\mathbf{F}, \mathbf{G}\} \subseteq \mathbf{U}^t$ and $\mathbf{X} \notin \mathbf{U}^t$, then the decision problem $\text{ATL}_{\text{Learn}}^2(\mathbf{U}^t, \emptyset, \mathbf{B}^l, n)$ is NP-hard.



■ **Figure 6** The structure $T_{no\ 1\ G \geq 2}$.



■ **Figure 7** The structure $T_{2, \{1\}, 2}$.

In the following, to ease the notations, the strategic operators $\langle\langle\emptyset\rangle\rangle$, $\langle\langle\{1\}\rangle\rangle$, $\langle\langle\{2\}\rangle\rangle$, $\langle\langle\{1, 2\}\rangle\rangle$ will simply be denoted \emptyset , 1, 2 and 1, 2 respectively. Consider an instance (l, C, k) of the hitting set problem. We follow the recipe of Subsection 4.2. Here, we want separating formulas to be promising, i.e. to only use the operators $1\mathbf{F}$, $2\mathbf{F}$ and $1\mathbf{G}$. To this end, all the structures we use are self-looping, thus making the operators $\emptyset\mathbf{F}$ and $1, 2\mathbf{G}$ useless.

► **Lemma 20.** *For all ATL-formulas ϕ and self-looping states q , we have: $q \models \phi$ if and only if $q \models \emptyset\mathbf{F}\phi$ if and only if $q \models 1, 2\mathbf{G}\phi$*

Proof. Since q is self-looping the coalition of agents $\{1, 2\}$ has a strategy s such that $\text{Out}(q, s) = \{q^\omega\}$. The lemma follows from the definition of the operators \mathbf{F} and \mathbf{G} . ◀

We also consider the two structures $T_p, T_{no\ 2\mathbf{G}}$, of Figure 5 satisfying the lemma below.

► **Lemma 21.** *For all ATL-formulas $\phi \in \text{ATL}(\{p\}, \mathbf{U}^t, \emptyset, \mathbf{B}^1, 0)$ accepting $T_p, T_{no\ 2\mathbf{G}}$ and rejecting $T_{2l+1;1,2}$, there is a promising formula $\phi' \in \text{ATL}(\{p\}, \mathbf{U}^t, \emptyset, \mathbf{B}^1, 0)$ with $|\phi'| \leq |\phi|$ that is equivalent to ϕ on self-looping structures.*

Proof sketch. Consider an ATL-formula ϕ' equivalent to ϕ with $|\phi'| \leq |\phi|$ and with at most one negation occurring before the proposition p . Since ϕ' accepts T_p , it follows that it is positive. By Lemma 20, we can remove the operators $1, 2\mathbf{G}$ and $\emptyset\mathbf{F}$ from ϕ' . Furthermore: ϕ' cannot use $\emptyset\mathbf{G}, 2\mathbf{G}$, since it accepts $T_{no\ 2\mathbf{G}}$, and it cannot use $1, 2\mathbf{F}$ since it rejects $T_{2l+1;1,2}$. It is therefore promising. ◀

We will also consider $T_{2l;1,2}$ as a positive structure, thus allowing us to focus on $(1, 2, 2l)$ -alternating formulas (recall Lemma 18). Then, we want to associate to a subset $H \subseteq [1, \dots, l]$ a promising $(1, 2, 2l)$ -alternating ATL-formula. To get an intuition, let us consider the turn-based structure $T_{no\ 1\mathbf{G} \geq t}$ for $t = 2$ of Figure 6. This structure $T_{no\ 1\mathbf{G} \geq t}$ is analogous to the structure $T_{2t;1,2}$ except that all Agent-2 states have an additional successor: the state q^{lose} that does not satisfy any positive formula. Back to the structure of Figure 6, because Agent 2 owns the states q_3^G, q_1^G , these states do not accept any positive ATL-formula of the shape $1\mathbf{G}\phi$. Therefore, for all $q \in \{q_4^G, q_2^G\}$ and positive ATL-formulas ϕ , we have $q \models 1\mathbf{F}1\mathbf{G}2\mathbf{F}\phi$ if and only if $q \models \phi$. This actually implies that a $(1, 2, 2l)$ -alternating formula ϕ accepts

$T_{\text{no } 1\mathbf{G} \geq 2}$ if and only if the sequence of operators $1\mathbf{F}2\mathbf{F}$ (without $1\mathbf{G}$ in between) occurs at least twice in ϕ (to go from $q_4^{\mathbf{G}}$ to $q_2^{\mathbf{G}}$ and then from $q_2^{\mathbf{G}}$ to q^{win}). In fact, we consider formulas that only use $1\mathbf{G}$ operators after $1\mathbf{F}$ and before $2\mathbf{F}$, as defined below. Such formulas satisfy the lemma that follows.

► **Definition 22.** For all $H \subseteq [1, \dots, l]$, we let $\phi(l, H, 2) \in \text{ATL}^2(\{p\}, \mathbf{U}^t, \emptyset, \mathbf{B}^l, 0)$ denote the ATL-formula defined by: $\phi(l, H, 2) := 1\mathbf{F}Q_12\mathbf{F} \cdots 1\mathbf{F}Q_l2\mathbf{F}p$ where, for all $1 \leq i \leq l$, we have $Q_i \in \{\epsilon, 1\mathbf{G}\}$ and $Q_i = 1\mathbf{G}$ iff $i \notin H$.

For all $1 \leq i \leq l+1$, we let $\phi_i(l, H, 2) := 1\mathbf{F}Q_i2\mathbf{F} \cdots 1\mathbf{F}Q_l2\mathbf{F}p$ (with $\phi_{l+1}(l, H, 2) := p$).

► **Lemma 23.** A promising $(1, 2, 2l)$ -alternating formula ϕ with $|\phi| \leq 3l + 1 - k$ rejects $T_{\text{no } 1\mathbf{G} \geq k+1}$ if and only if $\phi = \phi(l, H, 2)$ for some $H \subseteq [1, \dots, l]$ such that $|H| = k$.

Proof sketch. Since ϕ is $(1, 2, 2l)$ -alternating, it uses at least l operators $1\mathbf{F}$ and $2\mathbf{F}$. Thus, it can use at most $l - k$ operators $1\mathbf{G}$. In addition, ϕ accepts $T_{\text{no } 1\mathbf{G} \geq k+1}$ iff there are at least $k + 1$ occurrences of the sequence $1\mathbf{F}2\mathbf{F}$ in ϕ . Thus, ϕ uses each $l - k$ remaining operators $1\mathbf{G}$ between a different pair of successive $1\mathbf{F}, 2\mathbf{F}$ iff it rejects $T_{\text{no } 1\mathbf{G} \geq k+1}$. ◀

With $T_p, T_{\text{no } 2\mathbf{G}}, T_{2l:1,2}$ as positive structures and $T_{2l+1:1,2}, T_{\text{no } 1\mathbf{G} \geq k+1}$ as negative structures, we have achieved both Steps a and b. Let us turn to Step c. For all $C \subseteq [1, \dots, l]$, we define a turn-based structure $T_{l,C,2}$. An example is depicted in Figure 7 for $l = 2$. The structure $T_{l,C,2}$ features a sequence of states $q_1^1, q_1^2, \dots, q_l^1, q_l^2$ alternating between Agent-1 and Agent-2 states ending in a self-looping sink q^{lose} not labeled by p . However, the Agent-1 states q_i^1 for which $i \in C$ have a “testing state” q_i^{Test} as successor. That state is self-looping, and may branch to the self-looping sink q^{lose} or to the structure $T_{2(l-i):1,2}$. That state is such that $q_i^{\text{Test}} \models Q_i 2\mathbf{F} \phi_{i+1}(l, H, 2)$ iff $Q_i = \epsilon$ (iff $i \in H$). Furthermore, note that it is useless to “wait” at the state q_i^1 before branching to q_i^{Test} . Indeed, if for instance $Q_i = 1\mathbf{G}$ but $Q_{i+1} = \epsilon$, then it may seem that $q_i^{\text{Test}} \models \varphi'$, for $\varphi' := Q_{i+1}2\mathbf{F} \phi_{i+2}(l, H, 2)$ and therefore $q_i^1 \models \phi_i(l, H, 2) = 1\mathbf{F}Q_i2\mathbf{F}1\mathbf{F}\varphi'$. However, it is not the case because we do not have $q_i^{\text{Test}} \models \varphi'$, since $\phi_{i+2}(l, H, 2)$ is not $(1, 2, 2(l-i))$ -alternating, and thus it does not satisfy the structure $T_{2(l-i):1,2}$. Overall, we have the lemma below.

► **Lemma 24.** For all $C, H \subseteq [1, \dots, l]$, we have $T_{(l,C)} \models \phi(l, H, 2)$ if and only if $C \cap H = \emptyset$.

We have achieved Step c. Then, we let $\text{In}^{\text{ATL}^2} := (\{p\}, \mathcal{P}, \mathcal{N}, B)$ be an input of the decision problem $\text{ATL}_{\text{Learn}}^2(\mathbf{U}^t, \emptyset, \mathbf{B}^l, 0)$ where $\mathcal{P} := \{T_p, T_{\text{no } 2\mathbf{G}}, T_{2l:1,2}, T_{(l,C_i,2)} \mid 1 \leq i \leq n\}$, $\mathcal{N} := \{T_{2l+1:1,2}, T_{\text{no } 1\mathbf{G} \geq k+1}\}$, and $B := 3l + 1 - k$. By Lemmas 21, 23 and 24, In^{ATL^2} is a positive instance of $\text{ATL}_{\text{Learn}}^2(\mathbf{U}^t, \emptyset, \mathbf{B}^l, 0)$ iff (l, C, k) is a positive instance of Hit.

ATL² learning with $\mathbf{U}^t = \{\mathbf{F}\}$ or $\mathbf{U}^t = \{\mathbf{G}\}$. The ATL² learning problem is now in P.

► **Theorem 25.** For all sets of operators $\mathbf{U}^t \in \{\{\mathbf{F}\}, \{\mathbf{G}\}\}$, $\mathbf{B}^l \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$, and bounds $n \in \mathbb{N}$, the decision problem $\text{ATL}_{\text{Learn}}^2(\mathbf{U}^t, \emptyset, \mathbf{B}^l, n)$ is in P.

We focus on the case $\mathbf{U}^t = \{\mathbf{F}\}$, the other is analogous. Towards Step A, consider a formula $\phi \in \text{ATL}^2(\text{Prop}, \{\mathbf{F}\}, \emptyset, \mathbf{B}^l, 0)$ and the only proposition $p \in \text{Prop}$ occurring in ϕ . By Lemma 16, we can make the following observations: 1) If the operator $1, 2\mathbf{F}$ occurs in ϕ , then $\phi \equiv 1, 2\mathbf{F}p$; 2) Otherwise, if the only operator occurring in ϕ is $\emptyset\mathbf{F}$ then $\phi \equiv \emptyset\mathbf{F}p$; 3) Otherwise, ϕ is equivalent to a formula ϕ' alternating between the operators $1\mathbf{F}$ and $2\mathbf{F}$, with $|\phi'| \leq |\phi|$. These observations suggest the definition below, which satisfies the lemma that follows.

► **Definition 26.** For a set of propositions Prop , we define the set $\text{ATL}_{\mathbf{F}}^2(\text{Prop}) := \{\text{Qt} \cdot p \mid p \in \text{Prop}, \text{Qt} \in \text{Quant}_{\text{Alt}}^{\mathbf{F}}\}$ where $\text{Quant}_{\text{Alt}}^{\mathbf{F}} := \{\epsilon, \emptyset \mathbf{F}, 1, 2 \mathbf{F}, (1 \mathbf{F} \cdot 2 \mathbf{F})^*, (1 \mathbf{F} \cdot 2 \mathbf{F})^* \cdot 1 \mathbf{F}, (2 \mathbf{F} \cdot 1 \mathbf{F})^*, (2 \mathbf{F} \cdot 1 \mathbf{F})^* \cdot 2 \mathbf{F}\}$.

► **Lemma 27.** For a set of propositions Prop , and $\phi \in \text{ATL}^2(\text{Prop}, \{\mathbf{F}\}, \emptyset, \mathbf{B}^1, 0)$, there is an ATL-formula $\phi' \in \text{ATL}_{\mathbf{F}}^2(\text{Prop})$ such that $\phi \equiv \phi'$ and $|\phi'| \leq |\phi|$.

This concludes Step A since the number of formulas of size at most B in $\text{ATL}_{\mathbf{F}}^2(\text{Prop})$ is polynomial in B and $|\text{Prop}|$. As for Step B, in this case it is trivial since checking that an ATL-formula satisfies a structure can always be done in polynomial time.

Proof of P-hardness. In Table 1, we additionally state only that ATL^2 learning with $\mathbf{U}^t \in \{\{\mathbf{F}\}, \{\mathbf{G}\}\}$ is P-hard. The proof of this fact is actually very similar to the proof that CTL learning without the operator \mathbf{X} is NL-hard, except that the reduction is made from the problem of reachability in a turn-based game (which is P-complete [31]).

ATL³ learning with $\mathbf{U}^t \in \{\{\mathbf{F}\}, \{\mathbf{G}\}\}$. Let us consider ATL learning with one more agent, i.e. ATL^3 learning, still with $\mathbf{U}^t \in \{\{\mathbf{F}\}, \{\mathbf{G}\}\}$. The turn-based structures that we consider now use the set of agents $\text{Ag} = \{1, 2, 3\}$. The goal is to show the theorem below.

► **Theorem 28.** For all sets $\mathbf{U}^t \in \{\{\mathbf{F}\}, \{\mathbf{G}\}\}$, $\mathbf{B}^1 \subseteq \text{Op}_{\text{Bin}}^{\text{lg}}$, and bound $n \in \mathbb{N}$, the decision problem $\text{ATL}_{\text{Learn}}^3(\mathbf{U}^t, \emptyset, \mathbf{B}^1, n)$ is NP-hard.

We focus on the case $\mathbf{U}^t = \{\mathbf{F}\}$ (the case $\mathbf{U}^t = \{\mathbf{G}\}$ is analogous since the operators \mathbf{F} and \mathbf{G} have a dual behavior). Once again, let us consider an instance (l, C, k) of the problem Hit. We start right away by defining the ATL^3 -formula associated to a subset $H \subseteq [1, \dots, l]$.

► **Definition 29.** For $H \subseteq [1, \dots, l]$, we let $\phi(l, H, 3)$ denote the ATL^3 -formula defined by $\phi(l, H, 3) := 1 \mathbf{F} \langle \langle A_1 \rangle \rangle \mathbf{F} \cdots 1 \mathbf{F} \langle \langle A_l \rangle \rangle \mathbf{F} p$ where, for all $1 \leq i \leq l$, we have $A_i \in \{\{2\}, \{2, 3\}\}$ and $A_i = \{2, 3\} \mathbf{F}$ if and only if $i \in H$.

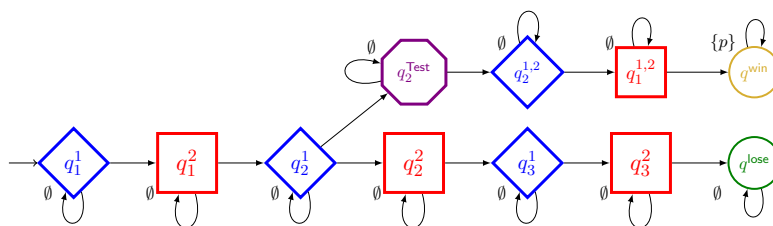
For $1 \leq i \leq l + 1$, we let $\phi_i(l, H, 3) := 1 \mathbf{F} \langle \langle A_i \rangle \rangle \mathbf{F} \cdots 1 \mathbf{F} \langle \langle A_l \rangle \rangle \mathbf{F} p$ (with $\phi_{l+1}(l, H, 3) = p$).

Toward Step a, we define $T_{2l+1:1,2}, T_{2(k+1):1,3}$ as negative structures, thus ensuring that a separating formula does not use an operator $\langle \langle A \rangle \rangle \mathbf{F}$ with $1, 2 \in A$, or $1, 3 \in A$. We also define $T_{2l:1,2}$ as a positive structure with the bound $B := 2l + 1$. That way, a separating formula is necessarily $(1, 2, 2l)$ -alternating and only uses the operators $1 \mathbf{F}, 2 \mathbf{F}$, and $2, 3 \mathbf{F}$.

► **Lemma 30.** If a formula $\phi \in \text{ATL}^3(\{\{p\}, \{\mathbf{F}\}, \emptyset, \mathbf{B}^1, n)$ with $|\phi| \leq 2l + 1$ accepts $T_{2l:1,2}$ and rejects $T_{2l+1:1,2}, T_{2(k+1):1,3}$, then there is some $H \subseteq [1, \dots, l]$ such that $\phi = \phi(l, H, 3)$.

Note that, if $|H| \geq k + 1$, then $\phi(l, H, 3)$ is $(1, 3, 2(k + 1))$ -alternating. Therefore, since $T_{2(k+1):1,3}$ is a negative structure, if $\phi(l, H, 3)$ is separating, then $|H| \leq k$, i.e. we have also achieved Step b. Let us now turn to Step c. For all $C \subseteq [1, \dots, l]$, we define the structure $T_{l,C,3}$. An example is given in Figure 8 with $l = 3$. This structure $T_{l,C,3}$ features a sequence of states $q_1^1, q_1^2, \dots, q_l^1, q_l^2$ alternating between Agent-1 and Agent-2 states ending in a self-looping sink q^{lose} . However, the Agent-1 states q_i^1 for which $i \in C$ have an Agent-3 “testing state” q_i^{test} as successor. That state is self-looping and also branches to the structure $T_{(l-i):1,2}$. Note that, given $r \geq i + 1$, the sub-formula $\phi_r(l, H, 3)$ is $(1, 2, l - r + 1)$ -alternating, and therefore satisfies the structure $T_{(l-i):1,2}$, iff $r = i + 1$. Thus, since q_i^{test} is an Agent-3 state, $q_i^{\text{test}} \models \langle \langle A_i \rangle \rangle \mathbf{F} \phi_{i+1}(l, H, 3)$ iff $3 \in A_i$ iff $i \in H$. Thus, we have the following lemma.

► **Lemma 31.** For all $C, H \subseteq [1, \dots, l]$, we have $T_{(l,C,3)} \models \phi(l, H, 3)$ if and only if $C \cap H \neq \emptyset$.



■ **Figure 8** The turn-based structure $T_{3,\{2\},3}$.

This concludes Step c. Overall, we let $\text{In}^{\text{ATL}^{(3),\mathbf{F}}} := (\{p\}, \mathcal{P}, \mathcal{N}, B)$ be an input of the decision problem $\text{ATL}_{\text{Learn}}^3(\{\mathbf{F}\}, \emptyset, \mathbf{B}^1, 0)$ where $\mathcal{P} := \{T_{2l:1,2}, T_{(l,C_i,3)} \mid 1 \leq i \leq n\}$, $\mathcal{N} := \{T_{2l+1:1,2}, T_{2(k+1):1,3}\}$, and $B := 2l + 1$. We have that $\text{In}^{\text{ATL}^{(3),\mathbf{F}}}$ is a positive instance of $\text{ATL}_{\text{Learn}}^3(\{\mathbf{F}\}, \emptyset, \mathbf{B}^1, 0)$ if and only if (l, C, k) is a positive instance of Hit.

5 Future Work

Within our setting, we have covered many cases, as can be seen in Table 1. That is why the complete version of this work [8] is already quite long. However, there are still some cases that we have not tackled. First, there is the case of ATL^2 learning with $\mathbf{U}^t \in \{\{\mathbf{F}, \neg\}, \{\mathbf{G}, \neg\}\}$. We believe that it behaves like the case $\mathbf{F}, \mathbf{G} \in \mathbf{U}^t$, but the proofs would entail many additional technical details, since replacing \mathbf{F} with $\neg \mathbf{G} \neg$ increases the size of the formulas.

More importantly, when considering a bounded amount of binary operators, we have not allowed binary temporal operators ($\mathbf{U}, \mathbf{R}, \mathbf{W}, \mathbf{M}$). Doing so would enhance the expressivity of the fragment that we consider, and we conjecture that we would obtain the same result as in this paper, with proofs that should be only moderately more involved.

On a more high level perspective, in this paper we have focused solely on solving exactly the learning problems and although we have found some relevant tractable cases, many are untractable. A promising research direction would be to look for tractable approximation algorithms, similarly to what is done in [27].

References

- 1 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002. doi:10.1145/585265.585270.
- 2 Dana Angluin. On the complexity of minimum inference of regular sets. *Inf. Control.*, 39(3):337–350, 1978. doi:10.1016/S0019-9958(78)90683-6.
- 3 M. Fareed Arif, Daniel Larraz, Mitziu Echeverria, Andrew Reynolds, Omar Chowdhury, and Cesare Tinelli. SYSLITE: syntax-guided synthesis of PLTL formulas from finite traces. In *FMCAD*, pages 93–103. IEEE, 2020. doi:10.34727/2020/ISBN.978-3-85448-042-6_16.
- 4 Ayca Balkan, Moshe Y. Vardi, and Paulo Tabuada. Mode-target games: Reactive synthesis for control applications. *IEEE Trans. Autom. Control.*, 63(1):196–202, 2018. doi:10.1109/TAC.2017.2722960.
- 5 Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory*, 24(3):384–386, 1978. doi:10.1109/TIT.1978.1055873.
- 6 Dines Bjørner and Klaus Havelund. 40 years of formal methods - some obstacles and some possibilities? In *FM*, volume 8442 of *Lecture Notes in Computer Science*, pages 42–61. Springer, 2014. doi:10.1007/978-3-319-06410-9_4.

- 7 Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotohi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC '16*, pages 1–10, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2883817.2883843.
- 8 Benjamin Bordais, Daniel Neider, and Rajarshi Roy. The complexity of learning temporal properties. *CoRR*, abs/2408.04486, 2024. doi:10.48550/arXiv.2408.04486.
- 9 Benjamin Bordais, Daniel Neider, and Rajarshi Roy. Learning branching-time properties in CTL and ATL via constraint solving. In André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *Formal Methods - 26th International Symposium, FM 2024, Milan, Italy, September 9-13, 2024, Proceedings, Part I*, volume 14933 of *Lecture Notes in Computer Science*, pages 304–323. Springer, 2024. doi:10.1007/978-3-031-71162-6_16.
- 10 Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6065–6073. ijcai.org, 2019. doi:10.24963/IJCAI.2019/840.
- 11 Alberto Camacho and Sheila A. McIlraith. Learning interpretable models expressed in linear temporal logic. In *ICAPS*, pages 621–630. AAAI Press, 2019. URL: <https://ojs.aaai.org/index.php/ICAPS/article/view/3529>.
- 12 Alberto Camacho, Eleni Triantafillou, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724. AAAI Press, 2017. doi:10.1609/AAAI.V31I1.11058.
- 13 Alessio Cecconi, Giuseppe De Giacomo, Claudio Di Ciccio, Fabrizio Maria Maggi, and Jan Mendling. Measuring the interestingness of temporal logic behavioral specifications in process mining. *Inf. Syst.*, 107:101920, 2022. doi:10.1016/J.IS.2021.101920.
- 14 William Chan. Temporal-logic queries. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 450–463. Springer, 2000.
- 15 Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logics of Programs, Workshop, Yorktown Heights, New York, USA, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981. doi:10.1007/BFB0025774.
- 16 Rüdiger Ehlers, Ivan Gavran, and Daniel Neider. Learning properties in $LTL \cap ACTL$ from positive examples only. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, pages 104–112. IEEE, 2020. doi:10.34727/2020/ISBN.978-3-85448-042-6_17.
- 17 Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In *ICRA*, pages 2020–2025. IEEE, 2005. doi:10.1109/ROBOT.2005.1570410.
- 18 Nathanaël Fijalkow and Guillaume Lagarde. The complexity of learning linear temporal formulas from examples. In Jane Chandlee, Rémi Eyraud, Jeff Heinz, Adam Jardine, and Menno van Zaanen, editors, *Proceedings of the 15th International Conference on Grammatical Inference, 23-27 August 2021, Virtual Event*, volume 153 of *Proceedings of Machine Learning Research*, pages 237–250. PMLR, 2021. URL: <https://proceedings.mlr.press/v153/fijalkow21a.html>.
- 19 Marie Fortin, Boris Konev, Vladislav Ryzhikov, Yury Savateev, Frank Wolter, and Michael Zakharyashev. Reverse engineering of temporal queries mediated by LTL ontologies. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3230–3238. ijcai.org, 2023. doi:10.24963/IJCAI.2023/360.

- 20 Jean-Raphaël Gaglione, Daniel Neider, Rajarshi Roy, Ufuk Topcu, and Zhe Xu. Maxsat-based temporal logic inference from noisy data. *Innov. Syst. Softw. Eng.*, 18(3):427–442, 2022. doi:10.1007/S11334-022-00444-8.
- 21 E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.
- 22 Antonio Ielo, Mark Law, Valeria Fionda, Francesco Ricca, Giuseppe De Giacomo, and Alessandra Russo. Towards ilp-based ltlf passive learning. In *Inductive Logic Programming: 32nd International Conference, ILP 2023, Bari, Italy, November 13–15, 2023, Proceedings*, pages 30–45, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-49299-0_3.
- 23 Neil Immerman. Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.*, 22(3):384–406, 1981. doi:10.1016/0022-0000(81)90039-8.
- 24 Jean Christoph Jung, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. Extremal separation problems for temporal instance queries. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 3448–3456. ijcai.org, 2024. URL: <https://www.ijcai.org/proceedings/2024/382>.
- 25 Xiao Li, Cristian Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 3834–3839. IEEE, 2017. doi:10.1109/IROS.2017.8206234.
- 26 Weilin Luo, Pingjia Liang, Jianfeng Du, Hai Wan, Bo Peng, and Delong Zhang. Bridging ltlf inference to GNN inference for learning ltlf formulae. In *AAAI*, pages 9849–9857. AAAI Press, 2022. doi:10.1609/AAAI.V36I9.21221.
- 27 Corto Mascle, Nathanaël Fijalkow, and Guillaume Lagarde. Learning temporal formulas from examples is hard. *CoRR*, abs/2312.16336, 2023. doi:10.48550/arXiv.2312.16336.
- 28 Sara Mohammadinejad, Jyotirmoy V. Deshmukh, Aniruddh Gopinath Puranic, Marcell Vazquez-Chanlatte, and Alexandre Donzé. Interpretable classification of time-series data using efficient enumerative techniques. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 9:1–9:10. ACM, 2020. doi:10.1145/3365365.3382218.
- 29 Daniel Neider and Ivan Gavran. Learning linear temporal properties. In Nikolaj S. Bjørner and Arie Gurfinkel, editors, *2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 - November 2, 2018*, pages 1–10. IEEE, 2018. doi:10.23919/FMCAD.2018.8603016.
- 30 Daniel Neider and Rajarshi Roy. *What Is Formal Verification Without Specifications? A Survey on Mining LTL Specifications*, pages 109–125. Springer Nature Switzerland, Cham, 2025. doi:10.1007/978-3-031-75778-5_6.
- 31 C.H. Papadimitriou. *Computational Complexity*. Theoretical computer science. Addison-Wesley, 1994. URL: <https://books.google.de/books?id=JogZAQAIAAJ>.
- 32 Nir Piterman, Amir Pnueli, and Yaniv Sa’ar. Synthesis of reactive(1) designs. In E. Allen Emerson and Kedar S. Namjoshi, editors, *Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006, Proceedings*, volume 3855 of *Lecture Notes in Computer Science*, pages 364–380. Springer, 2006. doi:10.1007/11609773_24.
- 33 Amir Pnueli. The temporal logic of programs. In *Proc. 18th Annu. Symp. Found. Computer Sci.*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.
- 34 Adrien Pommellet, Daniel Stan, and Simon Scatton. Sat-based learning of computation tree logic. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part I*, volume 14739 of *Lecture Notes in Computer Science*, pages 366–385. Springer, 2024. doi:10.1007/978-3-031-63498-7_22.

- 35 Ritam Raha, Rajarshi Roy, Nathanaël Fijalkow, and Daniel Neider. Scalable anytime algorithms for learning fragments of linear temporal logic. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 263–280, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-99524-9_14.
- 36 Ritam Raha, Rajarshi Roy, Nathanaël Fijalkow, Daniel Neider, and Guillermo A. Pérez. Synthesizing efficiently monitorable formulas in metric temporal logic. In *VMCAI (2)*, volume 14500 of *Lecture Notes in Computer Science*, pages 264–288. Springer, 2024. doi:10.1007/978-3-031-50521-8_13.
- 37 Heinz Riener. Exact synthesis of LTL properties from traces. In *FDL*, pages 1–6. IEEE, 2019. doi:10.1109/FDL.2019.8876900.
- 38 Rajarshi Roy, Dana Fisman, and Daniel Neider. Learning interpretable models in the property specification language. In *IJCAI*, pages 2213–2219. ijcai.org, 2020. doi:10.24963/IJCAI.2020/306.
- 39 Kristin Yvonne Rozier. Specification: The biggest bottleneck in formal methods and autonomy. In *VSTTE*, volume 9971 of *Lecture Notes in Computer Science*, pages 8–26, 2016. doi:10.1007/978-3-319-48869-1_2.
- 40 Dorsa Sadigh, Eric S. Kim, Samuel Coogan, S. Shankar Sastry, and Sanjit A. Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, pages 1091–1096. IEEE, 2014. doi:10.1109/CDC.2014.7039527.
- 41 Mojtaba Valizadeh, Nathanaël Fijalkow, and Martin Berger. LTL learning on gpus. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part III*, volume 14683 of *Lecture Notes in Computer Science*, pages 209–231. Springer, 2024. doi:10.1007/978-3-031-65633-0_10.
- 42 Hai Wan, Pingjia Liang, Jianfeng Du, Weilin Luo, Rongzhen Ye, and Bo Peng. End-to-end learning of ltlf formulae by faithful ltlf encoding. In *AAAI*, pages 9071–9079. AAAI Press, 2024. doi:10.1609/AAAI.V38I8.28757.
- 43 Andrzej Wasylkowski and Andreas Zeller. Mining temporal specifications from object usage. *Autom. Softw. Eng.*, 18(3-4):263–292, 2011. doi:10.1007/S10515-011-0084-1.

On Cascades of Reset Automata

Roberto Borelli  

University of Udine, Italy

Luca Geatti   

University of Udine, Italy

Marco Montali   

Free University of Bozen-Bolzano, Italy

Angelo Montanari   

University of Udine, Italy

Abstract

The Krohn-Rhodes decomposition theorem is a pivotal result in automata theory. It introduces the concept of *cascade product*, where two semiautomata, that is, automata devoid of initial and final states, are combined in a feed-forward fashion. The theorem states that any semiautomaton can be decomposed into a sequence of permutation-reset semiautomata. For the counter-free case, this decomposition consists entirely of reset components with two states each. This decomposition has significantly impacted recent research in various areas of computer science, including the identification of a class of transformer encoders equivalent to star-free languages and the conversion of Linear Temporal Logic formulas into past-only expressions (pastification).

The paper revisits the cascade product in the context of *reset automata*, thus considering each component of the cascade as a language acceptor. First, we give regular expression counterparts of cascades of reset automata. We then establish several expressiveness results, identifying hierarchies of languages based on the restriction of the height (number of components) of the cascade or of the number of states in each level. We also show that any cascade of reset automata can be transformed, with a quadratic increase in height, into a cascade that only includes two-state components. Finally, we show that some fundamental operations on cascades, like intersection, union, negation, and concatenation with a symbol to the left, can be directly and efficiently computed by adding a two-state component.

2012 ACM Subject Classification Theory of computation → Regular languages; Theory of computation → Automata extensions

Keywords and phrases Automata, Cascade products, Regular expressions, Krohn-Rhodes theory

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.20

Funding Luca Geatti and Angelo Montanari acknowledge the support from the 2024 Italian INdAM-GNCS project “Certificazione, monitoraggio, ed interpretabilità in sistemi di intelligenza artificiale”, ref. no. CUP E53C23001670001 and the support from the Interconnected Nord-Est Innovation Ecosystem (iNEST), which received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.5 – D.D. 1058 23/06/2022, ECS00000043). In addition, Marco Montali and Angelo Montanari acknowledges the support from the MUR PNRR project FAIR - Future AI Research (PE00000013) also funded by the European Union Next-GenerationEU.

Acknowledgements The authors would like to thank Alessio Mansutti for his valuable comments during the preparation of this paper.

1 Introduction

The Krohn-Rhodes decomposition theorem is a fundamental result both in automata theory and in semigroup algebra [12]. It relies on the concept of *cascade product* of two semiautomata, i.e., automata devoid of initial and final states, and thus, ultimately, edge-labeled graphs.



© Roberto Borelli, Luca Geatti, Marco Montali, and Angelo Montanari;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 20; pp. 20:1–20:22



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



In this setup, the first semiautomaton operates on an alphabet Σ , while the second one reads symbols belonging to the Cartesian product of Σ and the set of states of the first semiautomaton. The key feature of the cascade product, which extends the notion of direct product, is that the second semiautomaton transitions from state s to state s' by reading the pair (σ, q) if and only if the input symbol is σ and the first semiautomaton is in state q .

The Krohn-Rhodes theorem states that any semiautomaton can be decomposed into a *cascade* (i.e., a sequence of cascade products) of *permutation-reset semiautomata*.¹ In such semiautomata, each symbol of the alphabet induces a function on the set of states that is either a *permutation*, i.e., a bijective function, or a *reset*, that is, there is a specific state to which all other states are mapped into when reading that symbol. Crucially, if the semiautomaton is *counter-free*, that is, it does not contain non-trivial cycles [18], the Krohn-Rhodes theorem guarantees the existence of a decomposition that consists of *reset automata* only, i.e., automata where all symbols induce reset functions (as described above) or the identity function.

The Krohn-Rhodes theorem, in particular the decomposition of counter-free automata into reset automata, had a significant impact on some meaningful problems of current research in computer science. A notable example comes from Angluin et al. [1], who employ the Krohn-Rhodes decomposition theorem to prove that Linear Temporal Logic (LTL [19]) is equivalent to transformer encoders with hard attention and strict future masking (see also [13]). Specifically, they show how reset semiautomata can be encoded in **B-RASP**, a minimal programming language that compiles into transformers. Similarly, studies such as [21, 10, 11] utilized this theorem to analyze the sample complexity of cascades and the expressiveness of Recurrent Neural Networks without circular dependencies. Another example is provided by Maler [14, 15], who used the decomposition theorem to transform any formula of LTL, interpreted over finite words, into an equivalent formula using only past operators (see also [20]), a problem now known as *pastification* [2].

In this paper, we revisit the cascade product in the *reset automata* setting, i.e., language acceptors whose underlying semiautomaton is a reset. We address various expressiveness issues for cascade products by themselves and in relation to regular expressions. These results represent a necessary step towards a more efficient exploitation of Krohn-Rhodes decomposition in pastification, with the ultimate goal of lowering its current, triply exponential upper bound, which is far away from the known, singly exponential lower bound.

The paper consists of three main parts. In the first part, we address the question: given a cascade of reset automata, which is its corresponding regular expression? We begin by focusing on cascades of height 1, proving that the language corresponding to a reset automaton over the alphabet Σ is always of the form $J \cup (\Sigma^* \cdot R \cdot I^*)$, for some $I, R \subseteq \Sigma$, such that $I \cap R = \emptyset$ and either $J = I^*$ or $J = \emptyset$. Then, we extend the analysis to cascades of reset automata of arbitrary height. As a first step, we show that the last level can always be transformed into a two-state automaton, and then, by exploiting such a result, we derive the regular expression corresponding to a generic cascade of reset automata.

In the second part, we build on the previously obtained results and establish several expressiveness results about cascades of reset automata. We structure the analysis into three types of cascades:

- (i) short cascades (whose height is bounded by 2),
- (ii) narrow cascades (where each component has two states, but there is not a height limitation), and
- (iii) general cascades (with no limitations on the height or on the number of states per level).

¹ Formally, the decomposition is guaranteed to preserve a homomorphism from the cascade to the initial semiautomaton.

As for short cascades, we prove that any language \mathcal{L} over an alphabet Σ of cardinality k that is definable by a reset cascade of height 2 can also be defined by one where the first component has at most $k + 1$ states. Additionally, we show that increasing the number of states in the first component results in a strict increase in expressiveness: there exists a family of languages which are definable by a two-reset cascade whose first component has n states, but are not definable if the first component is restricted to $n - 1$ states. Similarly, for narrow cascades, we show that increasing the height results in an increase in expressiveness. These two results – the increase in the number of states in the first component for short cascades and the increase in height for narrow cascades – lead to two hierarchies (with infinitely many levels) of languages that are not definable at previous levels. Finally, we show that any general cascade can be transformed into one whose components have all 2 states, with an increase in height of at most a quadratic factor (relative to the height of the original cascade).

In the last part, we deal with closure properties of the languages recognized by reset cascades, and show that some fundamental operations can be computed in an efficient way. More precisely, we prove that the operations of intersection, union, negation, and concatenation with Σ to the left (“next operation”) all require the addition of one component with 2 states only.

The paper is structured as follows. Related work is discussed in Section 2. In Section 3, we provide some background knowledge. In Section 4, we introduce the cascades of reset automata, we state some basic results about them, and we provide a characterization of the languages that they recognize in terms of regular expressions. In Section 5, we present some expressiveness results for short, narrow, and general cascades of reset automata. Section 6 focus on closure properties and the efficient computation of some basic operations. Finally, Section 7 provides an assessment of the work done, and it outlines some directions for future research.

2 Related Work

The Krohn-Rhodes theorem and the cascade product turn out to be quite useful in understanding the structure and the expressiveness of finite-state systems, in particular in the context of automata and neural networks, and their connection to logic. Various recent contributions have leveraged this foundational theory to explore the expressiveness, modularity, and learning potential of automata in such a context.

A pivotal contribution in this area is the work by Maler on the cascade decomposition of semiautomata [14, 15]. It revisits the Eilenberg’s variant of the Krohn-Rhodes theorem [6] and offers a constructive proof that any semiautomaton can be decomposed into a cascade of elementary (permutation and reset) semiautomata. The paper introduces the *holonomy tree* as a data structure to represent cascade decompositions and an algorithm to build such a tree. Crucially, the algorithm carefully maps the permutations of the obtained cascade product to non-trivial cycles of the starting semiautomaton: this guarantees that, whenever the starting semiautomaton is counter-free, that is, devoid of non-trivial cycles, the generated cascade decomposition only consists of reset components. An exponential bound on the size of the cascade decomposition in terms of the size of the starting semiautomaton is given. This algorithm can be used to actually translate counter-free automata to temporal logic. More precisely, Maler shows how to translate any cascade product of reset semiautomata into a pure past LTL formula, that is, a formula featuring only past temporal modalities.

Together with the transformation of the future fragment of LTL, interpreted over finite words, into counter-free automata, this leads to a triply exponential upper bound to the problem of transforming pure-future LTL over finite words into pure-past LTL (pastification problem). Equivalently, in the case of LTL interpreted over infinite words, Maler shows how to use the proposed algorithm to normalize every LTL formula by mapping it into one belonging to the *Reactivity* class [16], at a cost of a triply exponential blowup. For both problems, that is, pastification and normalization, the best known lower bounds are singly exponential [3, 17].

The Krohn-Rhodes theorem has also been applied to analyze the complexity of semigroups, as shown in [9]. This study examines semigroups of upper triangular matrices over finite fields and establishes that the Krohn-Rhodes complexity of these semigroups corresponds to $n - 1$, where n is the matrix dimension. These results underline the deep connection between the algebraic structure of semigroups and their matrix representations, providing a measure of how intricate the cascade product representation needs to be for such semigroups.

In [8], the Krohn-Rhodes theorem is used to characterize piecewise testable and commutative languages. The authors define biased reset semiautomata, where the current state changes at most once, and characterize cascades $\mathcal{A} \circ \mathcal{B}$, where \mathcal{B} is a biased reset semiautomaton. Theorem 4.12 in Section 4 can be seen as a simplification and a generalization (to cascades of unbounded height) of such a characterization. Finally, the authors propose the notion of scope of a cascade, which is used to analyze the dot-depth of star-free languages.

In [21], Ronca builds on the Krohn-Rhodes theorem, proposing automata cascades as a structured and modular framework to describe complex systems. The resulting framework allows automata to be decomposed into components with specific functionalities, enabling fine-grained control of their expressiveness. By focusing on component-based decomposition, the study demonstrates that the sample complexity of learning automata cascades is linear in the number of components and their individual complexities, up to logarithmic factors. This contrasts with traditional state-centric perspectives, where sample complexity scales with the number of states, often limiting the feasibility of learning large systems. The relationships between the cascade product and neural networks are investigated in [10]. Recurrent Neural Cascades (RNCs) are a class of networks with acyclic connections, which naturally align with the cascade product of automata. By exploiting the Krohn-Rhodes theorem, the authors prove that RNCs capture star-free regular languages.

The Krohn-Rhodes theorem also underpins the exploration of transformer models in [13]. While transformers lack recurrence, the paper demonstrates that their layered architecture can simulate the cascade decomposition of finite automata. Leveraging Krohn-Rhodes theory, the authors show that shallow transformers can hierarchically approximate automata computations, enabling polynomial-sized and constant-depth shortcuts for specific automata.

In [1], Angluin et al. draws direct parallels between the expressive power of masked hard-attention transformers and star-free regular languages. These models, constrained by strict future masking, are shown to be equivalent to LTL and counter-free automata – both closely tied to the Krohn-Rhodes cascade framework. The study underscores how the structured limitations of these transformers, akin to a cascade decomposition, yield expressive yet computationally efficient models.

Together, these contributions extend the applicability of the Krohn-Rhodes theory to neural networks, transformers, and beyond, demonstrating the versatility of the cascade framework as a powerful principle in computation.

3 Background

A *semiautomaton* \mathcal{A} is a tuple (Σ, Q, δ) such that:

- (i) Σ is a (finite) alphabet;
- (ii) Q is a set of states;
- (iii) $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.

An *automaton* $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ is a semiautomaton extended with an initial state $q_0 \in Q$ and a set $F \subseteq Q$ of final states. With δ^* we denote the Kleene's closure of δ . We say that \mathcal{A} is *two-state* iff $|Q| = 2$.

Given an automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ and a (finite) word $\sigma := \langle \sigma_0, \dots, \sigma_n \rangle \in \Sigma^*$, the *run* $\tau \in Q^+$ induced by σ is a sequence $\langle q_0, q_1, \dots, q_{n+1} \rangle$ such that $\delta(q_i, \sigma_i) = q_{i+1}$, for all $0 \leq i \leq n$. We say that τ is *accepting* iff $q_{n+1} \in F$. A word $\sigma \in \Sigma^*$ is *accepted* by \mathcal{A} iff the run induced by σ is accepting. We define the *language of* \mathcal{A} , denoted by $\mathcal{L}(\mathcal{A})$, as the set of accepted words. Given a state $q \in Q$, let $\mathcal{L}_q(\mathcal{A})$ be the set of words inducing a run $\tau := \langle q_0, \dots, q_m \rangle$ with $q_m = q$. The classic *direct product* of automata is defined as follows.

► **Definition 3.1** (Direct product of automata). *Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{A}' = (\Sigma, Q', \delta', q'_0, F')$ be two automata. The direct product of \mathcal{A} and \mathcal{A}' , denoted by $\mathcal{A} \times \mathcal{A}'$, is the automaton $(\Sigma, Q \times Q', \delta'', (q_0, q'_0), F \times F')$ such that, for all $(q, q') \in Q \times Q'$ and for all $a \in \Sigma$, it holds that $\delta''((q, q'), a) = (\delta(q, a), \delta'(q', a))$.*

The cascade product of semiautomata is defined as follows.

► **Definition 3.2** (Cascade Product of semiautomata [15, 22]). *Let Σ be a finite alphabet and let $\mathcal{A} = (\Sigma, Q, \delta)$ and $\mathcal{A}' = (\Sigma \times Q, Q', \delta')$ be two semiautomata over the alphabets Σ and $\Sigma \times Q$, respectively. We define the cascade product between \mathcal{A} and \mathcal{A}' , denoted with $\mathcal{A} \circ \mathcal{A}'$, as the semiautomaton $(\Sigma, Q \times Q', \delta'')$ such that, for all $(q, q') \in Q \times Q'$ and for all $a \in \Sigma$:*

$$\delta''((q, q'), a) = (\delta(q, a), \delta'(q', (a, q)))$$

We will often simply use “*cascade*” for “*cascade product*”.

It is worth noticing that the cascade product of semiautomata is a generalization of the classic direct product: the latter can be recovered by imposing the alphabet of the second semiautomaton to be Σ (i.e., the alphabet of the first one) instead of $\Sigma \times Q$.

The cascade product is an *associative* operation, meaning that $(\mathcal{A} \circ \mathcal{A}') \circ \mathcal{A}''$ is the same semiautomaton as $\mathcal{A} \circ (\mathcal{A}' \circ \mathcal{A}'')$. We define the *height* of the product $\mathcal{A}_1 \circ \dots \circ \mathcal{A}_n$ as n .

We now introduce two classes of semiautomata, *reset* and *permutation*, depending on the form of their transitions. We first define the notion of *function induced by a symbol*.

► **Definition 3.3** (Function induced by a symbol). *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be a semiautomaton. For each symbol $a \in \Sigma$, we define the function induced by a in \mathcal{A} , denoted by $\tau_a^{\mathcal{A}}$ (or simply τ_a when \mathcal{A} is clear from the context), as the transformation $\tau_a : Q \rightarrow Q$ such that, for all $q \in Q$, it holds $\tau_a(q) = q'$ iff $\delta(q, a) = q'$.*

Reset and permutation functions are defined as follows.

► **Definition 3.4** (Reset and permutation functions). *Let $\tau : Q \rightarrow Q$. We say that τ is a reset function iff there exists $q' \in Q$ such that $\tau(q) = q'$, for all $q \in Q$. In this case, we say that τ is a reset on q' . If $\tau : Q \rightarrow Q$ is a bijection, then it is called a permutation.*

On the basis of the functions induced by the symbols of their alphabet, we define the following classes of semiautomata.

► **Definition 3.5** (Classes of semiautomata). Let $\mathcal{A} = (\Sigma, Q, \delta)$ be a semiautomaton. We say that \mathcal{A} is:

- a permutation-reset semiautomaton iff, for each $a \in \Sigma$, τ_a is either a permutation or a reset.
- a permutation semiautomaton iff, for each $a \in \Sigma$, τ_a is a permutation;
- a reset semiautomaton iff, for each $a \in \Sigma$, τ_a is either the identity function or a reset function;
- a pure-reset semiautomaton iff, for each $a \in \Sigma$, τ_a is a reset function.

We now introduce counter-free semiautomata [18]. Let $\sigma \in \Sigma^*$. From now on, we denote by $(\sigma)^i$ the word generated by concatenating i times the word σ to itself. A word $\sigma \in \Sigma^*$, with $\sigma \neq \varepsilon$, defines a *nontrivial cycle* in a semiautomaton $\mathcal{A} = (\Sigma, Q, \delta)$ if there exists a state $q \in Q$ such that:

- (i) $\delta^*(q, \sigma) \neq q$
- (ii) $\delta^*(q, (\sigma)^i) = q$, for some $i > 1$.

We say that a semiautomaton \mathcal{A} is *counter-free* if there are no words that define a nontrivial cycle. Counter-free automata recognize exactly the set of languages definable by *star-free* regular expressions, i.e., expressions devoid of Kleene's star. We denote this set by \mathcal{SF} .

A fundamental result in the field is the Krohn-Rhodes Cascade Decomposition Theorem. The theorem's initial formulation was expressed in the context of semigroups [12], and its automata-theoretic counterpart [14] can be articulated as follows.

► **Theorem 3.6** (The Krohn-Rhodes Cascade Decomposition Theorem [12, 14]). For each semiautomaton $\mathcal{A} = (\Sigma, Q, \delta)$, there exists a cascade product of semiautomata $\mathcal{C} := \mathcal{A}_1 \circ \mathcal{A}_2 \circ \dots \circ \mathcal{A}_n$ such that:

- (i) \mathcal{A}_i is a permutation-reset semiautomaton, for each $1 \leq i \leq n$;
- (ii) there is an homomorphism² from \mathcal{C} to \mathcal{A} ;
- (iii) if \mathcal{A} is counter-free, then \mathcal{A}_i is a two-state reset semiautomaton, for each $1 \leq i \leq n$.

4 Cascades of automata

In this section, we begin our study of the languages recognized by cascades of automata. We start by formally defining them and stating some basic properties. Then, we focus on cascades of reset automata, and provide a characterization of the languages they recognize in terms of regular expressions.

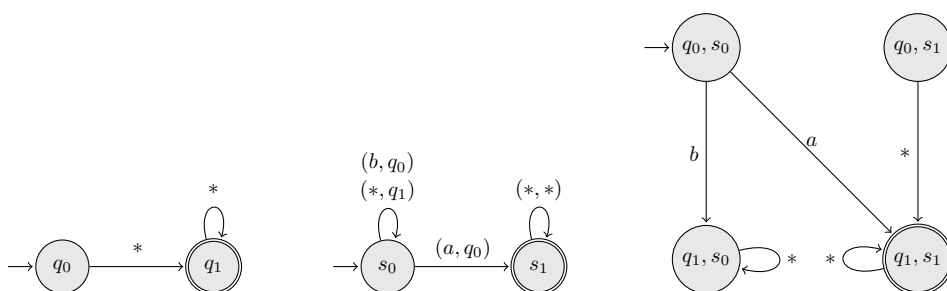
4.1 Definitions and basic properties

To begin with, we generalize the notion of cascade product of semiautomata (Definition 3.2) to automata.

► **Definition 4.1** (Cascade product of automata). Let Σ be a finite alphabet and let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{A}' = (\Sigma \times Q, Q', \delta', q'_0, F')$ be two automata over the alphabets Σ and $\Sigma \times Q$, respectively. We define the cascade product of \mathcal{A} and \mathcal{A}' , denoted by $\mathcal{A} \circ \mathcal{A}'$, as the automaton $(\Sigma, Q \times Q', \delta'', (q_0, q'_0), F \times F')$ where δ'' is defined as in Definition 3.2.

We say that a language \mathcal{L} is *definable* by a cascade \mathcal{C} iff $\mathcal{L} = \mathcal{L}(\mathcal{C})$. Figure 1 shows the cascade product of two reset automata defining the language $a \cdot \Sigma^*$.

² We refer to [12, 14] for a formal definition of homomorphism between semiautomata.



■ **Figure 1** The reset automaton \mathcal{A}_1 with set of states $Q = \{q_0, q_1\}$ over the alphabet $\Sigma = \{a, b\}$ (left). The reset automaton \mathcal{A}_2 over the alphabet $\Sigma \times Q$ (middle). The cascade product $\mathcal{A}_1 \circ \mathcal{A}_2$ over the alphabet Σ that recognizes the languages $a \cdot \Sigma^*$ (right).

In the following, we will use the term cascade to refer both to the component automata and to the resulting automaton.

We now show how to compute the language recognized by a cascade of automata on the basis of the languages recognized by its components. Let Σ_1 and Σ_2 be two alphabets. Let $\sigma^1 = \sigma_1^1 \dots \sigma_n^1 \in (\Sigma_1)^n$ and $\sigma^2 = \sigma_1^2 \dots \sigma_n^2 \in (\Sigma_2)^n$ be two words of length n . We define $\text{aug}(\sigma^1, \sigma^2) \in (\Sigma_1 \times \Sigma_2)^n$ as the word $(\sigma_1^1, \sigma_1^2) \dots (\sigma_n^1, \sigma_n^2)$.

▶ **Definition 4.2** (Language of \mathcal{B} at a state s over \mathcal{A}). Let $\mathcal{A} = \langle \Sigma, Q, \delta_A, q_0, F_A \rangle$ and $\mathcal{B} = \langle \Sigma \times Q, S, \delta_B, s_0, F_B \rangle$ be two automata. The language of \mathcal{B} at state $s \in S$ over \mathcal{A} , denoted by $\mathcal{L}_s(\mathcal{B})[\mathcal{A}]$, is defined as follows: the empty word only belongs to $\mathcal{L}_{s_0}(\mathcal{B})[\mathcal{A}]$; a word $\sigma = \sigma_1 \dots \sigma_k$, with $k \geq 1$, belongs to $\mathcal{L}_s(\mathcal{B})[\mathcal{A}]$ if

- (i) $\sigma_1 \dots \sigma_{k-1}$ induces a run $\tau = \langle q_0, q_1, \dots, q_{k-1} \rangle$ on \mathcal{A} ; and
- (ii) $\text{aug}(\sigma, \tau) \in \mathcal{L}_s(\mathcal{B})$.

The language of a cascade can be computed from those of its components as follows. Let $\mathcal{C} = \mathcal{A} \circ \mathcal{B}$ be a cascade. The words forcing \mathcal{C} to reach a state (q, s) are exactly those words such that:

- (i) they force \mathcal{A} to reach state q ; and
- (ii) they force \mathcal{B} to reach state s , when augmented with the run of \mathcal{A} .

▶ **Proposition 4.3** (Language of a cascade in terms of its components). Let $\mathcal{A} = \langle \Sigma, Q, \delta_A, q_0, F_A \rangle$ and $\mathcal{B} = \langle \Sigma \times Q, S, \delta_B, s_0, F_B \rangle$ be two automata. It holds that:

1. $\mathcal{L}_{(q,s)}(\mathcal{A} \circ \mathcal{B}) = \mathcal{L}_q(\mathcal{A}) \cap \mathcal{L}_s(\mathcal{B})[\mathcal{A}]$, for all states $q \in Q$ and $s \in S$;
2. $\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = \bigcup_{(q,s) \in F} \mathcal{L}_{(q,s)}(\mathcal{A} \circ \mathcal{B})$.

We now show that, as it happens with semiautomata, the direct product of automata is just a special case of the cascade product. To this end, we first define the notion of augmentation of an automaton.

▶ **Definition 4.4** (Augmentation). Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ and $\mathcal{A}' = \langle \Sigma', Q', \delta', q'_0, F' \rangle$ be two automata such that either $\Sigma' = \Sigma$ or $\Sigma' = \Sigma \times S$, for an arbitrary finite set S . We define the augmentation of \mathcal{A}' relative to \mathcal{A} , denoted by $\text{aug}(\mathcal{A}, \mathcal{A}')$, as the automaton $(\Sigma'', Q', \delta'', q'_0, F')$ such that:

- if $\Sigma' = \Sigma$, then $\Sigma'' := \Sigma \times Q$ and for all $q \in Q'$ and all $a \in \Sigma$, $\delta''(q, (a, *)) = \delta'(q, a)$;
- if $\Sigma' = \Sigma \times S$, then $\Sigma'' := \Sigma \times Q \times S$ and, for all $q \in Q'$ and for all $(a, s) \in \Sigma \times S$, it holds that $\delta''(q, (a, *, s)) = \delta'(q, (a, s))$.

Given a cascade $\mathcal{C} = \mathcal{A}'_1 \circ \dots \circ \mathcal{A}'_n$ over Σ , we define the augmentation of \mathcal{C} relative to \mathcal{A} , denoted by $\text{aug}(\mathcal{A}, \mathcal{C})$, as the cascade $\text{aug}(\mathcal{A}, \mathcal{A}'_1) \circ \dots \circ \text{aug}(\mathcal{A}, \mathcal{A}'_n)$.

The notion of augmentation can be generalized to a pair of cascades \mathcal{C} and \mathcal{C}' by treating \mathcal{C} as a single automaton: from now on, when we will refer to the cascade product of \mathcal{C} and \mathcal{C}' , we will interpret it as the cascade product of the automaton \mathcal{A} generated by \mathcal{C} and \mathcal{C}' .

The next proposition shows that direct product can be simulated by means of augmentation and cascade product.

► **Proposition 4.5** (Direct product by means of cascade product). *Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ be an automaton and let \mathcal{C} be a cascade over Σ . It holds that $\mathcal{A} \circ \text{aug}(\mathcal{A}, \mathcal{C}) = \mathcal{A} \times \mathcal{C}$.*

Furthermore, augmenting an automaton does not affect its property of being *reset* (or *permutation*), as stated by the following Proposition 4.6.

► **Proposition 4.6.** *Let Σ be a finite alphabet and let \mathcal{A} be an automaton over Σ or $\Sigma \times S$, for an arbitrary finite set S . If \mathcal{A} is a reset (resp., permutation) automaton, then, for any automaton \mathcal{A}' over Σ , $\text{aug}(\mathcal{A}', \mathcal{A})$ is a reset (resp., permutation) automaton.*

It follows that, in particular, if \mathcal{C} is a cascade of reset (resp., permutation) automata, then $\text{aug}(\mathcal{A}, \mathcal{C})$ is a cascade of reset (resp., permutation) automata. From Propositions 4.5 and 4.6, it follows directly that, given two cascades \mathcal{C} and \mathcal{C}' of height m and n of reset (resp., permutation) automata, there exists a cascade of height $m + n$ of reset (resp., permutation) automata for $\mathcal{L}(\mathcal{C}) \cap \mathcal{L}(\mathcal{C}')$. In Section 6, we will show how to compute other basic operations on cascades of resets.

4.2 Languages of cascades of resets

In this part, we characterize the language recognized by a cascade of reset automata in terms of regular expressions. We begin with the case of cascades of height 1 and then we move to cascades of unbounded height.

4.2.1 Cascades of height 1

The study of which regular expressions characterize height-1 cascades of resets coincides with the study of the languages recognized by *reset automata*. The following theorem gives a characterization of reset automata in terms of regular expressions.

► **Theorem 4.7** (The languages of reset automata). *Let Σ be a finite alphabet. A language $\mathcal{L} \subseteq \Sigma^*$ is recognized by a reset automaton if and only if $\mathcal{L} = J \cup (\Sigma^* \cdot R \cdot I^*)$ for some $I, R \subseteq \Sigma$ such that $I \cap R = \emptyset$ and either $J = I^*$ or $J = \emptyset$.*

Intuitively, an automaton reading a symbol that induces a reset function on a final state is forced to end up in that state, regardless of which state it was in before. Furthermore, it remains in that state if all subsequent symbols induce identity functions. In the case of words containing multiple resets on a final state, only the last of these symbols matters, resulting in words of the form $\Sigma^* \cdot R \cdot I^*$. The case of $J = I^*$ arises when the initial state is also final. In this scenario, to accept a word, the automaton does not need to read a symbol that induces a reset on a final state (since it is already there), but only needs to stay in the initial state.

A by-product of Theorem 4.7 is that any reset automaton is equivalent to one with two states, only one of which is final. The rationale is as follows:

- (i) the symbols in R induce a reset on the single final state;
- (ii) the symbols in I act as identities; and
- (iii) the symbols neither in R nor in I induce resets on the single non-final state.



■ **Figure 2** The reset automaton corresponding to a language of the form $J \cup (\Sigma^* \cdot R \cdot I^*)$ in the case $J = \emptyset$ (on the left) and in the case $J = I^*$ (on the right).

Moreover, the initial state is also the final state if and only if $J = I^*$. A graphical account is given in Figure 2.

► **Proposition 4.8.** *For every reset automaton, there exists an equivalent one with two states, exactly one of which is final.*

Theorem 4.7 allows us to establish a first connection between *Linear Temporal Logic on finite traces* (LTLf [5]) formulas and equivalent reset cascades. As highlighted in the introduction, the languages expressible in LTLf are exactly the star-free languages, that is, those languages that can be represented by regular expressions that do not use the Kleene star, or equivalently, by languages whose minimal automaton is counter-free [18]. By Krohn-Rhodes' theorem (Theorem 3.6), it follows that the languages definable in LTLf are precisely those expressible through cascades of resets. Given the relevance of reset cascade decomposition in problems such as *pastification* [2, 4] and *normalization* [7] of temporal logic formulas, it is crucial to understand which LTLf formulas can be expressed with cascades of a specific height. The following result shows that even simple formulas like p (the proposition letter “p” holds at the initial time point) or $p \cup q$ (there is a future point where “q” holds, and until then, “p” remains true) cannot be expressed with cascades of resets of height 1. In fact, the languages they recognize³ are respectively $\vec{p} \cdot \Sigma^*$ and $(\vec{p})^* \cdot \vec{q} \cdot \Sigma^*$, which are not of the form $J \cup (\Sigma^* \cdot R \cdot I^*)$, for any choice of R , I , and J . However, the formula $\text{F}p$ (there exists a point in the future where “p” holds) can be expressed with reset cascades of height 1, as its language is of the form $J \cup (\Sigma^* \cdot R \cdot I^*)$, choosing $R := \vec{p}$, $I := \Sigma \setminus \vec{p}$, and $J = \emptyset$.

► **Corollary 4.9.** *The languages defined by the LTLf formulas p and $p \cup q$ are not definable with height-1 cascades of resets.*

4.2.2 Cascades of unbounded-height

In this section we derive regular expressions for cascades of arbitrary height. As a first step, we show that a cascade of height h of reset automata, say $\mathcal{A}_1 \circ \dots \circ \mathcal{A}_h$, can be transformed into an equivalent cascade of the same height, still consisting of reset automata, where the last automaton (\mathcal{A}_h) has exactly two states, one of which is the only accepting state. This result forms the basis for Theorem 4.12, which provides the characterization of cascades of arbitrary height.

► **Lemma 4.10.** *Let \mathcal{A} be an automaton with set of states Q over the alphabet Σ , and let \mathcal{B} be a reset automaton over the alphabet $\Sigma \times Q$. There exists a 2-states reset automaton \mathcal{B}' , with exactly one final state, such that $\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = \mathcal{L}(\mathcal{A} \circ \mathcal{B}')$.*

³ Here, assuming a set of atomic propositions $\mathcal{AP} := \{p, q, r, \dots\}$, the languages of formulas over \mathcal{AP} are defined over the alphabet $\Sigma := 2^{\mathcal{AP}}$. Moreover, given any $p \in \mathcal{AP}$, we indicate with \vec{p} all the letters $a \in \Sigma$ such that $p \in a$.

20:10 On Cascades of Reset Automata

The proof of Lemma 4.10 heavily relies on the characterization of cascades of height 1. More precisely, since \mathcal{B} is a reset automaton over the alphabet $\Sigma \times Q$, by Proposition 4.8, there exists an equivalent reset automaton with two states (one of which is the only accepting state) over the same alphabet. Since \mathcal{B} is at the bottom of the cascade, the language of the cascade $\mathcal{A} \circ \mathcal{B}'$ is the same as the language of $\mathcal{A} \circ \mathcal{B}$. This is because there are no other automata below \mathcal{B} in the cascade that can exploit information about \mathcal{B} 's current state. As a matter of fact, in Section 5, we will prove that this is no longer true when applying the same procedure to \mathcal{A} : there exist languages definable by a cascade $\mathcal{A} \circ \mathcal{B}$, where \mathcal{A} has 3 states and \mathcal{B} has 2 states, that cannot be expressed if the number of states of \mathcal{A} is limited to 2.

Let us now introduce the notion of *filtered automaton*, which is obtained from a given automaton by removing (filtering) certain outgoing transitions and possibly changing its initial state.

► **Definition 4.11** (Filtered Automaton). *Let $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F \rangle$ be an automaton. A filter is pair (q, H) , where $q \in Q$ and $H \subseteq \Sigma \times Q$. The partial automaton \mathcal{A} , filtered by (q, H) , denoted as $\mathcal{A} \downarrow_H^q$, is the automaton $(Q, \Sigma, q, \delta', F)$ where $\delta'(q', \sigma) := \delta(q', \sigma)$ if $(\sigma, q') \in H$, or is undefined otherwise.*

Before formally stating Theorem 4.12, that characterizes the languages of cascades of unbounded-height, we give an intuitive account of it. Let $\mathcal{A} \circ \mathcal{B}$ be a cascade, with \mathcal{A} an automaton and \mathcal{B} a reset automaton, where, w.l.o.g. (Lemma 4.10), \mathcal{B} has only two states and exactly one final state. Any word accepted by $\mathcal{A} \circ \mathcal{B}$ must drive both \mathcal{A} and \mathcal{B} to an accepting state. Its language can be captured by analyzing the symbols inducing a reset function that leads to a final state of \mathcal{B} , and the symbols inducing identities in \mathcal{B} . The words in the language of $\mathcal{A} \circ \mathcal{B}$ are precisely those consisting of

- (i) a *prefix* that, for any symbol (σ, q) inducing a reset on a final state of \mathcal{B} , drives automaton \mathcal{A} to state q ;
- (ii) followed by the symbol $\sigma \in \Sigma$ (let q_σ be the state reached by \mathcal{A} after reading it);
- (iii) a *suffix* that forces \mathcal{B} to remain in its accepting state through its identity functions I_B , and forces \mathcal{A} to reach a final state starting from q_σ .

In addition, \mathcal{A} cannot transition from state q' when reading a symbol σ' if the pair (σ', q') does not belong to \mathcal{B} 's identity functions, as this would cause \mathcal{B} to leave its accepting state. Therefore, the suffix corresponds to the language of automaton \mathcal{A} , filtered by $(\delta_A(q, \sigma), I_B)$. This is formally expressed by the following theorem.

► **Theorem 4.12** (Languages of cascades of unbounded-height). *Let $\mathcal{A} = \langle \Sigma, Q = \{q_0, \dots, q_n\}, \delta_A, q_0, F_A \rangle$ be an automaton and let $\mathcal{B} = \langle \Sigma \times Q, \{s_0, s_1\}, \delta, s_0, \{s_f\} \rangle$, with $s_f \in \{s_0, s_1\}$, be a two-state reset automaton with one final state. It holds that:*

$$\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = M \cup \bigcup_{(\sigma, q) \in R_{s_f}} \mathcal{L}_q(\mathcal{A}) \cdot \sigma \cdot \mathcal{L}(\mathcal{A} \downarrow_{I_B}^{\delta_A(q, \sigma)})$$

where R_{s_f} is the set of symbols in $\Sigma \times Q$ that induce a reset function on state s_f , and $M := \mathcal{L}(\mathcal{A} \downarrow_{I_B}^{q_0})$ if $s_0 = s_f$ or $M := \emptyset$ otherwise.

Figure 3 gives an example of application of Theorem 4.12 to a cascade over the alphabet $\Sigma := \{a, b\}$ of two reset automata, \mathcal{A} (on the left) and \mathcal{B} (on the center), with two states each, recognizing the language $b^* \cdot a^+$. Using Theorem 4.12, we have that $\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = \mathcal{L}_{q_0}(\mathcal{A}) \cdot a \cdot \mathcal{L}(\mathcal{A} \downarrow_{I_B}^{q_1})$, where $\mathcal{A} \downarrow_{I_B}^{q_1}$ is the automaton obtained from \mathcal{A} filtered by the identities $I_B = \{(a, q_1), (b, q_0)\}$ of automaton \mathcal{B} . Since $\mathcal{L}_{q_0}(\mathcal{A}) = b^*$ and $\mathcal{L}(\mathcal{A} \downarrow_{I_B}^{q_1}) = a^*$, we obtain $\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = b^* \cdot a^+$. The following is a corollary of Theorem 4.12 in the case in which \mathcal{B} is a pure-reset automaton.

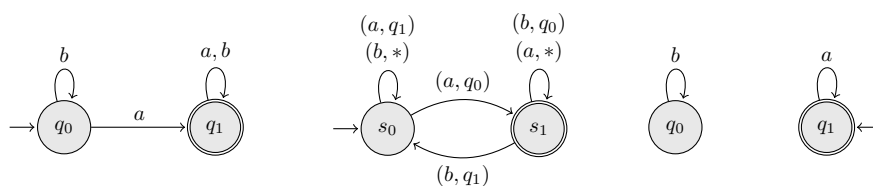


Figure 3 On the left and on the center, the reset automata \mathcal{A} and \mathcal{B} , respectively, for the cascade $\mathcal{A} \circ \mathcal{B}$ recognizing the language $b^* \cdot a^+$. On the right, the automaton $\mathcal{A} \downarrow_{I_B}^{s_1}$, where $I_B = \{(a, q_1), (b, q_0)\}$ are the identities of automaton \mathcal{B} .

► **Corollary 4.13.** *Let $\mathcal{A} = \langle \Sigma, \{q_0, \dots, q_n\}, \delta_A, q_0, F_A \rangle$ be an automaton and let $\mathcal{B} = \langle \Sigma \times Q, \{s_0, s_1\}, \delta, s_0, \{s_f\} \rangle$ be a two-state pure-reset automaton with one final state. It holds that*

$$\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = M \cup \bigcup_{\substack{(\sigma, q) \in R_{s_f} \\ \delta_A(q, \sigma) \in F_A}} \mathcal{L}_q(\mathcal{A}) \cdot \sigma$$

where $M := \epsilon$ if $s_0 = s_f$ and $q_0 \in F_A$, or $M := \emptyset$ otherwise.

It is worth noticing that the regular expressions in Theorem 4.12 and Corollary 4.13 refer to states of the cascade under consideration. This is a major difference with the characterization of reset cascades of height 1 (Theorem 4.7). In order to prove some undefinability results in the next section, we provide a characterization of the languages recognized by cascades of two-states resets of height 2 where the second component is pure-reset, based on regular expressions that do not refer to states of the cascade.

► **Lemma 4.14.** *Let $\mathcal{L} \subseteq \Sigma^*$ be a language. \mathcal{L} is definable by a cascade of height 2 in which the second component is pure-reset if and only if $\mathcal{L} = M \cup \bigcup_{i=1}^n K_i \cdot \sigma_i$ for some M, n, σ_i and K_i such that:*

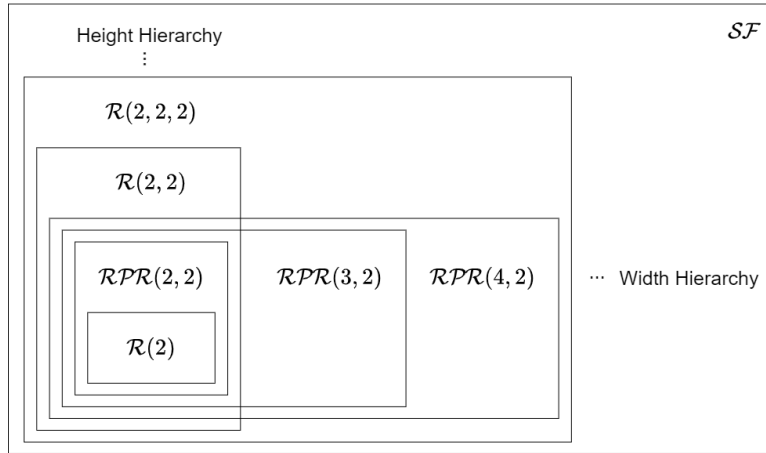
- (i) M is either ϕ or ϵ ;
- (ii) $0 \leq n \leq 2 \cdot |\Sigma|$;
- (iii) for all $i = 1 \dots n$ it holds $\sigma_i \in \Sigma$;
- (iv) there exists a language L recognizable by a two-state reset automaton such that for all $i = 1 \dots n$, either K_i is L or K_i is $\bar{L} = \Sigma^* \setminus L$.

Notice that the Lemma above can be easily extended to the case in which the first automaton in the cascade has k states, for some $k \geq 2$. This is done by relaxing (iv) and imposing that K_i can be chosen between k languages K_1, \dots, K_k such that $\{K_1, \dots, K_k\}$ is a partition of Σ^* and each K_i is definable by a cascade of resets of height 1. Constraint (ii) is also relaxed to $0 \leq n \leq k \cdot |\Sigma|$.

We will use Theorem 4.12, Corollary 4.13, and Lemma 4.14 in the next section to prove undefinability results of certain languages by cascades of a given height and with a specified number of states at each level.

5 Expressiveness results

In this section, we analyze the expressive power of various types of reset automaton cascades. We begin by defining several language classes, and subsequently structure our analysis into *short cascades* (where the height is constrained to at most two), *narrow cascades* (where the height is unbounded but each component contains two states), and *general cascades* (with no restrictions on either the height or the number of states).



■ **Figure 4** Summary of (some of) the results in Section 5.

► **Definition 5.1** (Classes \mathcal{R} and \mathcal{RPR}). Let $h \in \mathbb{N}^{>0}$ and let $k_1, \dots, k_h \in \mathbb{N}^{>1}$. We denote by $\mathcal{R}(k_1, \dots, k_h)$ the class of languages definable by a cascade $\mathcal{A}_1 \circ \dots \circ \mathcal{A}_h$ of reset automata such that \mathcal{A}_i has k_i states, for each $1 \leq i \leq h$. We denote by $\mathcal{RPR}(k_1, \dots, k_h)$ the subclass of $\mathcal{R}(k_1, \dots, k_h)$ where the last automaton (\mathcal{A}_h) is required to be pure-reset. For $h > 0$ and $k > 1$, we define $\mathcal{R}_k^h := \bigcup_{2 \leq k_1, \dots, k_h \leq k} \mathcal{R}(k_1, \dots, k_h)$ as the set of languages definable by a cascade of height h , where each component has at most k states. We define $\mathcal{R} := \bigcup_{h > 0, k > 1} \mathcal{R}_k^h$ as the set of languages definable by any cascade of reset automata. The classes \mathcal{RPR}_k^h and \mathcal{RPR} are defined analogously.

Figure 4 provides an overview of (some of) the results presented in this section. Specifically, it illustrates that increasing the cascade height and increasing the number of states at the first level lead to two distinct language hierarchies.

5.1 Short Cascades

We begin by considering *short* cascades, i.e. cascades of reset automata of height 2. As a first step, we start by comparing the classes $\mathcal{R}(2), \mathcal{RPR}(2, 2)$ and $\mathcal{R}(2, 2)$, and then we focus on $\mathcal{RPR}(k, 2)$ and $\mathcal{R}(k, 2)$ for every $k > 2$.

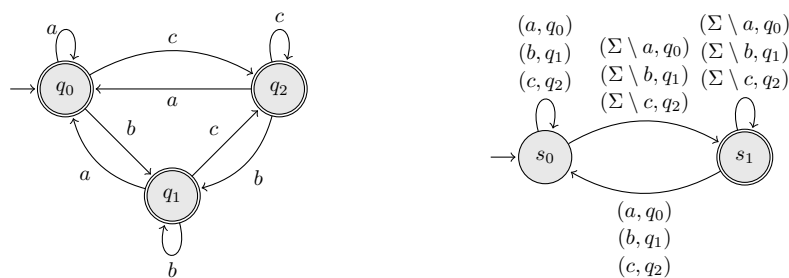
We already know that with a single pure-reset automaton we can recognize the set of all words ending with a certain symbol of the alphabet (this follows from Theorem 4.7 in the special case in which $I = \emptyset$). As an example, it holds that $\Sigma^*a \in \mathcal{RPR}(2)$. Now, if we introduce an additional pure-reset layer, we can effectively recognize the set of words ending with a *two-character suffix*. However, we also demonstrate that this is impossible using a single reset automaton.

► **Lemma 5.2.** Let $L = \Sigma^*aa$. It holds that:

- (i) $L \in \mathcal{RPR}(2, 2)$;
- (ii) $L \notin \mathcal{R}(2)$.

Lemma 5.2 shows that increasing the height of a cascade, even of height 1 and even with a pure-reset automaton, results into a gain of expressive power.

In the upcoming lemma, we demonstrate that, at the same height, prohibiting identities in the final layer results in a loss of expressive power. To illustrate this, let us consider the language $a \cdot \Sigma^*$. As shown in Figure 1, this language can be defined using a cascade of two reset



■ **Figure 5** On the left, the reset automaton \mathcal{A} and on the right the reset automaton \mathcal{B} such that, for $\Sigma = \{a, b, c\}$, the cascade $C := \mathcal{A} \circ \mathcal{B}$ accepts the language $\Sigma^* (\Sigma^2 \setminus \{aa, bb, cc\}) \cup \{b, c\}$, which precisely corresponds to the language L_3 described in Lemma 5.4. When viewed as a single automaton, C is also the minimal automaton for the language L_3 .

automata, with the last one specifically containing identities. Building upon Lemma 4.14, we further demonstrate that achieving the same language recognition is not possible when prohibiting identities in the final layer, no matter of the number of states of the first automaton.

- **Lemma 5.3.** *Let Σ be an alphabet with at least two symbols, let $L = a\Sigma^*$. It holds that:*
- (i) $L \in \mathcal{R}(2, 2)$;
 - (ii) $L \notin \mathcal{RPR}(k, 2)$, for every $k \geq 2$.

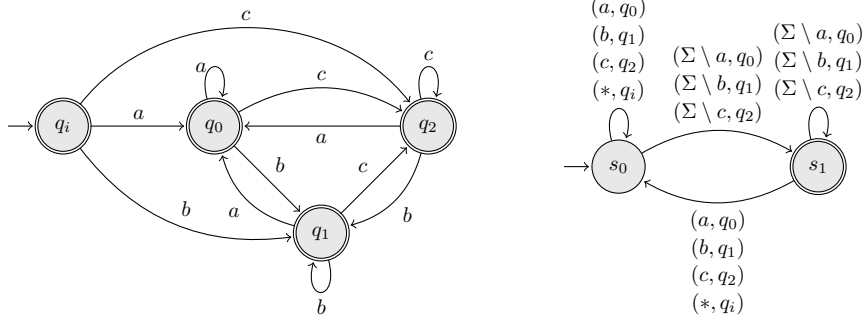
In Lemma 4.10, we have shown that the final component of a cascade can always be restricted to two states. A natural question arises: *Can the first component also be limited to just two states?* The answer is negative, as illustrated by the following example. Consider an alphabet of three symbols and the language L consisting of all words that end with two distinct symbols (e.g. $cb \in L$ but $aa \notin L$). As demonstrated in Figure 5, this language can be recognized by a cascade of two reset automata where the first component has three states. However, we will prove that it cannot be recognized if the first component has only two states. Intuitively, the first component’s role is to remember the second-to-last symbol, but with an alphabet of three symbols and only two states, this task becomes impossible. The following *Width-Hierarchy Lemma* formalizes this intuition, demonstrating the existence of an infinite hierarchy of languages that can be defined using cascades of two resets where the first component contains k states, but cannot be defined when the first component is restricted to $k - 1$ states.

- **Lemma 5.4 (Width-Hierarchy Lemma).** *For each $k > 2$, let $\Sigma = \{\sigma_0, \dots, \sigma_{k-1}\}$. Let $L_k = \Sigma^* (\Sigma^2 \setminus \bigcup_{0 \leq i < k} \sigma_i \sigma_i) \cup (\Sigma \setminus \sigma_0)$, it holds that:*
- (i) $L_k \in \mathcal{RPR}(k, 2)$;
 - (ii) $L_k \notin \mathcal{R}(k - 1, 2)$.

The following corollary (depicted in Figure 4) follows from Lemmas 5.2–5.4.

- **Corollary 5.5.** *It holds that:*
- $\emptyset \subsetneq \mathcal{R}(2) \subsetneq \mathcal{RPR}(2, 2) \subsetneq \mathcal{R}(2, 2)$;
 - $\mathcal{RPR}(3, 2) \not\subseteq \mathcal{R}(2, 2)$ and $\mathcal{R}(2, 2) \not\subseteq \mathcal{RPR}(3, 2)$.

Additionally, we prove that for a fixed alphabet Σ of cardinality k , any language over Σ expressible by a cascade of height 2 can also be expressed by a cascade of height 2 where the first component has at most $k + 1$ states. The intuition behind this is that, due to



■ **Figure 6** The cascade $C := \mathcal{A} \circ \mathcal{B}$ accepts the language $L'_3 := \Sigma^* (\Sigma^2 \setminus \{aa, bb, cc\})$. When treated as a single automaton, C consists of 8 states, in contrast to the minimal automaton for L'_3 , which has only 7 states.

the restriction of transitions to resets or identities, only a finite number of states can be reached from the initial state. This is formalized in the next lemma, which also establishes the optimality of the bound on the number of states of the first component.

► **Lemma 5.6.** *Let Σ be a finite alphabet with size $|\Sigma| = k$. Let L be a language such that $L \subseteq \Sigma^*$. For every $m \in \mathbb{N}^{>0}$, if $L \in \mathcal{R}(m, 2)$, then $L \in \mathcal{R}(k + 1, 2)$. Furthermore, there exists a language $L'_k \subseteq \Sigma^*$ such that $L'_k \in \mathcal{R}(k + 1, 2)$ but $L'_k \notin \mathcal{R}(k, 2)$.*

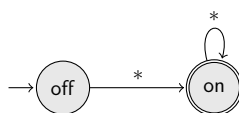
The language L'_k used to prove the optimality of the bound in Lemma 5.6 is defined as $\Sigma^* (\Sigma^2 \setminus \bigcup_{0 \leq i < k} \sigma_i \sigma_i)$. As an example, Figure 6 shows the case of L'_3 .

5.2 Narrow Cascades

Thus far, our discussion has centered around cascades composed of one or two components. Now, we shift our focus to *narrow cascades*, i.e. cascades of greater height but in which each component is restricted to have two states (i.e. \mathcal{R}_2^h). Just as we have seen that some languages cannot be expressed by cascades of height 1, we will demonstrate that for any given height, there exists a language that cannot be captured at that height, provided the components of the cascade are restricted to two states. We call this the *Height-Hierarchy Lemma*, and is a counterpart of the Width-Hierarchy Lemma (Lemma 5.4) focused on the height of cascades. It is based on the following family of languages: for each $h \geq 2$, we consider the language $L_h = \Sigma^{h-2} a \Sigma^*$, that is all words that contain symbol “a” precisely at position $h - 1$. The Height-Hierarchy Lemma below proves that, for any $h \geq 2$, the language L_h is *not* definable by cascades of two states reset automata of height less than h .

► **Lemma 5.7 (Height-Hierarchy Lemma).** *For each $h \geq 2$, let $L_h = \Sigma^{h-2} a \Sigma^*$. It holds that:*

- (i) $L_h \in \mathcal{R}_2^h$;
- (ii) $L_h \notin \mathcal{R}_2^{h-1}$.



■ **Figure 7** Switch automaton.

We briefly explain the intuition behind Lemma 5.7. Regarding point (i) $L_h \in \mathcal{R}_2^h$, the construction of the two-state reset cascade proceeds as follows. The base case corresponds to Figure 1, while the inductive step for height h involves the use of the two-state reset automaton \mathcal{A}_{switch} (illustrated in Figure 7) in cascade with the augmentation of the cascade for the case $h - 1$. Intuitively, \mathcal{A}_{switch} recognizes all words containing at least one symbol. Using in cascade $h - 2$ copies of \mathcal{A}_{switch} together with the cascade in Figure 1, corresponds exactly to the language $\Sigma^{h-2}a\Sigma^*$. In Figure 8, we provide an example of the construction for the case $h = 3$.

The proof that $L_h \notin \mathcal{R}_2^{h-1}$ is more involved and proceeds by induction on h . For the base case ($h = 2$), we have $L_2 = a\Sigma^*$. This case is verified by Lemma 5.3. For the inductive step, assume that the statement holds for every $i \leq h$. We need to prove that it also holds for $i = h + 1$. Suppose, by contradiction, that $L_{h+1} \in \mathcal{R}_2^h$. By definition, this would imply the existence of a cascade $C = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_h$ of two-state reset automata such that $\mathcal{L}(C) = L_{h+1} = \Sigma^{h-1}a\Sigma^*$. However, starting from C , the proof shows how to construct a new cascade $C' = \mathcal{B}_1 \circ \dots \circ \mathcal{B}_{h-1}$ of two-state reset automata such that $\mathcal{L}(C') = \Sigma^{h-2}a\Sigma^* = L_h$. The existence of C' contradicts the inductive hypothesis, which states that $L_h \notin \mathcal{R}_2^{h-1}$. Therefore, the assumption that $L_{h+1} \in \mathcal{R}_2^h$ must be false, and the cascade C cannot exist.

5.3 General Cascades

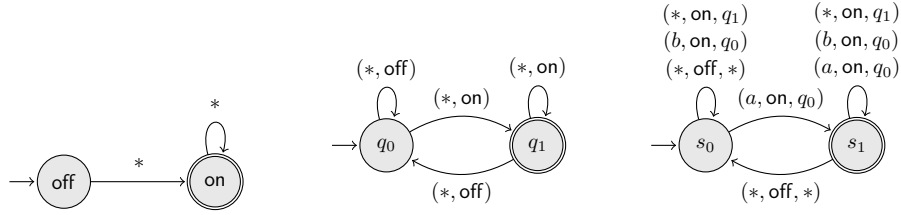
In this subsection, we examine cascades of reset automata without imposing restrictions on the number of states in each component or on the total number of components. In the previous part, Lemma 5.6 demonstrates that, for cascade of two resets over the alphabet Σ , the maximum expressiveness is achieved when the first component has $|\Sigma| + 1$ states. Here, we extend this result to cascades of unbounded height in the *Width-Collapse Lemma*, that provides lower bounds on the number of states in each component, for which adding states at certain levels does not affect expressiveness.

► **Lemma 5.8** (Width-Collapse Lemma). *Let Σ be a finite alphabet with $|\Sigma| = k \geq 2$. Let $L \subseteq \Sigma^*$ be a language. For any positive integers h and k_1, \dots, k_h , if $L \in \mathcal{R}(k_1, \dots, k_h, 2)$, then $L \in \mathcal{R}(f(1), \dots, f(h), 2)$, where $f(i) = \frac{k^{i+1}-1}{k-1}$.*

We now demonstrate how to transform *general cascades* into narrow cascades. Specifically, we show how any cascade of reset automata (of height h and with k_i states at level i , for each $i \in \{1, \dots, h\}$) can be transformed into an equivalent *narrow cascade* (i.e. made of two-state resets), at the cost of increasing its height at most by a factor of $2 + \sum_{i=1}^{h-1} \lceil \log_2(k_i) \rceil$. This result is based on two key points:

1. Given a general cascade, we can always append a pure-reset automaton at the end without altering its language;
2. the *Narrowing Lemma*, which we prove below, demonstrates that any cascade of reset automata, whose final component is pure-reset and containing a component \mathcal{A}_j with k_j states (and $k_j > 2$), can be transformed into a new cascade where \mathcal{A}_j is replaced by two new automata, with 2 and $\lceil \frac{k_j}{2} \rceil$ states each.

20:16 On Cascades of Reset Automata



■ **Figure 8** A cascade $C_3 = \mathcal{A}_1 \circ \mathcal{A}_2 \circ \mathcal{A}_3$ that recognizes the language $\Sigma a \Sigma^*$. The first two components enforce that any accepted word contains at least two symbols, as $\mathcal{L}(\mathcal{A}_1 \circ \mathcal{A}_2) = \Sigma \Sigma \Sigma^*$.

Instrumental to the Narrowing Lemma, the following result demonstrates that, given a general cascade whose last component is a pure-reset automaton, we can modify this last component to make all the states of the preceding components final, without altering the recognized language.

► **Lemma 5.9.** *Consider a cascade $\mathcal{A} \circ \mathcal{B}$ of automata, where \mathcal{B} is a two-state pure-reset automaton. Let \mathcal{A}' be the automaton obtained from \mathcal{A} by making all states final. Then, there exists a two-state pure-reset automaton \mathcal{B}' such that $\mathcal{L}(\mathcal{A} \circ \mathcal{B}) = \mathcal{L}(\mathcal{A}' \circ \mathcal{B}')$.*

The Narrowing Lemma is stated as follows.

► **Lemma 5.10 (Narrowing Lemma).** *Let $C = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_h$ be a cascade where \mathcal{A}_i is a reset automaton with k_i states for each $1 \leq i \leq h-1$, and \mathcal{A}_h is a pure-reset automaton. Let j be an index such that $1 \leq j \leq h-1$. Then, there exists a cascade C' of reset automata*

$$C' = \mathcal{A}'_1 \circ \dots \circ \mathcal{A}'_{j-1} \circ \mathcal{B}_1 \circ \mathcal{B}_2 \circ \mathcal{A}'_{j+1} \circ \dots \circ \mathcal{A}'_h$$

such that:

- (i) each \mathcal{A}'_i has k_i states for $i \neq j$;
- (ii) if \mathcal{A}_i is pure-reset (resp., reset), then also \mathcal{A}'_i is pure-reset (resp., reset), for $i \neq j$;
- (iii) \mathcal{B}_1 has 2 states and \mathcal{B}_2 has $\lceil \frac{k_j}{2} \rceil$ states; and
- (iv) $\mathcal{L}(C) = \mathcal{L}(C')$.

By iteratively applying the Narrowing Lemma to every component with more than two states, we obtain a procedure that, given a cascade of reset automata, produces an equivalent cascade where all components are two-state reset automata. Moreover, it is worth noticing that:

- (i) by Lemma 4.10, *w.l.o.g.* the last component of any cascades of reset (or pure-resets) has two states, and therefore the Narrowing Lemma does not need to be applied at the last level;
- (ii) if the final component of a cascade is not a pure-reset, a new pure-reset level can always be added without affecting the language of the cascade.

This leads to the following inclusions.

► **Corollary 5.11.** *For each positive h, k_1, \dots, k_h it holds that*

1. $\mathcal{RPR}(k_1, \dots, k_h) \subseteq \mathcal{RPR}_2^{H+1}$
 2. $\mathcal{R}(k_1, \dots, k_h) \subseteq \mathcal{R}_2^{H+2}$
- where $H = \lceil \log_2 k_1 \rceil + \dots + \lceil \log_2 k_{h-1} \rceil$.

Combining Lemma 5.8 and Corollary 5.11, we conclude that if a language L is recognized by a cascade of resets of height h , it can also be recognized by a cascade of height $\Theta(h^2)$ composed entirely of two-state resets.

► **Corollary 5.12.** *Let Σ be an alphabet such that $|\Sigma| = k \geq 2$ and let $L \subseteq \Sigma^*$ be a language. If L admits a cascade of reset automata of height h , then $L \in \mathcal{R}_2^H$ where $H \in \Theta(h^2)$. If $k = 2$, then $H = \frac{h^2+h+2}{2}$.*

Exploiting the bound for the case $|\Sigma| = 2$, we can prove undefinability of certain languages by *general cascades*, i.e. without any bound on their height nor on the number of states of its component. As an example, by Lemma 5.7, we know that the language $L = \Sigma^6 a \Sigma^*$ over the alphabet $\Sigma = \{a, b\}$ does not belong to the class \mathcal{R}_2^7 . If L could be recognized by a cascade of height $h = 3$, then it would also be recognized by a two-state cascade of height $H = \frac{h^2+h+2}{2} = 7$, leading to the following conclusion: with $\Sigma = \{a, b\}$, the language $\Sigma^6 a \Sigma^*$ does not admit any cascade of resets of height 3.

Building upon this reasoning, we can formulate the Generalized Height-Hierarchy Lemma. Unlike the original Height-Hierarchy Lemma, which focuses solely on two-state cascades, the generalized version addresses the undefinability of cascades in a broader context, encompassing general cascades.

► **Lemma 5.13 (Generalized Height-Hierarchy Lemma).** *Let h be a positive integer, and define $H = \frac{h^2+h+2}{2} + 1$. Consider the language $L_H \subseteq \Sigma^*$, where $L_H = \Sigma^{H-2} a \Sigma^*$ and Σ is a two-symbol alphabet. The language L_H cannot be recognized by any cascade of reset automata of height h , but it holds that $L_H \in \mathcal{R}_2^H$.*

6 Efficient closure properties of cascades of reset automata

In this section, we present an efficient method for computing specific closure properties of reset cascades. For instance, for the case in which the operation \otimes is binary, given two cascades of resets \mathcal{C} and \mathcal{C}' (made of only two-states components), we show how it is possible to compute a cascade of two-states resets that recognizes $\mathcal{L}(\mathcal{C}) \otimes \mathcal{L}(\mathcal{C}')$ by adding *at most one* two-state reset automaton (that, in this context, we call *brick*). We show this for the following operations:

- (i) intersection;
- (ii) complementation;
- (iii) union; and
- (iv) left-concatenation of Σ , i.e. given a language \mathcal{L} to compute $\Sigma \cdot \mathcal{L}$.⁴

Proposition 4.5 already shows that *intersection* can be implemented efficiently for cascades of resets: given two reset cascades \mathcal{C} and \mathcal{C}' (with m and n two-states components, respectively), there exists a cascade for $\mathcal{L}(\mathcal{C}) \cap \mathcal{L}(\mathcal{C}')$ with $m + n$ two-state resets.

Before showing the construction for the remaining operations, we give the following key definitions. We define the *finalized version* of an automaton \mathcal{A} , denoted with $\text{finv}(\mathcal{A})$, as the automaton obtained from \mathcal{A} by setting all its states as final. The definition naturally extends to cascades: the finalized version of \mathcal{C} , denoted with $\text{finv}(\mathcal{C})$, is defined as $\text{finv}(\mathcal{A}_1) \circ \dots \circ \text{finv}(\mathcal{A}_n)$. Clearly, if \mathcal{C} is a cascade of reset automata, $\text{finv}(\mathcal{C})$ is still a cascade of reset automata. We define the *reachability set* of an automaton relative to a set of states as follows.

► **Definition 6.1 (Reachability Set).** *Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ be an automaton. Let $P \subseteq Q$ be a set of states. The reachability set of \mathcal{A} with respect to P , denoted with $\text{RS}(\mathcal{A}, P)$, is the set $\{(\sigma, q) \in (\Sigma \times Q) : \delta(q, \sigma) \in P\}$. We denote with $\overline{\text{RS}(\mathcal{A}, P)}$ the set $(\Sigma \times Q) \setminus \text{RS}(\mathcal{A}, P)$. We write $\text{RS}(\mathcal{A})$ to refer to $\text{RS}(\mathcal{A}, F)$.*

⁴ It is worth noticing that this operation corresponds to compute the closure under the LTL *next* modality.



■ **Figure 9** The *negation brick* $\text{negb}(\mathcal{A})$ in the two cases: (a) $\epsilon \in \mathcal{L}(\mathcal{A})$ (b) $\epsilon \notin \mathcal{L}(\mathcal{A})$.

We now show how to efficiently compute the remaining closure properties.

Complementation

To compute complementation, we introduce the *negation brick*, whose structure is illustrated in Figure 9 and is formally defined here below.

► **Definition 6.2** (Negation brick). *Let $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$ be an automaton. The negation brick for \mathcal{A} , denoted with $\text{negb}(\mathcal{A})$, is the two-state pure-reset automaton $\langle \Sigma \times Q, \{n_0, n_1\}, \delta, n_0, \{n_f\} \rangle$ such that:*

- (i) *the final state n_f is n_1 if and only if $\epsilon \in \mathcal{L}(\mathcal{A})$;*
- (ii) *the function τ induced by symbols in $\text{RS}(\mathcal{A})$ maps all states in the non-final state, i.e. $\tau : \{n_0, n_1\} \mapsto \{n_0, n_1\} \setminus \{n_f\}$;*
- (iii) *the function τ' induced by symbols in $\overline{\text{RS}(\mathcal{A})}$ maps all states in the final one, i.e. $\tau' : \{n_0, n_1\} \mapsto \{n_f\}$.*

The intuition is that the negation brick, when appended to the end of a cascade \mathcal{C} , reaches its final state if and only if the underlying cascade \mathcal{C} is not in a final state. Consequently, by setting all the states of \mathcal{C} as final, we obtain a cascade that recognizes the complement of $\mathcal{L}(\mathcal{C})$, as proved by the following lemma.

► **Lemma 6.3.** *Let \mathcal{C} be a cascade of automata. The cascade $\mathcal{C}' := \text{finv}(\mathcal{C}) \circ \text{negb}(\mathcal{C})$ recognizes the language $\overline{\mathcal{L}(\mathcal{C})}$. Moreover, if \mathcal{C} is a cascade of reset automata, then so is \mathcal{C}' .*

Interestingly, if the cascade terminates with a pure-reset layer \mathcal{A} , this automaton can itself serve the function of the negation brick, without the need of an additional component.

► **Lemma 6.4.** *Let $\mathcal{C} = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_n$ be a cascade of automata such that \mathcal{A}_n is a pure-reset automaton. There exists a cascade $\mathcal{C}' = \mathcal{A}'_1 \circ \dots \circ \mathcal{A}'_n$ such that:*

- (i) $\mathcal{L}(\mathcal{C}') = \overline{\mathcal{L}(\mathcal{C})}$;
- (ii) *each automaton \mathcal{A}'_i has the same number of states as \mathcal{A}_i ;*
- (iii) *if \mathcal{A}_i is a reset (resp., pure-reset), then \mathcal{A}'_i is also a reset (resp., pure-reset).*

Union

Given two cascades \mathcal{C} and \mathcal{C}' of height m and n , respectively, since $\mathcal{L}(\mathcal{C}) \cup \mathcal{L}(\mathcal{C}') = \overline{\overline{\mathcal{L}(\mathcal{C})} \cap \overline{\mathcal{L}(\mathcal{C}')}}$, it is possible to build cascade for $\mathcal{L}(\mathcal{C}) \cup \mathcal{L}(\mathcal{C}')$ of height $m + n + 3$, using the previously discussed constructions. In this section, we present a more efficient construction that introduces only one additional component, referred to as the *union brick*, resulting in a cascade for $\mathcal{L}(\mathcal{C}) \cup \mathcal{L}(\mathcal{C}')$ of height $n + m + 1$.

► **Definition 6.5** (Union brick). *Let $\mathcal{A} = \langle \Sigma, Q_A, \delta_A, q_{0A}, F_A \rangle$ and $\mathcal{B} = \langle \Sigma, Q_B, \delta_B, q_{0B}, F_B \rangle$ be two automata. Let $U \subseteq Q_A \times Q_B$ the set of states $\{(q_A, q_B) : q_A \in F_A \vee q_B \in F_B\}$. Let $\mathcal{C} = \mathcal{A} \circ \text{aug}(\mathcal{A}, \mathcal{B})$. The union brick of \mathcal{A} and \mathcal{B} , denoted with $\text{unionb}(\mathcal{A}, \mathcal{B})$, is the two-state pure-reset automaton $\langle \Sigma \times Q, \{u_0, u_1\}, \delta, u_0, \{u_f\} \rangle$ such that:*

- (i) the final state u_f is u_0 if and only if $\epsilon \in \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$;
- (ii) the function τ induced by symbols in $\text{RS}(\mathcal{C}, U)$ maps all states in the final one, i.e. $\tau : \{n_0, n_1\} \mapsto \{n_f\}$;
- (iii) the function τ' induced by symbols in $\overline{\text{RS}(\mathcal{C}, U)}$ maps all states in the non-final state, i.e. $\tau : \{n_0, n_1\} \mapsto \{n_0, n_1\} \setminus \{n_f\}$.

Similarly to the case of complementation, when appended to the end of a cascade $\mathcal{C} \circ \text{aug}(\mathcal{C}, \mathcal{C}')$, the union brick reaches its final state if and only if either \mathcal{C} is in a final state or $\text{aug}(\mathcal{C}, \mathcal{C}')$ is in a final state. This leads to the following lemma.

► **Lemma 6.6.** *Let \mathcal{C} and \mathcal{C}' be two cascade of automata. The cascade $\mathcal{C}'' := \text{finv}(\mathcal{C} \circ \text{aug}(\mathcal{C}, \mathcal{C}')) \circ \text{unionb}(\mathcal{C}, \mathcal{C}')$ recognizes the language $\mathcal{L}(\mathcal{C}) \cup \mathcal{L}(\mathcal{C}')$. Moreover, if \mathcal{C} and \mathcal{C}' are cascades of reset automata, then so is \mathcal{C}'' .*

Also in this case, if one of the two automata corresponds to a cascade terminating with a pure-reset component \mathcal{A} , the union can be performed without the need for additional layers: the automaton \mathcal{A} effectively serves as the union brick.

Left-concatenation of Σ

Given a cascade \mathcal{C} , we demonstrate how to construct a cascade that recognizes the language $\Sigma \cdot \mathcal{L}(\mathcal{C})$, adding only one brick and guaranteeing that the property of being a reset cascade is preserved. As a by-product of this construction, we obtain that, given a cascade of resets of height h equivalent to an LTL formula ϕ (interpreted over finite words), it is possible to construct a cascades of resets for $\text{X}(\phi)$ of height $h + 1$, where X is the *next* modality of LTL.

We first define the *next version* of an automaton. The next version of an automaton \mathcal{A} , denoted as $\text{nextv}(\mathcal{A})$, is defined considering the Cartesian product between the alphabet of \mathcal{A} and the set $\{\text{off}, \text{on}\}$. Intuitively, if \mathcal{A} transitions from q to q' with a symbols σ , so does $\text{nextv}(\mathcal{A})$ with the symbol (σ, on) . On the contrary, all symbols (σ, off) force $\text{nextv}(\mathcal{A})$ to transition to the initial state. The formal definition of $\text{nextv}(\mathcal{A})$ is given here below.

► **Definition 6.7** (Next version of an automaton). *Let $\mathcal{A} = \langle \Sigma', Q, \delta, q_0, F \rangle$ be an automaton such that either $\Sigma' = \Sigma$ or $\Sigma' = \Sigma \times S$, for an arbitrary finite set S . We define the next version of \mathcal{A} , denoted as $\text{nextv}(\mathcal{A})$, as the automaton $(\Sigma'', Q, \delta', q_0, F)$ such that:*

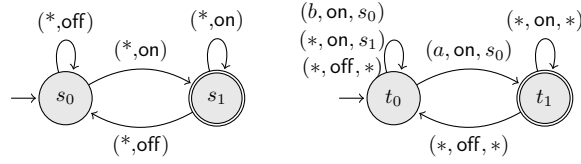
- if $\Sigma' = \Sigma$, then $\Sigma'' := \Sigma \times \{\text{off}, \text{on}\}$ and, for all $q \in Q$ and for all $a \in \Sigma$, it holds: $\delta'(q, (a, \text{on})) = \delta(q, a)$ and $\delta'(q, (*, \text{off})) = q_0$.
- if $\Sigma' = \Sigma \times S$, then $\Sigma'' := \Sigma \times \{\text{off}, \text{on}\} \times S$ and, for all $q \in Q$ and for all $(a, s) \in \Sigma \times S$, it holds that: $\delta'(q, (a, \text{on}, s)) = \delta(q, (a, s))$ and $\delta'(q, (*, \text{off}, *)) = q_0$.

Given a cascade $\mathcal{C} = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_n$ over Σ , we define the next version of \mathcal{C} , denoted with $\text{nextv}(\mathcal{C})$, as the cascade $\text{nextv}(\mathcal{A}_1) \circ \dots \circ \text{nextv}(\mathcal{A}_n)$ over $\Sigma \times \{\text{off}, \text{on}\}$.

Figure 10 shows the next versions of the automata in Figure 1. Crucially, computing the next version of an automaton does not alter its property of being a reset automaton.

► **Lemma 6.8.** *Let Σ be a finite alphabet and let \mathcal{A} be an automaton over Σ or over $\Sigma \times S$ for an arbitrary finite set S . If \mathcal{A} is reset automaton, then also $\text{nextv}(\mathcal{A})$ is a reset automaton.*

Now, given any cascade \mathcal{C} , to capture the language $\Sigma \cdot \mathcal{L}(\mathcal{C})$, it suffices to consider the automaton $\mathcal{A}_{\text{switch}}$ (depicted in Figure 7) with the next version of \mathcal{C} . In fact, considering that initially both $\mathcal{A}_{\text{switch}}$ and $\text{nextv}(\mathcal{C})$ are in their initial states (which, for $\mathcal{A}_{\text{switch}}$, is state off), reading the first input symbol σ forces:



■ **Figure 10** The next version of the two automata in the cascade of Figure 1.

- (i) \mathcal{A}_{switch} to transition to state on; and
- (ii) $nextv(\mathcal{C})$ to remain in its initial state, because the symbol it reads is (σ, off) .

After the first symbol and for all the rest of the input word, \mathcal{A}_{switch} remains in state on, while $nextv(\mathcal{C})$ operates like \mathcal{C} because it reads symbols of the form (σ', on) . As shown by the following lemma, this captures exactly $\Sigma \cdot \mathcal{L}(\mathcal{C})$.

► **Lemma 6.9.** *Let \mathcal{C} be a cascade of automata. The cascade $\mathcal{C}' := \mathcal{A}_{switch} \circ nextv(\mathcal{C})$ recognizes the language $\Sigma \cdot \mathcal{L}(\mathcal{C})$. Moreover, if \mathcal{C} is a cascade of reset automata, then so is \mathcal{C}' .*

From Lemma 5.7, it follows the optimality of the construction outlined in Lemma 6.9.

7 Conclusions and Future Work

In this paper, we investigated some fundamental properties of cascades of reset automata. Unlike the approach commonly followed in the literature, where the cascade product is restricted to semi-automata, we focused on the case of automata. This allowed us to study the properties of the recognized languages. As an initial step, we showed how to compute regular expressions equivalent to a cascade. Then, on the basis of such a transformation, we established some meaningful expressiveness results, in particular lower bounds to the height and to the minimum number of states per level of a cascade of resets for specific families of languages. Finally, we showed how to compute the closure of reset cascades under certain basic operations by adding at most one brick to the end of the cascade.

As for the future developments of the work, finding an efficient construction for the closure of reset cascades under the concatenation operation is undoubtedly a crucial direction. This would enable the design of an efficient approach to handling the *eventually* and *until* operators of LTL, providing, together with the results given in the last section of the paper, an efficient decomposition into reset cascades for full LTL. This would improve the triply-exponential upper bound to such a decomposition achieved by Maler’s algorithm [4, 15]. Last but not least, giving analogous expressiveness and closure results for permutation automata appears to be another promising avenue for further investigation.

References

- 1 Dana Angluin, David Chiang, and Andy Yang. Masked hard-attention transformers and boolean RASP recognize exactly the star-free languages. *CoRR*, abs/2310.13897, 2023. doi:10.48550/arXiv.2310.13897.
- 2 Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari. A Singly Exponential Transformation of LTL[X, F] into Pure Past LTL. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 65–74, 2023. doi:10.24963/KR.2023/7.

- 3 Alessandro Artale, Luca Geatti, Nicola Gigante, Andrea Mazzullo, and Angelo Montanari. Succinctness issues for LTLf and safety and cosafety fragments of LTL. *Information and Computation*, 302:105262, 2025. doi:10.1016/j.ic.2024.105262.
- 4 Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4959–4965, 2021.
- 5 Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 854–860. IJCAI/AAAI, 2013. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997>.
- 6 Samuel Eilenberg. *Automata, languages, and machines., B. Pure and applied mathematics.* Academic Press, 1976. URL: <https://www.worldcat.org/oclc/310535259>.
- 7 Javier Esparza, Rubén Rubio, and Salomon Sickert. Efficient Normalization of Linear Temporal Logic. *J. ACM*, 71(2):16:1–16:42, 2024. doi:10.1145/3651152.
- 8 Marcus Gelderie. Classifying regular languages via cascade products of automata. In *Language and Automata Theory and Applications: 5th International Conference, LATA 2011, Tarragona, Spain, May 26-31, 2011. Proceedings 5*, pages 286–297. Springer, 2011. doi:10.1007/978-3-642-21254-3_22.
- 9 Mark Kambites. On the krohn-rhodes complexity of semigroups of upper triangular matrices. *Int. J. Algebra Comput.*, 17(1):187–201, 2007. doi:10.1142/S0218196707003548.
- 10 Nadezda Alexandrovna Knorozova and Alessandro Ronca. On the expressivity of recurrent neural cascades. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10589–10596. AAAI Press, 2024. doi:10.1609/AAAI.V38I9.28929.
- 11 Nadezda Alexandrovna Knorozova and Alessandro Ronca. On the expressivity of recurrent neural cascades with identity. In Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco, editors, *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*, 2024. doi:10.24963/KR.2024/82.
- 12 Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.
- 13 Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: <https://openreview.net/forum?id=De4FYqjFueZ>.
- 14 Oded Maler. On the Krohn-Rhodes Cascaded Decomposition Theorem. In Zohar Manna and Doron A. Peled, editors, *Time for Verification, Essays in Memory of Amir Pnueli*, volume 6200 of *Lecture Notes in Computer Science*, pages 260–278. Springer, 2010. doi:10.1007/978-3-642-13754-9_12.
- 15 Oded Maler and Amir Pnueli. Tight Bounds on the Complexity of Cascaded Decomposition of Automata. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 672–682. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89589.
- 16 Zohar Manna and Amir Pnueli. A hierarchy of temporal properties (invited paper, 1989). In *Proceedings of the 9th annual ACM symposium on Principles of distributed computing*, pages 377–410, 1990. doi:10.1145/93385.93442.
- 17 Nicolas Markey. Temporal logic with past is exponentially more succinct, concurrency column. *Bull. EATCS*, 79:122–128, 2003.

20:22 On Cascades of Reset Automata

- 18 Robert McNaughton and Seymour A Papert. *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press, 1971.
- 19 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977. doi:10.1109/SFCS.1977.32.
- 20 Alessandro Ronca. The transformation logics. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 3549–3557. ijcai.org, 2024. URL: <https://www.ijcai.org/proceedings/2024/393>.
- 21 Alessandro Ronca, Nadezda Alexandrovna Knorozova, and Giuseppe De Giacomo. Automata cascades: Expressivity and sample complexity. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 9588–9595. AAAI Press, 2023. doi:10.1609/AAAI.V37I8.26147.
- 22 Karl-Heinz Zimmermann. On Krohn-Rhodes theory for semiautomata. *CoRR*, abs/2010.16235, 2020. arXiv:2010.16235.

Computability of Extender Sets in Multidimensional Subshifts

Antonin Callard ✉ 🏠 

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000, Caen, France

Léo Paviet Salomon ✉ 

Université de Lorraine, CNRS, Inria, LORIA, 54000, Nancy, France

Pascal Vanier ✉ 🏠 

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000, Caen, France

Abstract

Subshifts are sets of colorings of \mathbb{Z}^d defined by families of forbidden patterns. Given a subshift and a finite pattern, its extender set is the set of admissible completions of this pattern. It has been conjectured that the behavior of extender sets, and in particular their growth called *extender entropy* [10], could provide a way to separate the classes of sofic and effective subshifts. We prove here that both classes have the same possible extender entropies: exactly the Π_3 real numbers of $[0, +\infty)$.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Theory of computation → Models of computation

Keywords and phrases Symbolic dynamics, subshifts, extender sets, extender entropy, computability, sofic shifts, tilings

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.21

Related Version *Extended Version*: <https://doi.org/10.48550/arXiv.2401.07549>

Funding This research was partially funded by ANR JCJC 2019 19-CE48-0007-01.

Acknowledgements We are thankful to the referees for their many helpful remarks and suggestions.

1 Introduction

In dimension $d \in \mathbb{N}$, subshifts are sets of colorings of \mathbb{Z}^d where a family of patterns, *i.e.* colorings of finite portions of \mathbb{Z}^d , have been forbidden. They were originally introduced to discretize continuous dynamical systems [19]. One of the main families of subshifts that has been studied is the class of *subshifts of finite type* (SFTs), which can be defined with a finite family of forbidden patterns. This class has independently been introduced under the formalism of Wang tiles [25] in dimension 2 in order to study fragments of second order logic.

In dimension 1, *sofic subshifts* [26], which are obtained as letter-to-letter projections of SFTs, are studied mainly through their defining graphs. In dimension 2 and higher, SFTs (and thus sofic subshifts) can embed arbitrary Turing machine computations; as such, the main tool in the study of subshifts becomes computability theory. This led to the introduction of a new class of subshifts, the *effective subshifts*, which can be defined by computably enumerable families of forbidden patterns [13].

An important question in symbolic dynamics is thus to find criteria separating sofic from effective subshifts [15, 22, 12, 3]. In dimension 1, a subshift is sofic if it can be defined by a regular language of forbidden patterns: the Myhill–Nerode theorem states that these are exactly the languages that have finitely many Nerode congruence classes. In dimension 2 and higher, no such clear characterization exists. Indeed, many effective subshifts have been



© Antonin Callard, Léo Paviet Salomon, and Pascal Vanier;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 21; pp. 21:1–21:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



proved to be sofic, such as substitutive subshifts [20], or effective subshifts on $\{0, 1\}$ whose densities of symbols 1 are sublinear [6]; it even turns out that sofic subshifts of dimension $d + 1$ capture all the behaviors of effective shifts of dimension d [13, 8, 2].

All the methods used to prove some cases of non-soficity that are known by the authors revolve around a counting argument: only a linear amount of information may cross the border of an $n \times n$ square pattern (see for example [1, Proposition 9.4.5]). The most recent argument in this vein uses resource-bounded Kolmogorov complexity [7].

One formalization of this counting argument relies on *extender sets* of patterns [15], which can be considered as a higher-dimensional generalization of Nerode congruence classes: the extender set of a pattern p is the set of all configurations with a p -shaped hole that may extend p . For SFTs, the extender set of a given pattern is entirely determined by its boundary, which implies that the number of extender sets of an SFT cannot grow too quickly. For subshifts in dimension 1, [21, Lemma 3.4] proves the analog of Myhill-Nerode theorem: a subshift is sofic if and only if its number of extender sets of every size is bounded. In dimension 2 and higher, only sufficient conditions are known: for example, a subshift whose number of extender sets for patterns of size n^d is bounded by n must be sofic [21].

The study of the growth rate of the number of extender sets can be done asymptotically through the notion of the *extender entropy*, which is defined in a similar way to the classical notion of topological entropy [17]. Extender entropies in fact relate to the to notion of follower entropies [4], but are more robust in the sense that, despite it not being decreasing under factor map applications, the extender entropy of a subshift is still a conjugacy invariant.

In this paper, we achieve characterizations of the possible extender entropies in terms of computability, in the same vein as recent results on conjugacy invariants of subshifts [14, 18].

► **Theorem A.** *The set of extender entropies of \mathbb{Z} effective subshifts is exactly $\Pi_3 \cap [0, +\infty)$.*

► **Theorem B.** *The set of extender entropies of \mathbb{Z}^2 sofic subshifts is exactly $\Pi_3 \cap [0, +\infty)$.*

These results generalize to dimension $d \geq 2$ by Claim 8 and Corollary 12. While sofic subshifts were conjectured in [15] to have extender entropy zero, this was later disproved (see for example [7]); in fact, our characterization shows that the possible values are dense in $[0, +\infty)$. This also proves that extender entropies do not separate sofic from effective shifts.

	\mathbb{Z}	$\mathbb{Z}^d, d \geq 2$
SFT	{0} (Folklore: see Proposition 6)	
Sofic	{0} ([9, Theorem 1.1])	Π_3 (Theorem B)
Effective	Π_3 (Theorem A)	
Computable (\mathbb{Z} effective, \mathbb{Z}^d sofic)	Π_2 (Theorem 27)	
Sofic and minimal	{0} (Corollary 29)	
Effective and minimal	Π_1 (Corollary 30)	
Effective and 1-Mixing/Block-Gluing	Π_3 (Proposition 32)	Π_3 (Proposition 34)

■ **Figure 1** Sets of possible extender entropies for various classes of subshifts.

Finally, we also study extender entropies of subshifts constrained by some dynamical assumptions, such as minimality or mixingness. What is known by the authors at this stage can be summed up by the table Figure 1.

2 Definitions

2.1 Subshifts

Let \mathcal{A} denote a finite set of symbols and $d \in \mathbb{N}$ the dimension. A *configuration* is a coloring $x \in \mathcal{A}^{\mathbb{Z}^d}$, and the color of x at position $p \in \mathbb{Z}^d$ is denoted by x_p . A (d -dimensional) *pattern* over \mathcal{A} is a coloring $w \in \mathcal{A}^P$ for some set $P \subseteq \mathbb{Z}^d$ called its *support*¹. For any pattern w over \mathcal{A} of support P , we say that w *appears* in a configuration x (and we denote $w \sqsubseteq x$) if there exists $p_0 \in \mathbb{Z}^d$ such that $w_p = x_{p+p_0}$ for all $p \in P$.

The *shift functions* $(\sigma^t)_{t \in \mathbb{Z}^d}$ act on configurations as $(\sigma^t(x))_p = x_{p+t}$. For $t \in \mathbb{Z}^d$, a configuration x is *t-periodic* if $\sigma^t(x) = x$. We sometimes consider patterns or configuration by their restriction: for $S \subseteq \mathbb{Z}^d$ either finite or infinite, and $x \in \mathcal{A}^{\mathbb{Z}^d}$ a configuration (resp. w a pattern), we denote by $x|_S$ (resp. $w|_S$) the coloring of \mathcal{A}^S it induces on S .

► **Definition 1** (Subshift). *For any family of finite patterns \mathcal{F} , we define*

$$X_{\mathcal{F}} = \left\{ x \in \mathcal{A}^{\mathbb{Z}^d} \mid \forall w \in \mathcal{F}, w \not\sqsubseteq x \right\}$$

A set $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ is called a subshift if it is equal to some $X_{\mathcal{F}}$.

Given a subshift X and a finite support $P \subseteq \mathbb{Z}^d$, we define $\mathcal{L}_P(X)$ as the set of patterns w of support P that appear in the configurations of X . Such patterns are said to be *globally admissible* in X . We define the *language* of X as $\mathcal{L}(X) = \bigcup_{P \subseteq \mathbb{Z}^d \text{ finite}} \mathcal{L}_P(X)$. Slightly abusing notations, we denote $\mathcal{L}_n(X) = \mathcal{L}_{\llbracket 0, n-1 \rrbracket^d}(X)$ for $n \in \mathbb{N}$.

For $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ and $Y \subseteq \mathcal{B}^{\mathbb{Z}^d}$ two subshifts, $\varphi: X \rightarrow Y$ is a *factor map* if there exists some $N \subseteq \mathbb{Z}^d$ and $f: \mathcal{A}^N \rightarrow \mathcal{B}$ such that $\varphi(x)_p = f(x|_{p+N})$: then Y is a *factor* of X . X and Y are *conjugate* if there exists a bijective factor map $\varphi: X \rightarrow Y$ (called a conjugacy). Any object associated with subshifts that is preserved by conjugacy is a *conjugacy invariant*.

Subshifts can be classified as follows: a subshift is *of finite type* (SFT) if it is equal to $X_{\mathcal{F}}$ for some finite family \mathcal{F} of forbidden patterns; a subshift X is *effective* if it is equal to $X_{\mathcal{F}}$ for some computably enumerable family \mathcal{F} of forbidden patterns; and a subshift is *sofic* if it is a factor of some SFT, called its *SFT cover*. SFTs are sofic by definition, and sofic subshifts are effective.

Reciprocally, for a \mathbb{Z}^d subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, define the following *lifts*:

- *the periodic lift* $X^\uparrow = \{x^\uparrow \in \mathcal{A}^{\mathbb{Z}^{d+1}} \mid x \in X\}$, where $(x^\uparrow)|_{\mathbb{Z}^d \times \{i\}} = x$ for all $i \in \mathbb{Z}$;
- *the free lift* $X^\uparrow = \{y \in \mathcal{A}^{\mathbb{Z}^{d+1}} \mid \forall i \in \mathbb{Z}, y_{\mathbb{Z}^d \times \{i\}} \in X\}$.

If X is sofic (resp. effective), then both X^\uparrow and X^\uparrow are also sofic (resp. effective) since they can be defined by the same forbidden patterns. On the other hand:

► **Theorem 2** ([13], [2, Theorem 3.1], [8, Theorem 10]). *If X is an effective \mathbb{Z}^d subshift, then X^\uparrow is a sofic \mathbb{Z}^{d+1} subshift.*

Finally, most of our constructions will involve the notion of *layers*: for a subshift of a cartesian product $X \subseteq \prod_{i \in I} L_i$, the layers of X are the projections of X onto each of the L_i , which are often named for convenience. For $J \subseteq I$, we will denote by $\pi_{L_{j_1} \times L_{j_2} \times \dots}: \prod_{i \in I} L_i \mapsto \prod_{j \in J} L_j$ the cartesian projection.

¹ It is sometimes convenient to consider patterns up to the translation of their support. Usually, context will make it clear whether patterns are truly equal, or only up to a \mathbb{Z}^d translation.

2.2 Pattern Complexity and Extender Sets

The traditional notion of complexity is called *pattern complexity* and is defined by $N_X(n) = \mathcal{L}_n(X)$. The exponential growth rate of $|N_X(n)|$ is the *topological entropy*:

$$h(X) = \lim_{n \rightarrow +\infty} \frac{\log |N_X(n)|}{n^d}.$$

In this article, we focus on another notion of complexity based on extender sets:

► **Definition 3** (Extender set). *For $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ a d -dimensional subshift, $P \subseteq \mathbb{Z}^d$ and $w \in \mathcal{A}^P$ a pattern of support P , the extender set of w is the set*

$$E_X(w) = \{x \in \mathcal{A}^{\mathbb{Z}^d \setminus P} \mid x \sqcup w \in X\},$$

where $(x \sqcup w)_p = w_p$ if $p \in P$ and $(x \sqcup w)_p = x_p$ otherwise.

In other words, $E_X(w)$ is the set of all possible valid “completions” of the pattern w in X . For example, for two patterns with the same support w, w' , we have $E_X(w) \subseteq E_X(w')$ if and only if the pattern w can be replaced by w' every time it appears in any configuration of X .

In the case of \mathbb{Z} subshifts, extender sets are similar to the more classical notions of *follower* (resp. *predecessor*) *sets*, which are the set of right-infinite (resp. left-infinite) words that complete a finite given pattern (see for example [9]). Parallels can also be drawn with Nerode congruence classes.

For X a \mathbb{Z}^d subshift, denote $E_X(n) = \{E_X(w) \mid w \in \mathcal{L}_n(X)\}$ its set of extender sets. The *extender set sequence* $(|E_X(n)|)_{n \in \mathbb{N}}$ and its growth rate² are defined in [9]:

► **Definition 4** ([10, Definition 2.17]). *For a \mathbb{Z}^d subshift X , its extender entropy is*

$$h_E(X) = \lim_{n \rightarrow +\infty} \frac{\log |E_X(n)|}{n^d}.$$

This limit is well-defined by the multivariate subadditive lemma (see [5, Theorem 1]). In particular, $h_E(X) = \inf_{n \rightarrow +\infty} \frac{\log |E_X(n)|}{n^d}$, and $h_E(X)$ could actually be computed along any sequence of hyperrectangles that eventually fills \mathbb{Z}^d .

Examples

1. Let us consider $X = \mathcal{A}^{\mathbb{Z}^d}$ some full-shift in dimension d . Then X has maximal topological entropy, but $h_E(X) = 0$: indeed, for any two patterns $w, w' \in \mathcal{L}_n(X)$, we have $E_X(w) = E_X(w') = \{\mathcal{A}^{\mathbb{Z}^d \setminus \llbracket 0, n-1 \rrbracket^d}\}$; which implies that $|E_X(n)| = 1$ for every $n \in \mathbb{N}$.
2. Let us consider X a (strongly) periodic subshift: there exist $p_1, \dots, p_d \in \mathbb{N}$ such that, for every $x \in X$ and $i \leq d$, we have $\sigma^{p_i \cdot e_i}(x) = x$. Then X has zero topological entropy, and we also have $h_E(X) = 0$. Indeed, for $n \geq \max p_i$ and $w \in \mathcal{L}_n(X)$, w is the only pattern w' such that $E_X(w') = E_X(w)$; so that $|E_X(n)| = |\mathcal{L}_n(X)| \leq p \mathcal{A}^p$ for $p = \prod_i p_i$.

Some Properties

► **Theorem 5** (From [10] on \mathbb{Z} subshifts). *On \mathbb{Z}^d subshifts:*

- h_E is a conjugacy invariant.
- h_E is not necessarily decreasing under factor map.

² The authors define it for \mathbb{Z} subshifts, but the definition makes sense for higher dimensional shifts.

- h_E is additive under product (i.e. for X, Y two subshifts, $h_E(X \times Y) = h_E(X) + h_E(Y)$).
- h_E is upper bounded by h (i.e. for X a subshift, $h_E(X) \leq h(X)$).

For SFTs, the following proposition is folklore:

► **Proposition 6** ([15, Section 2]). *Let X be a d -dimensional SFT. Then $h_E(X) = 0$.*

Sketch of proof. In an SFT defined by adjacency constraints, the extender set of a pattern $w \in \mathcal{A}^{\llbracket 0, n-1 \rrbracket^d}$ is determined by its border; and there are at most $2^{O(n^{d-1})}$ such borders. ◀

By an analog of the Myhill-Nerode theorem, \mathbb{Z} sofic subshifts have extender entropy zero:

► **Proposition 7** ([21, Lemma 3.4]). *Let X be a 1-dimensional subshift. Then X is sofic if and only if $(|E_n(X)|)_{n \in \mathbb{N}}$ is uniformly bounded.*

2.3 Computability Notions

2.3.1 Arithmetical Hierarchy

The *arithmetical hierarchy* [24, Chapter 4] stratifies formulas of first-order arithmetic over \mathbb{N} by the number of their alternating unbounded quantifiers: for $n \in \mathbb{N}$, define

$$\begin{aligned} \Pi_n^0 &= \{\forall k_1, \exists k_2, \forall k_3, \dots \phi(k_1, \dots, k_n) \mid \phi \text{ only contains bounded quantifiers}\} \\ \Sigma_n^0 &= \{\exists k_1, \forall k_2, \exists k_3, \dots \phi(k_1, \dots, k_n) \mid \phi \text{ only contains bounded quantifiers}\}. \end{aligned}$$

A decision problem is said to be in Π_n^0 (resp. Σ_n^0) if its set of solutions $S \subseteq \mathbb{N}$ is described by a Π_n^0 (resp. Σ_n^0) formula: in other words, $\Pi_0^0 = \Sigma_0^0$ corresponds to the set of computable decision problems; Σ_1^0 is the set of computably enumerable decision problems, etc. . .

2.3.2 Arithmetical Hierarchy of Real Numbers

The *arithmetical hierarchy of real numbers* [27] stratifies real numbers depending on the difficulty of computably approximating them: for $n \geq 0$, define

$$\begin{aligned} \Sigma_n &= \{x \in \mathbb{R} \mid \{r \in \mathbb{Q} \mid r \leq x\} \text{ is a } \Sigma_n^0 \text{ set}\} \\ \Pi_n &= \{x \in \mathbb{R} \mid \{r \in \mathbb{Q} \mid r \geq x\} \text{ is a } \Sigma_n^0 \text{ set}\} = \{x \in \mathbb{R} \mid \{r \in \mathbb{Q} \mid r \leq x\} \text{ is a } \Pi_n^0 \text{ set}\}. \end{aligned}$$

In particular, $\Sigma_0 = \Pi_0$ is the set of computable real numbers, *i.e.* numbers that can be computably approximated up to arbitrary precision; Π_1 real numbers are also called right-computable, since they can be computably approximated from above; etc. . .

Alternatively, this hierarchy is also defined by the number of alternating limit operations needed to obtain a real number from the computable ones [27]. In other words, for $n \geq 1$:

$$\begin{aligned} \Sigma_n &= \left\{ \sup_{k_1 \in \mathbb{N}} \inf_{k_2 \in \mathbb{N}} \sup_{k_3 \in \mathbb{N}} \dots \beta_{k_1, \dots, k_n} \mid (\beta_{k_1, \dots, k_n})_{k_1, \dots, k_n \in \mathbb{N}} \in \mathbb{Q}^{\mathbb{N}^n} \text{ is computable} \right\} \\ \Pi_n &= \left\{ \inf_{k_1 \in \mathbb{N}} \sup_{k_2 \in \mathbb{N}} \inf_{k_3 \in \mathbb{N}} \dots \beta_{k_1, \dots, k_n} \mid (\beta_{k_1, \dots, k_n})_{k_1, \dots, k_n \in \mathbb{N}} \in \mathbb{Q}^{\mathbb{N}^n} \text{ is computable} \right\} \end{aligned}$$

3 Elementary Constructions on Extender Sets

The free lift

We use this construction to generalize results on \mathbb{Z} or \mathbb{Z}^2 subshifts to higher dimensions:

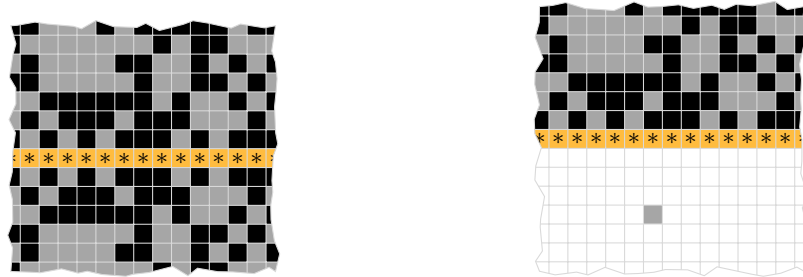
▷ **Claim 8.** For a subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, $h_E(X) = h_E(X^\uparrow)$.

Proof. Consider $X^\uparrow \subseteq \mathcal{A}^{\mathbb{Z}^{d+1}}$. Since each d -dimensional hyperplane of \mathbb{Z}^{d+1} contains an independent configuration, we have $|E_{X^\uparrow}(n)| = |E_X(n)|^n$ and $h_E(X^\uparrow) = h_E(X)$. ◀

The (semi)-mirror construction

▷ Claim 9. Let Y be any \mathbb{Z}^2 sofic subshift over an alphabet \mathcal{A} . There exists a \mathbb{Z}^2 sofic subshift Y_{mirror} such that $h_E(Y_{\text{mirror}}) = h(Y)$ ($= h(Y_{\text{mirror}})$).

A first idea to create one extender set per pattern of Y is the *mirror construction*: add a line of some special symbol $*$ to separate two half-planes; the upper half-plane contains a half-configuration of Y , while the lower half-plane contains its reflection by the line of $*$. As any two patterns of Y have distinct reflections, they generate different extender sets: this results in a subshift Y' verifying $h_E(Y') = h(Y)$. Unfortunately, Y' is not always sofic, see for example [1, Proposition 57].



(a) The (classical) mirror shift. (b) The semi-mirror shift.

■ **Figure 2** Example configurations of the mirror and semi-mirror subshifts.

To solve this non-soficness issue, the *semi-mirror with large discrepancy* from [7, Example 5''] reflects a single symbol instead of the whole upper-plane:

Sketch of proof. For $\mathcal{A}' = \mathcal{A} \cup \{\square, *\}$, define Y_{mirror} over the alphabet \mathcal{A}' as follows:

- Symbols $*$ must be aligned in a row, and there is at most one such row per configuration.
- If a row of $*$ appears in a configuration x , then the lower half-plane contains at most one non- \square position; and the upper half-plane must appear in a configuration of Y .
- If $x_{i,j} = *$ and $x_{i,j-k} \in \mathcal{A}$ for some $i \in \mathbb{Z}, j \in \mathbb{Z}, k \in \mathbb{N}$, then $x_{i,j+k} = x_{i,j-k}$. In other words, the only symbol of \mathcal{A} in the lower half-plane must be the mirror of the same symbol in the upper half-plane, as reflected by the horizontal row of $*$ symbols.

Then Y_{mirror} is sofic and $h_E(Y_{\text{mirror}}) = h(Y)$. Indeed, any two distinct patterns of Y must appear in Y_{mirror} and have distinct extender sets, since they can have different reflections. ◀

This construction shows that there exist subshifts with arbitrarily large extender entropy; and since every Π_1 real number is the topological entropy of some (SFT, thus) sofic subshift [14], every Π_1 number can be realized as the extender entropy of some sofic subshift. In particular, this further disproves the conjecture from [15] mentioned in the introduction.

4 Decision Problems on Extender Sets

4.1 Inclusion of Extender Sets

Let us consider the following decision problem:

EXTENDER-INCLUSION	
Input:	An effective subshift $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$, and $u, v \in \mathcal{L}(X)$,
Output:	Whether $E_X(u) \subseteq E_X(v)$.

► **Proposition 10.** *EXTENDER-INCLUSION is a Π_2^0 -complete problem.*

Proof of inclusion. As $E_X(u) \subseteq E_X(v)$ if and only if $\forall B \in \mathcal{A}^*, u \sqsubseteq B \implies (B \notin \mathcal{L}(X) \vee ((B \setminus u) \sqcup v) \in \mathcal{L}(X))$, we obtain inclusions: indeed, for X effective, deciding whether a pattern w belongs in $\mathcal{L}(X)$ is a Π_1^0 problem. ◀

Proof of Π_2^0 -hardness for \mathbb{Z} subshifts. We reduce the following known Π_2^0 problem³:

DET-REC-STATE	
Input:	A deterministic Turing Machine M , and a state q ,
Output:	Is q visited infinitely often by M during its run on the empty input?

Let (M, q) be an instance of this DET-REC-STATE. We construct an effective subshift X over the alphabet $\{0, 1, \square\}$ as follows:

- Symbols 0 and 1 cannot appear together in a configuration. The symbol 1 can only appear at most once in a configuration.
 - If two symbols 0 appear in a configuration at distance, say, $n > 0$, then the whole configuration is n -periodic; and if M enters q at least n' times, then we impose $n > n'$.
- As the rules above forbid an enumerable set of patterns, X is an effective subshift.

Finally, $E_X(0) \subseteq E_X(1)$ if and only if M enters q infinitely many times. Indeed, the symbol 0 can be extended either by semi-infinite lines of symbols \square , which also extend the symbol 1 ; or by configurations containing n -periodic symbols 0, which do *not* extend the symbol 1 because of the first rule. However, by the second rule, this n -periodic configuration exists if and only if M visits q less than n times. ◀

4.2 Computing the Number of Extender Sets

Let us determine the computational complexity of the problem “ $k \leq |E_X(n)|$ ”, when given a subshift X , some size n and some k . It is equivalent to the following:

$$\bigvee_{v_1, \dots, v_k \in \mathcal{L}_n(X)} \bigwedge_{1 \leq i < j \leq k} E_X(v_i) \neq E_X(v_j).$$

Since $v_i \in \mathcal{L}_n(X)$ is a $\Pi_1^0 \subseteq \Sigma_2^0$ problem and that the class of Σ_2^0 problems is stable by finite disjunctions and conjunctions, we conclude from Proposition 10 that:

► **Lemma 11.** *For an effective subshift X , “ $k \leq |E_X(n)|$ ” is a Σ_2^0 problem.*

4.3 Upper Computational Bounds on Extender Entropies

► **Corollary 12.** *For X an effective subshift, $h_E(X) \in \Pi_3$.*

Proof. Given X and n , the set $\{k \leq |E_X(n)|\}$ is a Σ_2^0 set if X is effective by Lemma 11. This implies that $\frac{\log |E_X(n)|}{n^d}$ is Σ_2 ; and since $h_E(X) = \inf_n \frac{\log |E_X(n)|}{n^d}$, we obtain $h_E(X) \in \Pi_3$ as the infimum of Σ_2 real numbers. ◀

5 Π_3 Extender Entropies for \mathbb{Z} Effective Subshifts

Let us focus on one-dimensional subshifts for the time being.

► **Theorem A.** *The set of extender entropies of \mathbb{Z} effective subshifts is exactly $\Pi_3 \cap [0, +\infty)$.*

³ It is equivalent to INF (does a given machine halt on infinitely many inputs?). See [24, Theorem 4.3.2].

In order to construct a subshift Z_α with $h_E(Z_\alpha) = \alpha$, we would like to have $|E_{Z_\alpha}(n)| \simeq 2^{\alpha n}$. To do so, we could create one extender set per pattern, and $2^{\alpha n}$ patterns of size n (as the semi-mirror in Section 3); however, since effective subshifts have Π_1 entropies, this would not realize the whole class of Π_3 numbers.

Yet, realizing the right number of patterns is the main idea behind the proof that follows: we just do not blindly create one extender set per pattern, but only separate extender sets when some conditions are met.

5.1 Preliminary: Encoding Integers With Configurations $\langle i \rangle_k$

Before we begin our construction, we fix a way to encode integers in configurations: to encode the integer $i \in \mathbb{N}$, we use configurations where a symbol $*$ is i -periodic, and the rest is blank.

More formally, consider the alphabet $\mathcal{A}_* = \{*, \sqcup\}$. Denote by $\langle i \rangle_{k_1}$ the i -periodic configuration $\langle i \rangle_{k_1} = \sigma^{k_1}(\dots \sqcup \underbrace{* \sqcup \dots \sqcup *}_{i+1 \text{ symbols}} \sqcup \dots \sqcup * \sqcup \dots)$ properly defined as $(\langle i \rangle_{k_1})_p = *$ if and only if $p = k_1 \bmod i$. A configuration $\langle i \rangle_{k_1}$ is said to *encode* the integer $i \in \mathbb{N}$. Considering the subshift all the configurations $\langle i \rangle_{k_1}$ for $i \in \mathbb{N}$ and $k_1 \leq i$ generate, we denote:

$$X_* = \bigcup_{i \in \mathbb{N}} \{ \langle i \rangle_{k_1} \in \mathcal{A}_*^{\mathbb{Z}} \mid k_1 \leq i \} \cup \langle \infty \rangle$$

where $\langle \infty \rangle = \{x \in \mathcal{A}_*^{\mathbb{Z}} \mid |x|_* \leq 1\}$ is the set of configurations having at most one symbol $*$. The configurations of $\langle \infty \rangle$ are said to be *degenerate*, and they appear when taking the closure of all $\langle i \rangle_{k_1}$.

5.2 Preliminary: Toeplitz Density in Periodic Configurations

Our construction will also need to build configurations with a controlled density of symbols, *i.e.* configurations on $\{0, 1\}$ where the number of symbols 1 in large patterns converges to some value: for some fixed α , we want to build configurations $x \in \{0, 1\}^{\mathbb{Z}}$ such that $\lim_{n \rightarrow +\infty} \frac{1}{n} \cdot |x|_{\llbracket [0, n-1] \rrbracket} = \alpha$. Several explicit constructions of such configurations and subshifts exist. We choose to work with *Toeplitz sequences*.

Toeplitz density words

Consider the ruler sequence $T = 12131214\dots$ defined by $T_n = \max\{m \in \mathbb{N} : 2^m \mid 2n\}$ (see OEIS A001511). For a given binary sequence $u = (u_n)_{n \in \mathbb{N}} \in \{0, 1\}^{\mathbb{N}}$, we consider its Toeplitzification $T(u) \in \{0, 1\}^{\mathbb{N}}$ defined as $T(u)_n = u_{T_n}$ for $n \in \mathbb{N}$.

In particular, for $\beta \in [0, 1]$ a real number and $(\beta_n)_{n \in \mathbb{N}}$ its proper binary expansion, we consider the word $T(\beta) = (\beta_{T_n})_{n \in \mathbb{N}} = \beta_1 \beta_2 \beta_1 \beta_3 \beta_1 \beta_2 \dots$. Denoting by $|w|_1$ the numbers of letters 1 in a binary word $w \in \{0, 1\}^*$ and by $|w|$ its length, we have:

▷ **Claim 13.** For $\beta \in [0, 1]$ and $w \sqsubseteq T(\beta)$ a factor of $T(\beta)$, we have $|w|_1 = \beta \cdot |w| + O(1)$.

Toeplitz density in periodic configurations

For our specific construction, let $\alpha \in [0, 1]$ and $i \in \mathbb{N}$, and consider the subshift $T_{\leq \alpha, i}$ composed of i -periodic configurations made of truncated Toeplitz words:

$$T_{\leq \alpha, i} = \{x \in \{0, 1\}^{\mathbb{Z}} \mid \exists \beta \leq \alpha, \exists k_1 \in \llbracket [0, i-1] \rrbracket, \forall p \in \mathbb{Z}, x_p = T(\beta)_{(p+k_1 \bmod i)}\}$$

We denote $T(\beta, i)_{k_1} \in \{0, 1\}^{\mathbb{Z}}$ the configuration defined by $(T(\beta, i)_{k_1})_p = T(\beta)_{(p+k_1 \bmod i)}$ for $p \in \mathbb{Z}$. Notice that, for any $\alpha \in [0, 1]$, $i \in \mathbb{N}$ and $n \in \mathbb{N}$, there are $|\mathcal{L}_n(T_{\leq \alpha, i})| = 2^{\log(\min(i, n)) + O(1)} \cdot O(\min(i, n))$ factors of length n in $T_{\leq \alpha, i}$.

▷ **Claim 14.** Let $\alpha \in [0, 1] \cap \Pi_1$. Then $T_{\leq \alpha, i}$ is an SFT, and a family of forbidden patterns realizing $T_{\leq \alpha, i}$ can be computably enumerated from α .

Proof. Consider $\alpha \in \Pi_1$: the set $\{r \in \mathbb{Q} \mid r > \alpha\}$ is computably enumerable. Thus, the following family \mathcal{F} of forbidden patterns that realizes $T_{\leq \alpha, i}$ is recursively enumerable: forbid finite pattern that are either not i -periodic, or do not respect the structure of the ruler sequence in an i -period; and inside an i -period, forbid patterns $r_{T_1} r_{T_2} r_{T_1} \dots \in \{0, 1\}^i$ that encode the finite expansion of a rational $r = \sum_{k=1}^{\log i} r_k 2^{-k}$ if r is such that $r > \alpha$. ◁

5.3 Construction: the Effective \mathbb{Z} Subshift Z_α

Let us now begin the construction to prove Theorem A. Let $\alpha \in \Pi_3$ be a positive real number, $\alpha = \inf_i \sup_j \alpha_{i,j}$ for some computable sequence $(\alpha_{i,j})$ of Π_1 real numbers. We can assume $\alpha \leq 1$ since extender entropy is additive under cartesian products, and using [27, Lemma 3.1] we can assume that $(\alpha_{i,j})_{i,j \in \mathbb{N}^2}$ satisfies some monotonicity properties: for all i , $(\alpha_{i,j})_{j \in \mathbb{N}}$ is weakly increasing towards some α_i ; and the sequence $(\alpha_i)_{i \in \mathbb{N}}$ is weakly decreasing towards α .

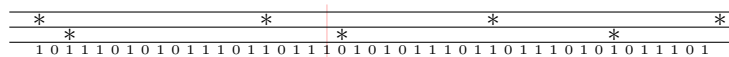
Auxiliary subshift Z'_α

We create an auxiliary subshift Z'_α on the following three layers:

1. *First layer L_1 :* We take $L_1 = X_*$ to encode integers $i \in \mathbb{N}$. Intuitively, i will denote which Σ_2 number α_i is approximated in the configuration.
2. *Second layer L_2 :* We also set $L_2 = X_*$ to encode integers $j \in \mathbb{N}$, $j \geq i$. Intuitively, j will denote which Π_1 number $\alpha_{i,j}$ is approximated in the configuration.
3. *Density layer L_d :* We define the density layer as $L_d = \{0, 1\}^{\mathbb{Z}}$. Whenever the first two layers are non-degenerate, this layer will be restricted to densities $\lesssim \alpha_{i,j}$. Since the real numbers $\alpha_{i,j}$ are Π_1 , the subshifts $T_{\leq \alpha_{i,j}, i}$ are effective from the numbers $\alpha_{i,j}$.

such that Z'_α is defined as:

$$Z'_\alpha = \left\{ (z^{(1)}, z^{(2)}, z^{(d)}) \in L_1 \times L_2 \times L_d \mid z^{(2)} \in \langle \infty \rangle \right\} \\ \cup \bigcup_{i \in \mathbb{N}} \bigcup_{j \geq i} \left\{ (z^{(1)}, z^{(2)}, z^{(d)}) \in L_1 \times L_2 \times L_d \mid \exists k_1, k_2 \in \mathbb{N}, \right. \\ \left. z^{(1)} = \langle i \rangle_{k_1}, z^{(2)} = \langle j \rangle_{k_2} \text{ and } \exists \beta \leq \alpha_{i,j}, z^{(d)} = T(\beta, i)_{k_1} \right\}$$



■ **Figure 3** A proper configuration: L_d contains a Toeplitz encoding of $\overline{.1010}^2 = \frac{5}{8}$. $z = (\langle 15 \rangle_{11}, \langle 18 \rangle_1, T(\frac{5}{8}, 15)_{10})$. The vertical red line indicates the origin.

▷ **Claim 15.** The \mathbb{Z} subshift Z'_α is an effective subshift.

Proof. Since the subshift X_* is effective, the conditions on the first two layers L_1 and L_2 are straightforward to enforce. Furthermore, since the $\alpha_{i,j}$ are Π_1 real numbers enumerated by a single machine, by Claim 14 we can obtain Z'_α as follows: a pattern $w = (w^{(1)}, w^{(2)}, w^{(d)}) \in \mathcal{L}(X_*) \times \mathcal{L}(X_*) \times \{0, 1\}^n$ is forbidden whenever both $w^{(1)}$ and $w^{(2)}$ contain at least two symbols $*$ (so that $w^{(1)}$ encodes an integer $i \in \mathbb{N}$, $w^{(2)}$ encodes an integer $j \geq i$) and $w^{(d)}$ contains a pattern forbidden in $T_{\leq \alpha_{i,j}, i}$. ◁

21:10 Computability of Extender Sets in Multidimensional Subshifts

A configuration $z = (\langle i \rangle_{k_1}, \langle j \rangle_{k_2}, T(\beta, i)_{k_1}) \in Z'_\alpha$ is said to be *proper*. A configuration $z = (z^{(1)}, z^{(2)}, \cdot) \in Z'_\alpha$ with $z^{(2)} \in \langle \infty \rangle$ is said to be *degenerate*. Thus, we separate patterns into two categories: whenever $w \in \mathcal{L}(Z'_\alpha)$ only appears in degenerate configurations, we call it a *degenerate pattern*; if w can appear in a proper configuration, we call it a *proper pattern*.

On the one hand, degenerate patterns of Z'_α do not contribute much to the number of extender sets, despite being exponentially many:

▷ **Claim 16.** Let $n \in \mathbb{N}$, and consider $D_E(n) = \{E_{Z'_\alpha}(w) \mid w \in \mathcal{L}_n(Z'_\alpha) \text{ degenerate}\}$, the set of extender sets of degenerate patterns of size n . Then $|D_E(n)| = O(n^3)$.

Proof. Let $u, v \in \mathcal{L}_n(Z'_\alpha)$ be two degenerate patterns. Whenever $u^{(1)} = v^{(1)}$ and $u^{(2)} = v^{(2)}$, we have $E_{Z'_\alpha}(u) = E_{Z'_\alpha}(v)$ because the density layer of such patterns can be anything. Since at most a single symbol $*$ can appear on the second layer of degenerate patterns, by counting possibilities for their first layers we obtain $|D_E(n)| = O(n^3)$. ◁

On the other hand, all proper patterns of Z'_α belong to distinct extender sets:

▷ **Claim 17.** Let $u, v \in \mathcal{L}_n(Z'_\alpha)$ be two distinct proper patterns. Then $E_{Z'_\alpha}(u) \neq E_{Z'_\alpha}(v)$.

Proof. Let $u \in \mathcal{L}_n(Z'_\alpha)$ be a proper pattern. It can be extended into a whole proper configuration $z = (\langle i \rangle_{k_1}, \langle j \rangle_{k_2}, z^{(d)}) \in Z'_\alpha$ such that $z|_{\llbracket 0, n-1 \rrbracket} = u$. By definition, z is periodic of period $i \cdot j$: thus, $z|_{\llbracket 0, n-1 \rrbracket}$ is entirely determined by $z|_{\llbracket n, i \cdot j + n - 1 \rrbracket}$, and $z|_{\mathbb{Z} \setminus \llbracket 0, n-1 \rrbracket}$ can only extend the pattern u itself. ◁

However, there are only polynomially many distinct proper patterns of a given size in Z'_α . The next section will nevertheless create a subshift Z_α with the correct (exponential) amount of proper patterns, thanks to the following remark:

▷ **Claim 18.**

- For an integer $i \in \mathbb{N}$ and a proper configuration $z \in Z'_\alpha$ such that $z^{(1)} = \langle i \rangle_{k_1}$, an i -period of the density layer $z^{(d)}$ contains at most $\alpha_i \cdot i + O(1)$ symbols 1.
- For integers $n \in \mathbb{N}$ and $i \geq n$, and a proper configuration $z \in Z'_\alpha$ such that $z^{(1)} = \langle i \rangle_{k_1}$, a factor of length n of the density layer $z^{(d)}$ contains at most $\alpha_n \cdot n + O(1)$ symbols 1.

Proof. This follows from Claim 13 and the monotonicity of the sequence $(\alpha_{i,j})_{i,j \in \mathbb{N}^2}$. ◁

Free bits in the subshift Z_α

To create the desired exponential number of extender sets, we create the subshift Z_α by adding *free bits* on top of the symbols 1 of the density layer. Informally, if there were $\beta \cdot i + O(1)$ symbols 1 in an i -period of the density layer in Z'_α , adding free bits on top of the symbols 1 creates $2^{\beta \cdot i + O(1)}$ patterns in Z_α . Thus, we add a fourth layer to Z'_α :

4. *Free layer L_f* : We define the free layer as $L_f = \{\sqcup, 0, 1\}^{\mathbb{Z}}$. Given the synchronizing map $\pi_{\text{sync}}: \{\sqcup, 0, 1\} \rightarrow \{0, 1\}$ defined as $\pi_{\text{sync}}(0) = \pi_{\text{sync}}(1) = 1$ and $\pi_{\text{sync}}(\sqcup) = 0$, we say that two configurations $z^{(d)} \in L_d$ and $z^{(f)} \in L_f$ are *synchronized* if $\pi_{\text{sync}}(z^{(f)}) = z^{(d)}$.

and we define Z_α as:

$$Z_\alpha = \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(f)}) \in L_1 \times L_2 \times L_d \times L_f \mid z^{(1)} \in \langle \infty \rangle \text{ or } z^{(2)} \in \langle \infty \rangle \right\} \\ \cup \bigcup_{i \in \mathbb{N}} \bigcup_{j \geq i} \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(f)}) \in L_1 \times L_2 \times L_d \times L_f \mid \exists k_1, k_2 \in \mathbb{N}, \right. \\ \left. z^{(1)} = \langle i \rangle_{k_1}, z^{(2)} = \langle j \rangle_{k_2}, \pi_{\text{sync}}(z^{(f)}) = z^{(d)}, \right. \\ \left. \exists \beta \leq \alpha_{i,j}, z^{(d)} = T(\beta, i)_{k_1} \text{ and } z^{(f)} \text{ is } i\text{-periodic} \right\}.$$

▷ Claim 19. The \mathbb{Z} subshift Z_α is effective.

Proof. In addition to the forbidden patterns of Z'_α , forbid patterns $w = (w^{(1)}, w^{(2)}, w^{(d)}, w^{(f)})$ for which $w^{(1)}$ and $w^{(2)}$ both contain two symbols $*$ (in which case, denote by i the distance between two symbols $*$ in $w^{(1)}$), but $w^{(f)}$ is either not synchronized with $w^{(d)}$ or not i -periodic. ◁

We extend the terminology from Z'_α to Z_α and call *proper* the configurations of Z_α that encode integers $i \in \mathbb{N}$ and $j \geq i$ on their first two layers, and *degenerate* those who do not. Similarly, a pattern is *proper* if it can be extended into a proper configuration, and *degenerate* if it only extends into degenerate configurations.

Since the free layer is required to be i -periodic only in proper configurations, Claims 16 and 17 both extend from Z'_α to Z_α by the very same arguments:

▷ Claim 20.

- For $n \in \mathbb{N}$, consider $D_E(n) = \{E_{Z_\alpha}(w) \mid w \in \mathcal{L}_n(Z_\alpha) \text{ degenerate}\}$. Then $D_E(n) = O(n^2)$.
- Let $u, v \in \mathcal{L}_n(Z_\alpha)$ be two distinct proper patterns. Then $E_{Z_\alpha}(u) \neq E_{Z_\alpha}(v)$.

► **Lemma 21.** Let $P(n) = \{w \in \mathcal{L}_n(Z_\alpha) \mid w \text{ is proper}\}$. Then

$$2^{n \cdot \alpha_n + O(1)} \leq P(n) \leq \text{poly}(n) \cdot \sum_{i=1}^n 2^{\alpha_i \cdot i + O(1)}.$$

Proof: lower bound. Consider the patterns $w' = (\langle n \rangle_0, \langle j \rangle_0, T(\alpha_{n,j}, n)_0) \upharpoonright_{[0, n-1]}$ in Z'_α for $j \geq n$: the number of symbols 1 in the density layer $w'^{(d)}$ of such w' is $\alpha_{n,j} \cdot n + O(1)$ by Claim 13. Since $\alpha_{n,j} \rightarrow \alpha_n$, by taking $j \geq n$ large enough we obtain a proper pattern $w' \in \mathcal{L}_n(Z'_\alpha)$ such that its density layer $w'^{(d)}$ contains $\alpha_n \cdot n + O(1)$ symbols 1.

Thus, we obtain $2^{\alpha_n \cdot n + O(1)}$ proper patterns $w \in \mathcal{L}_n(Z_\alpha)$ such that $\pi_{L_1 \times L_2 \times L_d}(w) = w'$ (since each symbol 1 in $w^{(d)}$ leads to two distinct patterns in the free layer L_f). ◀

Proof: upper bound. To overestimate the number of proper patterns $|P(n)|$, we consider the restrictions $w' = z' \upharpoonright_{[0, n-1]}$ for z' ranging in the proper configurations of Z'_α (consider all values of $\langle i \rangle_{k_1}, \langle j \rangle_{k_2}$ and of n -factors in $y^{(d)}$), and bound the number of symbols 1 in each case: by Claim 18,

- If $i \leq n$, an i -period of the density layer $w'^{(d)}$ contains less than $\alpha_i \cdot i + O(1)$ symbols 1.
- For $i > n$, $w'^{(d)}$ contains less than $\alpha_n \cdot n + O(1)$ symbols 1.

Since each symbol 1 in an i -period of the density layer results in two distinct patterns in the free layer, and there are less than $O(i^2)$ possibilities for such periods, we obtain:

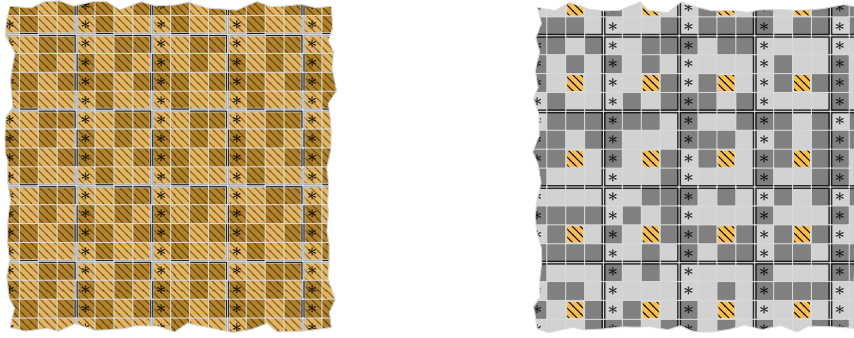
$$\begin{aligned} P(n) &\leq \sum_{i=1}^n \sum_{k_1=0}^{i-1} \sum_{j=1}^n \sum_{k_2=0}^{j-1} O(i^2) \cdot 2^{\alpha_i \cdot i + O(1)} + \sum_{k_1=0}^n \sum_{k_2=0}^n O(n^2) \cdot 2^{\alpha_n \cdot n + O(1)} \\ &\leq \text{poly}(n) \cdot \sum_{i=1}^n 2^{\alpha_i \cdot i + O(1)}. \end{aligned} \quad \blacktriangleleft$$

Combining Lemma 21 with Claim 20, we obtain by taking the limit over $\alpha_n \rightarrow \alpha$ that $h_E(Z_\alpha) = \alpha$, which concludes the proof.

6 Π_3 Extender Entropies for \mathbb{Z}^2 Sofic Subshifts

We now want to extend Theorem A to multidimensional sofic shifts: an idea could be to replace i -periodic words on \mathbb{Z} in the previous construction with (i, i) -periodic squares on \mathbb{Z}^2 . Unfortunately, such a subshift cannot be sofic⁴.

Yet, making configurations *periodic* is not necessary to ensure that two proper patterns u and v have distinct extender sets: it is enough to have a configuration that *witnesses* the difference between u and v (by extending one but not the other). This was already illustrated in the semi-mirror shift (see Section 3): instead of mirroring the whole half-plane (which is not sofic), non-deterministically reflecting a single bit from the upper to the lower half-plane is actually enough, since each bit can be reflected individually in some configuration. In this section, we use this idea to prove (see Figure 4):



(a) Whole (i, i) -periodic squares of free bits. (b) A single (i, i) -periodic free bit.

■ **Figure 4** The periodized area is highlighted in color ■ and hatched. To make the figure readable, symbols for free bits are $\{\square, \blacksquare\}$ instead of $\{b, b'\}$.

► **Theorem B.** *The set of extender entropies of \mathbb{Z}^2 sofic subshifts is exactly $\Pi_3 \cap [0, +\infty)$.*

6.1 Preliminary: Marking Offsets With Configurations $[2i]_{m_1, m_2}$

In our construction, we will need to mark some positions $(m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$. To do so, we consider the alphabet $A_m = \{\square, \blacksquare\}$. Denote by $[2i]_{m_1, m_2}$ the $(2i, 2i)$ -periodic configuration formally defined as $([2i]_{m_1, m_2})_p = \blacksquare$ if and only if $p = (m_1, m_2) \bmod (2i, 2i)$. We say that a symbol \blacksquare is a *marker*.

For a configuration $x = [2i]_{m_1, m_2}$ with $(m_1, m_2) \in \llbracket 0, 2i - 1 \rrbracket^2$, we say that a position $p \in \mathbb{Z}^2$ is *marked* if $p \in (m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$. This lattice has unit cells of size $i \times i$ instead of $2i \times 2i$: this is voluntary. In particular, some marked positions $p \in \mathbb{Z}^2$ satisfy $x_p = \square$.

Considering the subshift generated by all the configurations $[i]_{m_1, m_2}$, we define:

$$\mathcal{G} = \bigcup_{i \in \mathbb{N}} \{[i]_{m_1, m_2} \mid (m_1, m_2) \in \llbracket 0, i - 1 \rrbracket^2\} \cup [\infty]$$

where $[\infty] = \{x \in A_m^{\mathbb{Z}^2} \mid |x|_{\blacksquare} \leq 1\}$ is the set of configurations having at most one marker symbol \blacksquare : these are the configurations that appear when taking the closure of all $[i]_{m_1, m_2}$.

⁴ The argument proving that the classical mirror subshift cannot be sofic still applies here: there would be $2^{O(i^2)}$ distinct $i \times i$ patterns, but only $2^{O(i)}$ borders in the SFT cover.

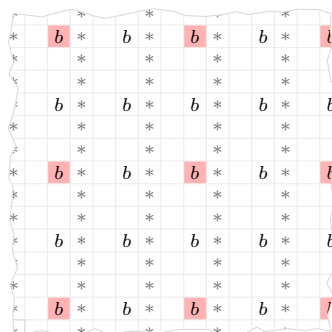
6.2 Construction: the Sofic \mathbb{Z}^2 Subshift Y_α

Let us now begin a construction to prove Theorem B. We use the notations introduced in the proof of Theorem A: we fix $\alpha \in [0, 1] \cap \Pi_3$ such that $\alpha = \inf_i \sup_j \alpha_{i,j}$ for $\alpha_{i,j}$ a computable sequence of Π_1 real numbers (we assume the same monotonicity properties). We define a subshift Y_α on the following five layers:

- *Lifted layers:* We define the first three layers of Y_α as $L_1^\uparrow \times L_2^\uparrow \times L_d^\uparrow$, where L_1, L_2 and L_d are the three layers of the subshift Z'_α defined in the proof of Theorem A.
- *Marker layer L_m :* We define $L_m = \mathcal{G}$ to mark positions $p \in (m_1 + i\mathbb{Z}, m_2 + i\mathbb{Z})$.
- *Free layer L_f :* We also define the free layer by $L_f = \{\sqcup, 0, 1\}^{\mathbb{Z}^2}$.

and we define Y_α as (see Figure 5 for an illustration):

$$Y_\alpha = \left\{ (y^{(1)\uparrow}, y^{(2)\uparrow}, y^{(d)\uparrow}, y^{(m)}, y^{(f)}) \in L_1^\uparrow \times \langle \infty \rangle^\uparrow \times L_d^\uparrow \times L_m \times L_f \mid \right. \\ \left. \forall i \in \mathbb{N}, (\exists k_1 \in \mathbb{N}, y^{(1)} = \langle i \rangle_{k_1} \iff \exists m_1, m_2 \in \mathbb{N}, y^{(m)} = [2i]_{m_1, m_2}) \right\} \\ \cup \bigcup_{i \in \mathbb{N}} \bigcup_{j \geq i} \left\{ (y^{(1)\uparrow}, y^{(2)\uparrow}, y^{(d)\uparrow}, y^{(m)}, y^{(f)}) \in L_1^\uparrow \times L_2^\uparrow \times L_d^\uparrow \times L_m \times L_f \mid \exists k_1, m_1, m_2, k_2 \in \mathbb{N}, \right. \\ \left. y^{(1)} = \langle i \rangle_{k_1}, y^{(m)} = [2i]_{m_1, m_2}, y^{(2)} = \langle j \rangle_{k_2}, \pi_{\text{sync}}(y^{(f)}) = y^{(d)\uparrow}, \right. \\ \left. \exists \beta \leq \alpha_{i,j}, y^{(d)} = T(\beta, i)_{k_1} \text{ and } y^{(f)}|_{(m_1+i\mathbb{Z}) \times (m_2+i\mathbb{Z})} \text{ is constant} \right\}.$$



■ **Figure 5** Projection of a proper configuration on $L_1^\uparrow \times L_m \times L_f$. The symbols $*$ are on L_1^\uparrow , the symbols \blacksquare on L_m the symbols 1 on L_f . All the other bits of L_f (not drawn here) are free.

Extending the terminology from Z_α to Y_α , we call *proper* the configurations of Y_α that encode integers $i \in \mathbb{N}$ and $j \geq i$ on their first two layers, and *degenerate* those which do not. Additionally we say that a pattern is *proper* if it can be extended into a proper configuration, and *degenerate* otherwise. We say that two proper patterns $u, v \in \mathcal{L}_n(Y_\alpha)$ are *similar* if they are equal on their first four layers (i.e. $\pi_{L_1^\uparrow \times L_2^\uparrow \times L_d^\uparrow \times L_m}(u) = \pi_{L_1^\uparrow \times L_2^\uparrow \times L_d^\uparrow \times L_m}(v)$).

▷ **Claim 22.** Two similar proper patterns $u, v \in \mathcal{L}_n(Y_\alpha)$ have distinct extender sets if and only if there exists a proper configuration y that extends u and that *marks* a position $p \in \llbracket 0, n-1 \rrbracket^2$ such that $u_p^{(f)} \neq v_p^{(f)}$.

We would very much like an analog of Claim 20: unfortunately, not all proper patterns generate distinct extender sets. Indeed, by the previous claim, similar proper patterns generate distinct extender sets only when the positions at which they differ can be *marked* by an extending configuration (this depends on the relative position of an $n \times n$ window covering the four quadrants of a $2i \times 2i$ square, etc...). Yet, we do not need precise considerations to count the number of extender sets, and simply prove the following bounds:

► **Lemma 23.** *Let $P_E(n) = \{E_{Y_\alpha}(w) \in \mathcal{L}_n(Y_\alpha) \mid w \text{ is proper}\}$. Then*

$$2^{\alpha_n \cdot n^2 + O(n)} \leq P_E(n) \leq \text{poly}(n) \cdot \sum_{i=0}^n 2^{\alpha_i \cdot i^2 + O(i)}.$$

Proof: lower bound. For $j \geq n$, consider the set $J_j = \{(y \in Y_\alpha \mid y^{(1)} = \langle n \rangle_0, y^{(2)} = \langle j \rangle_0, y^{(d)} = T(\alpha_{n,j}, n))\}$. The number of symbols 1 in an $(n \times n)$ -period in the density layer of such configurations is $\alpha_{n,j} \cdot n^2 + O(n)$ by Claim 13. Since $\alpha_{n,j} \rightarrow \alpha_n$, by taking $j \geq n$ large enough we obtain a set $J = J_j$ of proper configurations y whose density layer $y^{(d)}$ contains $\alpha_n \cdot n^2 + O(n)$ symbols 1 in an $(n \times n)$ -period.

Considering the free layer of such patterns, there are at least $2^{\alpha_n \cdot n^2 + O(n)}$ distinct patterns in the finite set $W = \{y|_{\llbracket 0, n-1 \rrbracket^2} \mid y \in J_j \text{ and } y^{(m)}|_{\llbracket 0, n-1 \rrbracket^2} = \square^{\llbracket 0, n-1 \rrbracket^2}\}$, and we claim that they all generate distinct extender sets. Indeed, for any two distinct patterns $u, v \in W$, there exists a position $p \in \llbracket 0, n-1 \rrbracket^2$ such that $u_p^{(f)} \neq v_p^{(f)}$; and there exists a configuration $y \in J_j$ that extends u with $y^{(m)} = [2n]_{p+(n,n)}$: in particular, y marks the position p .⁵ By Claim 22, we obtain $E_{Y_\alpha}(u) \neq E_{Y_\alpha}(v)$. This proves that $P_E(n) \geq 2^{\alpha_n \cdot n^2 + O(n)}$. ◀

Proof: upper bound. We proceed as with the \mathbb{Z} effective subshift Z_α : to bound the cardinality of $P_E(n)$, we consider the restrictions $w = y|_{\llbracket 0, n-1 \rrbracket^2}$ for y ranging in the proper configurations of Y_α (for all values of $\langle i \rangle_{k_1}$, $\langle j \rangle_{k_2}$, $T(\beta, i)$ and $[2i]_{m_1, m_2}$), and count free layers by Claim 18:

- If $i \leq n$, an $i \times i$ square of the density layer $w^{(d)}$ contains less than $\alpha_i \cdot i^2 + O(i)$ symbols 1.
- If $i > n$, the density layer $w^{(d)}$ contains less than $\alpha_n \cdot n^2 + O(n)$ symbols 1.

Finally, when summing over all these cases, we overestimate the number of extender sets generated by the free layer by assuming that each position $p \in \llbracket 0, i-1 \rrbracket^2$ containing a symbol 1 on the density layer can be marked by a proper configuration y extending the pattern (while only a subset of such positions can be marked):

$$\begin{aligned} P_E(n) &\leq \sum_{i=1}^n \sum_{k_1=0}^{i-1} \sum_{j=1}^n \sum_{k_2=0}^{j-1} O(i^4) \cdot 2^{\alpha_i \cdot i^2 + O(i)} + \sum_{k_1=0}^n \sum_{k_2=0}^n O(n^4) \cdot 2^{\alpha_n \cdot n^2 + O(n)} \\ &\leq \text{poly}(n) \cdot \sum_{i=1}^n 2^{\alpha_i \cdot i^2 + O(i)}. \end{aligned} \quad \blacktriangleleft$$

By taking the limit $\alpha = \lim_n \alpha_n$, we obtain that $h_E(Y_\alpha) = \alpha$. Thus, we are left to prove:

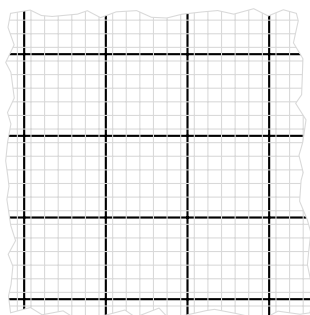
► **Claim 24.** The subshift Y_α is a sofic subshift.

This proof is very standard and unsurprising, yet is included for the sake of exhaustiveness.

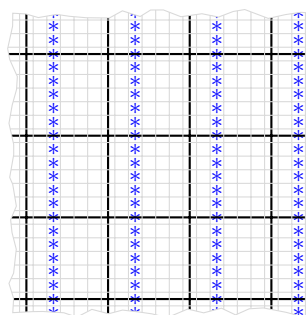
Sketch of proof. First, we introduce a grid subshift. Let us denote by Y_{grid} the subshift on the alphabet $\{+, -, | \}$ defined as the closure of all the square grid configurations (see Figure 6a). It is a sofic subshift: by enforcing the continuity of black lines between adjacent positions, we obtain an irregular grid; to obtain a regular square grid, we make each cross $+$ send diagonals in the SFT cover (since diagonals can only go through a cross, the grid becomes regular).

Let us now synchronize Y_{grid} with L_1^\uparrow : we define $Y_{\text{grid}^*} \subseteq L_1^\uparrow \times Y_{\text{grid}}$ the set of configurations $(x^{(1)\uparrow}, x^{(g)})$ such that $x^{(g)}$ has mesh $i \times i$ if and only if $x^{(1)}$ encodes some $i \in \mathbb{N}$ (see Figure 6b).

⁵ Markers were chosen to be $(2i, 2i)$ -periodic for this reason: we need to be able to mark a position $p \in \llbracket 0, i-1 \rrbracket^2$ in a configuration without seeing a marker in the square $\llbracket 0, i-1 \rrbracket^2$.



(a) A square grid configuration of mesh $i \times i$.



(b) Vertical blue columns of symbols $*$ are i -periodic, the square grid has mesh $i \times i$.

■ **Figure 6** Two configurations using grids.

▷ **Claim 25.** $Y_{\text{grid}*}$ is a \mathbb{Z}^2 sofic subshift.

Sketch of proof. Using areas of colors in the SFT cover, ensure that exactly one black vertical line in Y_{grid} can appear between two vertical lines of symbols $*$ in L_1^\uparrow . ◁

Let us now prove that Y_α is a \mathbb{Z}^2 sofic subshift. Intuitively, it follows from Theorem 2: Y_α is a “decorated version” of Z_α^\uparrow . The most tricky step is in the periodicity condition: periodicity of a free bit in $L^{(f)}$ should only be enforced whenever both layers $y^{(1)}$ and $y^{(2)}$ do not belong in $\langle \infty \rangle^\uparrow$, *i.e.* whenever they both actually encode some integers $i \in \mathbb{N}$ and $j \in \mathbb{N}$.

To proceed, we slightly alter the \mathbb{Z} subshift Z'_α to define a new subshift Z''_α : it contains an additional layer L_p (the *proper* layer) that can take two values (either $\mathfrak{p}^\mathbb{Z}$ or $\mathfrak{d}^\mathbb{Z}$), and is forced to be $\mathfrak{p}^\mathbb{Z}$ whenever both the first and second layer do encode integers:

$$Z''_\alpha = \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(p)}) \in L_1 \times L_2 \times L_d \times \{\mathfrak{p}^\mathbb{Z}, \mathfrak{d}^\mathbb{Z}\} \mid z^{(2)} \in \langle \infty \rangle^\uparrow \right\} \\ \cup \bigcup_{i \in \mathbb{N}} \bigcup_{j \geq i} \left\{ (z^{(1)}, z^{(2)}, z^{(d)}, z^{(p)}) \in L_1 \times L_2 \times L_d \times \{\mathfrak{p}^\mathbb{Z}\} \mid \exists k_1, k_2 \in \mathbb{N}, \right. \\ \left. z^{(1)} = \langle i \rangle_{k_1}, z^{(2)} = \langle j \rangle_{k_2} \text{ and } \exists \beta \leq \alpha_{i,j}, z^{(d)} = T(\beta, i)_{k_1} \right\}.$$

By a slight alteration of Claim 15, the subshift Z''_α is effective whenever α is a Π_3 real number. By Theorem 2, the \mathbb{Z}^2 subshift Z''_α^\uparrow is thus sofic. Then, we use the proper layer to enforce periodicity of a free bit in $L^{(f)}$ only whenever $y^{(p)} = \mathfrak{p}^{\mathbb{Z}^2}$, and define Y'_α as:

$$Y'_\alpha = \left\{ (y^{(1)\uparrow}, y^{(2)\uparrow}, y^{(d)\uparrow}, y^{(p)\uparrow}, y^{(g)}, y^{(f)}) \in Z''_\alpha^\uparrow \times Y_{\text{grid}} \times \{\sqcup, 0, 1\}^{\mathbb{Z}^2} \mid \right. \\ \left. (y^{(1)\uparrow}, y^{(g)}) \in Y_{\text{grid}*}, \quad \pi_{\text{sync}}(y^{(f)}) = y^{(d)\uparrow}, \right. \\ \left. \exists b \in \{\sqcup, 0, 1\}, \forall p \in \mathbb{Z}^2, y^{(p)} = \mathfrak{p}^\mathbb{Z} \wedge y_p^{(g)} = \vdash \implies y_p^{(f)} = b \right\}$$

▷ **Claim 26.** Y'_α is a \mathbb{Z}^2 sofic subshift.

Sketch of proof. By the previous paragraph, the first four layers are sofic; and by Claim 25, the synchronization $Y_{\text{grid}*}$ of L_1^\uparrow and Y_{grid} is sofic. To make a free bit periodic, one can carry a unique symbol $b_{\text{grid}} \in \{\sqcup, 0, 1\}$ along the black lines of Y_{grid} in an SFT cover, and enforce the following: on positions at which a cross symbol \vdash appears on the grid layer $y^{(g)}$, and a symbol \mathfrak{p} appears on the proper layer $y^{(p)\uparrow}$, the free bit in $y^{(f)}$ is then made equal to the symbol b_{grid} . ◁

21:16 Computability of Extender Sets in Multidimensional Subshifts

We can now prove that Y_α is sofic. Indeed, fix an SFT cover of Y'_α in which we color cross symbols \dagger into two colors alternatingly: let us say, red and blue. On each horizontal and vertical line of the grid layer Y_{grid} , crosses are now alternating between red and blue. We claim that we obtain the subshift Y_α by projecting this SFT cover as follows:

- Erase the proper layer.
 - Projection of the grid layer: red crosses become \blacksquare , and all other symbols become \square .
- Indeed, projecting the grid layer as mentioned creates the marker layer L_m . The only condition that remains to be checked is the (i, i) -periodicity condition on a free bit.

Notice that in Y'_α , both cases $y^{(p)\dagger} = \mathbf{p}^{\mathbb{Z}^2}$ and $y^{(p)\dagger} = \mathbf{d}^{\mathbb{Z}^2}$ are possible whenever $y^{(1)} \in \langle \infty \rangle$ or $y^{(2)} \in \langle \infty \rangle$, so that erasing the proper layer in the projection merges the two cases together and removes the periodicity enforced a free bit of $y^{(f)}$; while whenever $y^{(1)} = \langle i \rangle_{k_1}$ and $y^{(2)} = \langle j \rangle_{k_2}$, only the case $y^{(p)} = \mathbf{p}^{\mathbb{Z}^2}$ is allowed: so that, when projecting, the periodicity condition is still enforced. \triangleleft

7 Realizing Extender Entropies: Computable Subshifts

A subshift X is said to be *computable* if its language $\mathcal{L}(X)$ is decidable. Following the proofs from Section 4, one proves that extender entropies of computable subshifts are Π_2 real numbers. We prove the converse inclusion and obtain:

► **Theorem 27.** *The set of extender entropies of computable \mathbb{Z} effective subshifts (resp. computable \mathbb{Z}^2 sofic subshifts) is exactly $\Pi_2 \cap [0, +\infty)$.*

Sketch of proof. We slightly alter our previous constructions. The subshift Z'_α constructed in Theorem A might not be computable whenever $\alpha \in \Pi_3$, since, given some $i, j \in \mathbb{N}$ and some factor of $T(\beta, i)$, it might be undecidable to know whether $\beta \leq \alpha_{i,j}$ when $\alpha_{i,j} \in \Pi_1$.

Yet, when taking $\alpha = \inf_i \alpha_i = \inf_i \sup_j \alpha_{i,j} \in \Pi_2$ for $(\alpha_{i,j})$ a computable sequence, the previous problem becomes decidable; thus, the subshift Z'_α is computable. Both proofs, on \mathbb{Z} and \mathbb{Z}^2 , then go through without any other modification. \blacktriangleleft

8 Extender Sets of Minimal Subshifts

Minimality is a general dynamical notion; in our context, a subshift is *minimal* if it contains no nonempty proper subshift. Extender sets are much easier in minimal subshifts and do not even depend on the computability of the language:

► **Proposition 28.** *Let X be a minimal subshift over \mathbb{Z}^d . Then for any $n > 0$ and any patterns $u, v \in \mathcal{L}_n(X)$, $E_X(u) \subseteq E_X(v) \iff u = v$.*

Proof. Let $u, v \in \mathcal{L}_n(X)$ and suppose that $E_X(u) \subseteq E_X(v)$. Then any appearance of u in a configuration can be replaced by v : by iterating the process while ordering patterns lexicographically (see [23, Lemma 2.2] for the complete argument), we obtain by compactness a configuration of X in which u does not appear, which contradicts minimality. \blacktriangleleft

This implies that $h_E(X) = h(X)$ if X is minimal. Since minimal sofic subshifts have zero entropy (folklore, see [11, Proposition 6.1]), and minimal effective subshifts have arbitrary Π_1 entropy (consider [16, Theorem 4.77] with computable sequences $(k_n)_{n \in \mathbb{N}}$), we obtain:

- **Corollary 29.** *The extender entropy of a minimal \mathbb{Z}^d sofic subshift is always 0.*
- **Corollary 30.** *The set of extender entropies of minimal \mathbb{Z}^d effective subshifts is exactly $\Pi_1 \cap [0, +\infty)$.*

9 Extender Sets of Subshifts With Mixing Properties

9.1 Mixing \mathbb{Z} Subshifts

Mixingness is another dynamical notion. In the context of \mathbb{Z} subshifts, mixingness intuitively implies that for any pair of admissible words, there exists a configuration containing both of them at arbitrary positions, provided they are sufficiently far apart:

► **Definition 31** (Mixing subshift). *A \mathbb{Z} subshift X is mixing if*

$$\forall n > 0, \exists N > 0, \forall u, v \in \mathcal{L}_n(X), \forall k \geq N, \exists w \in \mathcal{L}_k(X), uwv \in \mathcal{L}(X).$$

We say that X is $f(n)$ -mixing for some function f if N can be taken equal to $f(n)$ in the previous definition. When f is constant $f(n) = N$, we simply write that X is N -mixing.

One could expect that strong mixing conditions would restrict the behaviors of extender sets: indeed, all the examples we mentioned so far either have strong mixing properties (the full shift, \mathbb{Z} SFTs. . .) and zero extender entropy, or have positive extender entropy but are far from mixing (periodicity, reflected positions, . . .). However, we show in this section that even very restrictive mixing properties do not imply anything on extender entropies.

► **Proposition 32.** *Let X be a one-dimensional subshift. There exists a 1-mixing subshift $X_\#$ with $h_E(X) = h_E(X_\#)$. (Furthermore, if X was effective, then $X_\#$ can be taken effective.)*

Proof. Let $X \subseteq \mathcal{A}^{\mathbb{Z}}$ be a subshift, and $\alpha = h_E(X)$. Denote $\mathcal{F} = \mathcal{A}^* \setminus \mathcal{L}(X)$. Let us define a subshift $X_\#$ over the alphabet $\mathcal{A} \sqcup \{\#\}$ (assuming that $\#$ is a free symbol not in \mathcal{A}) by the same family of forbidden patterns \mathcal{F} : configurations of $X_\#$ are composed of (possibly infinite) words of $\mathcal{L}(X)$ separated by the *safe symbol* $\#$. Then $X_\#$ is 1-mixing, as for any $u, v \in \mathcal{L}(Y)$, we have $u\#v \in \mathcal{L}(Y)$.

We are left with proving that $h_E(X_\#) = h_E(X)$. First, we need to introduce the notion of *follower* and *predecessor sets*: in X , the *follower* and *predecessor sets* are respectively defined as $F_X(w) = \{x \in \mathcal{A}^{\mathbb{N}} \mid wx \sqsubseteq X\}$ and $P_X(w) = \{x \in \mathcal{A}^{-\mathbb{N}} \mid xw \sqsubseteq X\}$. In other words, the follower set (resp. predecessor set) of some word w correspond to the set of right-infinite (resp. left-infinite) sequences x such that ux (resp. xw) appears in some configuration of X .

Let $n \geq 0$. We prove that:

$$|E_X(n)| \leq |E_{X_\#}(n)| \leq |E_X(n)| + \sum_{i+j < n} |P_X(i)||F_X(j)|$$

Lower bound. The lower bound holds simply because if x extends a pattern $w \in \mathcal{L}(X)$ but not $w' \in \mathcal{L}(X)$, then x also belongs in $X_\#$ and still extends w but not w' in $X_\#$, so that $E_{X_\#}(w) \neq E_{X_\#}(w')$. \triangleleft

Upper bound. For the rightmost inequality, we need to distinguish some cases according to whether a pattern contains a $\#$ or not.

■ Let $w \in \mathcal{L}_n(X_\#)$ that does not contain a symbol $\#$. Then

$$E_{X_\#}(w) = E_X(w) \cup \bigcup_{l, r \in \mathcal{A}^* \mid lwr \in \mathcal{L}(X)} \{(x\#l, r\#x') \mid x, x' \text{ admissible in } X_\#\}$$

So, for $w, w' \in \mathcal{L}_n(X_\#)$. So, for $w, w' \in \mathcal{L}_n(X_\#)$ that do not contain a symbol $\#$, we have $E_{X_\#}(w) = E_{X_\#}(w')$ if and only if $E_X(w) = E_X(w')$.

- Let $w \in \mathcal{L}_n(X_\#)$ containing at least a symbol $\#$, and let $i \leq j$ be the first and last positions in w at which a symbol $\#$ appear. Let $l, r \in \mathcal{A}^*$ be respectively $w_{\llbracket 0, i-1 \rrbracket}$ and $w_{\llbracket j+1, n-1 \rrbracket}$. Since $\#$ is a safe symbol, $E_{X_\#}(w)$ is entirely determined by $(P_X(l), F_X(r))$:

$$E_{X_\#}(w) = (P_X(l) \times F_X(r)) \cup \bigcup_{\substack{l', r' \in \mathcal{A}^* \\ l' \cdot l, r \cdot r' \in \mathcal{L}(X)}} \{(y \# l', r' \# y') \mid y, y' \text{ admissible in } X_\#\}.$$

Doing a disjunction on these two cases, and over the pairs $i + j < n$ in the second case (and abusing notations again by denoting $P_X(i) = \{P_X(w) \mid w \in \mathcal{L}_i(X)\}$ and $F_X(j) = \{F_X(w) \mid w \in \mathcal{L}_j(X)\}$) we obtain:

$$|E_{X_\#}(n)| \leq |E_X(n)| + \sum_{i+j < n} |P_X(i)| |F_X(j)| \quad \triangleleft$$

As $|P_X(n)| \leq |E_X(n)|$ and $|F_X(n)| \leq |E_X(n)|$, and that $|E_X(n)| = 2^{\alpha n + o(n)}$, we obtain $2^{\alpha n + o(n)} \leq |E_{X_\#}(n)| \leq \text{poly}(n) \cdot 2^{\alpha n + o(n)}$, and conclude that $h_E(X_\#) = \alpha$. \blacktriangleleft

9.2 Block-gluing \mathbb{Z}^d Subshifts

There exists various mixing notions in higher dimension. We formulate our results for *block-gluing* subshifts:

► **Definition 33.** Let $X \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift, and $f: \mathbb{N} \rightarrow \mathbb{N}$ be a (weakly) increasing function. We say that X is f -block-gluing if

$$\forall p, q \in \mathcal{L}_n(X), \forall k \geq n + f(n), \forall u \in \mathbb{Z}^d, \|u\|_\infty \geq k \implies (p \cup \sigma^u(q)) \in \mathcal{L}(X)$$

Said differently, X is f -block-gluing if any two square patterns of size n can appear at any position as long as they are placed with a gap of size at least $f(n)$ between them. As with Definition 31, we will simply write N -block-gluing for constant gluing distance ($f: n \rightarrow N$).

► **Proposition 34.** For any $\alpha \in \Pi_3 \cap [0, +\infty)$, there exists an effective and 1-block-gluing \mathbb{Z}^d subshift $Z_{\alpha, \#}$ such that $h_E(Z_{\alpha, \#}) = \alpha$.

Proof. Notice that the free lift of a 1-block-gluing \mathbb{Z}^d subshift to \mathbb{Z}^{d+1} is also 1-block-gluing. By Claim 8, the free lift preserves the extender entropy: thus, we reduce to the one-dimensional case. We conclude by combining the previous Proposition 32 with Theorem A. \blacktriangleleft

References

- 1 Nathalie Aubrun, Sebastián Barbieri, and Emmanuel Jeandel. *About the Domino Problem for Subshifts on Groups*, pages 331–389. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-69152-7_9.
- 2 Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *Acta applicandae mathematicae*, 126:35–63, 2013. doi:10.1007/s10440-013-9808-5.
- 3 Sebastián Barbieri, Mathieu Sablik, and Ville Salo. Soficity of free extensions of effective subshifts. *Discrete and Continuous Dynamical Systems*, 45(4):1117–1149, 2025. doi:10.3934/dcds.2024125.
- 4 Jérôme Buzzi. Subshifts of quasi-finite type. *Inventiones mathematicae*, 2003. doi:10.1007/s00222-004-0392-1.

- 5 Silvio Capobianco. Multidimensional cellular automata and generalization of Fekete’s lemma. *Discrete Mathematics & Theoretical Computer Science (DMTCS)*, 10(3):95–104, 2008. doi:10.46298/dmtcs.442.
- 6 Julien Destombes. *Algorithmic Complexity and Soficness of Shifts in Dimension Two*. PhD thesis, Université de Montpellier, 2023. arXiv:2309.12241, doi:10.48550/arXiv.2309.12241.
- 7 Julien Destombes and Andrei Romashchenko. Resource-bounded Kolmogorov complexity provides an obstacle to soficness of multidimensional shifts. *Journal of Computer and System Sciences*, 128:107–134, 2022. doi:10.1016/j.jcss.2022.04.002.
- 8 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *Journal of Computer and System Sciences*, 78(3):731–764, 2012. doi:10.1016/j.jcss.2011.11.001.
- 9 Thomas French. Characterizing follower and extender set sequences. *Dynamical Systems*, 31(3):293–310, 2016. doi:10.1080/14689367.2015.1111865.
- 10 Thomas French and Ronnie Pavlov. Follower, predecessor, and extender entropies. *Monatshefte für Mathematik*, 188:495–510, 2019. doi:10.1007/s00605-018-1224-5.
- 11 Silvère Gangloff. *Algorithmic complexity of growth-type invariants of SFT under dynamical constraints*. PhD thesis, Aix-Marseille Université, 2018. URL: <http://www.theses.fr/2018AIXM0231/document>.
- 12 Pierre Guillon and Emmanuel Jeandel. Infinite communication complexity, 2015. arXiv:1501.05814.
- 13 Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones Mathematicae*, 176(1):2009, April 2009. doi:10.1007/s00222-008-0161-7.
- 14 Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics*, 171(3):2011–2038, 2010. doi:10.4007/annals.2010.171.2011.
- 15 Steve Kass and Kathleen Madden. A sufficient condition for non-soficness of higher-dimensional subshifts. *Proceedings of the American Mathematical Society*, 141(11):3803–3816, 2013. doi:10.1090/S0002-9939-2013-11646-1.
- 16 Petr Kůrka. *Topological and Symbolic Dynamics*. Société mathématique de France, 2003.
- 17 Douglas Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge university press, 2021.
- 18 Tom Meyerovitch. Growth-type invariants for \mathbb{Z}^d subshifts of finite type and arithmetical classes of real numbers. *Inventiones Mathematicae*, 184(3), 2010. doi:10.1007/s00222-010-0296-1.
- 19 Harold Marston Morse and Gustav Arnold Hedlund. Symbolic Dynamics. *American Journal of Mathematics*, 60(4):815–866, October 1938. doi:10.2307/2371264.
- 20 Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d’Analyse Mathématique*, 53(1):139–186, 1989. doi:10.1007/bf02793412.
- 21 Nic Ormes and Ronnie Pavlov. Extender sets and multidimensional subshifts. *Ergodic Theory and Dynamical Systems*, 36(3):908–923, 2016. doi:10.1017/etds.2014.71.
- 22 Ronnie Pavlov. A class of nonsocif multidimensional shift spaces. *Proceedings of the American Mathematical Society*, 141(3):987–996, 2013. doi:10.1090/S0002-9939-2012-11382-6.
- 23 Anthony N. Quas and Paul B. Trow. Subshifts of multi-dimensional shifts of finite type. *Ergodic Theory and Dynamical Systems*, 20(3):859–874, 2000. doi:10.1017/S0143385700000468.
- 24 Robert I Soare. *Turing computability: Theory and applications*, volume 300. Springer, 2016. doi:10.1007/978-3-642-31933-4.
- 25 Hao Wang. Proving theorems by Pattern Recognition II. *Bell Systems technical journal*, 40:1–41, 1961. doi:10.1002/j.1538-7305.1961.tb03975.x.
- 26 Benjamin Weiss. Subshifts of finite type and sofic systems. *Monatshefte für Mathematik*, 77:462–474, 1973. doi:10.1007/BF01295322.
- 27 Xizhong Zheng and Klaus Weihrauch. The arithmetical hierarchy of real numbers. *Mathematical Logic Quarterly: Mathematical Logic Quarterly*, 47(1):51–65, 2001. doi:10.1002/1521-3870(200101)47:1<51::AID-MALQ51>3.0.CO;2-W.

CMSO-Transducing Tree-Like Graph Decompositions

Rutger Campbell ✉

Discrete Mathematics Group, Institute for Basic Science, Daejeon, South Korea

Bruno Guillon ✉

Université Clermont Auvergne, Clermont Auvergne INP, LIMOS, CNRS, Clermont-Ferrand, France

Mamadou Moustapha Kanté ✉ 

Université Clermont Auvergne, Clermont Auvergne INP, LIMOS, CNRS, Clermont-Ferrand, France

Eun Jung Kim ✉ 

KAIST, Daejeon, South Korea

CNRS, France

Noleen Köhler ✉ 

University of Leeds, UK

Abstract

We show that given a graph G we can CMSO-transduce its modular decomposition, its split decomposition and its bi-join decomposition. This improves results by Courcelle [Logical Methods in Computer Science, 2006] who gave such transductions using order-invariant MSO, a strictly more expressive logic than CMSO. Our methods more generally yield C_2 MSO-transductions of the canonical decomposition of weakly-partitive set systems and weakly-bipartitive systems of bipartitions.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases MSO-transduction, MSO-definability, graph decompositions

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.22

Related Version *Extended Version:* [arXiv:2412.04970](https://arxiv.org/abs/2412.04970) [6]

Funding *Rutger Campbell:* Supported by the Institute for Basic Science (IBS-R029-C1).

Mamadou Moustapha Kanté: Supported by the French National Research Agency (ANR-18-CE40-0025-01 and ANR-20-CE48-0002).

1 Introduction

A decomposition of a graph, especially a tree-like decomposition, is a result of recursive separations of a graph and is extremely useful for investigating combinatorial properties such as colourability, and for algorithm design. Such a decomposition also plays a fundamental role when one wants to understand the relationship between logic and a graph class. Different notions of the complexity of a separation motivate different ways to decompose, such as tree-decomposition, branch-decomposition, rank-decomposition and carving-decomposition. Furthermore, some important graph classes can be defined through the tree-like decomposition they admit; cographs with cotrees and distance-hereditary graphs with split decompositions being prominent examples.

For a logic L , an L -transduction is a non-deterministic map from a class of relational structures to a new class of relational structures using L -formulas. Transducing a tree-like decomposition is of particular interest. Notably, transducing a decomposition of a graph implies that any property that is definable using a decomposition, is also definable on graphs that admit such a decomposition. Moreover, tree-like decompositions can be



© Rutger Campbell, Bruno Guillon, Mamadou Moustapha Kanté, Eun Jung Kim, and Noleen Köhler;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 22; pp. 22:1–22:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



often represented by labelled trees, for which the equivalence of recognisability by a tree automaton and definability in CMSO is well known [8]. Hence, it is an interesting question to consider what kind of graph decompositions can be transduced using L-transductions for some extension L of MSO.

In a series of papers [8, 9, 10, 12], Courcelle investigated the relationship between the graph properties that can be defined in an extension of MSO and the graph properties that can be recognized by a tree automaton. In particular, for graphs of bounded treewidth, Courcelle’s theorem states that any property that is definable in the logic MSO with modulo counting predicate, denoted CMSO, can be recognized by a tree automaton [8]. Combining this result with the linear time algorithm for computing tree-decompositions [1], yields that CMSO model-checking can be done in linear time on graphs of bounded treewidth. The converse statement – whether recognisability by a tree automaton implies definability in CMSO on graphs of bounded treewidth – was conjectured by Courcelle in [8] and finally settled by Bojańczyk and Pilipczuk [4]. The key step to obtain this result is obtaining a tree-decomposition of a graph via an MSO-transduction, a strategy which was initially proposed in [9] and is now standard.

The obvious next question is whether an analogous result can be proved for graphs of bounded clique-width and for more general combinatorial objects, most notably, matroids representable over a fixed finite field and of bounded branch-width. Due to the above-mentioned strategy, the key challenge is to produce corresponding tree-like decompositions by MSO-transduction. It is known that clique-width decompositions can be MSO-transduced for graphs of bounded linear clique-width [3]. However, it is unknown if clique-width decompositions can be MSO-transduced in general. In fact, this question remains open even for distance-hereditary graphs, which are precisely graphs of rank-width 1 (thus, of constant clique-width).

Besides tree-decompositions, the problem of transducing cotrees, and in general hierarchical decompositions such as modular decompositions and split decompositions were considered in the literature [10, 11, 12, 14]. In [10], Courcelle provides transductions using order-invariant MSO for cographs and modular decompositions of graphs of bounded modular width. Order-invariant MSO allows the use of a linear order of the set of vertices and is more expressive than CMSO [20]. The applicability of these transductions was later generalized using the framework of weakly-partitive set systems¹ to obtain order-invariant MSO-transductions of modular and split decompositions [12]. It was left as an open question whether one can get rid of the order (see for instance [13] where an overview of the result on hierarchical decompositions was given).

1.1 Our results

In this paper, we consider decompositions given by separations that do not overlap with any other separations of the same type. We view separations of a given kind as a “set system”. A set system consists of a set U , the universe, and a set \mathcal{S} of subsets of U . Two sets overlap if they have non-empty intersection but neither of them is contained in the other. If no two elements in a set system (U, \mathcal{S}) overlap, i.e. the set system is *laminar*, then the subset relation in (U, \mathcal{S}) can be described by a rooted tree T , called the *laminar tree* of (U, \mathcal{S}) . For any set system (U, \mathcal{S}) we can look at the subset of strong sets, i.e., sets that

¹ Weakly-partitive set systems are set systems enjoying some nice closure properties, which were then used to show that some set systems allow canonical tree representations, see for instance the thesis by Montgolfier and Rao [17, 24].

do not overlap with any other set, and the laminar tree T they induce. Additionally, we consider systems of bipartitions \mathcal{B} of a universe U . Two bipartitions of U overlap if neither side of one of the bipartition is contained in either side of the other bipartition. In case (U, \mathcal{B}) has no overlapping bipartitions, then (U, \mathcal{B}) can be described by an undirected tree, also called *laminar tree*, in which bipartitions correspond to edge cuts. We can define the concept of strong bipartitions equivalently and consider the laminar tree induced by the strong bipartitions in (U, \mathcal{B}) .

Given a graph G we can consider the set system $(V(G), \mathcal{M})$ where \mathcal{M} is the set of all modules in G , or the system of bipartition $(V(G), \mathcal{S})$ where \mathcal{S} contains all splits in G , or the system of bipartitions $(V(G), \mathcal{B})$ where \mathcal{B} contains all bi-joins in G . We obtain the modular decomposition/cotree/split decomposition/bi-join decomposition by equipping the laminar tree of the suitable set system/system of bipartitions with some additional structure. The additional structure allows us to recover the graph from the respective decomposition.

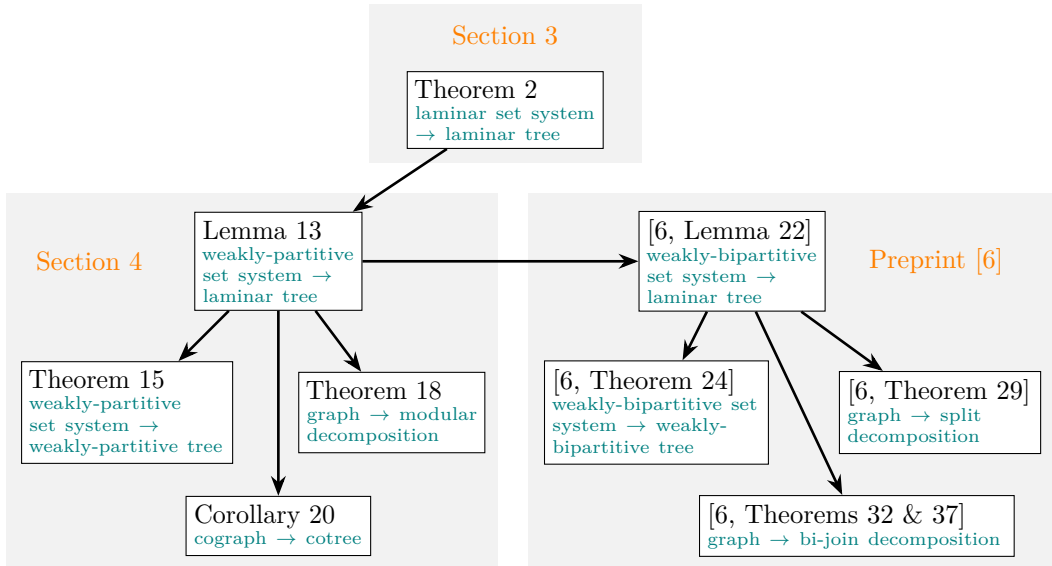
Abstractly, the set systems/systems of bipartitions mentioned are instances of *weakly-partitive set systems/weakly-bipartitive systems of bipartitions*. Roughly speaking, if a set system (U, \mathcal{S}) is well behaved, i.e. (U, \mathcal{S}) is weakly-partitive (definition in Section 2), then there is a labelling λ of T and a partial order $<$ of its nodes such that (U, \mathcal{S}) is completely described by $(T, \lambda, <)$ [7, 24]. A similar statement holds for systems of bipartitions [17].

We show the following, wherein each item λ is a suitable labelling of the nodes of the laminar tree T , $<$ is a partial ordering of its nodes, and F is an additional edge relation defined only on pairs of siblings in T . A visualization how results depend on each other is given in Figure 1. Due to space constraints, some proofs are omitted, but they are available in the extended preprint [6].

► **Theorem 1.** *There are non-deterministic C_2 MSO-transductions τ_1, \dots, τ_7 such that:*

1. *For any laminar set system (U, \mathcal{S}) , $\tau_1(U, \mathcal{S})$ is non-empty and every output in $\tau_1(U, \mathcal{S})$ is a laminar tree T of (U, \mathcal{S}) (Theorem 2);*
2. *For any graph G , $\tau_2(G)$ is non-empty and every output in $\tau_2(G)$ is a modular decomposition (T, F) of G (Theorem 18);*
3. *For any cograph G , $\tau_3(G)$ is non-empty and every output in $\tau_3(G)$ is a cotree (T, λ) of G (Corollary 20);*
4. *For any graph G , $\tau_4(G)$ is non-empty and every output in $\tau_4(G)$ is a split decomposition (T, F) of G [6, Theorem 29];*
5. *For any graph G , $\tau_5(G)$ is non-empty and every output in $\tau_5(G)$ is a bi-join decomposition (T, F) of G [6, Theorems 32 & 37];*
6. *For any weakly-partitive set system (U, \mathcal{S}) , $\tau_6(U, \mathcal{S})$ is non-empty and every output in $\tau_6(U, \mathcal{S})$ is a weakly-partitive tree $(T, \lambda, <)$ of (U, \mathcal{S}) (Theorem 15);*
7. *For any weakly-bipartitive set system (U, \mathcal{B}) , $\tau_7(U, \mathcal{B})$ is non-empty and every output in $\tau_7(U, \mathcal{B})$ is a weakly-bipartitive tree $(T, \lambda, <)$ of (U, \mathcal{B}) [6, Theorem 24].*

The key step in obtaining these transductions is to transduce the laminar tree T of a set system (U, \mathcal{S}) , namely Theorem 2. The crux here is to find a suitable representative of each node of T amongst the elements of U and a non-deterministic coloring which allows the assignment of representatives to nodes by means of a C_2 MSO-formula. It should be mentioned that a similar result is claimed in the preprint [2], where a proof sketch designing a C_3 MSO-transduction is described. Once the laminar tree is obtained, the additional relations for each decomposition can be obtained using a deterministic MSO-transduction. Notice that for each of these transductions, there exists an inverse deterministic MSO-transduction, namely a transduction that from the tree-like decomposition outputs the original structure.



■ **Figure 1** Overview of the various transductions in the paper. An arrow from x to y indicates that result x is used in the proof of result y .

1.2 Organization

The paper is organized as follows. In Section 2 we introduce terminology and notation needed. In Section 3 we prove Theorem 2. In Section 4 we provide the proof of Theorems 15 and 18 and obtain Corollary 20.

2 Preliminaries

2.1 Graphs, trees, set systems

Graphs. We use standard terminology of graph theory, and we fix some notations. Given a *directed graph* G , its sets of *vertices* and *edges* are denoted by $V(G)$ and $E(G)$, respectively. We denote by uv an edge $(u, v) \in E(G)$. An undirected graph is no more than a directed graph for which $E(G)$ is symmetric (*i.e.*, $uv \in E(G) \iff vu \in E(G)$). The notions of *paths*, *connected components*, *etc.* are defined as usual. Given a subset Z of $V(G)$, we denote by $G[Z]$ the sub-graph of G induced by Z .

Trees. A *tree* is a connected undirected graph without cycles. In the context of trees, we use a slightly different terminology than for graphs. In particular, vertices are called *nodes*, nodes of degree at most 1 are called *leaves*, and nodes of degree greater than 1 are called *inner*. The set of leaves is denoted $L(T)$; thus the set of inner nodes is $V(T) \setminus L(T)$. For a node t of a tree T and a neighbor s of t , we denote by T_s^t the connected component of $T - t$ containing s . We sometimes consider *rooted trees*, namely trees with a distinguished node, called the *root*. Rooted trees enjoy a natural orientation of their edges toward the root, which induces the usual notions of *parent*, *child*, *sibling*, *ancestor* and *descendant*. Hence, we represent a rooted tree by a set of nodes with an ancestor/descendant relationship (instead of specifying the root). We use the convention that every node is one of its own ancestors and descendants. We refer to ancestors (resp. descendants) of a node that are not the node itself as *proper ancestors* (resp. *proper descendants*). For a node t of a rooted tree T , we denote by T_t the subtree of T rooted in t (*i.e.*, the restriction of T to the set of descendants of t).

Set systems and laminar trees. A *set system* is a pair (U, \mathcal{S}) where U is a finite set, called the *universe*, and \mathcal{S} is a family of subsets of U where $\emptyset \notin \mathcal{S}$, $U \in \mathcal{S}$, and $\{a\} \in \mathcal{S}$ for every $a \in U$.² Two sets X and Y *overlap* if they are neither disjoint nor related by containment. A set system (U, \mathcal{S}) is said to be *laminar* (aka *overlap-free*) when no two sets from \mathcal{S} overlap. By extension, a set family \mathcal{S} is *laminar* if $(\bigcup \mathcal{S}, \mathcal{S})$ is a laminar set system (note this also requires that $\emptyset \notin \mathcal{S}$, $\bigcup \mathcal{S} \in \mathcal{S}$, and $\{a\} \in \mathcal{S}$ for every $a \in \bigcup \mathcal{S}$).

A laminar family \mathcal{S} of subsets of U naturally defines a rooted tree where the nodes are the sets from \mathcal{S} , the root is U , and the ancestor relation corresponds to set inclusion. We call this rooted tree the *laminar tree of U induced by \mathcal{S}* (or *laminar tree of (U, \mathcal{S})*). In this rooted tree, the leaves are the singletons $\{x\}$ for $x \in U$, which we identify with the elements themselves. That is to say, $L(T) = U$. Laminar trees have the property that each inner node has at least two children. Observe also that the size of a laminar tree is linearly bounded in the size of the universe.

2.2 Logic and transductions

We use *relational structures* to model both graphs and the various tree-like decompositions used in this paper. In order to concisely model set systems we use the more general notion of *extended relational structures*, namely *relational structure* extended with set predicate names. Such structures also naturally arise as outputs of *MSO-transductions* defined below.

Define an (*extended*) *vocabulary* to be a set of symbols, each being either a *relation* name R with associated arity $\text{ar}(R) \in \mathbb{N}$, or a *set predicate* name P with associated arity $\text{ar}(P) \in \mathbb{N}$. Set predicate names are aimed to describe relations between sets, *e.g.*, one may have a unary set predicate for selecting finite sets of even size, or a binary set predicate for selecting pairs of disjoint sets. We use capital R or names starting with a lowercase letter (*e.g.*, *edge*, *ancestor*, *t-edge*) for relation names, and capital P or uppercase names (*e.g.*, *SET*, *C₂*) for set predicate names. To refer to an arbitrary symbol of undetermined kind, we use capital Q . A *relational vocabulary* is an extended vocabulary in which every symbol is a relation name.

Let Σ be a vocabulary. An *extended relational structure over Σ* (Σ -*structure*) is a structure $\mathbb{A} = \langle U_{\mathbb{A}}, (Q_{\mathbb{A}})_{Q \in \Sigma} \rangle$ consisting on the one hand of a set $U_{\mathbb{A}}$ called *universe*, and on the other hand, for each symbol Q in Σ , an *interpretation $Q_{\mathbb{A}}$ of Q* , which is a relation of arity $\text{ar}(Q)$ either over the universe if Q is a relation name, or over the family of subsets of the universe if Q is a set predicate name. When Σ is not extended, \mathbb{A} is simply a *relational structure*.

Given a Σ -structure \mathbb{A} and, for some vocabulary Γ , a Γ -structure \mathbb{B} , we write $\mathbb{A} \sqsubseteq \mathbb{B}$ if $\Sigma \subseteq \Gamma$, $U_{\mathbb{A}} \subseteq U_{\mathbb{B}}$ and for each symbol Q in Σ , $Q_{\mathbb{A}} = Q_{\mathbb{B}}$.³ We write $\mathbb{A} \sqcup \mathbb{B}$ to denote the $(\Sigma \cup \Gamma)$ -structure consisting of the universe $U_{\mathbb{A}} \cup U_{\mathbb{B}}$ and, for each symbol $Q \in \Sigma \cup \Gamma$, the interpretation $Q_{\mathbb{A} \sqcup \mathbb{B}}$ which is $Q_{\mathbb{A}}$, $Q_{\mathbb{B}}$, or $Q_{\mathbb{A}} \cup Q_{\mathbb{B}}$ according to whether Q belongs to $\Sigma \setminus \Gamma$, to $\Gamma \setminus \Sigma$, or to $\Sigma \cap \Gamma$.⁴

To describe properties of (extended) relational structures, we use *monadic second order logic* (MSO) and refer for instance to [15, 19, 22, 25] for the definition of MSO on extended relational structures such as matroids or set systems in general. This logic allows quantification

² Though these restrictions on \mathcal{S} are not usual for set systems, it is convenient for our contribution and it does not significantly impact the generality of set systems: every family \mathcal{F} of subsets of U can be associated with a set system (U, \mathcal{S}) where $\mathcal{S} = (\mathcal{F} \setminus \{\emptyset\}) \cup \{U\} \cup \{\{a\} \mid a \in U\}$.

³ We require equality here (in particular only elements or subsets of $U_{\mathbb{A}}$ are related in $Q_{\mathbb{B}}$). This differs from classical notions of inclusions of relational structures which typically require equality only on the restriction of the universe to $U_{\mathbb{A}}$, *i.e.*, $Q_{\mathbb{A}} = Q_{\mathbb{B}/U_{\mathbb{A}}}$, *e.g.*, in order to correspond to induced graphs.

⁴ We do not require \mathbb{A} and \mathbb{B} to be disjoint structures whence we may have $\mathbb{A} \not\sqsubseteq \mathbb{A} \sqcup \mathbb{B}$.

both over single elements of the universe and over subsets of the universe. We also use *counting* MSO (CMSO), which is the extension of MSO with, for every positive integer p , a unary set predicate C_p that checks whether the size of a subset is divisible by p or not. We only use C_2 . As usual, lowercase variables indicate first-order-quantified variables, while uppercase variables indicate monadic-quantified variables. For a formula ϕ , we write, e.g., $\phi(x, y, X)$ to indicate that the variables x , y , and X belong to the set of *free variables of ϕ* , namely, the set of variables occurring in ϕ that are not bound to a quantifier within ϕ . A *sentence* is a formula without free variables.

We now fix some (extended) vocabularies that we will use.

Graphs. To model both graphs, unrooted trees, and directed graphs, we use the relational vocabulary $\{\text{edge}\}$ where edge is a relation name of arity 2. A (directed) graph $G = (V, E)$ is modeled as the $\{\text{edge}\}$ -structure \mathbb{G} with universe $U_{\mathbb{G}} = V$ and interpretation $\text{edge}_{\mathbb{G}} = E$. In particular if G is undirected, then $\text{edge}_{\mathbb{G}}$ is symmetric.

Rooted trees. We use the relational vocabulary $\{\text{ancestor}\}$ to model rooted trees where ancestor is a relation name of arity 2. A rooted tree T is modeled as the $\{\text{ancestor}\}$ -structure \mathbb{T} with universe $U_{\mathbb{T}} = V(T)$ and the interpretation $\text{ancestor}_{\mathbb{T}}$ being the set of pairs (u, v) such that u is an ancestor of v in T . It is routine to define FO-formulas over this vocabulary to express the binary relations *parent*, *child*, *proper ancestor*, (*proper*) *descendant*, as well as the unary relations *leaf* and *root*.

Set systems. To model set systems, we use the extended vocabulary $\{\text{SET}\}$ where SET is a set predicate name of arity 1. A set system $S = (U, \mathcal{S})$ is thus naturally modeled as the $\{\text{SET}\}$ -structure \mathbb{S} with universe $U_{\mathbb{S}} = U$ and interpretation $\text{SET}_{\mathbb{S}} = \mathcal{S}$.

Transductions

Let Σ and Γ be two extended vocabularies. A Σ -to- Γ *transduction* is a set τ of pairs formed by a Σ -structure, call the *input*, and a Γ -structure, called the *output*. We write $\mathbb{B} \in \tau(\mathbb{A})$ when $(\mathbb{A}, \mathbb{B}) \in \tau$. When for every pair $(\mathbb{A}, \mathbb{B}) \in \tau$ we have $\mathbb{A} \sqsubseteq \mathbb{B}$, we call τ an *overlay transduction*. Some transductions can be defined by means of MSO- or C_2 MSO-formulas. This leads to the notion of MSO- and C_2 MSO-*transductions*. Following the presentation of [3], for L denoting MSO or C_2 MSO, define an L -*transduction* to be a transduction obtained by composing a finite number of *atomic L-transductions* of the following kinds.

Filtering. An overlay transduction specified by an L -sentence ϕ over the input vocabulary Σ , which discards the inputs that do not satisfy ϕ and keeps the other unchanged. Hence, it defines a partial function (actually, a partial identity) from Σ -structures to Σ -structures.

Universe restriction. A transduction specified by an L -formula ϕ over the input vocabulary Σ , with one free first-order variable, which restricts the universe to those elements that satisfy ϕ . The output vocabulary is Σ and the interpretation of every relation (resp. every predicate) in the output structure is defined as the restriction of its interpretation in the input structure to those tuples of elements satisfying ϕ (resp. tuples of sets of elements that satisfy ϕ). This defines a total function from Σ -structures to Σ -structures.

Interpretation. A transduction specified by a family $(\phi_Q)_{Q \in \Gamma}$ over the input vocabulary Σ where Γ is the output vocabulary and each ϕ_Q has $\text{ar}(Q)$ free variables which are first-order if Q is a relation name and monadic if it is a set predicate name. The transduction outputs the Γ -structure that has the same universe as the input structure and in which each relation or predicate Q is interpreted as those set of tuples that satisfy ϕ_Q . This defines a total function from Σ -structures to Γ -structures.

Copying. An overlay transduction parametrized by a positive integer k that adds k copies of each element to the universe. The output vocabulary consists in the input vocabulary Σ extended with k binary relational symbols $(\text{copy}_i)_{i \in [k]}$ interpreted as pairs of elements (x, y)

saying that “ y is the i -th copy of x ”. The interpretation of the relations (resp. predicates) of the input structure are preserved, on original elements. This defines a total function from Σ -structures to Γ -structures, where $\Gamma = \Sigma \cup \{\text{copy}_i \mid i \in [k]\}$.

Colouring. An overlay transduction that adds a new unary relation $\text{color} \notin \Sigma$ to the structure. Any possible interpretation yields an output; indeed the interpretation is chosen non-deterministically. The interpretation of the relations (resp. predicates) of the input structure are preserved. Hence, it defines a total (non-functional) relation from Σ -structures to Γ -structures where $\Gamma = \Sigma \cup \{\text{color}\}$ (providing $\text{color} \notin \Sigma$).

We say an L-transduction is *deterministic* if it does not use colouring, it is *non-deterministic* otherwise. By definition, deterministic L-transductions define functions.

3 Transducing the laminar-tree

In this section, we present an overlay C_2 MSO-transduction that takes as input a laminar set system and outputs the laminar tree it induces.

► **Theorem 2.** *Let Σ be an extended vocabulary, including a unary set predicate name SET and not including the binary relational symbol ancestor. There exists a non-deterministic C_2 MSO-transduction τ such that, for each laminar set system (U, \mathcal{S}) represented as the {SET}-structure \mathbb{S} and inducing the laminar tree T with $L(T) = U$, and for each Σ -structure \mathbb{A} with $\mathbb{S} \sqsubseteq \mathbb{A}$, $\tau(\mathbb{A})$ is non-empty and every output in $\tau(\mathbb{A})$ is equal to some {ancestor}-structure \mathbb{T} representing T .*

Since the sets from \mathcal{S} are precisely the sets of leaves of the subtrees of T , there is an MSO-transduction which is the inverse of the above C_2 MSO-transduction; that is to say, given an {ancestor}-structure \mathbb{T} representing the laminar tree, it outputs the original set system \mathcal{S} . Namely,

1. An interpretation $\Phi_{\text{SET}}(S)$ that is true for a set S when there exists an element a such that an element x is in S if and only if a is an ancestor of x .
2. The filtering $\phi(x)$ keeping leaves only.

We prove Theorem 2 in Section 3.2, using the key tools developed in Section 3.1, that allow us to represent each inner node of T with a pair of leaves from its subtree, while keeping the number of inner nodes represented by a given leaf bounded.

3.1 Inner node representatives

In this section, the root of any rooted tree is always an inner node (unless the tree is a unique node). We fix a rooted tree T , in which every inner node has at least two children (a necessary assumption that is satisfied by laminar trees), and we let V denote the set of its nodes ($V = V(T)$) and $L \subseteq V$ denote the subset of its leaves ($L = L(T)$).

Let $S \subseteq V \setminus L$, and let (π, σ) be a pair of injective mappings from S to L . We say that the pair (π, σ) *identifies* S if for each $s \in S$, s is the least common ancestor of $\pi(s)$ and $\sigma(s)$. For $s \in S$ and $x \in V$, we say that x is *s-requested in* (π, σ) if x lies on the path from $\pi(s)$ to $\sigma(s)$ (namely, on either of the paths from $\pi(s)$ to s and from $\sigma(s)$ to s). We say that x is *requested in* (π, σ) if it is s -requested for some s . The pair (π, σ) has *unique request* if every node x of T is requested at most once in (π, σ) , *i.e.*, there exists at most one $s \in S$ such that x is s -requested in (π, σ) . Note that if (π, σ) has unique request then the paths from $\pi(s)$ to $\sigma(s)$ are pairwise disjoint for all the $s \in S$. We now state a few basic observations.

► **Remark 3.** Let (π, σ) identifying some subset S of $V \setminus L$.

1. The reversed pair (σ, π) also identifies S and has unique request whenever (π, σ) does.
2. If $S' \subset S$, then $(\pi|_{S'}, \sigma|_{S'})$ identifies S' , and has unique request if (π, σ) does.
3. For each $s \in S$, s is s -requested in (π, σ) .
4. If (σ, π) has unique request, then $\pi(S)$ and $\sigma(S)$ are disjoint subsets of L .

► **Lemma 4.** *Let (π, σ) identifying some subset S of $V \setminus L$ with unique request. Then for each $a \in \pi(S)$ the node $\pi^{-1}(a)$ is the least ancestor of a which belongs to S .*

Proof. Let $a \in \pi(S)$ and let $s = \pi^{-1}(a)$. By definition, s is an ancestor of a which belongs to S . Let y be the least ancestor of a that is contained in S . As s is an ancestor of a belonging to S , y must be a descendant of s . Hence, y is s -requested in (π, σ) . Additionally, by Item 3 of Remark 3, y is y -requested in (π, σ) . Since (π, σ) has unique request, $y = s$. Thus, s is the least ancestor of a which belongs to S . ◀

Notice that, by Item 1 of Remark 3, a similar result holds for each $b \in \sigma(S)$. It follows that the sets $\pi(S)$ and $\sigma(S)$ characterize (π, σ) .

► **Lemma 5.** *Let $S \subseteq V \setminus L$, and (π, σ) and (π', σ') be two pairs of injections from S to L identifying S with unique request. If $\pi(S) = \pi'(S)$ and $\sigma(S) = \sigma'(S)$ then $\pi = \pi'$ and $\sigma = \sigma'$.*

Proof. Let $s \in S$, $a = \pi(s)$, and $s' = \pi'^{-1}(a)$. Both s and s' are the least ancestor of a which belongs to S , hence $s = s'$ and $\pi'(s) = a$. Thus $\pi = \pi'$. By Item 1 of Remark 3, we also obtain that $\sigma = \sigma'$. ◀

Let A and B be two disjoint subsets of L . We call such a pair (A, B) a *bi-colouring*. We say that (A, B) *identifies* S if there exists a pair (π, σ) identifying S with unique request such that $\pi(S) = A$ and $\sigma(S) = B$. By the previous lemma, for a fixed set $S \subseteq V \setminus L$ the pair (π, σ) identifying S is unique when it exists. We will also prove that S is actually uniquely determined from (A, B) . Before that, we state the following technical lemma.

► **Lemma 6.** *Let (A, B) identify some subset S of $V \setminus L$ through a pair (π, σ) of injections having unique request. Then, for each inner node x , exactly one of the three following cases holds:*

1. $x \notin S$, x is not requested in (π, σ) , and for each child c of x , $|A \cap V(T_c)| = |B \cap V(T_c)|$;
2. $x \notin S$, x is requested in (π, σ) , and there exists one leaf $z \in (A \cup B) \cap V(T_x)$ such that, for each child c of x , $|(A \setminus \{z\}) \cap V(T_c)| = |(B \setminus \{z\}) \cap V(T_c)|$;
3. $x \in S$, x is requested in (π, σ) , and there exists two leaves $a \in A \cap V(T_x)$ and $b \in B \cap V(T_x)$ such that, for each child c of x , $|(A \setminus \{a\}) \cap V(T_c)| = |(B \setminus \{b\}) \cap V(T_c)|$ and $\{a, b\} \not\subseteq V(T_c)$.

Proof. Let x be an inner node. We consider the set $S' = S \setminus (V(T_x) \setminus \{x\})$ of all nodes which are not proper descendants of x and the restrictions π' and σ' of, respectively, π and σ to S' . By Item 2 of Remark 3 (π', σ') identify S' with unique request. We denote $A' = \pi'(S')$ and $B' = \sigma'(S')$, thus (A', B') identify S' . Let $A'_x = A' \cap V(T_x)$ and $B'_x = B' \cap V(T_x)$ be the sets of representatives contained in T_x of nodes in S' . Let a be an element of A'_x . The element $s_a = \pi^{-1}(a)$ is an ancestor of x because it is an ancestor of $a \in V(T_x)$ and belongs to S' whence not to $V(T_x) \setminus \{x\}$. Therefore, x is s_a -requested. As (π', σ') has unique request, there exists at most one element in A'_x . Similarly, B'_x has size at most 1. Moreover, $A'_x \cap B'_x = \emptyset$ by Item 4 of Remark 3. We thus we have three cases:

Case $A'_x \cup B'_x = \emptyset$. Then there is no $s \in S'$ such that $\pi(s) \in V(T_x)$ or $\sigma(s) \in V(T_x)$. Hence, x is not requested in (π, σ) , in particular, $x \notin S$.

Let c be a child of x . If $|A \cap V(T_c)| \neq |B \cap V(T_c)|$, then there exists $a \in (A \cup B) \cap V(T_c)$ such that, assuming without loss of generality that $a \in A$ and denoting $s_a = \pi^{-1}(a)$, $\sigma(s_a) \notin V(T_c)$. Hence, s_a is a proper ancestor of c , thus, equivalently, an ancestor of x , implying that x is s_a -requested in (π, σ) . This contradicts the above argument. So, $|A \cap V(T_c)| = |B \cap V(T_c)|$ for each child c of x .

Case $A'_x \cup B'_x = \{a\}$. Assume, without loss of generality, that $a \in A$, and denote $s_a = \pi^{-1}(a)$ and $b = \sigma(s_a)$. We have that s_a is a proper ancestor of x , since it is the least common ancestor of $a \in V(T_x)$ and $b \notin V(T_x)$. Thus, x is s_a -requested and $s_a \neq x$ so $x \notin S$.

Let c be a child of x . If $|A \cap V(T_c)| \neq |B \cap V(T_c)|$, then it means that there exists $z \in (A \cup B) \cap V(T_c)$, such that either $z \in A$ and $s_z = \pi^{-1}(z) \notin V(T_c)$, or $z \in B$ and $s_z = \sigma^{-1}(z) \notin V(T_c)$. In both cases, s_z is a proper ancestor of c , whence an ancestor of x . Thus, x is s_z -requested in (π, σ) . However, x is s_a -requested in (π, σ) which has unique request, hence $s_z = s_a$. If $z \in B$, it follows that $z = b$, which contradicts the fact $b \notin V(T_x)$. Hence, $z \in A$ and thus, $z = \pi(s_a) = a$. Therefore, $|(A \setminus \{a\}) \cap V(T_c)| = |B \cap V(T_c)|$ for each child c of x .

Case $A'_x = \{a\}$ and $B'_x = \{b\}$. Let $s_a = \pi^{-1}(a)$ and $s_b = \sigma^{-1}(b)$. The node x is s_a -requested and s_b -requested in (π, σ) , so, by the unique request property, $s_a = s_b$. Since s_a is the least common ancestor of a and b , it belongs to $V(T_x)$ whence $s_a = x$ implying $x \in X$. Let c be a child of x . If $|A \cap V(T_c)| \neq |B \cap V(T_c)|$, then there exists $z \in (A \cup B) \cap V(T_c)$ such that either $z \in A$ and $s_z = \pi^{-1}(z) \notin V(T_c)$, or $z \in B$ and $s_z = \sigma^{-1}(z) \notin V(T_c)$. In both cases, s_z is a proper ancestor of c , whence an ancestor of x , and thus x is s_z -requested in (π, σ) . However, x is x -requested in (π, σ) which has unique request, hence $s_z = x$ and thus $z \in \{a, b\}$. Therefore, $|(A \setminus \{a\}) \cap V(T_c)| = |(B \setminus \{b\}) \cap V(T_c)|$ for each child c of x . Because the least common ancestor of a and b is x , there is no child c of x containing both a and b as leaves, *i.e.*, $\{a, b\} \not\subseteq V(T_c)$.

This concludes the proof of the statement. \blacktriangleleft

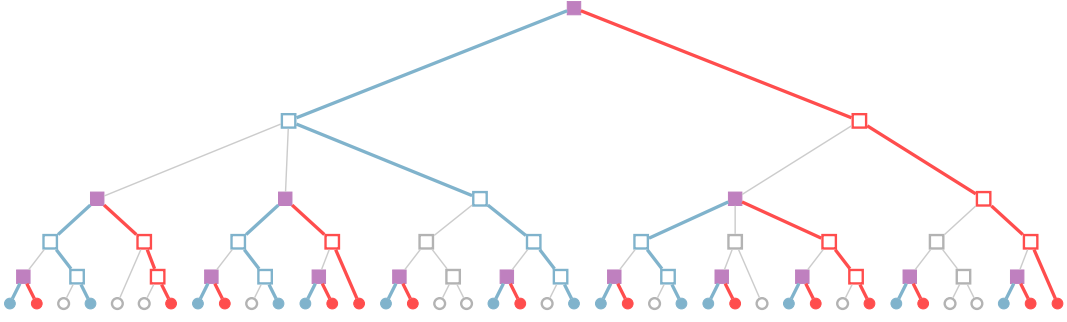
It follows that for each bi-colouring (A, B) , there exists at most one set S of inner nodes identified by (A, B) .

► Lemma 7. *Let (A, B) be two disjoint subsets of L and let S and S' be two subsets of $V \setminus L$. If (A, B) identify both S and S' , then $S = S'$.*

Proof. We proceed by contradiction and thus assume $S \neq S'$. Let $s \in S \setminus S'$. By Lemma 6, the number of children c of s for which the set $(A \cup B) \cap V(T_c)$ has odd size is 2 since $s \in S$, while it should also be less than 2 since $s \notin S'$. A contradiction. \blacktriangleleft

When (A, B) identifies a set S , we call *A-representative* (resp. *B-representative*) of $s \in S$ the leaf $\pi(s) \in A$ (resp. $\sigma(s) \in B$), where (π, σ) witnessing that (A, B) identifies S . An example of bi-colouring identifying a subset of inner nodes is given in Figure 2.

Not every set of inner nodes has a bi-colouring identifying it. To ensure that such a pair exists, we consider *thin sets of inner nodes*. While thin sets always have bi-colourings identifying them, it is also guaranteed that the set of inner nodes can be partitioned into only 4 thin sets. A subset $X \subseteq V \setminus L$ is *thin* when, for each $x \in X$ not being the root, on the one hand, the parent p_x of x does not belong to X , and on the other hand, x admits at least one sibling (including possible leaves) that does not belong to X . Having a thin set X allows to find branches avoiding it.



■ **Figure 2** Illustration of a bi-colouring (A, B) identifying some set $S \subseteq V \setminus L$ in a binary tree T . Leaves from A are coloured blue, leaves from B are coloured red, and inner nodes from S are marked purple. Furthermore, the two paths connecting an inner node $s \in S$ to its A - and B -representatives are coloured blue and red, respectively.

► **Lemma 8.** *Let $X \subseteq V \setminus L$ and $s \in V \setminus X$. If X is thin, then there exists a leaf $t \in V(T_s)$ such that the path from t to s avoids X (i.e., none of the nodes along this path belong to X).*

Proof. If T_s has height 0, then s is a leaf and taking $t = s$ trivially gives the expected path. Otherwise, s is an inner node and, because X is thin, s has at least one child c_s not belonging to X . By induction, there is a path from some leaf $t \in V(T_{c_s})$ to c_s avoiding X and, since $s \notin X$, this path could be extended into a path from t to s avoiding X . ◀

The previous lemma allows to identify every thin set.

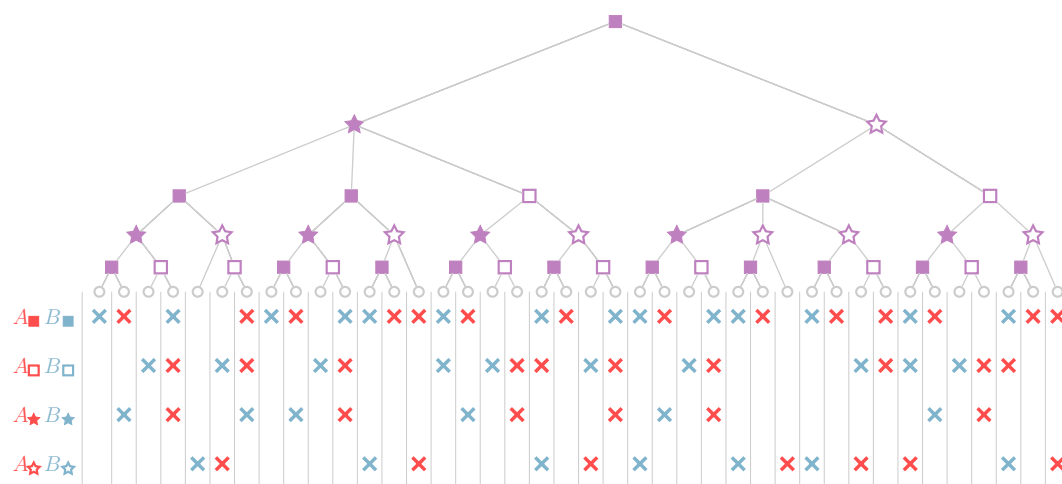
► **Lemma 9.** *If X is a thin set, then there exists a pair (π, σ) of injections from X to L that has unique request and that identifies X .*

Proof. We proceed by induction on the size of X . If $X = \emptyset$, the result is trivial. Let $n \in \mathbb{N}$ and suppose that for every thin set of size n there exists a pair of injections identifying it with unique request. Let X be a thin set of size $n + 1$, and let $s \in X$ be of minimal depth. Clearly, $X \setminus \{s\}$ is thin and thus there exists, by induction, a pair (π, σ) of injections from $X \setminus \{s\}$ to L identifying $X \setminus \{s\}$ with unique request. Since X is thin and $s \in X$, we can find two distinct children c_a and c_b of s not belonging to X .⁵ Then, by Lemma 8, there exists a leaf $a \in V(T_{c_a})$ (resp. a leaf $b \in V(T_{c_b})$) such that the path from a to c_a (resp. from b to c_b) avoids X . In particular, for each node y along these paths, since y has no ancestor that belongs to X but s , y is not requested in (π, σ) . Hence, extending π (resp. σ) in such a way that, besides mapping each $x \in X \setminus \{s\}$ to $\pi(x)$ (resp. to $\sigma(x)$), it maps s to a (resp. to b), we obtain a pair $(\hat{\pi}, \hat{\sigma})$ of injections from X to L that identifies X with unique request. ◀

A family $F = (A_1, B_1), \dots, (A_n, B_n)$ of bi-colourings *identifies* a set $S \subseteq V \setminus L$, if there exists a partition (S_1, \dots, S_n) of S such that, for each $i \in [n]$, (A_i, B_i) identifies S_i . Whenever $S = V \setminus L$ we say that F *identifies* T . A collection of subsets of $V \setminus L$ is *thin* if each of its subsets is thin. We now show that there exists a thin 4-partition.

► **Lemma 10.** *There exists a thin 4-partition of $V \setminus L$.*

⁵ Remember that every inner node of T has at least two children (including possible leaves).



■ **Figure 3** Illustration of a partition of the inner nodes into 4 thin sets indicated by different shapes/fillings. The four colourings identifying each thin set are indicated in the table below the leaves.

Proof. We build such a thin 4-partition as follows. First, consider the partition (D_e, D_o) of $V \setminus L$ in which D_e (resp. $D_o = V \setminus (D_e \cup L)$) is the set of all inner nodes of even (resp. odd) depth. Second, arbitrarily fix one child c_x of x for each inner node x , and consider the set $C = \{c_x \mid x \in V \setminus L\} \setminus L$, inducing a partition (C, \bar{C}) of $V \setminus L$ where $\bar{C} = V \setminus (L \cup C)$. By refining these two bi-partitions, we obtain a 4-partition which is thin by construction.⁵ ◀

Hence, four bi-colourings are enough to identify T . For an example see Figure 3.

► **Corollary 11.** *There exists a family of four bi-colourings identifying T .*

Proof. The result immediately follows from Lemmas 9 and 10. ◀

3.2 The transduction

The goal of this section is to prove Theorem 2, that is, to design a C_2 MSO-transduction that produces the laminar tree induced by an input laminar set system. We fix a laminar set system (U, \mathcal{S}) , represented by a $\{\text{SET}\}$ -structure \mathbb{S} . Before proving the theorem, we make the following basic observation. On \mathbb{S} , we can define two MSO-formulas $\text{desc}(X, Y)$ and $\text{child}(X, Y)$ expressing that in the laminar tree induced by \mathcal{S} , X and Y are nodes and X is a descendant or a child of Y , respectively:

$$\begin{aligned} \text{desc}(X, Y) &:= \text{SET}(X) \wedge \text{SET}(Y) \wedge X \subseteq Y; \\ \text{child}(X, Y) &:= \text{desc}(X, Y) \wedge X \neq Y \wedge \forall Z \left((\text{desc}(Z, Y) \wedge Z \neq Y) \rightarrow \text{desc}(Z, X) \right). \end{aligned}$$

The key point of our construction consists in defining a C_2 MSO-formula $\text{repr}_{A,B}(a, X)$ which, assuming a bi-colouring (A, B) of the universe modeled as disjoint unary relations and identifying a subset S of inner nodes of T , is satisfied exactly when $X \in S$ and a is its A -representative.

► **Lemma 12.** *Let (A, B) be a bi-colouring of $L(T)$ identifying a subset S of inner nodes of T . There exists a C_2 MSO-formula $\text{repr}_{A,B}(a, X)$ that is satisfied exactly when X is an inner node of T that belongs to S and is A -represented by a .*

22:12 CMSO-Transducing Tree-Like Graph Decompositions

Proof. First, we define a formula $\phi_{A,B}(X)$ that, under the above assumption, is satisfied exactly when X belongs to S . According to Lemma 6, this happens if and only if X is a node of T (i.e., $\text{SET}(X)$ is satisfied) and there exists $a \in X \cap A$ and $b \in X \cap B$ such that for each child Z of X , $\{a, b\} \not\subseteq Z$ and the set $(Z \setminus \{a, b\}) \cap (A \cup B)$ has even size. This property is easily expressed in $C_2\text{MSO}$, using the MSO-formula $\text{child}(X, Y)$ defined previously, as well as the predicate SET :

$$\phi_{A,B}(X) := \text{SET}(X) \wedge \exists a \exists b \left[a \in (X \cap A) \wedge b \in (X \cap B) \wedge \forall Z \left(\text{child}(Z, X) \rightarrow \left(\{a, b\} \not\subseteq Z \wedge C_2((Z \setminus \{a, b\}) \cap (A \cup B)) \right) \right) \right].$$

Now, we can easily define $\text{repr}_{A,B}(a, X)$ based on Lemma 4:

$$\text{repr}_{A,B}(a, X) := \phi_{A,B}(X) \wedge a \in (X \cap A) \wedge \forall Z \subsetneq X \left(a \in Z \rightarrow \neg \phi_{A,B}(Z) \right).$$

This concludes the proof. ◀

We are now ready to prove the theorem.

Proof of Theorem 2. The $C_2\text{MSO}$ -transduction is obtained by composing the following atomic $C_2\text{MSO}$ -transductions. The transduction makes use of the formulas $\text{repr}_{A,B}(a, X)$ given by Lemma 12.

1. Guess a family of four bi-colourings $(A_i, B_i)_{i \in [4]}$ identifying T (which exists by Corollary 11).
2. Copy the input graph four times, thus introducing four binary relations $(\text{copy}_i)_{i \in [4]}$ where $\text{copy}_i(x, y)$ indicates that x is the i -th copy of the original element y .
3. Filter the universe keeping only the original elements as well as the i -th copy of each vertex a for which there exists X such that $\text{repr}_{A_i, B_i}(a, X)$.
4. Define the relation $\text{ancestor}(x, y)$ so that it is satisfied exactly when there exist x', X, i , and Y such that, on the one hand $\text{desc}(Y, X)$ and $\text{copy}_i(x, x') \wedge \text{repr}_{A_i, B_i}(x', X)$, and, on the other hand, either y is an original element and $Y = \{y\}$, or there exists y' and j such that $\text{copy}_j(y, y') \wedge \text{repr}_{A_j, B_j}(y', Y)$. ◀

4 Transducing modular decompositions

The set of modules of a directed graph is a specific example of a particular type of set system, a “weakly-partitive set system” (and a “partitive set system” in the case of an undirected graph). In this section, we first give a general $C_2\text{MSO}$ -transduction to obtain the canonical tree-like decomposition of a weakly-partitive set system from the set system itself. We then show how to obtain the modular decomposition of a graph via a $C_2\text{MSO}$ -transduction as an application.

4.1 Transducing weakly-partitive trees

A set system (U, \mathcal{S}) is *weakly-partitive* if for every two overlapping sets $X, Y \in \mathcal{S}$, the sets $X \cup Y$, $X \cap Y$, $X \setminus Y$, and $Y \setminus X$ belong to \mathcal{S} . It is *partitive* if, moreover, for every two overlapping sets $X, Y \in \mathcal{S}$, their symmetric difference, denoted $X \triangle Y$, also belongs to \mathcal{S} . By extension, a set family \mathcal{S} is called *weakly-partitive* or *partitive* whenever $(\bigcup \mathcal{S}, \mathcal{S})$ is a set system which is weakly-partitive or partitive, respectively (note these also requires that $\emptyset \notin \mathcal{S}$, $\bigcup \mathcal{S} \in \mathcal{S}$, and $\{a\} \in \mathcal{S}$ for every $a \in \bigcup \mathcal{S}$).

A member of a set system (U, \mathcal{S}) is said to be *strong* if it does not overlap any other set from \mathcal{S} . The sub-family $\mathcal{S}_!$ of strong sets of \mathcal{S} is thus laminar by definition. Hence, it induces a laminar tree T . By extension, we say that T is *induced by the set system* (U, \mathcal{S}) (or simply by \mathcal{S}). The next result extends Theorem 2, by showing that T can be C_2 MSO-transduced from \mathcal{S} .

► **Lemma 13.** *Let Σ be an extended vocabulary, including a set unary predicate name SET and not including the binary relational symbol ancestor. There exists a non-deterministic C_2 MSO-transduction τ such that, for each set system (U, \mathcal{S}) represented as the {SET}-structure \mathbb{S} and inducing the laminar tree T with $L(T) = U$, and for each Σ -structure \mathbb{A} with $\mathbb{S} \sqsubseteq \mathbb{A}$, $\tau(\mathbb{A})$ is non-empty and every output in $\tau(\mathbb{A})$ is equal to some {ancestor}-structure \mathbb{T} representing T .*

Proof. On the {SET}-structure \mathbb{S} , it is routine to define an MSO-formula $\phi_{\text{SET!}}(Z)$ that identifies those subsets $Z \subseteq U$ that are strong members of \mathcal{S} :

$$\phi_{\text{SET!}}(Z) := \text{SET}(Z) \wedge \forall X \left((\text{SET}(X) \wedge X \cap Z \neq \emptyset) \rightarrow (X \subseteq Z \vee Z \subseteq X) \right).$$

Hence, we can design an MSO-interpretation that outputs the $\Sigma \cup \{\text{SET!}\}$ -structure corresponding to \mathbb{A} equipped with the set unary predicate SET! that selects strong members of \mathcal{S} . Thus, up to renaming the predicates SET! and SET, by Theorem 2, we can produce, through a C_2 MSO-transduction, the $\Sigma \cup \{\text{ancestor}\}$ -structure $\mathbb{A} \sqcup \mathbb{T}$ where \mathbb{T} is the tree-structure modeling the laminar tree T induced by $\mathcal{S}_!$ with $L(T) = U$ (once the output obtained, the interpretation drops the set predicate SET! which is no longer needed). ◀

Clearly, the laminar tree T of a weakly-partitive set system (U, \mathcal{S}) does not characterize (U, \mathcal{S}) . However, as shown by the below theorem, a labeling of its inner nodes and a controlled partial ordering of its nodes are sufficient to characterize all the sets of \mathcal{S} . For Z a set equipped with a partial order $<$ and X a subset of Z , we say that X is a *<-interval* whenever $<$ defines a total order on X and for every $a, b \in X$ and every $c \in Z$, $a < c < b$ implies $c \in X$.

► **Theorem 14** ([7, 24]). *Let \mathcal{S} be a weakly-partitive family, $\mathcal{S}_!$ be its subfamily of strong sets, and T be the laminar tree it induces. There exists a total labeling function λ from the set $V(T) \setminus L(T)$ of inner nodes of T to the set {degenerate, prime, linear}, and, for each inner node $t \in \lambda^{-1}(\text{linear})$, a linear ordering $<_t$ of its children, such that every inner node having exactly two children is labeled by degenerate and the following two conditions are satisfied:*

- for each $X \in \mathcal{S} \setminus \mathcal{S}_!$, there exists $t \in V(T)$ and a subset \mathcal{C} of children of t such that $X = \bigcup_{c \in \mathcal{C}} L(T_c)$ and either $\lambda(t) = \text{linear}$ and \mathcal{C} is a $<_t$ -interval, or $\lambda(t) = \text{degenerate}$;
- conversely, for each inner node t and each non-empty subset \mathcal{C} of children of t , if either $\lambda(t) = \text{linear}$ and \mathcal{C} is a $<_t$ -interval, or $\lambda(t) = \text{degenerate}$, then $\bigcup_{c \in \mathcal{C}} L(T_c) \in \mathcal{S}$.

Furthermore, T and λ are uniquely determined from \mathcal{S} , and, for each inner node t of T labeled by linear, only two orders $<_t$ are possible, one being the inverse of the other (indeed, inverting an order $<$ does preserve the property of being a $<$ -interval). Hence, every weakly-partitive family \mathcal{S} is characterized by a labeled and partially-ordered tree $(T, \lambda, <)$ where T is the laminar tree induced by the subfamily $\mathcal{S}_!$ of strong sets of \mathcal{S} , $\lambda : V(T) \setminus L(T) \rightarrow \{\text{degenerate, prime, linear}\}$ is the labeling function, and $<$ is the partial order $\bigcup_{t \in \lambda^{-1}(\text{linear})} <_t$ over $V(T)$. As, up-to inverting some of the $<_t$ orders, $(T, \lambda, <)$ is unique, we abusively call it *the weakly-partitive tree induced by \mathcal{S}* . Conversely, a weakly-partitive tree characterizes the unique weakly-partitive set system which induced it.

We naturally model a weakly-partitive tree $(T, \lambda, <)$ of a partitive set system (U, \mathcal{S}) by the {ancestor, degenerate, betweenness}-structure \mathbb{T} of universe $U_{\mathbb{T}} = V(T)$ such that $\langle V(T), \text{ancestor}_{\mathbb{T}} \rangle \sqsubseteq \mathbb{T}$ models T with $L(T) = U$, $\text{degenerate}_{\mathbb{T}}$ is a unary relation which selects

the inner nodes of T of label degenerate, *i.e.*, $\text{degenerate}_{\mathbb{T}} = \lambda^{-1}(\text{degenerate})$, and $\text{betweeness}_{\mathbb{T}}$ is a ternary relation selecting triples (x, y, z) satisfying $x < y < z$ or $z < y < x$. (Although it is possible, through a non-deterministic MSO-transduction, to define $<$ from $\text{betweeness}_{\mathbb{T}}$, the use of $\text{betweeness}_{\mathbb{T}}$ rather than $<_{\mathbb{T}}$ ensures unicity of the output weakly-partitive tree.) The inner nodes of T that are labeled by **linear** can be recovered through an MSO-formula as those inner nodes whose children are related by **betweeness**. The inner nodes labeled by **prime** can be recovered through an MSO-formula as those inner nodes that are labeled neither by **degenerate** nor by **linear**. Using Theorem 14, it is routine to design an MSO-transduction which takes as input a weakly-partitive tree and outputs the weakly-partitive set system which induced it. The inverse C_2 MSO-transduction is the purpose of the next result. The proof is omitted due to space constraints.

► **Theorem 15.** *There exists a non-deterministic C_2 MSO-transduction τ such that, for every weakly-partitive set system (U, \mathcal{S}) represented as the $\{\text{SET}\}$ -structure \mathbb{S} and inducing the weakly-partitive tree $(T, \lambda, <)$ represented as the $\{\text{ancestor, degenerate, betweeness}\}$ -structure \mathbb{T} , we have $\mathbb{T} \in \tau(\mathbb{S})$ and every output in $\tau(\mathbb{S})$ is a weakly-partitive tree of (U, \mathcal{S}) .*

If \mathcal{S} is partitive then the weakly-partitive tree it induces enjoys a simple form, and is truly unique. Indeed, the label **linear** and, thus, the partial order $<$, are not needed.

► **Theorem 16** ([7]). *Let \mathcal{S} be a weakly-partitive family and $(T, \lambda, <)$ be the weakly-partitive tree it induces. If \mathcal{S} is partitive, then $\lambda^{-1}(\text{linear}) = \emptyset$ and $<$ is empty.*

Hence, in case of a partitive set systems (U, \mathcal{S}) , we can consider the simpler object (T, λ) , called *the partitive tree induced by \mathcal{S}* (or *the partitive tree of (U, \mathcal{S})*) in which λ maps $V(T) \setminus L(T)$ to $\{\text{degenerate, prime}\}$. As a direct consequence of Theorems 15 and 16, we can produce, through a C_2 MSO-transduction, the partitive tree induced by a partitive set system and naturally modeled by an $\{\text{ancestor, degenerate}\}$ -structure.

► **Corollary 17.** *There exists a non-deterministic C_2 MSO-transduction τ such that, for each partitive set system (U, \mathcal{S}) represented as the $\{\text{SET}\}$ -structure \mathbb{S} and inducing the partitive tree (T, λ) represented as the $\{\text{ancestor, degenerate}\}$ -structure \mathbb{T} , we have $\mathbb{T} \in \tau(\mathbb{S})$ and every output in $\tau(\mathbb{S})$ is a partitive tree of (U, \mathcal{S}) .*

4.2 Application to modular decomposition

Let G be a directed graph and let $M \subseteq V(G)$. We say that M is a *module* (of G) if for every $u \notin M$ and every $v, w \in M$, $uv \in E(G) \iff uw \in E(G)$ and $vu \in E(G) \iff wu \in E(G)$. Clearly, the empty set, $V(G)$, and all the singletons $\{x\}$ for $x \in V(G)$ are modules; they are called *the trivial modules* of G . We say a non-empty module M is *maximal* if it is not properly contained in any non-trivial module. Let M and M' be two disjoint non-empty modules of G . Considering the edges that go from M to M' , namely edges from the set $(M \times M') \cap E(G)$, we have two possibilities: either it is empty, or it is equal to $M \times M'$. We write $M \not\rightarrow M'$ in the former case and $M \rightarrow M'$ in the latter. (It is of course possible to have both $M \rightarrow M'$ and $M' \rightarrow M$.) A *modular partition* of G is a partition $\mathcal{P} = \{M_1, \dots, M_\ell\}$ of $V(G)$ such that every M_i is a non-empty module. A modular partition $\mathcal{P} = \{M_1, \dots, M_\ell\}$ is called *maximal* if it is non-trivial and every M_i is strong and maximal. Note that every graph has exactly one maximal modular partition.

A *modular decomposition* of G is a rooted tree T in which the leaves are the vertices of G , and for each inner node $t \in T$, t has at least two children and the set $L(T_t)$ is a module of G . In a modular decomposition T of G , for each inner node $t \in V(T)$ with children c_1, \dots, c_r ,

the family $\mathcal{P}_t = \{L(T_{c_1}), \dots, L(T_{c_r})\}$ is a modular partition of $G[V(T_t)]$. When each such partition is maximal, the decomposition is unique and it is called *the maximal modular decomposition* of G . The maximal modular decomposition T of G alone is not sufficient to characterize G . However, enriching T with, for each inner node t with children c_1, \dots, c_j , the information of which pair of modules $(L(T_{c_i}), L(T_{c_j}))$ is such that $L(T_{c_i}) \rightarrow L(T_{c_j})$, yields a unique canonical representation of G . Formally, the *enriched modular decomposition* of G is the pair (T, F) where T is the maximal modular decomposition of G (with $L(T) = V(G)$) and $F \subset V(T) \times V(T)$ is a binary relation, that relates a pair (s, t) of nodes of T , denoted $st \in F$, exactly when s and t are siblings and $L(T_s) \rightarrow L(T_t)$. The elements of F are called *m-edges*.

It should be mentioned that the family of all non-empty modules of G is known to be weakly-partitive (or even partitive when G is undirected). In particular, the maximal modular decomposition T of G is the laminar tree induced by the family of strong modules. Hence Theorem 15 could be used to produce a partially-ordered and labeled tree which displays all the modules of G . However, this weakly-partitive tree is not sufficient for being able to recover the graph G from it. We now prove how to obtain the enriched modular decomposition of G through a C_2 MSO-transduction.

To model enriched modular decompositions as relational structures we use the relational vocabulary $\{\text{ancestor}, \text{m-edge}\}$ where *ancestor* and *m-edge* are two binary relation names. An enriched modular decomposition (T, F) of a graph G is modeled by the $\{\text{ancestor}, \text{m-edge}\}$ -structure \mathbb{M} with universe $U_{\mathbb{M}} = V(T)$, $\text{ancestor}_{\mathbb{M}}$ being the set of pairs (s, t) for which s is an ancestor of t in T , and $\text{m-edge}_{\mathbb{M}}$ being the set of all pairs (s, t) such that $st \in F$ (in particular, s and t are siblings in T). We use Lemma 13 in order to transduce the maximal modular decomposition of a graph.

► **Theorem 18.** *There exists a non-deterministic C_2 MSO-transduction τ such that for every directed graph G represented as the $\{\text{edge}\}$ -structure \mathbb{G} , $\tau(\mathbb{G})$ is non-empty and every output in $\tau(\mathbb{G})$ is equal to some $\{\text{ancestor}, \text{m-edge}\}$ -structure \mathbb{M} representing the enriched modular decomposition (T, F) of G .*

Proof. Let G be a graph represented by the $\{\text{edge}\}$ -structure \mathbb{G} . Several objects are associated to G , and each of them can be described by a structure:

- let \mathcal{M} be the family of non-empty modules of G and let \mathbb{S} be the $\{\text{SET}\}$ -structure modeling the weakly-partitive set system $(V(G), \mathcal{M})$ with $U_{\mathbb{S}} = V(G)$;
- let T be the laminar tree induced by the weakly-partitive family \mathcal{M} and let \mathbb{T} be the $\{\text{ancestor}\}$ -structure modeling it with $U_{\mathbb{T}} = V(T)$ and $L(T) = V(G) \subset U_{\mathbb{T}}$;
- let F be the m-edge relation, namely the subset of $V(T) \times V(T)$ such that (T, F) is the maximal modular decomposition of G , and let \mathbb{M} be the $\{\text{ancestor}, \text{m-edge}\}$ -structure modeling it with $\mathbb{T} \sqsubset \mathbb{M}$ and $U_{\mathbb{T}} = U_{\mathbb{M}}$.

Our C_2 MSO-transduction is obtained by composing the three following transductions:

- τ_1 : an MSO-interpretation which outputs the $\{\text{edge}, \text{SET}\}$ -structure $\mathbb{G} \sqcup \mathbb{S}$ from \mathbb{G} ;
- τ_2 : the non-deterministic C_2 MSO-transduction given by Lemma 13 which produces the $\{\text{edge}, \text{SET}, \text{ancestor}\}$ -structure $\mathbb{G} \sqcup \mathbb{S} \sqcup \mathbb{T}$ from $\mathbb{G} \sqcup \mathbb{S}$;
- τ_3 : an MSO-interpretation which outputs the $\{\text{ancestor}, \text{m-edge}\}$ -structure \mathbb{M} from $\mathbb{G} \sqcup \mathbb{S} \sqcup \mathbb{T}$.

In order to define τ_1 it is sufficient to observe that there exists an MSO-formula $\phi_{\text{SET}}(Z)$ with one monadic free-variable, which is satisfied exactly when Z is a non-empty module of G . Then, since τ_2 is given by Lemma 13, it only remains to define τ_3 . Given an inner node t , we can select, within MSO, the set $L(T_t)$ of leaves of the subtree rooted in t . We

thus assume a function `leafset`, with one first-order free-variable which returns the set of leaves of the subtree rooted at the given node. Equipped with this function, we can define the MSO-formula $\phi_{\text{m-edge}}$ which selects pairs (s, r) of siblings such that $sr \in F$. Remember that this happen exactly when there exist $u \in L(T_s)$ and $v \in L(T_r)$ such that $uv \in E(G)$. Hence, $\phi_{\text{m-edge}}$ could be defined as:

$$\phi_{\text{m-edge}}(s, r) := s \neq r \wedge \exists t (\text{parent}(t, s) \wedge \text{parent}(t, r)) \wedge \exists x \exists y (x \in \text{leafset}(s) \wedge y \in \text{leafset}(r) \wedge \text{edge}(x, y)).$$

Once defined, the MSO-interpretation τ_3 simply drops all non-necessary relations and predicates (namely `edge` and `SET`) and keeps only the ancestor and `m-edge` relations. ◀

Notice that it is routine to design a deterministic MSO-transduction which, given an $\{\text{ancestor}, \text{m-edge}\}$ -structure \mathbb{M} representing an enriched modular decomposition of some directed graph G , produces the $\{\text{edge}\}$ -structure \mathbb{G} representing G .

Cographs

Let G be a graph, (T, F) be its modular decomposition, and $(T, \lambda, <)$ be the weakly-partitive tree induced by the (weakly-partitive) family of its modules. Let t be an inner node of T , let \mathcal{C} be its set of children, and let C be the graph $(V(T) \setminus L(T), F)[\mathcal{C}]$ induced by F on the set of children of t . It can be checked that, if $\lambda(t) = \text{degenerate}$ then C is either a clique or an independant, and if $\lambda(t) = \text{linear}$ then C is a *tournament consistent with $<_t$* (i.e., for every $x, y \in \mathcal{C}$, xy is an edge of C if and only if $x <_t y$) or with the inverse of $<_t$. In the former case, we can refine the `degenerate` label into `series` and `parallel` labels, thus expressing that C is a clique or an independant, respectively. In the latter case, up-to reversing $<_t$, we can ensure that C is a tournament consistent with $<_t$. This yields a refined weakly-partitive tree $(T, \gamma, <)$, where γ maps inner nodes to $\{\text{series}, \text{parallel}, \text{prime}, \text{linear}\}$ and $<$ is the order $\bigcup_{t \in \gamma^{-1}(\text{linear})} <_t$ which ensures that tournaments are consistent with the corresponding $<_t$. Notice that this labeled and partially-ordered tree is now uniquely determined from G . Moreover, edges from F that connect children of a node not labeled by `prime` can be recovered from the so-refined weakly-partitive tree. In particular, if no nodes of T is labeled by `prime`, (T, F) and thus G is fully characterized by $(T, \gamma, <)$. Graphs for which this property holds are known as *directed cographs*, and can be described by the refined weakly-partitive tree, called *cotree*, explained above and formalized in the following statement.

► **Theorem 19.** *Let G be a directed cograph and let T be the laminar tree induced by the family of its strong modules. There exists a unique total labeling λ from the set $V(T) \setminus L(T)$ of inner nodes of T to the set $\{\text{series}, \text{parallel}, \text{linear}\}$ of labels, and, for each inner node $t \in \lambda^{-1}(\text{linear})$, a unique linear ordering $<_t$ of its children, such that every inner node having exactly two children is labeled `series` or `parallel`, and the following condition is satisfied:*

- *for every two leaves x and y of T , denoting by t their least common ancestor, xy is an edge of G if and only if either t is labeled by `linear` and $x <_t y$, or t is labeled by `series`.*

We naturally model cotrees as $\{\text{ancestor}, \text{series}, \varepsilon\text{-ord}\}$ -structures as follow. A cotree $(T, \gamma, <)$ is modeled by \mathbb{C} where $U_{\mathbb{C}} = V(T)$, $\text{series}_{\mathbb{C}} = \gamma^{-1}(\text{series})$, and $\varepsilon\text{-ord}_{\mathbb{C}} = \{(x, y) \mid x < y\}$. The nodes that are labeled by `linear` could be recovered as those inner nodes whose children are related by $\varepsilon\text{-ord}$, while the nodes that are labeled by `parallel` could be recovered as those inner nodes which are labeled neither by `series` nor by `linear`. Based on Theorem 19 and as a consequence of Theorem 18, we can design a C_2MSO -transduction which produces the cotree of a cograph G from G .

► **Corollary 20.** *There exists a non-deterministic C_2 MISO-transduction τ such that, for each cograph G modeled by the $\{\text{edge}\}$ -structure \mathbb{G} , $\tau(\mathbb{G})$ is non-empty and every output in $\tau(\mathbb{G})$ is equal to some $\{\text{ancestor, series, } \varepsilon\text{-ord}\}$ -structure \mathbb{C} representing the cotree $(T, \gamma, <)$ of G .*

5 Conclusion

We provide transductions for obtaining tree-like graph decompositions such as modular decompositions, cotrees, split decompositions and bi-join decompositions from a graph using CMSO. This improves upon results of Courcelle [10, 12] who gave such transductions for ordered graphs. In a more general settings, we also obtain CMSO-transductions outputting weakly-partitive and weakly-bipartitive trees of weakly-partitive and weakly-bipartitive systems (Items 6 and 7 of Theorem 1). It is worth mentioning that the latter transduction can be also used to CMSO-transduce canonical decompositions of other structures such as Tutte’s decomposition of matroids or generally split-decompositions of submodular functions [16] or modular decompositions of 2-structures [18] or of hypergraphs [21]. As shown by the application given in [12] for transducing Whitney’s isomorphism class of a graph, a line of research is to more investigate which structures can be CMSO-transduced from a graph or a set system by using the transductions from Theorem 1. Also, naturally, the question arises whether counting is necessary or whether MSO is sufficient to transduce such decompositions. Furthermore, our results include that transducing rank decompositions for graphs of rank-width 1 is possible using CMSO. But it is not known whether rank-decompositions can be transduced in general using some suitable extension of MSO. Nevertheless, a corollary of our results is that CMSO-transducing rank-decompositions can be now reduced to consider CMSO-transducing rank-decompositions of prime graphs wrt either modular decomposition or split-decomposition as if a graph has rank-width at least 2, then its rank-width is equal to the rank-width of its prime induced graphs wrt either modular or split decomposition. Thus, our results imply that, for any fixed k , there is a CMSO-transduction that computes a clique-decomposition of small width for any graph belonging to a graph class whose prime graphs have sizes bounded by k or prime graphs have linear clique-width bounded by k , e.g., many H -free graphs have small prime graphs or have small linear clique-width (see for instance [5] or [23] for prominent such examples).

References

- 1 Hans L Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234, 1993. doi:10.1145/167088.167161.
- 2 Mikołaj Bojańczyk. The category of mso transductions. *CoRR*, May 2023. arXiv:2305.18039v1.
- 3 Mikołaj Bojańczyk, Martin Grohe, and Michał Pilipczuk. Definable decompositions for graphs of bounded linear cliquewidth. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: <https://lmcs.episciences.org/7125>.
- 4 Mikołaj Bojańczyk and Michał Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS ’16, New York, NY, USA, July 5-8, 2016*, pages 407–416. ACM, 2016. doi:10.1145/2933575.2934508.
- 5 Andreas Brandstädt, Feodor F. Dragan, Hoàng-Oanh Le, and Raffaele Mosca. New graph classes of bounded clique-width. *Theory Comput. Syst.*, 38(5):623–645, 2005. doi:10.1007/S00224-004-1154-6.

- 6 Rutger Campbell, Bruno Guillon, Mamadou Moustapha Kanté, Eun Jung Kim, and Noleen Köhler. CMSO-transducing tree-like graph decompositions, 2024. doi:10.48550/arXiv.2412.04970.
- 7 Michel Chein, Michel Habib, and Marie-Catherine Maurer. Partitive hypergraphs. *Discrete mathematics*, 37(1):35–50, 1981. doi:10.1016/0012-365X(81)90138-2.
- 8 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
- 9 Bruno Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991. doi:10.1016/0304-3975(91)90387-H.
- 10 Bruno Courcelle. The monadic second-order logic of graphs X: linear orderings. *Theor. Comput. Sci.*, 160(1&2):87–143, 1996. doi:10.1016/0304-3975(95)00083-6.
- 11 Bruno Courcelle. The monadic second-order logic of graphs XI: hierarchical decompositions of connected graphs. *Theor. Comput. Sci.*, 224(1-2):35–58, 1999. doi:10.1016/S0304-3975(98)00306-5.
- 12 Bruno Courcelle. The monadic second-order logic of graphs XVI: Canonical graph decompositions. *Logical Methods in Computer Science*, 2, 2006. doi:10.2168/LMCS-2(2:2)2006.
- 13 Bruno Courcelle. Canonical graph decompositions. Talk, 2012. Available at <https://www.labri.fr/perso/courcell/Conferences/ExpoCanDecsJuin2012.pdf>.
- 14 Bruno Courcelle. The atomic decomposition of strongly connected graphs. Technical report, Université de Bordeaux, 2013. Available at <https://www.labri.fr/perso/courcell/ArticlesEnCours/AtomicDecSubmitted.pdf>.
- 15 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic*. Cambridge University Press, 2009. doi:10.1017/cbo9780511977619.
- 16 William H Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):214–228, 1982.
- 17 Fabien de Montgolfier. *Décomposition modulaire des graphes. Théorie, extension et algorithmes*. Phd thesis, Université Montpellier II, LIRMM, 2003.
- 18 Andrzej Ehrenfeucht, Tero Harju, and Grzegorz Rozenberg. *The Theory of 2-Structures – A Framework for Decomposition and Transformation of Graphs*. World Scientific, 1999. URL: <http://www.worldscibooks.com/mathematics/4197.html>.
- 19 Daryl Funk, Dillon Mayhew, and Mike Newman. Tree automata and pigeonhole classes of matroids: I. *Algorithmica*, 84(7):1795–1834, 2022. doi:10.1007/S00453-022-00939-7.
- 20 Tobias Ganzow and Sasha Rubin. Order-invariant MSO is stronger than counting MSO in the finite. In Susanne Albers and Pascal Weil, editors, *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings*, volume 1 of *LIPICs*, pages 313–324. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2008. doi:10.4230/LIPICs.STACS.2008.1353.
- 21 Michel Habib, Fabien de Montgolfier, Lalla Mouatadid, and Mengchuan Zou. A general algorithmic scheme for combinatorial decompositions with application to modular decompositions of hypergraphs. *Theor. Comput. Sci.*, 923:56–73, 2022. doi:10.1016/J.TCS.2022.04.052.
- 22 Petr Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Comb. Theory B*, 96(3):325–351, 2006. doi:10.1016/J.JCTB.2005.08.005.
- 23 Michaël Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theor. Comput. Sci.*, 377(1-3):260–267, 2007. doi:10.1016/J.TCS.2007.03.043.
- 24 Michaël Rao. *Décompositions de graphes et algorithmes efficaces*. Phd thesis, Université Paul Verlaine – Metz, 2006.
- 25 Yann Strozecki. Monadic second-order model-checking on decomposable matroids. *Discret. Appl. Math.*, 159(10):1022–1039, 2011. doi:10.1016/J.DAM.2011.02.005.

How to Play the Accordion: Uniformity and the (Non-)Conservativity of the Linear Approximation of the λ -Calculus

Rémy Cerda   

Aix-Marseille Université, CNRS, I2M, France
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France

Lionel Vaux Auclair   

Aix-Marseille Université, CNRS, I2M, France

Abstract

Twenty years after its introduction by Ehrhard and Regnier, differentiation in λ -calculus and in linear logic is now a celebrated tool. In particular, it allows to write the Taylor formula in various λ -calculi, hence providing a theory of linear approximations for these calculi. In the standard λ -calculus, this linear approximation is expressed by results stating that the (possibly) infinitary β -reduction of λ -terms is simulated by the reduction of their Taylor expansion: in terms of rewriting systems, the resource reduction (operating on Taylor approximants) is an extension of the β -reduction.

In this paper, we address the converse property, conservativity: are there reductions of the Taylor approximants that do not arise from an actual β -reduction of the approximated term? We show that if we restrict the setting to finite terms and β -reduction sequences, then the linear approximation is conservative. However, as soon as one allows infinitary reduction sequences this property is broken. We design a counter-example, the Accordion. Then we show how restricting the reduction of the Taylor approximants allows to build a conservative extension of the β -reduction preserving good simulation properties. This restriction relies on uniformity, a property that was already at the core of Ehrhard and Regnier's pioneering work.

2012 ACM Subject Classification Theory of computation \rightarrow Rewrite systems; Theory of computation \rightarrow Lambda calculus; Theory of computation \rightarrow Proof theory; Theory of computation \rightarrow Linear logic

Keywords and phrases program approximation, quantitative semantics, lambda-calculus, linear approximation, Taylor expansion, conservativity

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.23

Related Version A short abstract of a preliminary version of this work has been presented at the 7th International Workshop on Trends in Linear Logic and its Applications (TLLA'23). Most of this work has also appeared in the first author's PhD thesis, with the noticeable difference that Theorem 27 was only proved in a qualitative setting.

Previous Version (Short abstract presented at TLTA '23): <https://lipn.univ-paris13.fr/TLTA/2023/abstracts/05-Final.pdf>

Previous Version (PhD thesis version, see Chapter 5): <https://hal.science/te1-04664728> [7]

Funding *Rémy Cerda*: Partially supported by the French ANR project *RECIPROG* (ANR-21-CE48-019).

Lionel Vaux Auclair: Partially supported by the French ANR projects *LambdaComb* (ANR-21-CE48-0017), *RECIPROG* (ANR-21-CE48-019), and *PPS* (ANR-19-CE48-0014).

1 Introduction

The traditional approach to program approximation in a functional setting consists in describing the total information that a (potentially non-terminating) program can produce by the supremum of the finite pieces of information it can produce in finite time. This



© Rémy Cerda and Lionel Vaux Auclair;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 23; pp. 23:1–23:21



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



continuous approximation is tightly related to the Scott semantics of λ -calculi [29]: the Böhm tree of a term (or equivalently its semantics) is the limit of the approximants produced by hereditary head reduction [21, 32, 3].

More recently, Ehrhard and Regnier introduced the differential λ -calculus and differential linear logic [16, 17], following ideas rooted in the semantics of linear logic [20, 13, 14]. This suggested the renewed approach of *linear* approximation of functional programs. In this setting, a program (i.e. a λ -term) is approximated by multilinear (or “polynomial”) programs, obtained by iterated differentiation at zero. Using this differential formalism, the Taylor formula yields a weighted sum $\mathcal{T}(M)$ of all multilinear approximants of a λ -term M , producing the same total information as M , *via* normalization. More precisely, Ehrhard and Regnier’s “commutation” theorem [19, 18] ensure that the normal form of the Taylor expansion of M is the Taylor expansion of the Böhm tree of M :

$$\text{nf}(\mathcal{T}(M)) = \mathcal{T}(\text{BT}(M)) \tag{1}$$

(and a Böhm tree is uniquely determined by its Taylor expansion). This approach subsumes the previous one [2]. In addition, it allows for characterising quantitative properties of programs (e.g. complexity [11]), which is a key benefit of linearity. This approximation technique has been fruitfully applied to many languages, richer than the plain λ -calculus: nondeterministic [31], probabilistic [10], extensional [5], call-by-value [23], and call-by-push-value [15, 9] calculi, as well as for Parigot’s $\lambda\mu$ -calculus [1]. The interplay of the operational and Taylor approximations also suggests a broader notion of a approximation of a computation process [26, 12].

Another benefit of linear approximation is that it can approximate not only β -normalisation (the information ultimately produced by a program) but β -reduction (the “information flow” along program execution). In particular, Equation (1) can be refined into

$$M \longrightarrow_{\beta}^* N \Rightarrow \mathcal{T}(M) \longrightarrow_{\text{r}} \mathcal{T}(N), \tag{2}$$

where $\longrightarrow_{\text{r}}$ denotes the so-called “resource” reduction acting linearly on approximants. As highlighted by our previous work [8, 7], this can even be extended to

$$M \longrightarrow_{\beta}^{\infty} N \Rightarrow \mathcal{T}(M) \longrightarrow_{\text{r}} \mathcal{T}(N) \tag{3}$$

if one extends the λ -calculus with infinite λ -terms and an infinitary closure of the β -reduction, which is a way to encompass infinite computations and their limits [22].

This paper is interested in the converse of Equations (2) and (3): is the linear approximation of the λ -calculus conservative? In other terms, we ask whether every resource reduction from $\mathcal{T}(M)$ to $\mathcal{T}(N)$ corresponds to a β -reduction sequence from M to N .

We show that the finite β -reduction of finite λ -terms is conservatively approximated (Section 3), but we are able to design a counter-example to conservativity (the Accordion **A**) as soon as we want to approximate infinitary β -reductions (Section 4). However, we introduce a *uniform* linear approximation allowing for the same good properties as the standard one, while enjoying conservativity (Section 5).

2 Preliminaries

In this section, we briefly recall the linear approximation of the λ -calculus, following its refined presentation in [7]. We first recall the definition of the λ -calculus, as well as its “001” infinitary extension: this is the version of the infinitary λ -calculus that fits the formalism

of both continuous and linear approximations as they are usually presented (Section 2.1). Then we present the resource λ -calculus, i.e. a linear variant of the λ -calculus (there are no duplications or erasures of subterms during the reduction) enjoying strong confluence and normalisation properties (Section 2.2). Finally, the linear approximation relies on the Taylor expansion, that maps a λ -term to a sum of resource terms, in a way such that the reduction of λ -terms is simulated by the reduction of the resource approximants (Section 2.3).

2.1 Finite and infinitary λ -calculi

We give a brief presentation of the 001-infinitary λ -calculus. A more detailed exposition and a general account of infinitary λ -calculi can be found in [7, 4]. From now on, we fix a countable set \mathcal{V} of variables.

► **Definition 1.** *The set Λ of (finite) λ -terms is the set \mathcal{X} defined by the inductive rules:*

$$\frac{x \in \mathcal{V}}{x \in \mathcal{X}} \text{ (}\mathcal{V}\text{)} \quad \frac{x \in \mathcal{V} \quad M \in \mathcal{X}}{\lambda x.M \in \mathcal{X}} \text{ (}\lambda\text{)} \quad \frac{M \in \mathcal{X} \quad N \in \mathcal{X}}{(M)N \in \mathcal{X}} \text{ (}\@)\text{)}$$

The set Λ^{001} of **001-infinitary λ -terms** is the set \mathcal{X} defined by the rules (\mathcal{V}) , (λ) , and:

$$\frac{M \in \mathcal{X} \quad \triangleright N \in \mathcal{X}}{(M)N \in \mathcal{X}} \text{ (}\@^{001}\text{)} \quad \frac{N \in \mathcal{X}}{\triangleright N \in \mathcal{X}} \text{ (}\triangleright\text{)}$$

where the rule (\triangleright) is treated coinductively: infinite derivations are allowed provided each infinite branch crosses infinitely often this coinductive rule.

This means that Λ^{001} contains the infinitary λ -terms whose syntax tree contains only infinite branches entering infinitely often the argument side of an application.

Notice that we use Krivine's notation for applications [25], i.e. we parenthesise functions instead of arguments. We abbreviate the application of a term to successive arguments $(\dots((M)N_1)\dots)N_k$ as $(M)N_1 \dots N_k$, which is obtained by nesting applications on the left: this allows to use parentheses more sparingly, which will be a great relief later on. By contrast, $(M_1)(M_2)\dots(M_k)N$ is obtained by nesting applications on the right. A typical example of a term in Λ^{001} is $(x)^\omega := (x)(x)(x)\dots$. Observe also that there is an immediate inclusion $\Lambda \subseteq \Lambda^{001}$. On the contrary, neither $\lambda x_0.\lambda x_1.\lambda x_2.\dots$ nor $((\dots)x_2)x_1)x_0$ are allowed in Λ^{001} .

In practice we only consider infinitary terms having finitely many free variables, which allows us to consider them up to α -equivalence (i.e. renaming of bound variables) – as one usually does when dealing with λ -terms, and as we will do implicitly in all this paper. This enables us to define capture-avoiding substitution in the usual way, and we denote by $M[N/x]$ the term obtained by substituting N to x in M . We refer to [6] for a more careful and detailed presentation. These sets come equipped with the following dynamics.

► **Definition 2.** *The relation $\longrightarrow_\beta \subset \Lambda^{001} \times \Lambda^{001}$ of β -reduction is defined by the rules:*

$$\frac{}{(\lambda x.M)N \longrightarrow_\beta M[N/x]} \text{ (}\beta\text{)} \quad \frac{P \longrightarrow_\beta P'}{\lambda x.P \longrightarrow_\beta \lambda x.P'} \text{ (}\lambda_\beta\text{)}$$

$$\frac{P \longrightarrow_\beta P'}{(P)Q \longrightarrow_\beta (P')Q} \text{ (}\@l_\beta\text{)} \quad \frac{Q \longrightarrow_\beta Q'}{(P)Q \longrightarrow_\beta (P)Q'} \text{ (}\@r_\beta\text{)}$$

► **Definition 3.** The relation $\longrightarrow_{\beta}^{001} \subset \Lambda^{001} \times \Lambda^{001}$ of **001-infinitary β -reduction** is defined by the rules:

$$\frac{M \longrightarrow_{\beta}^* x}{M \longrightarrow_{\beta}^{001} x} (\mathcal{V}_{\beta}^{001}) \quad \frac{M \longrightarrow_{\beta}^* \lambda x.P \quad P \longrightarrow_{\beta}^{001} P'}{M \longrightarrow_{\beta}^{001} \lambda x.P'} (\lambda_{\beta}^{001})$$

$$\frac{M \longrightarrow_{\beta}^* (P)Q \quad P \longrightarrow_{\beta}^{001} P' \quad \triangleright Q \longrightarrow_{\beta}^{001} Q'}{M \longrightarrow_{\beta}^{001} (P')Q'} (\@_{\beta}^{001}) \quad \frac{Q \longrightarrow_{\beta}^{001} Q'}{\triangleright Q \longrightarrow_{\beta}^{001} Q'} (\triangleright)$$

where $\longrightarrow_{\beta}^*$ denotes the reflexive-transitive closure of \longrightarrow_{β} .

Infinitary β -reduction can be understood as allowing an infinite number of β -reduction steps, as long as the β -redexes are fired inside increasingly nested arguments of applications. This is formalised in the following result:

► **Theorem 4 (stratification).** Given $M, N \in \Lambda^{001}$, there is a reduction $M \longrightarrow_{\beta}^{001} N$ iff there exists a sequence of terms $(M_d) \in (\Lambda^{001})^{\mathbb{N}}$ such that for all $d \in \mathbb{N}$,

$$M = M_0 \longrightarrow_{\beta \geq 0}^* M_1 \longrightarrow_{\beta \geq 1}^* M_2 \longrightarrow_{\beta \geq 2}^* \dots \longrightarrow_{\beta \geq d-1}^* M_d \longrightarrow_{\beta \geq d}^{001} N,$$

where $\longrightarrow_{\beta \geq d}^*$ and $\longrightarrow_{\beta \geq d}^{001}$ denote β -reductions occurring inside (at least) d nested arguments of applications. Formally, **β -reduction at minimum depth d** is defined by:

$$\frac{M \longrightarrow_{\beta} M'}{M \longrightarrow_{\beta \geq 0} M'} (\mathcal{V}_{\beta \geq 0}) \quad \frac{P \longrightarrow_{\beta \geq d+1} P'}{\lambda x.P \longrightarrow_{\beta \geq d+1} \lambda x.P'} (\lambda_{\beta \geq d+1})$$

$$\frac{P \longrightarrow_{\beta \geq d+1} P'}{(P)Q \longrightarrow_{\beta \geq d+1} (P')Q} (\@l_{\beta \geq d+1}) \quad \frac{Q \longrightarrow_{\beta \geq d} Q'}{(P)Q \longrightarrow_{\beta \geq d+1} (P)Q'} (\@r_{\beta \geq d+1})$$

and **001-infinitary β -reduction at minimum depth d** is defined by:

$$\frac{M \longrightarrow_{\beta}^{001} M'}{M \longrightarrow_{\beta \geq 0}^{001} M'} (\mathcal{V}_{\beta \geq 0}^{001}) \quad \frac{}{x \longrightarrow_{\beta \geq d+1}^{001} x} (\mathcal{V}_{\beta \geq d+1}^{001})$$

$$\frac{P \longrightarrow_{\beta \geq d+1}^{001} P'}{\lambda x.P \longrightarrow_{\beta \geq d+1}^{001} \lambda x.P'} (\lambda_{\beta \geq d+1}^{001}) \quad \frac{P \longrightarrow_{\beta \geq d+1}^{001} P' \quad Q \longrightarrow_{\beta \geq d}^{001} Q'}{(P)Q \longrightarrow_{\beta \geq d+1}^{001} (P')Q} (\@_{\beta \geq d+1}^{001}).$$

A typical (and even motivating) example of an infinitary β -reduction involves the fix-point combinator $Y := \lambda f.(\lambda x.(f)(x)x)\lambda x.(f)(x)x$. It consists in the reduction $(Y)M \longrightarrow_{\beta}^{001} (M)^{\omega}$ corresponding to the sequence $(Y)M \longrightarrow_{\beta \geq 0}^* (M)(Y)M \longrightarrow_{\beta \geq 1}^* (M)(M)(Y)M \longrightarrow_{\beta \geq 2}^* \dots$. On the contrary, the infinite reduction sequence $\Omega \longrightarrow_{\beta} \Omega \longrightarrow_{\beta} \Omega \longrightarrow_{\beta} \dots$, where $\Omega := (\lambda x.(x)x)\lambda x.(x)x$, does not give rise to a 001-infinitary reduction because the redexes are fired at top-level all the way. On the other hand, each finite reduction sequence $\Omega \longrightarrow_{\beta}^* \Omega$ induces a reduction $\Omega \longrightarrow_{\beta}^{001} \Omega$, but only because $\longrightarrow_{\beta}^{001}$ contains $\longrightarrow_{\beta}^*$ (see [8], Lemma 2.13).

2.2 The resource λ -calculus

The resource λ -calculus is the target language of the linear approximation of the λ -calculus. We recall its construction, and we refer to [31, 7] for more details. The main intuition behind this calculus is that arguments become *finite multisets*, and that $(\lambda x.s)[t_1, \dots, t_n]$ will reduce to a term obtained by substituting *linearly* one t_i for each occurrence of x in s . The different matchings of the t_i 's and the occurrences of x are superposed by a sum operator; if a wrong number of t_i 's is provided, the term collapses to the empty sum.

Given a set \mathcal{X} , we denote by $!\mathcal{X}$ the set of finite multisets of elements of \mathcal{X} . A multiset is denoted by $\bar{x} = [x_1, \dots, x_n]$, with its elements in an arbitrary order. Multiset union is denoted multiplicatively, by $\bar{x} \cdot \bar{y}$. Accordingly, the empty multiset is denoted by 1. We may also write $[x_1^{k_1}, \dots, x_m^{k_m}]$ to indicate multiplicities: this is the same as $[x_1]^{k_1} \cdot \dots \cdot [x_m]^{k_m}$.

► **Definition 5.** The set Λ_r of **resource terms** is defined by the rules:

$$\frac{x \in \mathcal{V}}{x \in \Lambda_r} (\mathcal{V}) \quad \frac{x \in \mathcal{V} \quad s \in \Lambda_r}{\lambda x.s \in \Lambda_r} (\lambda) \quad \frac{s \in \Lambda_r \quad \bar{t} \in !\Lambda_r}{(s)\bar{t} \in \Lambda_r} (@!)$$

and is implicitly quotiented by α -equivalence. Multisets in $!\Lambda_r$ are called **resource monomials**. To denote indistinctly Λ_r or $!\Lambda_r$, we write $(!)\Lambda_r$.

Given a semiring \mathbb{S} and a set \mathcal{X} , we denote by $\mathbb{S}^{\mathcal{X}}$ the set of possibly infinite linear combinations of elements of \mathcal{X} with coefficients in \mathbb{S} , considered as formal weighted sums. Given a sum $\mathbf{S} \in \mathbb{S}^{\mathcal{X}}$, its support $|\mathbf{S}|$ is the set of all elements of \mathcal{X} bearing a non-null coefficient. We also denote by $\mathbb{S}^{(\mathcal{X})}$ the sub-semimodule of $\mathbb{S}^{\mathcal{X}}$ of all sums having a finite support.

We use the following syntactic sugar. The empty sum $\sum_{x \in \mathcal{X}} 0 \cdot x$ is denoted by 0. The one-element sum $\sum_{x \in \mathcal{X}} \delta_{x,y} \cdot x$ is assimilated to y , yielding an inclusion $\mathcal{X} \subseteq \mathbb{S}^{\mathcal{X}}$. Sums can be summed, i.e. $\sum_{x \in \mathcal{X}} a_x \cdot x + \sum_{x \in \mathcal{X}} b_x \cdot x = \sum_{x \in \mathcal{X}} (a_x + b_x) \cdot x$. It is also convenient to extend by linearity all the constructors of the calculus to sums of resource terms, i.e.

$$\begin{aligned} \lambda x. \left(\sum_{i \in I} a_i \cdot s_i \right) &:= \sum_{i \in I} a_i \cdot \lambda x.s_i, \\ \left(\sum_{i \in I} a_i \cdot s_i \right) \sum_{j \in J} b_j \cdot \bar{t}_j &:= \sum_{i \in I} \sum_{j \in J} a_i b_j \cdot (s_i)\bar{t}_j, \\ \left[\sum_{i \in I} a_i \cdot s_i \right] \cdot \sum_{j \in J} b_j \cdot \bar{t} &:= \sum_{i \in I} \sum_{j \in J} a_i b_j \cdot [s_i] \cdot \bar{t}_j. \end{aligned} \quad (4)$$

► **Definition 6.** For all $u \in (!)\Lambda_r$, $\bar{t} = [t_1, \dots, t_n] \in !\Lambda_r$ and $x \in \mathcal{V}$, the **multilinear substitution** of x by \bar{t} in u is the finite sum $s\langle \bar{t}/x \rangle \in \mathbb{N}^{(!)\Lambda_r}$ defined by

$$s\langle \bar{t}/x \rangle := \begin{cases} \sum_{\sigma \in \mathfrak{S}(n)} u[t_{\sigma(1)}/x_1, \dots, t_{\sigma(n)}/x_n] & \text{if } x \text{ occurs } n \text{ times in } u \\ 0 & \text{otherwise} \end{cases}$$

where x_1, \dots, x_n is an arbitrary enumeration of the occurrences of x in u , and $u[t_{\sigma(1)}/x_1, \dots]$ denotes the result of the (capture-avoiding) substitution of each x_i by the corresponding $t_{\sigma(i)}$.

► **Definition 7.** The relation $\longrightarrow_r \subset \mathbb{N}^{(!)\Lambda_r} \times \mathbb{N}^{(!)\Lambda_r}$ of **resource β -reduction** is defined using the auxiliary relation $\longrightarrow_r \subset (!)\Lambda_r \times \mathbb{N}^{(!)\Lambda_r}$ generated by the rules

$$\frac{}{(\lambda x.s)\bar{t} \longrightarrow_r s\langle \bar{t}/x \rangle} (\beta_r) \quad \frac{s \longrightarrow_r S'}{\lambda x.s \longrightarrow_r \lambda x.S'} (\lambda_r) \quad \frac{s \longrightarrow_r S'}{(s)\bar{t} \longrightarrow_r (S')\bar{t}} (@l_r)$$

$$\frac{\bar{t} \longrightarrow_r \bar{T}'}{(s)\bar{t} \longrightarrow_r (s)\bar{T}'} (@r_r) \quad \frac{s \longrightarrow_r S'}{[s] \cdot \bar{t} \longrightarrow_r [S'] \cdot \bar{t}} (!_r)$$

as well as the **lifting rule**

$$\frac{u_1 \longrightarrow_r U'_1 \quad \forall i \geq 2, u_i \overset{?}{\longrightarrow}_r U'_i}{\sum_{i=1}^n u_i \longrightarrow_r \sum_{i=1}^n U'_i} (\Sigma_r)$$

where $\overset{?}{\longrightarrow}_r$ is the reflexive closure of \longrightarrow_r .

From now on, we fix a semiring \mathbb{S} . We consider \mathbb{N} as a subset of \mathbb{S} through the map $n \mapsto 1 + \dots + 1$ (notice however that it might not be an injection), and we suppose that \mathbb{S} “has fractions”, i.e. for all non-null $n \in \mathbb{N}$ there is some $\frac{1}{n} \in \mathbb{S}$ such that $n \times \frac{1}{n} = 1$. This is the case of the semirings \mathbb{Q}_+ and \mathbb{R}_+ of non-negative rational (resp. real) numbers, but also of the semiring \mathbb{B} of boolean values (equipped with the logical “or” and “and” operations).

► **Definition 8.** *Given a set \mathcal{X} and a semiring \mathbb{S} , a family of sums $(\mathbf{S}_i)_{i \in I} \in (\mathbb{S}^{\mathcal{X}})^I$ is **summable** when each $x \in \mathcal{X}$ bears a non-null coefficient in finitely many of the \mathbf{S}_i . If this is the case then $\sum_{i \in I} \mathbf{S}_i$ is a well-defined sum.*

► **Definition 9.** *The relation $\longrightarrow_r \subset \mathbb{S}^{(!)\Lambda_r} \times \mathbb{S}^{(!)\Lambda_r}$ of **pointwise resource reduction** is defined by saying that there is a reduction $\mathbf{U} \longrightarrow_r \mathbf{V}$ whenever there are summable families $(u_i)_{i \in I} \in ((!)\Lambda_r)^I$ and $(V_i)_{i \in I} \in (\mathbb{N}^{(!)\Lambda_r})^I$ such that*

$$\mathbf{U} = \sum_{i \in I} a_i \cdot u_i, \quad \mathbf{V} = \sum_{i \in I} a_i \cdot V_i \quad \text{and} \quad \forall i \in I, u_i \longrightarrow_r^* V_i.$$

Notice that whereas \longrightarrow_r reduces finite sums with integer coefficients, \longrightarrow_r reduces arbitrary sums with arbitrary coefficients.

2.3 Linear approximation and the conservativity problems

We recall the definition of the Taylor expansion of λ -terms, and the approximation theorems it enjoys. Again, a detailed presentation can be found in [31], and in [7] for the adaption to infinitary λ -calculi. In the latter setting, we shall start with the following unusual definition.

► **Definition 10.** *The **Taylor expansion** is the map $\mathcal{T} : \Lambda^{001} \rightarrow \mathbb{S}^{\Lambda_r}$ defined by*

$$\mathcal{T}(M) := \sum_{s \in \Lambda_r} \mathcal{T}(M, s) \cdot s,$$

where the coefficient $\mathcal{T}(M, s)$ is defined by induction on $s \in \Lambda_r$ as follows:

$$\begin{aligned} \mathcal{T}(x, x) &:= 1 \\ \mathcal{T}(\lambda x.P, \lambda x.s) &:= \mathcal{T}(P, s) \\ \mathcal{T}\left((P)Q, (s)[t_1^{k_1}, \dots, t_m^{k_m}]\right) &:= \mathcal{T}(P, s) \times \prod_{i=1}^m \frac{\mathcal{T}(Q, t_i)^{k_i}}{k_i!}, \quad \text{the } t_i \text{'s being pairwise distinct} \\ \mathcal{T}(M, s) &:= 0 \quad \text{in all other cases.} \end{aligned}$$

Let us stress a crucial observation: whenever $s \in |\mathcal{T}(M)|$, the value of $\mathcal{T}(M, s)$ does not depend on M , hence $\mathcal{T}(M)$ is uniquely determined by its support [19].

Using the notation from Equation (4), we obtain the following description of the Taylor expansion. This is usually how the definition is presented for finite λ -terms, but since it is not a valid coinductive definition we had to provide Definition 10 in the infinitary setting.

► **Lemma 11** ([7], Corollary 4.7). *For all variables $x \in \mathcal{V}$ and terms $P, Q \in \Lambda^{001}$,*

$$\mathcal{T}(x) = x \quad \mathcal{T}(\lambda x.P) = \lambda x.\mathcal{T}(P) \quad \mathcal{T}((P)Q) = (\mathcal{T}(P))\mathcal{T}(Q)^\dagger,$$

where the operation of **promotion** is defined for all $\mathbf{S} \in \mathbb{S}^{\Lambda_r}$ by $\mathbf{S}^\dagger := \sum_{n \in \mathbb{N}} \frac{1}{n!} \cdot [\mathbf{S}]^n$.

We defined a map \mathcal{T} taking λ -terms to weighted sums of approximants. This induces an approximation of the λ -calculus, thanks to the following theorems expressing the fact that the reduction of the approximants can simulate the reduction of the approximated term.

► **Theorem 12** ([31], Lemma 7.6). *For $M, N \in \Lambda$, if $M \longrightarrow_\beta^* N$ then $\mathcal{T}(M) \longrightarrow_r \mathcal{T}(N)$.*

► **Theorem 13** ([7], Theorem 4.56). *For $M, N \in \Lambda^{001}$, if $M \longrightarrow_\beta^{001} N$ then $\mathcal{T}(M) \longrightarrow_r \mathcal{T}(N)$.*

In particular, the latter theorem encompasses the ‘‘Commutation theorem’’ [19, 18], which is usually presented as the cornerstone of the linear approximation of the λ -calculus: the normal form of $\mathcal{T}(M)$ is equal to the Taylor expansion of the Böhm tree of M (which is a notion of infinitary β -normal form of M), i.e. normalisation commutes with approximation¹.

► **Definition 14.** *Let (A, \longrightarrow_A) and (B, \longrightarrow_B) be two reduction systems. The latter is an **extension** of the former if:*

1. *there is an injection $i : A \hookrightarrow B$,*
2. *\longrightarrow_A **simulates** \longrightarrow_B through i , i.e. $\forall a, a' \in A$, if $a \longrightarrow_A a'$ then $i(a) \longrightarrow_B i(a')$.*

*This extension is said to be **conservative**² if $\forall a, a' \in A$, if $i(a) \longrightarrow_B i(a')$ then $a \longrightarrow_A a'$.*

Theorems 12 and 13 can be reformulated using this definition, thanks to the fact that $\mathcal{T} : \Lambda^{001} \rightarrow \mathbb{S}^{\Lambda_r}$ is injective [8, Lemma 5.18]:

- Theorem 12 tells that $(\mathbb{S}^{\Lambda_r}, \longrightarrow_r)$ simulates $(\Lambda, \longrightarrow_\beta^*)$,
- Theorem 13 tells that $(\mathbb{S}^{\Lambda_r}, \longrightarrow_r)$ simulates $(\Lambda^{001}, \longrightarrow_\beta^{001})$,

which leads us to the problems we tackle in this paper.

▷ Problem 15. Is $(\mathbb{S}^{\Lambda_r}, \longrightarrow_r)$ conservative wrt. $(\Lambda, \longrightarrow_\beta^*)$?

▷ Problem 16. Is $(\mathbb{S}^{\Lambda_r}, \longrightarrow_r)$ conservative wrt. $(\Lambda^{001}, \longrightarrow_\beta^{001})$?

3 Conservativity wrt. the finite λ -calculus

In this first section, we give a positive answer to Problem 15:

► **Theorem 17** (conservativity). *For all $M, N \in \Lambda$, if $\mathcal{T}(M) \longrightarrow_r \mathcal{T}(N)$ then $M \longrightarrow_\beta^* N$.*

We adapt a proof technique by Kerinec and the second author [24], who used it to prove that the algebraic λ -calculus is a conservative extension of the usual λ -calculus. Their proof relies on a relation \vdash , called ‘‘mashup’’ of β -reductions, relating λ -terms (from the ‘‘small world’’) to their algebraic reducts (in the ‘‘big world’’). In our setting, $M \vdash s$ when s is an approximant of a reduct of M .

► **Definition 18.** *The **mashup** relation $\vdash \subset \Lambda \times \Lambda_r$ is defined by the following rules:*

$$\frac{M \longrightarrow_\beta^* x}{M \vdash x} \quad \frac{M \longrightarrow_\beta^* \lambda x.P \quad P \vdash s}{M \vdash \lambda x.s}$$

$$\frac{M \longrightarrow_\beta^* (P)Q \quad P \vdash s \quad Q \vdash \bar{t}}{M \vdash (s)\bar{t}} \quad \frac{M \vdash t_1 \quad \dots \quad M \vdash t_n}{M \vdash [t_1, \dots, t_n]}$$

It is extended to $\Lambda \times \mathbb{S}^{\Lambda_r}$ by the following rule:

$$\frac{\forall i \in I, M \vdash s_i}{M \vdash \sum_{i \in I} a_i \cdot s_i}$$

for any index set I and coefficients $a_i \in \mathbb{S}$ such that the sum exists.

¹ To be rigorous, Theorem 13 must first be extended to a variant of the β -reduction called $\beta\perp$ -reduction. We remain allusive here, and refer to [8, 7] for more details.

² Notice that our definition varies from the one chosen by the *Terese* [30, § 1.3.21], where the conservativity of \longrightarrow_B wrt. \longrightarrow_A is defined as a property of the conversions $=_A$ and $=_B$ they generate. We prefer to distinguish between a conservative extension of a reduction (‘‘in the small world, the big reduction reduces the same people to the same people’’) and a conservative extension of the corresponding conversion.

23:8 How to Play the Accordion

► **Lemma 19.** For all $M \in \Lambda$, $M \vdash \mathcal{T}(M)$.

Proof. Take any $s \in |\mathcal{T}(M)|$. By an immediate induction on s , $M \vdash s$ follows from the rules of Definition 18 (where all the assumptions \rightarrow_{β}^* are just taken to be equalities). ◀

► **Lemma 20.** For all $M, N \in \Lambda$ and $\mathbf{S} \in \mathbb{S}^{\Lambda_r}$, if $M \rightarrow_{\beta}^* N$ and $N \vdash \mathbf{S}$ then $M \vdash \mathbf{S}$.

Proof. Take any $s \in |\mathbf{S}|$, then $N \vdash s$. By an immediate induction on s , $M \vdash s$ follows from the rules of Definition 18 (where the assumptions $M \rightarrow_{\beta}^* \dots$ follow from the corresponding $M \rightarrow_{\beta}^* N \rightarrow_{\beta}^* \dots$). ◀

► **Lemma 21.** For all $M, N \in \Lambda$, $x \in \mathcal{V}$, $s \in \Lambda_r$ and $\bar{t} \in !\Lambda_r$, if $M \vdash s$ and $N \vdash \bar{t}$ then $\forall s' \in |s(\bar{t}/x)|$, $M[N/x] \vdash s'$.

Proof. Assume M and N are given and show the following equivalent result by induction on s : if $M \vdash s$ then for all \bar{t} such that $N \vdash \bar{t}$ and for all $s' \in |s(\bar{t}/x)|$, $M[N/x] \vdash s'$. ◀

► **Lemma 22.** For all $M \in \Lambda$ and $\mathbf{S}, \mathbf{T} \in \mathbb{S}^{\Lambda_r}$, if $M \vdash \mathbf{S}$ and $\mathbf{S} \rightarrow_r \mathbf{T}$ then $M \vdash \mathbf{T}$.

Proof. Let us first show that for all $M \in \Lambda$ and $s \in \Lambda_r$ and $T \in \mathbb{N}^{(\Lambda_r)}$, if $M \vdash s \rightarrow_r T$ then $\forall t \in |T|$, $M \vdash t$. We do so by induction on $s \rightarrow_r T$. When $s = (\lambda x.u)\bar{v}$ is a redex, there exists a derivation:

$$\frac{M \rightarrow_{\beta}^* (P)Q \quad \frac{P \rightarrow_{\beta}^* \lambda x.P' \quad P' \vdash u}{P \vdash \lambda x.u} \quad Q \vdash \bar{v}}{M \vdash (\lambda x.u)\bar{v}}$$

By Lemma 21 with $P' \vdash u$, $Q \vdash \bar{v}$, for all $t \in |u(\bar{v}/x)|$, we obtain $P'[Q/x] \vdash t$. Finally, since $M \rightarrow_{\beta}^* (\lambda x.P')Q \rightarrow_{\beta} P'[Q/x]$, we conclude by Lemma 20. The other cases of the induction follow immediately by lifting to the context.

As a consequence, we can easily deduce the following steps:

- if $M \vdash s \rightarrow_r T$ then $M \vdash T$, for all $M \in \Lambda$, $s \in \Lambda_r$ and $T \in \mathbb{N}^{(\Lambda_r)}$,
- if $M \vdash S \rightarrow_r T$ then $M \vdash T$, for all $M \in \Lambda$ and $S, T \in \mathbb{N}^{(\Lambda_r)}$,
- if $M \vdash S \rightarrow_r^* T$ then $M \vdash T$, for all $M \in \Lambda$ and $S, T \in \mathbb{N}^{(\Lambda_r)}$,

which leads to the result. ◀

Before we state the last lemma of the proof, recall that there is a canonical injection $[-]_r : \Lambda \rightarrow \Lambda_r$ defined by:

$$[x]_r := x \quad [\lambda x.P]_r := \lambda x.[P]_r \quad [(P)Q]_r := ([P]_r) [[Q]_r]$$

and such that for all $N \in \Lambda$, $[N]_r \in |\mathcal{T}(N)|$.

► **Lemma 23.** For all $M, N \in \Lambda$, if $M \vdash \mathcal{T}(N)$ then $M \rightarrow_{\beta}^* N$.

Proof. If $M \vdash \mathcal{T}(N)$, then in particular $M \vdash [N]_r$. We proceed by induction on N :

- If $N = x$, then $M \vdash x$ so $M \rightarrow_{\beta}^* x$ by definition.
- If $N = \lambda x.P'$, then $M \vdash \lambda x.[P']_r$, i.e. there is a $P \in \Lambda$ such that $M \rightarrow_{\beta}^* \lambda x.P$ and $P \vdash [P']_r$. By induction, $P \rightarrow_{\beta}^* P'$, thus $M \rightarrow_{\beta}^* \lambda x.P' = N$.
- If $N = (P')Q'$, then $M \vdash ([P']_r) [[Q']_r]$ i.e. there are $P, Q \in \Lambda$ such that $M \rightarrow_{\beta}^* (P)Q$, $P \vdash [P']_r$ and $Q \vdash [[Q']_r]$. By induction, $P \rightarrow_{\beta}^* P'$ and $Q \rightarrow_{\beta}^* Q'$, thus $M \rightarrow_{\beta}^* (P')Q' = N$. ◀

Proof of Theorem 17. Suppose that $\mathcal{T}(M) \rightarrow_r \mathcal{T}(N)$. By Lemma 19 we obtain $M \vdash \mathcal{T}(M)$, hence by Lemma 22 $M \vdash \mathcal{T}(N)$. We can conclude with Lemma 23. ◀

4 Non-conservativity wrt. the infinitary λ -calculus

The previous theorem relied on the excellent properties of the Taylor expansion of finite λ -terms: a single (well-chosen) term $[M]_{\mathcal{r}} \in |\mathcal{T}(M)|$ is enough to characterise M , and a single (again, well-chosen) sequence of resource reducts of some $s \in |\mathcal{T}(M)|$ suffices to characterise any sequence $M \rightarrow_{\beta}^* N$. These properties are not true any more when considering more complicated settings, like the 001-infinitary λ -calculus. This does not only make the “mashup” proof technique fail, but also enables us to give a negative answer to Problem 16.

4.1 Failure of the “mashup” technique

Let us first describe where we hit an obstacle if we try to reproduce the proof we have given in the finite setting, which will make clearer the way we later build a counterexample.

First, it is not obvious what the mashup relation should be: we could just use the relation \vdash defined on $\Lambda^{001} \times \Lambda_{\mathcal{r}}$ by the same set of rules as in Definition 18, or define an infinitary mashup \vdash_{001} by the rules

$$\frac{M \rightarrow_{\beta}^{001} x}{M \vdash_{001} x} \quad \frac{M \rightarrow_{\beta}^{001} \lambda x.P \quad P \vdash_{001} s}{M \vdash_{001} \lambda x.s}$$

$$\frac{M \rightarrow_{\beta}^{001} (P)Q \quad P \vdash_{001} s \quad Q \vdash_{001} \bar{t}}{M \vdash_{001} (s)\bar{t}} \quad \frac{M \vdash_{001} t_1 \quad \dots \quad M \vdash_{001} t_n}{M \vdash_{001} [t_1, \dots, t_n]}$$

and extend it to $\mathbb{S}^{\Lambda_{\mathcal{r}}}$ accordingly. In fact, this happens to define the same relation.

► **Lemma 24.** *For all $M \in \Lambda^{001}$ and $s \in \Lambda_{\mathcal{r}}$, $M \vdash_{001} s$ iff $M \vdash s$.*

Proof. The inclusion $\vdash \subseteq \vdash_{001}$ is immediate. Let us show the converse. First, observe that the proof of Lemma 20 can be easily extended in order to show that for all $M, N \in \Lambda^{001}$ and $s \in \Lambda_{\mathcal{r}}$, if $M \rightarrow_{\beta}^{001} N \vdash_{001} s$ then $M \vdash_{001} s$. Then we proceed by induction on s .

- If $M \vdash_{001} x$, then $M \rightarrow_{\beta}^{001} x$, i.e. $M \rightarrow_{\beta}^* x$, and finally $M \vdash x$.
- If $M \vdash_{001} \lambda x.u$, then there is a derivation:

$$\frac{\frac{M \rightarrow_{\beta}^* \lambda x.P \quad P \rightarrow_{\beta}^{001} P'}{M \rightarrow_{\beta}^{001} \lambda x.P'} \quad P' \vdash_{001} u}{M \vdash_{001} \lambda x.u}}$$

Since $P \rightarrow_{\beta}^{001} P' \vdash_{001} u$, we have $P \vdash_{001} u$, and by induction on u we obtain $P \vdash u$. With $M \rightarrow_{\beta}^* \lambda x.P$, this yields $M \vdash \lambda x.u$.

- The case of $M \vdash_{001} (u)\bar{v}$ is similar. ◀

As a consequence, Lemmas 19–22 can be easily extended to $\rightarrow_{\beta}^{001}$ and \vdash_{001} . We have already explained how the proof of this can be done for Lemma 20; for the other ones, one just needs to observe that the proofs are all by induction on resource terms or on some inductively defined relation, hence replacing \rightarrow_{β}^* with $\rightarrow_{\beta}^{001}$ does not change anything (and neither does replacing \vdash with \vdash_{001} , thanks to Lemma 24).

The failure of the infinitary “mashup” proof occurs in the extension of Lemma 23. Indeed, this proof crucially relies on the existence of an injection $[-]_{\mathcal{r}} : \Lambda \rightarrow \Lambda_{\mathcal{r}}$, whereas for Λ^{001} there is only the counterpart $[-]_{\mathcal{r}, -} : \Lambda^{001} \times \mathbb{N} \rightarrow \Lambda_{\mathcal{r}}$ defined by

23:10 How to Play the Accordion

$$\begin{aligned} [x]_{r,d} &:= x & [(P)Q]_{r,0} &:= ([P]_{r,0}) 1 \\ [\lambda x.P]_{r,d} &:= \lambda x.[P]_{r,d} & [(P)Q]_{r,d+1} &:= ([P]_{r,d+1}) [[Q]_{r,d}]. \end{aligned}$$

Now, if we suppose that $M \vdash \mathcal{T}(N)$ and we want to show that $M \xrightarrow{\beta}^{001} N$, we cannot rely any more on the fact that $M \vdash [N]_r$, but only on the fact that $\forall d \in \mathbb{N}$, $M \vdash [N]_{r,d}$. This makes the induction fail. For instance, for the case where N is an abstraction $\lambda x.P'$, we obtain a d -indexed sequence of derivations

$$\frac{M \xrightarrow{\beta}^* \lambda x.P_d \quad P_d \vdash [P']_{r,d}}{M \vdash [N]_{r,d} = [\lambda x.P']_{r,d}}$$

but nothing tells us that the terms P_d and reductions $M \xrightarrow{\beta}^* \lambda x.P_d$ are coherent! This failure is what enables us to design a counterexample.

4.2 The Accordion

In this section, we define 001-infinitary λ -terms \mathbf{A} and $\bar{\mathbf{A}}$ and show that they form a counterexample not only to the 001-infinitary counterpart of Lemma 23, but also to the conservativity property in the infinitary setting.

► **Notation 25.** We denote as follows the usual representation of booleans, an “applicator” $\langle - \rangle$, and the Church encodings of integers and of the successor function:

$$\begin{aligned} \mathbf{T} &:= \lambda x.\lambda y.x & \mathbf{F} &:= \lambda x.\lambda y.y & \langle M \rangle &:= \lambda b.(b)M \\ \mathbf{n} &:= \lambda f.\lambda x.(f)^n x & \mathbf{Succ} &:= \lambda n.\lambda f.\lambda x.(n) f (f)x \end{aligned}$$

► **Definition 26.** The *Accordion* λ -term is defined as $\mathbf{A} := (\mathbf{P})\mathbf{0}$, where:

$$\mathbf{P} := (\mathbf{Y}) \lambda \phi.\lambda n. (\langle \mathbf{T} \rangle) ((n) \langle \mathbf{F} \rangle) \mathbf{Q}_{\phi,n} \quad \mathbf{Q}_{\phi,n} := (\mathbf{Y}) \lambda \psi.\lambda b. ((b) \langle \phi \rangle) (\mathbf{Succ}) n \psi.$$

We also define $\bar{\mathbf{A}} := (\langle \mathbf{T} \rangle) (\langle \mathbf{F} \rangle)^\omega$.

Let us show how this term behaves (and why we named it the Accordion). There exist terms \mathbf{P}'' (which is nothing but the first head reduct of \mathbf{P}) and \mathbf{Q}_n (for all $n \in \mathbb{N}$) such that the following reductions hold:

$$\begin{array}{cccccccc} \mathbf{A} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \mathbf{P}'' \quad \mathbf{0} \end{array} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \langle \mathbf{T} \rangle \quad \mathbf{Q}_0 \end{array} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \mathbf{P}'' \quad \mathbf{1} \end{array} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \langle \mathbf{T} \rangle \quad @ \\ \quad \quad \backslash \\ \quad \quad \langle \mathbf{F} \rangle \quad \mathbf{Q}_1 \end{array} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \mathbf{P}'' \quad \mathbf{n} \end{array} & \xrightarrow{\beta}^* & \begin{array}{c} @ \\ / \quad \backslash \\ \langle \mathbf{T} \rangle \quad @ \\ \quad \quad \backslash \\ \quad \quad \langle \mathbf{F} \rangle \quad @ \\ \quad \quad \quad \vdots \\ \quad \quad \quad @ \\ \quad \quad \quad \backslash \\ \quad \quad \quad \langle \mathbf{F} \rangle \quad \mathbf{Q}_n. \end{array} \end{array}$$

This means that:

1. for any $d \in \mathbb{N}$, \mathbf{A} reduces to terms \mathbf{A}_d that are similar to $\bar{\mathbf{A}}$ up to depth d (and, as a consequence, any finite approximant of $\bar{\mathbf{A}}$ is a reduct of approximants of \mathbf{A});
2. but this is not a valid infinitary reduction because we *need* to reduce a redex at depth 0 to obtain $\mathbf{A}_d \xrightarrow{\beta}^* \mathbf{A}_{d+1}$, thus the stratification property (Theorem 4) is violated: the depth of the reduced redexes does not tend to the infinity.

Our definition of \mathbf{A} and $\bar{\mathbf{A}}$ was entirely guided by this specification. More concretely:

- when fed with a Church integer argument \mathbf{n} , the term \mathbf{P}'' produces a term mimicking $\bar{\mathbf{A}}$ up to the n -th copy of $\langle \mathbf{F} \rangle$, the latter being applied to $\mathbf{Q}_n = \mathbf{Q}_{\mathbf{P}'', \mathbf{n}}$;
- the applicator $\langle - \rangle$ enforces a kind of call-by-value discipline, giving control to the argument (observe that $\langle \langle M \rangle \rangle N \rightarrow_{\beta} (N)M$);
- $\mathbf{Q}_{\mathbf{P}'', \mathbf{n}}$ eats up boolean arguments \mathbf{F} , until it is fed with a boolean \mathbf{T} (marking the root of the tree), at which point it restores \mathbf{P}'' , applied to the next Church integer.

In particular, this dynamics (\mathbf{A} is “stretched” and “compressed” over and over) justifies the name “Accordion”.

To be a counterexample to conservativity, \mathbf{A} actually has to satisfy a stronger property: *all reduction paths* starting from \mathbf{A} should have this “accordion” behaviour. A thorough analysis of the dynamics will allow us to establish this, and obtain:

► **Theorem 27.** (i) $\mathcal{T}(\mathbf{A}) \rightarrow_{\mathbf{r}} \mathcal{T}(\bar{\mathbf{A}})$, but (ii) there is no reduction $\mathbf{A} \rightarrow_{\beta}^{001} \bar{\mathbf{A}}$.

This theorem improves on the results from the first author’s PhD thesis [7, Theorem 5.12], where only the qualitative setting was treated (i.e. when $\mathbb{S} = \mathbb{B}$). Non-conservativity in the general case was presented as Conjecture 5.15, which is thereby solved.

4.3 Proof of the counterexample

In this (highly technical) section, we prove Theorem 27: a reader already satisfied with the above intuitions might prefer to skip it, and jump to Section 5. The key ingredient in the proof are the following well-known notions as well as the associated factorization property, due to Mitschke [28, cor. 5].

► **Definition 28.** A λ -term $M \in \Lambda^{001}$ has two possible *head forms*:

- either the form $\lambda x_1 \dots \lambda x_m. (y)M_1 \dots M_n$, called **head normal form (HNF)**,
- or the form $\lambda x_1 \dots \lambda x_m. (\lambda x. P)QM_1 \dots M_n$, where $(\lambda x. P)Q$ is called the **head redex**.

As a consequence, a β -reduction $M \rightarrow_{\beta} N$ reduces:

- either a head redex: it is a **head reduction**, denoted by $M \rightarrow_h N$,
- or any other redex: it is an **internal reduction**, denoted by $M \rightarrow_i N$.

► **Lemma 29** (head-internal decomposition). For all $M, N \in \Lambda$ such that $M \rightarrow_{\beta}^* N$, there exists an $M' \in \Lambda$ such that $M \rightarrow_h^* M' \rightarrow_i^* N$.

Let us also introduce some abbreviations³:

$$\begin{aligned} \mathbf{P}' &:= \lambda \phi. \lambda n. (\langle \mathbf{T} \rangle) (\langle n \rangle \langle \mathbf{F} \rangle) \mathbf{Q}_{\phi, n} & \mathbf{P}'' &:= (\lambda x. (\mathbf{P}') (x) x) \lambda x. (\mathbf{P}') (x) x & \mathbf{Q}_n &:= \mathbf{Q}_{\mathbf{P}'', (\text{Succ})^n 0} \\ \mathbf{Q}'_n &:= \lambda \psi. \lambda b. (\langle b \rangle \langle \mathbf{P}'' \rangle) (\text{Succ})^{n+1} 0 \psi & \mathbf{Q}''_n &:= (\lambda x. (\mathbf{Q}'_n) (x) x) \lambda x. (\mathbf{Q}'_n) (x) x. \end{aligned}$$

Using these definitions, Figure 1 describes the head reduction path starting from \mathbf{A} .

Proof of Theorem 27, item (i). For all $d \in \mathbb{N}$, we define:

- $\bar{\mathbf{A}}_d := (\langle \mathbf{T} \rangle) (\langle \mathbf{F} \rangle)^d \mathbf{Q}_n$. As a consequence of the reduction described in Figure 1, in particular its step 7, there are reductions $\mathbf{A} \rightarrow_{\beta}^* \bar{\mathbf{A}}_0 \rightarrow_{\beta}^* \bar{\mathbf{A}}_1 \rightarrow_{\beta}^* \bar{\mathbf{A}}_2 \rightarrow_{\beta}^* \dots$. By Theorem 12, we obtain

$$\mathcal{T}(\mathbf{A}) \rightarrow_{\mathbf{r}} \mathcal{T}(\bar{\mathbf{A}}_0) \rightarrow_{\mathbf{r}} \mathcal{T}(\bar{\mathbf{A}}_1) \rightarrow_{\mathbf{r}} \mathcal{T}(\bar{\mathbf{A}}_2) \rightarrow_{\mathbf{r}} \dots \quad (29)$$

³ Notice that the \mathbf{Q}_n we define here are slightly different from those in the example reduction described above, but they play the same role.

The first step is:		$\longrightarrow_h^* \left((Y)Q'_n \right) \underbrace{F \dots F}_n T$	(16)
$A = ((Y)P')0 \longrightarrow_h (P'')0$		$\longrightarrow_h (Q''_n) F \dots F T$	(17)
Then, for each $n \in \mathbb{N}$, we do the following head reduction steps:		$\longrightarrow_h ((Q'_n)Q''_n) F \dots F T$	(18)
$(P'')(Succ)^n 0$		$\longrightarrow_h (\lambda b. ((b)(P'')(Succ)^{n+1} 0) Q''_n) F \dots F T$	(19)
$\longrightarrow_h ((P')P'')(Succ)^n 0$	(5)	$\longrightarrow_h ((\lambda x. \lambda y. y)(P'')(Succ)^{n+1} 0) Q''_n \underbrace{F \dots F}_{n-1} T$	(20)
$\longrightarrow_h (\lambda n. ((T)) ((n)(F) Q_{P'',n}) (Succ)^n 0$	(6)	$\longrightarrow_h ((\lambda y. y)Q''_n) F \dots F T$	(21)
$\longrightarrow_h (T) (((Succ)^n 0) (F) Q_n$	(7)	$\longrightarrow_h (Q''_n) F \dots F T$	(22)
$\longrightarrow_h (Succ)^n 0 (F) Q_n T$	(8)		
$\longrightarrow_h (\lambda f. \lambda x. (((Succ)^{n-1} 0) f) (f)x) (F) Q_n T$	(9)		
$\longrightarrow_h (\lambda x. (((Succ)^{n-1} 0) (F) ((F)x) Q_n) T$	(10)	and by repeating steps (18) to (22):	
$\longrightarrow_h ((Succ)^{n-1} 0 (F) ((F)Q_n) T$	(11)	$\longrightarrow_h^* (Q''_n) T$	(23)
and by repeating steps (9) to (11):		$\longrightarrow_h ((Q'_n)Q''_n) T$	(24)
$\longrightarrow_h^* ((0)(F) ((F))^n Q_n) T$	(12)	$\longrightarrow_h (\lambda b. ((b)(P'')(Succ)^{n+1} 0) Q''_n) T$	(25)
$\longrightarrow_h ((\lambda x.x) ((F))^n Q_n) T$	(13)	$\longrightarrow_h ((\lambda x. \lambda y.x)(P'')(Succ)^{n+1} 0) Q''_n$	(26)
$\longrightarrow_h ((\lambda b.(b)F) ((F))^{n-1} Q_n) T$	(14)	$\longrightarrow_h (\lambda y.(P'')(Succ)^{n+1} 0) Q''_n$	(27)
$\longrightarrow_h (((F))^{n-1} Q_n) F T$	(15)	$\longrightarrow_h (P'')(Succ)^{n+1} 0$	(28)
and by repeating step (15):		which brings us back to step (5).	

■ **Figure 1** Exhaustive head reduction of the Accordion. We highlight the fired head redexes.

- $\mathcal{T}'_d(\bar{A}) := \mathcal{T}(((T))((F))^d \perp)$, where \perp is a constant such that $\mathcal{T}(\perp) := 0$ (this is just a trick to “cut” the Taylor expansion at some point), as well as

$$\begin{cases} \mathcal{T}_0(\bar{A}) := \mathcal{T}'_0(\bar{A}) \\ \mathcal{T}_{d+1}(\bar{A}) := \mathcal{T}'_{d+1}(\bar{A}) - \mathcal{T}'_d(\bar{A}) = \sum_{s \in |\mathcal{T}'_{d+1}(\bar{A})| \setminus |\mathcal{T}'_d(\bar{A})|} \mathcal{T}(s, \bar{A}) \cdot s. \end{cases}$$

By construction (using the observation that the coefficient of $s \in |\mathcal{T}(M)|$ does not depend on M), we obtain:

$$\mathcal{T}(\bar{A}_d) = \mathcal{T}_d(\bar{A}) + \mathbf{S}_d, \text{ for some } \mathbf{S}_d \text{ such that } |\mathcal{T}_d(\bar{A})| \cap |\mathbf{S}_d| = \emptyset \quad (30)$$

$$\mathcal{T}(\bar{A}) = \sum_{n \in \mathbb{N}} \mathcal{T}_d(\bar{A}) \quad (31)$$

Before we use this material to prove the theorem, we need to make the following crucial observation:

$$\forall s \in \mathcal{T}_d(\bar{A}), \forall k > 0, \nexists t \in \mathcal{T}_{d+k}(\bar{A}), s \longrightarrow_r^* t + T \quad (32)$$

for some $T \in \mathbb{N}^{(\Lambda_r)}$. This is due to the fact that terms in $\mathcal{T}(\bar{A})$ cannot see their (applicative) depth increase through resource reduction.

Now we start with Equation (29), having $\mathcal{T}(A) \longrightarrow_r \mathcal{T}(\bar{A}_0) \longrightarrow_r \mathcal{T}(\bar{A}_1)$. Thanks to Equation (30), this can be rewritten as $\mathcal{T}(A) \longrightarrow_r \mathcal{T}_0(\bar{A}) + \mathbf{S}_0 \longrightarrow_r \mathcal{T}_1(\bar{A}) + \mathbf{S}_1$. Equation (32) allows to say that only \mathbf{S}_0 contributes to $\mathcal{T}_1(\bar{A})$ in the second reduction. If we leave $\mathcal{T}_0(\bar{A})$ untouched and only reduce \mathbf{S}_0 , we obtain $\mathcal{T}_0(\bar{A}) + \mathbf{S}_0 \longrightarrow_r \mathcal{T}_0(\bar{A}) + \mathcal{T}_1(\bar{A}) + \mathbf{S}'_1$ for some \mathbf{S}'_1 that is part of \mathbf{S}_1 . If we keep applying Equations (30) and (32) and we iterate the process, we obtain:

$$\mathcal{T}(\mathbf{A}) \twoheadrightarrow_{\mathbf{r}} \mathcal{T}_0(\bar{\mathbf{A}}) + \mathbf{S}_0 \twoheadrightarrow_{\mathbf{r}} \mathcal{T}_0(\bar{\mathbf{A}}) + \mathcal{T}_1(\bar{\mathbf{A}}) + \mathbf{S}'_1 \twoheadrightarrow_{\mathbf{r}} \dots \twoheadrightarrow_{\mathbf{r}} \sum_{d=0}^N \mathcal{T}_d(\bar{\mathbf{A}}) + \mathbf{S}'_N \quad (33)$$

for all $N \in \mathbb{N}$. For each $s \in |\mathcal{T}(\mathbf{A})|$, this can be turned into⁴:

$$s \twoheadrightarrow_{\mathbf{r}}^* T_{s,0} + S_{s,0} \twoheadrightarrow_{\mathbf{r}}^* T_{s,0} + T_{s,1} + S_{s,1} \twoheadrightarrow_{\mathbf{r}}^* \dots \twoheadrightarrow_{\mathbf{r}}^* \sum_{d=0}^N T_{s,d} + S_{s,N} \quad (34)$$

for some $T_{s,d}, S_{s,d} \in \mathbb{N}^{(\Lambda_r)}$ satisfying $\mathcal{T}_d(\bar{\mathbf{A}}) = \sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot T_{s,d}$. In fact:

- There are only finitely many d 's such that $T_{s,d} \neq 0$ (this is due to the fact that a resource terms has only finitely many reducts [31, Lemma 3.13]).
- \mathbf{A} has no head normal form as demonstrated in Figure 1, which entails that $\mathcal{T}(\mathbf{A}) \twoheadrightarrow_{\mathbf{r}} 0$ [8, Theorem 5.6]. Since \mathbf{S}'_d only contains reducts of terms in $\mathcal{T}(\mathbf{A})$, this means that we can reduce $\mathbf{S}'_N \twoheadrightarrow_{\mathbf{r}}^* 0$.

As a consequence, $s \twoheadrightarrow_{\mathbf{r}}^* \sum_{d \in \mathbb{N}} T_{s,d}$ and we can conclude:

$$\mathcal{T}(\mathbf{A}) = \sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot s \twoheadrightarrow_{\mathbf{r}} \sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot \sum_{d \in \mathbb{N}} T_{s,d} = \sum_{d \in \mathbb{N}} \mathcal{T}_d(\bar{\mathbf{A}}) = \mathcal{T}(\bar{\mathbf{A}})$$

by Equation (31). ◀

Proof of Theorem 27, item (ii). We suppose that there is a reduction $\mathbf{A} \xrightarrow{\beta}^{001} \bar{\mathbf{A}}$ and we show that this leads to a contradiction. By Theorem 4 and Lemma 29, there exists respectively a sequence of terms $\mathbf{A}_d \in \Lambda$ and a term $\mathbf{A}'_0 \in \Lambda$ such that there are reductions

$$\mathbf{A} \xrightarrow{h}^* \mathbf{A}'_0 \xrightarrow{i}^* \mathbf{A}_1 \xrightarrow{\beta \geq 1}^* \mathbf{A}_d \xrightarrow{\beta \geq d}^{001} \bar{\mathbf{A}}.$$

\mathbf{A}'_0 and $\bar{\mathbf{A}}$ must have the same head form, i.e. there must be $M, N \in \Lambda$ such that $\mathbf{A}'_0 = (\lambda b.M)N$. The exhaustive description of the head reducts of \mathbf{A} detailed in Figure 1 allows to observe that this only happens in four cases (corresponding to steps 6, 7, 25 and 27 in Figure 1):

⁴ This inference might not be possible for an arbitrary reduction sequence, because the obtained reductions (34) occur in $\mathbb{N}^{(\Lambda_r)}$ (with integer coefficients only) while the original reductions (33) occur in \mathbb{S}^{Λ_r} (possibly with rational coefficients): if for some $s \in \mathbf{S}'_d$ the original reduction $\mathbf{S}'_d \twoheadrightarrow_{\mathbf{r}} \mathcal{T}_{d+1}(\bar{\mathbf{A}}) + \mathbf{S}'_{d+1}$ consists in doing $s = \frac{1}{3}s + \frac{2}{3}s \twoheadrightarrow_{\mathbf{r}} \frac{1}{3}S' + \frac{2}{3}S''$, we will not be able to retrieve a reduction $s \twoheadrightarrow_{\mathbf{r}}^* \dots$ of the desired shape.

But the reductions in \mathbb{S}^{Λ_r} we consider are not arbitrary: Equation (29) was obtained by simulating a sequence of β -reductions *via* Theorem 12, so that we can apply uniformity. With the notations to be introduced in Section 5, using Corollary 34 we obtain reductions $\mathbf{S}'_d \twoheadrightarrow_{\mathbf{r}}^* \mathcal{T}_{d+1}(\bar{\mathbf{A}}) + \mathbf{S}'_{d+1}$ instead of $\mathbf{S}'_d \twoheadrightarrow_{\mathbf{r}} \mathcal{T}_{d+1}(\bar{\mathbf{A}}) + \mathbf{S}'_{d+1}$. These reductions can only be derived as follows:

$$\frac{\sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot \underbrace{\sum_{i=1}^{n_{s,d}} s_{s,d,i}}_{\mathbf{S}_{s,d}} \xrightarrow{\mathbf{r}}^* \sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot \left(\underbrace{\sum_{i=1}^{n_{s,d}} T_{s,d+1,i}}_{\mathbf{T}_{s,d+1}} + \underbrace{\sum_{i=1}^{n_{s,d}} S_{s,d+1,i}}_{\mathbf{S}_{s,d+1}} \right)}{\sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot \sum_{i=1}^{n_{s,d}} s_{s,d,i} \xrightarrow{\mathbf{r}}^* \sum_{s \in \Lambda_r} \mathcal{T}(s, \mathbf{A}) \cdot \left(\sum_{i=1}^{n_{s,d}} T_{s,d+1,i} + \sum_{i=1}^{n_{s,d}} S_{s,d+1,i} \right)}$$

the premise of which allows to build a reduction $S_{s,d} \twoheadrightarrow_{\mathbf{r}}^* T_{s,d+1} + S_{s,d+1}$.

23:14 How to Play the Accordion

1. $A'_0 = (\lambda n. \langle T \rangle) ((n) \langle F \rangle) Q_{P'',n} (\text{Succ})^n 0$,
2. $A'_0 = \langle T \rangle (((\text{Succ})^n 0) \langle F \rangle) Q_n$,
3. $A'_0 = (\lambda b. ((b)(P''))(\text{Succ})^{n+1} 0) Q''_n T$,
4. $A'_0 = (\lambda y. (P'')) (\text{Succ})^{n+1} 0) Q''_n$,

for some $n \in \mathbb{N}$ (in the following, n denotes this specific integer appearing in A'_0). In particular, for one of these possible values of A'_0 there must be a reduction

$$A'_0 \longrightarrow_i^* A_{n+4} \longrightarrow_{\beta \geq n+4}^{001} \bar{A}.$$

Since A_{n+4} and \bar{A} are identical up to applicative depth $n+3$, we can write $A_{n+4} = \langle T \rangle (\langle F \rangle)^{n+1} M$ for some $M \in \Lambda$ such that $M \longrightarrow_{\beta}^{001} (\langle F \rangle)^\omega$ (we need to go up to depth $n+3$ since $\langle T \rangle$ and $\langle F \rangle$ are themselves of applicative depth 2). Finally, there must be a reduction

$$A'_0 \longrightarrow_i^* \langle T \rangle (\langle F \rangle)^{n+1} M.$$

For each of the possible cases for A'_0 , let us show that this is impossible. The easy cases are:

Case 1, step (6) Such a reduction would imply that $(\text{Succ})^n 0 \longrightarrow_{\beta}^* (\langle F \rangle)^{n+1} M$. However $(\text{Succ})^n 0 \longrightarrow_{\beta}^* n$, which is in β -normal form, while $(\langle F \rangle)^{n+1} M$ has no normal form. We conclude by confluence of the finite λ -calculus.

Case 3, step (25) Immediate because T is in normal form.

Case 4, step (27) Such a reduction would imply that $\lambda y. (P'') (\text{Succ})^{n+1} 0 \longrightarrow_{\beta}^* \langle T \rangle = \lambda y. (y) T$, and therefore that $(P'') (\text{Succ})^{n+1} 0$ has a HNF $(y) T$. This is impossible, as detailed in the exhaustive head reduction of A in Figure 1.

The remaining case concerns the reduct $\langle T \rangle (((\text{Succ})^n 0) \langle F \rangle) Q_n$. It is the only “non-degenerate” one, in the sense that it is where the accordion-like behaviour of A is illustrated: the sub-term $\langle T \rangle$ here is really “the same” as the one appearing at the root of \bar{A} but we need to reduce this sub-term at some point (i.e. to “compress” the Accordion). Thus there can be no 001-infinitary reduction towards \bar{A} . The formal proof of this case, i.e. of the impossibility of $(\text{Succ})^n 0 \langle F \rangle Q_n \longrightarrow_{\beta}^* (\langle F \rangle)^{n+1} M$, is given by Lemma 31 below. \blacktriangleleft

► **Lemma 30.** *For all $k \in \mathbb{N}$, $n \in \mathbb{N}$ and $M \in \Lambda$, there is no reduction*

$$\langle F \rangle^k Q_n \longrightarrow_{\beta}^* \langle F \rangle^{k+1} M.$$

Proof. We proceed by induction on k . First, take $k=0$ and suppose there is a reduction $Q_n \longrightarrow_{\beta}^* \langle F \rangle M$. By Lemma 29, there are $R, R' \in \Lambda$ such that

$$Q_n \longrightarrow_h^* (\lambda b. R) R' \longrightarrow_i^* \langle F \rangle M = (\lambda b. (b) F) M.$$

An exhaustive head reduction of Q_n gives the possible values of R and R' :

$$\begin{aligned} Q_n &= (Y) Q'_n \\ &\longrightarrow_h (\lambda x. (Q'_n)(x) x) \lambda x. (Q'_n)(x) x \\ &\longrightarrow_h (\lambda \psi. \lambda b. ((b)(P''))(\text{Succ})^{n+1} 0) \psi) Q''_n \\ &\longrightarrow_h \lambda b. ((b)(P''))(\text{Succ})^{n+1} 0) Q''_n, \end{aligned}$$

the last reduct being in HNF, which leaves only the first three possibilities. In any of those three cases, $R \longrightarrow_{\beta}^* (b) F$ (modulo renaming of b by α -conversion) is impossible by immediate arguments, so that $(\lambda b. R) R' \longrightarrow_i^* \langle F \rangle M$ cannot hold.

If $k \geq 1$, let us again suppose that there is a reduction $(\langle F \rangle)^k Q_n \rightarrow_{\beta}^* (\langle F \rangle)^{k+1} M$. Lemma 29 states that there are $R, R' \in \Lambda$ such that

$$(\langle F \rangle)^k Q_n \rightarrow_h^* (\lambda b. R) R' \rightarrow_i^* (\lambda b. (b)F) (\langle F \rangle)^k M.$$

An exhaustive head reduction of $(\langle F \rangle)^k Q_n$ gives the possible values of R and R' (we write only the reduction steps corresponding to the well-formed reducts – see the details in the detailed head reduction of \mathbf{A} , steps (15) and following):

$$\begin{aligned} (\langle F \rangle)^k Q_n &= (\lambda b. (b)F) (\langle F \rangle)^{k-1} Q_n \\ &\rightarrow_h^* (\lambda b. ((b)(P'')(\text{Succ})^{n+1} 0) Q_n'') F \\ &\rightarrow_h^* (\lambda y. y) Q_n'' \\ &\rightarrow_h Q_n'' \end{aligned}$$

In the first case, a reduction $(\lambda b. (b)F) (\langle F \rangle)^{k-1} Q_n \rightarrow_i^* (\lambda b. (b)F) (\langle F \rangle)^k M$ is impossible because it would imply that $(\langle F \rangle)^{k-1} Q_n \rightarrow_{\beta}^* (\langle F \rangle)^k M$, which is impossible by induction. The second and third cases are impossible by immediate arguments; the fourth case has already been explored (Q_n'' is exactly the term from the second line of the reduction of Q_n above). ◀

► **Lemma 31.** *For all $n \in \mathbb{N}$, $k \in [0, n]$ and $M \in \Lambda$, there is no reduction:*

$$(\text{Succ})^{n-k} 0 \langle F \rangle (\langle F \rangle)^k Q_n \rightarrow_{\beta}^* (\langle F \rangle)^{n+1} M.$$

Proof. We proceed by induction on $n - k$. The base case is $k = n$: if there is a reduction $(0) \langle F \rangle (\langle F \rangle)^n Q_n \rightarrow_{\beta}^* (\langle F \rangle)^{n+1} M$, then by Lemma 29 there are terms $R, R' \in \Lambda$ such that

$$(0) \langle F \rangle (\langle F \rangle)^n Q_n \rightarrow_h^* (\lambda b. R) R' \rightarrow_i^* (\lambda b. (b)F) (\langle F \rangle)^n M.$$

Observe that

$$(0) \langle F \rangle (\langle F \rangle)^n Q_n \rightarrow_h (\lambda x. x) (\langle F \rangle)^n Q_n \rightarrow_h (\langle F \rangle)^n Q_n$$

hence, because $\lambda x. x$ is in β -normal form and by Lemma 30, we reach a contradiction.

If $k < n$ and there is a reduction $(\text{Succ})^{n-k} 0 \langle F \rangle (\langle F \rangle)^k Q_n \rightarrow_{\beta}^* (\langle F \rangle)^{n+1} M$, then again by Lemma 29 there are terms $R, R' \in \Lambda$ such that

$$(\text{Succ})^{n-k} 0 \langle F \rangle (\langle F \rangle)^k Q_n \rightarrow_h^* (\lambda b. R) R' \rightarrow_i^* (\lambda b. (b)F) (\langle F \rangle)^n M.$$

Observe that

$$\begin{aligned} (\text{Succ})^{n-k} 0 \langle F \rangle (\langle F \rangle)^k Q_n &\rightarrow_h (\lambda f. \lambda x. (\text{Succ})^{n-k-1} 0 f(f)x) \langle F \rangle (\langle F \rangle)^k Q_n \\ &\rightarrow_h (\lambda x. (\text{Succ})^{n-k-1} 0 \langle F \rangle (\langle F \rangle)x) (\langle F \rangle)^k Q_n \\ &\rightarrow_h (\text{Succ})^{n-k-1} 0 \langle F \rangle (\langle F \rangle)^{k+1} Q_n \end{aligned}$$

The first reduct does not have the expected head form. In the second case, $(\lambda b. R) R' \rightarrow_i^* (\lambda b. (b)F) (\langle F \rangle)^n M$ would imply that $(\langle F \rangle)^k Q_n \rightarrow_{\beta}^* (\langle F \rangle)^n M$, which is impossible by Lemma 30 because $k < n$. In the third case, apply the induction hypothesis. ◀

5 The missing ingredient: Uniformity

The fact that the simulation of $\rightarrow_{\beta}^{001}$ by \rightarrow_r via the Taylor expansion is not conservative confirms that the pointwise reduction \rightarrow_r , even if needed in order to express the pointwise normal form of a sum through the resource reduction, weakens the dynamics of the β -reduction by allowing to reduce resource approximants along reductions paths that do not correspond to an actual reduction of the approximated term. As already underlined by Ehrhard and Regnier in their seminal work [19], *uniformity* is what gives the linear approximation all its robustness; this will also be the case for our study.

► **Definition 32.** *The relation $\circ \subset (!)\Lambda_r \times (!)\Lambda_r$ of **coherence** is defined by the rules:*

$$\frac{}{x \circ x} \quad \frac{s \circ s'}{\lambda x.s \circ \lambda x.s'} \quad \frac{s \circ s' \quad \bar{t} \circ \bar{t}'}{(s)\bar{t} \circ (s)\bar{t}'}$$

$$\frac{\forall i \in \{1, \dots, m\}, \forall j \in \{1, \dots, n\}, t_i \circ t'_j}{[t_1, \dots, t_m] \circ [t'_1, \dots, t'_n]} \quad (m, n \in \mathbb{N})$$

For $\mathbf{S}, \mathbf{T} \in \mathbb{S}^{(!)\Lambda_r}$, we write $\mathbf{S} \circ \mathbf{T}$ whenever $\forall s \in |\mathbf{S}|, \forall t \in |\mathbf{T}|, s \circ t$.

► **Definition 33.** *Given an index set I and a depth $d \in \mathbb{N}$, we define a relation $\curvearrowright_{r \geq d} \subset (!)\Lambda_r^I \times (\mathbb{N}^{(!)\Lambda_r})^I$ by the following rules:*

$$\frac{\forall i, j, s_i \circ s_j \quad \forall i, j, \bar{t}_i \circ \bar{t}_j}{((\lambda x.s_i)\bar{t}_i)_{i \in I} \curvearrowright_{r \geq 0} (s_i(\bar{t}_i/x))_{i \in I}} \quad \frac{(s_i)_{i \in I} \curvearrowright_{r \geq d} (S'_i)_{i \in I}}{(\lambda x.s_i)_{i \in I} \curvearrowright_{r \geq d} (\lambda x.S'_i)_{i \in I}}$$

$$\frac{(s_i)_{i \in I} \curvearrowright_{r \geq d} (S'_i)_{i \in I} \quad \forall i, j, \bar{t}_i \circ \bar{t}_j}{((s_i)\bar{t}_i)_{i \in I} \curvearrowright_{r \geq d} ((S'_i)\bar{t}_i)_{i \in I}} \quad \frac{(t_{i,j})_{\substack{i \in I \\ 1 \leq j \leq k_i}} \curvearrowright_{r \geq d} (T'_{i,j})_{\substack{i \in I \\ 1 \leq j \leq k_i}}}{([t_{i,1}, \dots, t_{i,k_i}])_{i \in I} \curvearrowright_{r \geq d} ([T'_{i,1}, \dots, T'_{i,k_i}])_{i \in I}}$$

$$\frac{\forall i, j, s_i \circ s_j \quad (\bar{t}_i)_{i \in I} \curvearrowright_{r \geq 0} (\bar{T}'_i)_{i \in I}}{((s_i)\bar{t}_i)_{i \in I} \curvearrowright_{r \geq 0} ((s_i)\bar{T}'_i)_{i \in I}} \quad \frac{\forall i, j, s_i \circ s_j \quad (\bar{t}_i)_{i \in I} \curvearrowright_{r \geq d} (\bar{T}'_i)_{i \in I}}{((s_i)\bar{t}_i)_{i \in I} \curvearrowright_{r \geq d+1} ((s_i)\bar{T}'_i)_{i \in I}}$$

The relation $\curvearrowright_{r \geq d} \subset \mathbb{S}^{(!)\Lambda_r} \times \mathbb{S}^{(!)\Lambda_r}$ of **uniform resource reduction at minimum depth d** is defined by

$$\frac{(u_i)_{i \in I} \curvearrowright_{r \geq d} (U'_i)_{i \in I}}{\sum_{i \in I} a_i u_i \curvearrowright_{r \geq d} \sum_{i \in I} a_i U'_i.}$$

We denote $\curvearrowright_{r \geq 0}$ and $\curvearrowright_{r \geq 0}$ simply by \curvearrowright_r and \curvearrowright_r , and call the latter **uniform resource reduction**.

The intuition behind \curvearrowright_r is that:

- it can only reduce “uniform” sums, i.e. sums containing resource terms that all have the same shape (formally, sums \mathbf{S} such that $\mathbf{S} \circ \mathbf{S}$),
- each reduction step of a sum is a “bundle” of resource reduction steps occurring at the same address in the elements of the sum (\curvearrowright_r is an inductive reformulation of Midez’ Γ -reduction [27]).

This allows to capture only the reductions of some $\mathcal{T}(M)$ that correspond to a β -reduction of M , as we will formally show. In fact, all the pointwise reductions \longrightarrow_r occurring in the proof of Theorem 12 are already instances of the particular case $\xrightarrow{*_r}$, hence the following reformulation.

► **Corollary 34** (of Theorem 12; [7], Lemma 4.50). *For all $M, N \in \Lambda$, if $M \longrightarrow_{\beta \geq d} N$ then $\mathcal{T}(M) \xrightarrow{*_r \geq d} \mathcal{T}(N)$.*

Let us show how this property can be used to build a conservative simulation of $\longrightarrow_{\beta}^{001}$. The simulating reduction needs to be:

- a restriction of \longrightarrow_r , because we want to eliminate the non-uniform reductions that cannot be turned into actual β -reductions,
- an extension of $\xrightarrow{*_r}$, because we want to be able to simulate not only finite, but also infinitary reductions.

The way we proceed is guided by the stratification property (Theorem 4).

► **Notation 35.** *The (applicative) **depth** of a resource term is the integer defined by*

$$\begin{aligned} \text{depth}(x) &:= 0 & \text{depth}((s)\bar{t}) &:= \max(\text{depth}(s), 1 + \text{depth}(\bar{t})) \\ \text{depth}(\lambda x.s) &:= \text{depth}(s) & \text{depth}([t_1, \dots, t_n]) &:= \max_{1 \leq i \leq n} \text{depth}(t_i). \end{aligned}$$

For all sum $\sum_{i \in I} a_i \cdot s_i \in \mathbb{S}^{\Lambda_r}$ and integer $d \in \mathbb{N}$, we write $\left(\sum_{i \in I} a_i \cdot s_i \right)_{< d} := \sum_{\substack{i \in I \\ \text{depth}(s_i) < d}} a_i \cdot s_i$.

► **Definition 36.** *The relation $\xrightarrow{*_r}^{\infty} \subset \mathbb{S}^{(!)\Lambda_r} \times \mathbb{S}^{(!)\Lambda_r}$ of **infinitary uniform resource reduction** is defined by writing $\mathbf{U} \xrightarrow{*_r}^{\infty} \mathbf{V}$ whenever there is a sequence $(\mathbf{U}_d)_{d \in \mathbb{N}}$ such that*

$$\mathbf{U}_0 = \mathbf{U} \quad \forall d \in \mathbb{N}, \mathbf{U}_d \xrightarrow{*_r \geq d} \mathbf{U}_{d+1} \quad \forall d \in \mathbb{N}, (\mathbf{U}_d)_{< d} = (\mathbf{V})_{< d}.$$

By design, $\xrightarrow{*_r}^{\infty}$ simulates the stratification of an infinitary β -reduction, hence the following property.

► **Corollary 37** (of Theorem 4 and Corollary 34). *For all $M, N \in \Lambda^{001}$, if $M \longrightarrow_{\beta}^{001} N$ then $\mathcal{T}(M) \xrightarrow{*_r}^{\infty} \mathcal{T}(N)$.*

Proof. We need to define a sequence $(\mathbf{U}_d)_{d \in \mathbb{N}}$ as in Definition 36. By stratification (Theorem 4), we obtain a sequence $(M_d)_{d \in \mathbb{N}}$ and we can define $\mathbf{U}_d := \mathcal{T}(M_d)$. The conclusion follows by Corollary 34 and by the fact that whenever $M \longrightarrow_{\beta \geq d}^{001} N$, then $(\mathcal{T}(M))_{< d} = (\mathcal{T}(N))_{< d}$. ◀

As announced, this simulation enjoys a converse conservativity property.

► **Theorem 38** (conservativity). *For $M, N \in \Lambda^{001}$, if $\mathcal{T}(M) \xrightarrow{*_r}^{\infty} \mathcal{T}(N)$ then $M \longrightarrow_{\beta}^{001} N$.*

The proof of the theorem goes as follows.

► **Lemma 39.** *For all $M, N \in \Lambda^{001}$ and $d \in \mathbb{N}$, if $\mathcal{T}(M) \xrightarrow{*_r \geq d} \mathcal{T}(N)$ then $M \longrightarrow_{\beta \geq d} N$.*

Proof. By an immediate induction on the reduction $(s)_{s \in |\mathcal{T}(M)|} \xrightarrow{*_r \geq d} (T_s)_{s \in |\mathcal{T}(M)|}$ induced by $\mathcal{T}(M) \xrightarrow{*_r \geq d} \mathcal{T}(N)$. ◀

► **Lemma 40.** *For all $M \in \Lambda^{001}$ and $\mathbf{S} \in \mathbb{S}^{\Lambda_r}$, if $\mathcal{T}(M) \xrightarrow{*_r} \mathbf{S}$ then there exists an $M' \in \Lambda^{001}$ such that $\mathbf{S} = \mathcal{T}(M')$.*

23:18 How to Play the Accordion

Proof. By an immediate induction on the reduction $(s)_{s \in |\mathcal{T}(M)|} \xrightarrow{r \geq d} (T_s)_{s \in |\mathcal{T}(M)|}$ induced by $\mathcal{T}(M) \xrightarrow{r \geq d} \mathbf{S}$. The base case relies on the same substitution lemma (Lemma 4.8 of [31]) as the proof of Theorem 12. ◀

► **Lemma 41.** Consider families $(u_i)_{i \in I} \in ((!) \Lambda_r)^I$ and $(V_i)_{i \in I} \in (\mathbb{N}^{(!) \Lambda_r})^I$ such that $(u_i)_{i \in I} \xrightarrow{r} (V_i)_{i \in I}$. For all $i, j \in I$, if $u_i = u_j$ then $V_i = V_j$.

Proof. By induction on $(u_i)_{i \in I} \xrightarrow{r} (V_i)_{i \in I}$. ◀

In particular, this lemma allows to change the index set I when writing a reduction $(u_i)_{i \in I} \xrightarrow{r} (V_i)_{i \in I}$, as soon as no u_i (and corresponding V_i) is erased or created – but duplications and erasures of duplicates are allowed.

► **Lemma 42.** For all $\mathbf{S}, \mathbf{T} \in \mathbb{S}^{\Lambda_r}$, if $\mathbf{S}^! \xrightarrow{r} \mathbf{T}^!$ then $\mathbf{S} \xrightarrow{r} \mathbf{T}$.

Proof. Suppose that $\mathbf{S}^! \xrightarrow{r} \mathbf{T}^!$. Thanks to Lemma 41, there is a derivation

$$\frac{\frac{\frac{(s_i)_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in |\mathbf{S}| \\ 1 \leq i \leq n}} \xrightarrow{r} (T_{s_i})_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in |\mathbf{S}| \\ 1 \leq i \leq n}}}{([s_1, \dots, s_n])_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in |\mathbf{S}|}} \xrightarrow{r} ([T_{s_1}, \dots, T_{s_n}])_{\substack{n \in \mathbb{N} \\ s_1, \dots, s_n \in |\mathbf{S}|}}}}{\sum_{n \in \mathbb{N}} \sum_{s_1, \dots, s_n \in |\mathbf{S}|} \underbrace{\frac{\prod_{i=1}^n a_{s_i}}{n!} \cdot [s_1, \dots, s_n]}_{\mathbf{S}^!} \xrightarrow{r} \sum_{n \in \mathbb{N}} \sum_{s_1, \dots, s_n \in |\mathbf{S}|} \underbrace{\frac{\prod_{i=1}^n a_{s_i}}{n!} \cdot [T_{s_1}, \dots, T_{s_n}]}_{\mathbf{T}^!}}$$

with $\mathbf{S} = \sum_{s \in |\mathbf{S}|} a_s \cdot s$. By Lemma 41 again, the hypothesis of the derivation is equivalent to $(s)_{s \in |\mathbf{S}|} \xrightarrow{r} (T_s)_{s \in |\mathbf{S}|}$, hence we can derive:

$$\frac{(s)_{s \in |\mathbf{S}|} \xrightarrow{r} (T_s)_{s \in |\mathbf{S}|}}{\mathbf{S} \xrightarrow{r} \sum_{s \in |\mathbf{S}|} a_s \cdot T_s.}$$

To see that $\mathbf{T} = \sum_{s \in |\mathbf{S}|} a_s \cdot T_s$, observe that

$$\begin{aligned} & \text{the coefficient of } t \text{ in } \mathbf{T} \\ &= \text{the coefficient of } [t] \text{ in } \mathbf{T}^! \\ &= \text{the coefficient of } [t] \text{ in } \sum_{n \in \mathbb{N}} \sum_{s_1, \dots, s_n \in |\mathbf{S}|} \frac{\prod_{i=1}^n a_{s_i}}{n!} \cdot [T_{s_1}, \dots, T_{s_n}] \\ &= \sum_{s \in |\mathbf{S}|} a_s \times \text{the coefficient of } t \text{ in } T_s \\ &= \text{the coefficient of } t \text{ in } \sum_{s \in |\mathbf{S}|} a_s \cdot T_s, \end{aligned}$$

which concludes the proof. ◀

Proof of theorem 38. Suppose that there is a sequence $(\mathbf{S}_d)_{d \in \mathbb{N}}$ such that

$$\mathbf{S}_0 = \mathcal{T}(M) \quad \forall d \in \mathbb{N}, \mathbf{S}_d \xrightarrow{r \geq d}^* \mathbf{S}_{d+1} \quad \forall d \in \mathbb{N}, (\mathbf{S}_d)_{< d} = (\mathcal{T}(N))_{< d}.$$

By Lemma 40 there is a sequence of terms $(M_d)_{d \in \mathbb{N}}$ such that $\forall d \in \mathbb{N}, \mathbf{S}_d = \mathcal{T}(M_d)$. We can take $M_0 = M$, and our hypotheses yield

$$\forall d \in \mathbb{N}, \mathcal{T}(M_d) \xrightarrow{r \geq d}^* \mathcal{T}(M_{d+1}) \tag{35}$$

$$\forall d \in \mathbb{N}, (\mathcal{T}(M_d))_{< d} = (\mathcal{T}(N))_{< d}. \tag{36}$$

For any sequence $(M_d)_{d \in \mathbb{N}}$ such that Equations (35) and (36) hold, we build a reduction $M_0 \rightarrow_{\beta}^{001} N$ by nested induction and coinduction on N .

- Case $N = x$. $(\mathcal{T}(M_1))_{<1} = (\mathcal{T}(N))_{<1} = x$ hence also $\mathcal{T}(M_1) = x$. As a consequence, $\mathcal{T}(M_0) \xrightarrow{r} x$ so by Lemma 39 $M_0 \xrightarrow{\beta}^* x$, which leads to the conclusion.
- Case $N = \lambda x.P'$. For all $d \geq 1$, $(\mathcal{T}(M_d))_{<d} = (\mathcal{T}(N))_{<d} = \lambda x.(\mathcal{T}(P'))_{<d}$ hence there is a term $P_d \in \Lambda^{001}$ such that $M_d = \lambda x.P_d$. We also define $P_0 := P_1$, so that $M_0 \xrightarrow{\beta}^* \lambda x.P_0$ by Equation (35) and Lemma 39.

The sequence $(P_d)_{d \in \mathbb{N}}$ satisfies Equations (35) and (36) wrt. P' , hence by induction we can build a reduction $P_0 \rightarrow_{\beta}^{001} P'$. We conclude with the rule (λ_{β}^{001}) from Definition 3.

- Case $N = (P')Q'$. For all $d \geq 1$, $(\mathcal{T}(M_d))_{<d} = (\mathcal{T}(N))_{<d} = ((\mathcal{T}(P'))_{<d})(\mathcal{T}(Q'))_{<d-1}$ hence there are terms $P_d, Q_d \in \Lambda^{001}$ such that $M_d = (P_d)Q_d$. We also define $P_0 := P_1$, so that $M_0 \xrightarrow{\beta}^* (P_0)Q_1$ by Equation (35) and Lemma 39.

By Equation (35), for all $d \geq 1$ there are reductions

$$\mathcal{T}(P_d) \xrightarrow{r \geq d}^* \mathcal{T}(P_{d+1}) \quad \text{and} \quad \mathcal{T}(Q_d)^! \xrightarrow{r \geq d-1}^* \mathcal{T}(Q_{d+1})^!.$$

From the first reduction we deduce that the sequence $(P_d)_{d \in \mathbb{N}}$ satisfies Equations (35) and (36) wrt. P' , hence by induction we can build a reduction $P_0 \rightarrow_{\beta}^{001} P'$. From the second reduction, by Lemma 42 we deduce that the sequence $(Q_{d+1})_{d \in \mathbb{N}}$ satisfies Equations (35) and (36) wrt. Q' : we apply rule $(@_{\beta}^{001})$ and proceed coinductively, through the guard (\triangleright) , to establish $Q_1 \rightarrow_{\beta}^{001} Q'$. ◀

In particular, observe that there is no reduction $\mathbf{A} \xrightarrow{r}^{\infty} \bar{\mathbf{A}}$: in the sequence of reductions given in Equation (29) in the proof of Theorem 27, item 1, all steps $\mathcal{T}(\mathbf{A}_d) \xrightarrow{r} \mathcal{T}(\mathbf{A}_{d+1})$ can be turned into $\mathcal{T}(\mathbf{A}_d) \xrightarrow{r}^* \mathcal{T}(\mathbf{A}_{d+1})$ (as explained in Footnote 4), but not into $\mathcal{T}(\mathbf{A}_d) \xrightarrow{r \geq d} \mathcal{T}(\mathbf{A}_{d+1})$ because there is always a reduction step occurring at depth 0.

We finally obtained a conservative approximation of the 001-infinitary λ -calculus. As a conclusive remark, let us mention that we did not take any \perp -reductions into account, though they are needed if one wants to simulate the reductions $M \rightarrow_{\beta \perp}^{001} \text{BT}(M)$ corresponding to Ehrhard and Regnier's commutation theorem. These reductions could be taken into account by adding the following rule:

$$\frac{\forall i, j, u_i \circ u_j \quad \forall i, u_i \xrightarrow{r}^* 0}{(u_i)_{i \in I} \xrightarrow{r \perp} (0)_{i \in I}}$$

to Definition 33. One would then be able to provide a conservative simulation of $\rightarrow_{\beta \perp}^{001}$ by $\xrightarrow{r \perp}^{\infty}$.

The question naturally arises whether this approach is transferrable to the richer λ -calculi already endowed with a linear approximation (as listed in the introduction). This remains unclear, since most of these settings are non-uniform, i.e. it is not true any more that $\mathcal{T}(M) \circ \mathcal{T}(M)$ in general. Investigating how existing techniques used to tame non-uniformity, e.g. in [31], can be exploited to address the conservativity problem in richer settings, remains an open line of research.

References

- 1 Davide Barbarossa. Resource approximation for the $\lambda\mu$ -calculus. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, 2022. doi:10.1145/3531130.3532469.

- 2 Davide Barbarossa and Giulio Manzonetto. Taylor Subsumes Scott, Berry, Kahn and Plotkin. In *47th Symposium on Principles of Programming Languages*, 2020. doi:10.1145/3371069.
- 3 Henk P. Barendregt. *The Lambda Calculus*. Elsevier, Amsterdam, 2 edition, 1984.
- 4 Henk P. Barendregt and Giulio Manzonetto. *A Lambda Calculus Satellite*. Number 94 in Mathematical logic and foundations. College publications, 2022.
- 5 Lison Blondeau-Patissier, Pierre Clairambault, and Lionel Vaux Auclair. Extensional Taylor Expansion, 2024. arXiv:2305.08489v2.
- 6 Rémy Cerda. Nominal algebraic-coalgebraic data types, with applications to infinitary λ -calculi. To appear in the proceedings of the 12th International Workshop on Fixed Points in Computer Science (FICS'24).
- 7 Rémy Cerda. *Taylor Approximation and Infinitary λ -Calculi*. PhD thesis, Aix-Marseille Université, 2024. URL: <https://hal.science/te1-04664728>.
- 8 Rémy Cerda and Lionel Vaux Auclair. Finitary Simulation of Infinitary β -Reduction via Taylor Expansion, and Applications. *Logical Methods in Computer Science*, 19, 2023. doi:10.46298/LMCS-19(4:34)2023.
- 9 Jules Chouquet and Christine Tasson. Taylor expansion for call-by-push-value. In *28th EACSL Annual Conference on Computer Science Logic*, 2020. doi:10.4230/LIPICS.CSL.2020.16.
- 10 Ugo Dal Lago and Thomas Leventis. On the Taylor Expansion of Probabilistic Lambda Terms. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction*, 2019. doi:10.4230/LIPICS.FSCD.2019.13.
- 11 Daniel de Carvalho. Execution time of λ -terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science*, 28(7):1169–1203, 2017. doi:10.1017/s0960129516000396.
- 12 Aloÿs Dufour and Damiano Mazza. Böhm and Taylor for All! In *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*, 2024. doi:10.4230/LIPICS.FSCD.2024.29.
- 13 Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5):579–623, 2002. doi:10.1017/s0960129502003729.
- 14 Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005. doi:10.1017/S0960129504004645.
- 15 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming*, 2016. doi:10.1145/2967973.2968608.
- 16 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003. doi:10.1016/S0304-3975(03)00392-X.
- 17 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Electronic Notes in Theoretical Computer Science*, 123:35–74, 2005. doi:10.1016/j.entcs.2004.06.060.
- 18 Thomas Ehrhard and Laurent Regnier. Böhm Trees, Krivine's Machine and the Taylor Expansion of Lambda-Terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Logical Approaches to Computational Barriers*, pages 186–197. Springer, 2006. doi:10.1007/11780342_20.
- 19 Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2):347–372, 2008. doi:10.1016/j.tcs.2008.06.001.
- 20 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.
- 21 Martin Hyland. A syntactic characterization of the equality in some models for the lambda calculus. *Journal of the London Mathematical Society*, s2-12:361–370, 1976. doi:10.1112/jlms/s2-12.3.361.
- 22 Richard Kennaway, Jan Willem Klop, Ronan Sleep, and Fer-Jan de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175(1):93–125, 1997. doi:10.1016/S0304-3975(96)00171-5.

- 23 Axel Kerinec, Giulio Manzonetto, and Michele Pagani. Revisiting Call-by-value Böhm trees in light of their Taylor expansion. *Logical Methods in Computer Science*, 16:1860–5974, 2020. URL: <https://lmcs.episciences.org/6638>, doi:10.23638/LMCS-16(3:6)2020.
- 24 Axel Kerinec and Lionel Vaux Auclair. The algebraic λ -calculus is a conservative extension of the ordinary λ -calculus, 2023. arXiv:2305.01067, doi:10.48550/arXiv.2305.01067.
- 25 Jean-Louis Krivine. *Lambda-calcul, types et modèles*. Masson, 1990.
- 26 Damiano Mazza. An axiomatic notion of approximation for programming languages and machines, 2021. Unpublished. URL: <https://www.lipn.fr/~mazza/papers/ApxAxiom.pdf>.
- 27 Jean-Baptiste Midez. *Une étude combinatoire du lambda-calcul avec ressources uniforme*. PhD thesis, Aix-Marseille Université, 2014. URL: <http://www.theses.fr/2014AIXM4093>.
- 28 Gerd Mitschke. The standardization theorem for λ -calculus. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 25:29–31, 1979. doi:10.1002/malq.19790250104.
- 29 Dana Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1–2):411–440, 1993. doi:10.1016/0304-3975(93)90095-b.
- 30 Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.
- 31 Lionel Vaux. Normalizing the Taylor expansion of non-deterministic λ -terms, via parallel reduction of resource vectors. *Logical Methods in Computer Science*, 15:9:1–9:57, 2019. doi:10.23638/LMCS-15(3:9)2019.
- 32 Christopher P. Wadsworth. Approximate reduction and lambda calculus models. *SIAM Journal on Computing*, 7(3):337–356, 1978. doi:10.1137/0207028.

A Deterministic Approach to Shortest Path Restoration in Edge Faulty Graphs

Keerti Choudhary  

Department of Computer Science and Engineering, IIT Delhi, India

Rishabh Dhiman 

Department of Computer Science and Engineering, IIT Delhi, India

Abstract

Afek, Bremner-Barr, Kaplan, Cohen, and Merritt (PODC'01) in their seminal work on shortest path restorations demonstrated that after a single edge failure in a graph G , a *replacement shortest path* between any two vertices s and t , which avoids the failed edge, can be represented as the concatenation of two original shortest paths in G . They also showed that we cannot associate a canonical¹ shortest path between the vertex pairs in G that consistently allows for the replacement path (in the surviving graph) to be represented as a concatenation of these canonical paths. Recently, Bodwin and Parter (PODC'21) proposed a randomized tie-breaking scheme for selecting canonical paths for the “ordered” vertex pairs in graph G with the desired property of representing the replacement shortest path as a concatenation of canonical shortest-paths provided for ordered pairs.

An interesting open question is whether it is possible to provide a deterministic construction of canonical paths in an efficient manner. We address this question in our paper by presenting an $O(mn)$ time deterministic algorithm to compute a canonical path family $\mathcal{F} = \{P_{x,y}, Q_{x,y} \mid x, y \in V\}$ comprising of two paths per (unordered) vertex pair. Each replacement is either a PQ-path (of type $P_{x,y} \circ Q_{y,z}$), a QP-path, a QQ-path, or a PP-path. Our construction is fairly simple and is a straightforward application of independent spanning trees. We also present various applications of family \mathcal{F} in computing fault-tolerant structures.

2012 ACM Subject Classification Theory of computation \rightarrow Data structures design and analysis; Mathematics of computing \rightarrow Graph algorithms

Keywords and phrases Fault-tolerant Data-structures, Shortest Path Restoration, Replacement path

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.24

Funding *Keerti Choudhary*: The author is supported in part by Google India Algorithms Research grant 2021.

1 Introduction

In their seminal work on shortest path restorations, Afek, Bremner-Barr, Kaplan, Cohen, and Merritt [1] studied the question of how the structure of shortest paths in a graph changes when edges fail. They showed that, after failure of any single edge in a graph, a replacement shortest path between any two vertices s and t avoiding a failed edge can be represented as the concatenation of two original shortest paths in the graph.

► **Theorem 1** (Afek et al. [1]). *For any vertices s, t and any failing edge e in G , a replacement shortest path between s and t in $G \setminus e$ is a concatenation of two original shortest paths in G , namely, $\pi(s, x)$ and $\pi(x, t)$, where vertex x is a function of s, t , and e .*

Various works in the past have employed the restoration path theory to develop efficient solutions for problems such as distance sensitivity oracles [2, 7, 8, 9], replacement paths [6, 5], fault-tolerant distance preservers [4, 5], routing schemes [5], among others.

¹ In Afek et al. (1991), the term “set of base paths” is used to refer to what is termed as “canonical paths” here.

An f -fault-tolerant distance preserver for a graph G is a sparse subgraph H such that, after failure of at most f edges, the distances between a pre-specified set of vertex pairs in the surviving part of H still match the corresponding distances in the surviving part of G . They are formally defined as follows.

► **Definition 2** (Fault Tolerant Distance Preservers). *For a graph $G = (V, E)$ and demand pairs P , a subgraph $H = (V, E_H \subseteq E)$ of G is an f -fault tolerant distance preserver (f -preserver) for P if for any subset F of E of size at most f , we have*

$$\text{dist}(s, t, G \setminus F) = \text{dist}(s, t, H \setminus F), \quad \text{for all } (s, t) \in P.$$

For a set $S \subseteq V$, we say that H is a **source-wise distance preserver** for S if $P = S \times V$; and a **subset distance preserver** for S if $P = S \times S$.

Parter and Peleg [15] showed that for any n -vertex unweighted undirected graph, there exists a 1-fault-tolerant $S \times V$ preserver with $O(|S|^{1/2}n^{3/2})$ edges. They also showed that this bound is existentially tight. Specifically, there exist n -vertex graphs and a set of sources S such that any $S \times V$ fault-tolerant preserver must have $\Omega(|S|^{1/2}n^{3/2})$ edges. Later, Parter [14] extended this result by providing an upper bound of $O(|S|^{1/3}n^{5/3})$ for dual fault-tolerant $S \times V$ preservers in undirected graphs. Gupta and Khan [11] extended this to directed graphs. For general f , Bodwin, Grandoni, Parter, and Williams [4] presented a construction of an f -fault-tolerant $S \times V$ preserver with $\tilde{O}(|S|^{1/2^f}fn^{2-1/2^f})$ edges.

For the problem of computing subset distance preservers, Bodwin, Choudhary, Parter and Shahar [3] presented a construction of 1-fault-tolerant $S \times S$ distance preserver with $\tilde{O}(|S|n)$ edges. Recently, Bodwin and Parter [5] extended this work to general f failures by providing a construction of $(f + 1)$ -fault-tolerant $S \times S$ preserver with $\tilde{O}(fn^{2-1/2^f}|S|^{1/2^f})$ edges, for any $f \geq 0$. These constructions for $S \times S$ preservers are obtained by employing the work of Afek, Bremler-Barr, Kaplan, Cohen, and Merritt [1] (see Theorem 1) on the structure of shortest paths in fault-prone graphs.

To exploit Theorem 1, Bodwin and Parter [5] introduced a tie-breaking scheme for selecting shortest paths in graph $G \setminus F$. Specifically, they used the Isolation lemma [13] to show that if each edge in G is assigned a random integer weight from the range $[1, n^{f+c+2}]$, then with probability $1 - 1/n^c$, for any pair of vertices s and t , and for any set F of edges, where $|F| \leq f$, exactly one of the shortest paths from s to t in $G \setminus F$ will be the unique shortest path in the modified edge weighted graph $G \setminus F$.

The main drawback of this tie-breaking scheme is that it is randomized and requires an additional $n^{O(f)}$ time to derandomize. This raises the following natural question:

Question: Is it possible to bypass randomness and have an efficient deterministic construction of the canonical path family?

We address this question by presenting a deterministic algorithm that runs in $O(mn)$ time to compute a canonical path family $\mathcal{F} = \{P_{x,y}, Q_{x,y} \mid x, y \in V\}$ comprising of two paths for each (unordered) vertex pair in G . We show that each replacement path in G can take the form of a PQ-path (of type $P_{x,y} \circ Q_{y,z}$), a QP-path, a QQ-path, or a PP-path. Our construction is not only straightforward but also represents a simple yet effective application of independent spanning trees [10].

We present in this paper an efficient construction of the family \mathcal{F} (see Theorem 10) and also explore various applications of these structures.

Sourcewise Distance Preservers

We introduce a stronger notion of fault-tolerant distance preserving subgraphs, which we call $(f, 1)$ -preservers.

► **Definition 3** ($(f, 1)$ -Preservers). *For a graph $G = (V, E)$ and a set of demand pairs P , a subgraph $H = (V, E_H \subseteq E)$ of G is an $(f, 1)$ -preserver if, for every pair $(s, t) \in P$, every set F of edges of size at most f , and every edge e in G that satisfies $\text{dist}(s, t, G \setminus F) = \text{dist}(s, t, G \setminus (F \cup e))$, we have*

$$\text{dist}(s, t, H \setminus (F \cup e)) = \text{dist}(s, t, G \setminus (F \cup e)).$$

Note that for any $f \geq 0$, an $(f + 1)$ -preserver is also an $(f, 1)$ -preserver. However, an f -preserver is not necessarily an $(f, 1)$ -preserver.

Our first contribution is a construction of $(f, 1)$ -preservers in polynomial time that matches the size of the current best construction of f -preservers given by Bodwin, Grandoni, Parter, and Williams [4].

► **Theorem 4.** *Let $f \geq 1$ be a positive integer. For any undirected, unweighted n -vertex graph $G = (V, E)$ with a set of source vertices $S \subseteq V$, we can compute a $(f, 1)$ -source-wise preserver for G with $\tilde{O}(fn^{2-1/2^f} |S|^{1/2^f})$ edges in $O(fmn)$ time.*

Further, for $f = 0$, we can compute a $(0, 1)$ -source-wise preserver for G with $O(|S|n)$ edges in $O(m + n)$ time.

Subset Distance Preservers

We provide the following relation between $(f, 1)$ -preservers and $(f + 1)$ -fault-tolerant subset distance preservers.

► **Theorem 5.** *In any undirected graph $G = (V, E)$, a source-wise $(f, 1)$ -preserver H with respect to a set S is an $(f + 1)$ -fault-tolerant subset distance preserver for pairs in $S \times S$.*

Combined with the construction of $(f, 1)$ -preservers, this gives us a deterministic algorithm to compute subset distance preservers in polynomial time.

► **Theorem 6.** *For any n -vertex graph $G = (V, E)$, a set of source vertices $S \subseteq V$, and a fixed non-negative integer f , there is an $(f + 1)$ -fault-tolerant distance preserver of G for pairs in $S \times S$ with $\tilde{O}((f + 1)n^{2-1/2^f} |S|^{1/2^f})$ edges which can be computed in $O(fmn)$ time when $f \geq 1$, and $O(|S|m)$ time when $f = 0$.*

Prior to our work, Bodwin et al. [5] gave construction for such preservers that takes $O(n^{2-1/2^f} |S|^{1/2^f})$ space and requires $O(mn)$ computation time in the randomized setting. However, the deterministic variant of their algorithm had a time complexity of $n^{O(f)}$, which is polynomial only for constant values of f . Our work improves upon this by providing a deterministic algorithm with a time complexity of $O((f + 1)mn)$, almost matching the efficiency of the randomized version of [5].

Fault-tolerant Distance Labeling Scheme

A distance labeling scheme is an assignment of bit-string labels to each node of G such that we can recover $\text{dist}(s, t, G)$ by looking at only the labels at s and t . An f -fault-tolerant distance labeling scheme allows us to compute the distances even when a set F of edges of size at most f fail.

While it is possible to store the entire graph within the labels, our objective is to find a labeling of subquadratic size. Bodwin et al. [5] gave an f -fault-tolerant distance labeling scheme that requires $O(n^{2-1/2^f} \log n)$ bits per vertex and can be computed in $O(mn)$ time in the randomized setting, and in $n^{O(f)}$ time deterministically. We offer an improvement to this by employing our $(f, 1)$ -preservers, which allows for the computation of the labels deterministically in polynomial time.

► **Theorem 7.** *For any fixed nonnegative integer $f \geq 0$, and n -vertex unweighted undirected graph, there is an $(f + 1)$ -fault-tolerant distance labeling scheme that assigns each vertex a label of $O((f + 1)n^{2-1/2^f} \log n)$ bits that can be computed in $O(fmn)$ time each when $f \geq 1$, and $O(m)$ time when $f = 0$.*

2 Preliminaries

Given a directed graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, the following notations and definitions will be used throughout the paper. We omit the term G when graph is clear from context.

- $\text{dist}(x, y, G)$: Distance of node y from x in the graph G .
- $G \setminus F$: The graph obtained by removing the edges that lie in F from the graph G .
- $G \cup H$: The graph obtained by taking a union of the vertices and edges of G and H .
- $E(P)$: The edges lying in path P .
- $T[x, y]$: The path from vertex x to vertex y in tree T .
- $P[x, y]$: The subpath of path P lying between vertices x, y , assuming x precedes y on P .
- $\text{IN-EDGES}(v, G)$: The set of all edges incoming to v in G .
- (s, v) -cut-edge: An edge which lies on all (s, v) -paths and hence whose removal disconnects v from s .
- (s, v) -distance-cut edge: An edge that lies on all the (s, v) shortest paths and hence whose removal increases the distance from s to v .
- $\text{LASTE}(P)$: the last edge lying on the path P .

All graphs in this paper are undirected and unweighted, unless stated otherwise.

3 Fault Tolerant Preservers

3.1 Preservers for Source-wise setting

► **Theorem 8.** *Any undirected or directed graph $G = (V, E)$ with a positive integer $f \geq 1$ and a set $S \subseteq V$ of sources has a source-wise $(f, 1)$ -preserver H with $\tilde{O}(f|S|^{1/2^f} n^{2-1/2^f})$ edges, which can be computed in $O(fmn)$ time.*

We also improve the running time for $f = 0$.

► **Theorem 9.** *Any undirected or directed graph $G = (V, E)$ and a set $S \subseteq V$ of sources has a source-wise $(0, 1)$ -preserver H with $\tilde{O}(|S|n)$ edges which can be computed in $O(|S|m)$ time.*

We first establish the following theorem.

► **Theorem 10.** *Any directed/undirected graph $G = (V, E)$ with a destination vertex $t \in V$ can be processed in $O(m + n)$ time to implicitly compute, for each $s \in V$, a pair of (s, t) -shortest paths, denoted as $P_{s,t}$ and $Q_{s,t}$, which intersect solely at the (s, t) -distance-cut edges.*

Furthermore, if G is undirected then for any $s, t \in V$ and any $e \in E$, there exists a vertex $w \in V$ such that at least one of the four paths obtained by concatenating paths from the family $\{P_{s,w}, Q_{s,w}\} \times \{P_{w,t}, Q_{w,t}\}$ forms a replacement shortest path between s and t in $G \setminus e$.

In order to prove Theorem 10, we use the concept of independent trees. Given a directed graph G and a designated source r , a pair of trees T_1, T_2 rooted at r are said to be *independent trees*, if for each $v \neq r$, the paths from v to r in T_1 and T_2 intersect only at the (v, r) -cut-edges.

► **Theorem 11** (Georgiadis and Tarjan [10]). *Given a directed graph G and a designated source r , a pair of independent trees T_1, T_2 rooted at r are computable in $O(m + n)$ time.*

We are now ready to prove Theorem 10.

Proof of Theorem 10. We begin by proving the first claim. Consider a directed acyclic graph (DAG) $D = (V, E_D \subseteq E)$ containing only those edges $e = (x, y) \in E$ for which $\text{dist}(x, t, G) = \text{dist}(y, t, G) + 1$. Let T_1 and T_2 be a pair of independent trees rooted at t in D . Then, for any vertex $v \in V$, the paths from v to t in T_1 and T_2 intersect only at the (v, t) -cut edges in D . By Theorem 11, the time required to compute T_1 and T_2 is $O(m + n)$.

For each $s \in V$, we define $P_{s,t}$ and $Q_{s,t}$ to be the paths $T_1[s, t]$ and $T_2[s, t]$, respectively. Note that there is a one-to-one correspondence between (v, t) -cut edges in D and (v, t) -distance-cut edges in G . Therefore, $P_{s,t}$ and $Q_{s,t}$ are the shortest (s, t) -paths in G , and they intersect only at the edges whose failure increases the (s, t) -distance in G .

Next, we prove the second claim. By Theorem 1, for any pair $s, t \in V$ and any failing edge e , there exists a vertex w such that:

1. $\text{dist}(s, w, G \setminus e) = \text{dist}(s, w, G)$,
2. $\text{dist}(w, t, G \setminus e) = \text{dist}(w, t, G)$,
3. $\text{dist}(s, t, G \setminus e) = \text{dist}(s, w, G) + \text{dist}(w, t, G)$.

This implies that e is not a distance cut-edge for the pairs (s, w) and (w, t) . Therefore, at least one of the paths from $P_{s,w}, Q_{s,w}$ must avoid e , and similarly, at least one of the paths from $P_{w,t}, Q_{w,t}$ must avoid e .

Thus, we conclude that at least one of the four paths obtained by concatenating paths from the family $\{P_{s,w}, Q_{s,w}\} \times \{P_{w,t}, Q_{w,t}\}$ forms a replacement shortest path between s and t in $G \setminus e$. ◀

Proof of Theorem 9. For a source $s \in S$, let T_1^s and T_2^s be the independent trees constructed in the proof of Theorem 10. Then the graph $\bigcup_{s \in S} (T_1^s \cup T_2^s)$ is a $(0, 1)$ -preserver. The correctness is a corollary of the earlier proof. ◀

In order to compute H in Theorem 8 we associate a set E_t of edges incident to each node $t \in V$, so that the graph $H = \bigcup_{t \in V} E_t$ obtained by taking the union of edges lying in E_t is an $(f, 1)$ -preserver.

Our construction is inspired by FT-BFS algorithm of Bodwin et al. [4], and the running time matches the time taken by their construction of f -FT-BFS. Let S be the source set. For each $s \in S$, we compute a pair of (s, t) -shortest-paths, say $P_{s,t}$ and $Q_{s,t}$, using Theorem 10.

Now we set $L = \sqrt{8f|S|n \log n}$, and compute a uniformly random subset R of $V \setminus \{t\}$ of size L . Further, for $s \in S$, let

$$W_s = \{u \in V \mid 1 \leq \text{dist}(u, t, P_{s,t} \cup Q_{s,t}) \leq 8nf \log n / L\}.$$

Finally, with respect to $(\bigcup_{s \in S} W_s) \cup R$ as the source, we compute an $(f - 1, 1)$ -preserver for the graph $G_0 = (V, E \setminus \bigcup_{s \in S} (E(P_{s,t}) \cup E(Q_{s,t})))$ and designate E_t as the set of in-edges of t lying in this $(f - 1, 1)$ -preserver. To compute a $(0, 1)$ -preserver, we simply take E_t to be the last edges of $E(P_{s,t})$ and $E(Q_{s,t})$. This completes the description of our algorithm. For pseudocode see Algorithm 1.

■ **Algorithm 1** COMPUTE-INCIDENT-EDGES(G, S, t, f).

```

1 if  $f = 0$  then
2   | Return  $\{\text{LASTE}(P_{s,t}), \text{LASTE}(Q_{s,t}) \mid s \in S\}$ ;
3 end
4  $L \leftarrow \sqrt{8f|S|n \log n}$ ;
5  $R \leftarrow$  uniformly random subset of  $V \setminus \{t\}$  of size  $L$ ;
6 for  $s \in S$  do
7   |  $(P_{s,t}, Q_{s,t}) \leftarrow$  Pair of  $(s, t)$ -shortest-paths computed using Theorem 10;
8   |  $W_s \leftarrow \{u \in V \mid 1 \leq \text{dist}(u, t, P_{s,t} \cup Q_{s,t}) \leq 8nf \log n/L\}$ 
9 end
10  $G_0 \leftarrow (V, E \setminus \bigcup_{s \in S} E(P_{s,t}) \cup E(Q_{s,t}))$ ;
11 Return COMPUTE-INCIDENT-EDGES( $G_0, (\bigcup_{s \in S} W_s) \cup R, t, f - 1$ );
    
```

► **Lemma 12** (Correctness). *The graph $H = (V, \bigcup_{t \in V} E_t)$, where E_t are sets computed using Algorithm 1, is an $(f, 1)$ -preserver of G with respect to the source S .*

Proof. We prove the correctness using induction on the value of f . Consider a source node $s \in S$, a set F of failing edges of size t most f , and an edge $e \in E \setminus F$ satisfying $\text{dist}(s, t, G \setminus F) = \text{dist}(s, t, G \setminus F \cup e)$. For $f = 0$, note that $F = \emptyset$, and on the failure of e , at least one of $P_{s,t}$ or $Q_{s,t}$ must remain intact. Therefore, we focus on the case where $|F| \geq 1$.

Let $P_{s,t,F}$ and $Q_{s,t,F}$ denote an *arbitrary* pair of (s, t) -shortest paths in $G \setminus F$ intersecting only at (s, v) -distance-cut edges. Further, let x and y be respectively the last vertices in $P_{s,t,F}$ and $Q_{s,t,F}$ lying in the structure $\bigcup_{r \in S} (P_{r,t} \cup Q_{r,t}) \setminus t$, where, $P_{r,t}$ and $Q_{r,t}$ are (r, t) -shortest-paths computed using Theorem 10. Observe that we can assume $(P_{s,t} \cup Q_{s,t})$ contains at least one edge from the set F , as otherwise, it suffices to keep the last edges of $P_{s,t}$, $Q_{s,t}$ in the set E_t .

Next, we compute a vertex \bar{x} lying in $P_{s,t,F}$ as follows. If x lies in W_s , then simply set $\bar{x} = x$. If x does not lie in W_s , then $|P_{s,t,F}[x, t]| = \text{dist}(x, t, G \setminus F) \geq \text{dist}(x, t, G) > L$, implying that with a high probability R contains a vertex of path $P_{s,t,F}[x, t]$. So, in this case, \bar{x} is set as an arbitrary vertex of R lying in $P_{s,t,F}[x, t]$. In a similar manner, \bar{y} lying in $Q_{s,t,F}$ can be computed. It must be noted that the internal vertices of suffixes $P_{s,t,F}[\bar{x}, t]$ and $Q_{s,t,F}[\bar{y}, t]$ are disjoint from the structure $\bigcup_{r \in S} (P_{r,t} \cup Q_{r,t}) \setminus t$.

Observe that on the failure of e in the graph $G \setminus F$, at least one of the paths $P_{s,t,F}$ or $Q_{s,t,F}$ must be intact. Without loss of generality assume that $P_{s,t,F}$ does not contain e . Due to sub-structure property of shortest paths, we have $\text{dist}(\bar{x}, t, G \setminus F) = \text{dist}(\bar{x}, t, G \setminus (F \cup e))$. Thus, $P_{s,t,F}[s, \bar{x}]$ concatenated with an (x, t) -shortest-path in $G \setminus F$ that is disjoint from e gives us an (s, t) -shortest-path in $G \setminus F \cup e$. Such an (\bar{x}, t) -shortest-path lies in $\bigcup_{t \in V} E_t$ as it contains incoming edges of t in an $(f - 1, 1)$ preserver of $(V, E \setminus \bigcup_{r \in S} E(P_{r,t}) \cup E(Q_{r,t}))$ with respect to $\bar{x} \in (\bigcup_{s \in S} W_s) \cup R$. ◀

In order to bound the size of E_t , we establish a recurrence relation on the size of E_t in the following lemma.

► **Lemma 13.** *Let $\mathcal{M}(f, z)$ denote an upper bound on the in-degree of a vertex t in an $(f, 1)$ -preserver of an n -vertex graph with respect to a source set comprising z vertices. Then,*

$$\mathcal{M}(f, z) \leq \mathcal{M}(f - 1, 3\sqrt{8fzn \log n}) .$$

Proof. In Algorithm 1, the size of the set $|W_s|$ is at most $2(8nf \log n/L)$ by Theorem 10, and $|R|$ is exactly L by definition. Since $\sum_{s \in S} |W_s| = 2L$, we have $|(\bigcup_{s \in S} W_s) \cup R| \leq 3L$. By correctness of the algorithm, we get the required recurrence. ◀

► **Lemma 14** (Size Analysis). $\mathcal{M}(f, |S|) = \tilde{O}(f|S|^{1/2^f} n^{1-1/2^f})$, and therefore, the number of edges in the $(f, 1)$ -preserver is $\tilde{O}(f|S|^{1/2^f} n^{2-1/2^f})$.

Proof. Let $\alpha_f = 3\sqrt{8f \log n}$, and $|S|$ be z . By using $\alpha_f \geq \alpha_{f-1}$ and that $\mathcal{M}(f, x)$ is an increasing function in x , we can resolve the recurrence in Lemma 13 as follows,

$$\mathcal{M}(f, z) \leq \mathcal{M}(0, \alpha_f^{2^{-1/2^{f-1}}} z^{1/2^f} n^{1-1/2^f}).$$

By the algorithm's description $\mathcal{M}(0, z) \leq 2z$, therefore,

$$\mathcal{M}(f, z) \leq 2\alpha_f^{2^{-1/2^{f-1}}} z^{1/2^f} n^{1-1/2^f} \leq 2\alpha_f^2 z^{1/2^f} n^{1-1/2^f} = \tilde{O}(f|S|^{1/2^f} n^{1-1/2^f}). \quad \blacktriangleleft$$

► **Lemma 15** (Running Time). *Algorithm 1 can be made to run in $O(fmn)$ time.*

Proof. Rather than explicitly computing $(P_{s,t}, Q_{s,t})$ in the algorithm, we compute the two trees T_1, T_2 once using Theorem 10 in $O(m+n)$. We discard all edges from T_1 and T_2 which don't lie on an s - t path, $\cup_{s \in S} W_s$ can then be computed as $\{u \in V \mid 1 \leq \text{dist}(u, t, T_1 \cup T_2) \leq 8nf \log n/L\}$ which takes $O(n)$ time. In the final iteration, we simply compute a $(0, 1)$ -preserver which takes $O(m+n)$ time. We require $f+1$ iterations of this and run it for each node t , hence, the algorithm takes $O(fmn)$ time. \blacktriangleleft

3.2 Subset Distance Preservers

In this subsection, we will provide an efficient construction of subset distance preserver. In particular, we will prove the following result.

► **Theorem 16.** *Given an n -vertex graph $G = (V, E)$, a set of source vertices $S \subseteq V$, and a nonnegative integer f , there is an $(f+1)$ -fault-tolerant distance preserver of G for all pairs in $S \times S$ on $\tilde{O}((f+1)n^{2-1/2^f} |S|^{1/2^f})$ edges which can be computed in $O((f+1)mn)$ time.*

Further, the computation time can be reduced to $O(|S|m)$ when $f = 0$.

We make use of the shortest path restoration theory by Afek et al. [1] to first prove the following.

► **Lemma 17.** *For any undirected graph $G = (V, E)$ and any nonnegative integer f , a source-wise $(f, 1)$ -preserver H of G with respect to a source set S is an $(f+1)$ -fault-tolerant subset distance preserver for pairs in $S \times S$.*

Proof. Let (F, e) be a pair such that $F \subseteq E$, has at most f edges and e is an edge in $E \setminus F$. Applying Theorem 1 on $G \setminus F$, for any $s, t \in S$ and failing edge e there exists a vertex w such that, (i) $\text{dist}(s, w, G \setminus (F \cup e)) = \text{dist}(s, w, G \setminus F)$, (ii) $\text{dist}(w, t, G \setminus (F \cup e)) = \text{dist}(w, t, G \setminus F)$, and (iii) $\text{dist}(s, t, G \setminus (F \cup e)) = \text{dist}(s, w, G \setminus F) + \text{dist}(w, t, G \setminus F)$.

For any $s, t \in S$ and the corresponding node w , $\text{dist}(s, w, H \setminus (F \cup e)) = \text{dist}(s, w, G \setminus F)$ and $\text{dist}(w, t, H \setminus (F \cup e)) = \text{dist}(w, t, G \setminus F)$ by Definition 3, since H is an $(f, 1)$ -preserver with respect to source S . Therefore,

$$\begin{aligned} \text{dist}(s, t, H \setminus (F \cup e)) &\leq \text{dist}(s, w, H \setminus (F \cup e)) + \text{dist}(w, t, H \setminus (F \cup e)) \\ &= \text{dist}(s, w, G \setminus F) + \text{dist}(w, t, G \setminus F) \\ &= \text{dist}(s, t, G \setminus (F \cup e)). \end{aligned}$$

However, since H is a subgraph of G , $\text{dist}(s, t, H \setminus (F \cup e)) \geq \text{dist}(s, t, G \setminus (F \cup e))$ as well, which implies $\text{dist}(s, t, H \setminus (F \cup e)) = \text{dist}(s, t, G \setminus (F \cup e))$. \blacktriangleleft

As a corollary of Theorem 8, Theorem 9, and Lemma 17, we get the construction in Theorem 16.

4 Other Applications

4.1 Distance Labeling Schemes

In this section, we make use of our $(f, 1)$ -preserver to provide an alternate construction for distance labels of sub-quadratic size.

► **Theorem 18.** *For any fixed nonnegative integer $f \geq 0$, and n -vertex unweighted undirected graph, there is an $(f + 1)$ -fault-tolerant distance labeling scheme that assigns each vertex a label of $O((f + 1)n^{2-1/2^f} \log n)$ bits that can be computed in $O(fmn)$ time each when $f \geq 1$, and $O(m)$ time when $f = 0$.*

Proof. Let H_s be an $(f, 1)$ -preserver with respect to source $\{s\}$. At each node s , store H_s as the label. Since the number of edges in H_s is $O((f + 1)n^{2-1/2^f})$ and it takes $O(\log n)$ bits to describe an edge, each label takes $O((f + 1)n^{2-1/2^f} \log n)$ bits.

To compute $\text{dist}(s, t, G \setminus F)$ for any $|F| \leq f + 1$, we read the labels of s and t to determine H_s and H_t , then union them together to get $H_{st} = H_s \cup H_t$. We then simply find the distance in $H_{st} \setminus F$.

H_{st} is a $(f, 1)$ -preserver with respect to the source $\{s, t\}$. By Lemma 17, H_{st} is an $(f + 1)$ -fault-tolerant distance-preserver for pairs in $\{s, t\} \times \{s, t\}$. Thus, $\text{dist}(s, t, H_{st} \setminus F) = \text{dist}(s, t, G \setminus F)$ as desired. ◀

4.2 Subset Replacement Path Algorithm

In the Subset Replacement Path problem (SUBSET-RP), the input is a graph $G = (V, E)$ and a set of source vertices S , and for every pair of vertices $s, t \in S$ and failing edge $e \in E$, report $\text{dist}(s, t, G \setminus e)$.

We modify the algorithm by Bodwin et al. [5] to use our $(0, 1)$ -preserver from Theorem 9 to solve the SUBSET-RP problem.

► **Theorem 19.** *Given a graph $G = (V, E)$ and a set of source vertices S , we can solve the SUBSET-RP problem in $O(|S|m) + \tilde{O}(|S|^2n)$ time in the word-RAM model.*

To this end, we will make use of the following result whose proof we will omit as we only make a black box use of it.

► **Theorem 20** (Hershberger and Suri [12]). *When $|S| = 2$, there is an algorithm that solve SUBSET-RP(G, S) in time $\tilde{O}(m + n)$.*

We now propose Algorithm 2 to solve the general problem.

► **Algorithm 2** Algorithm for SUBSET-RP(G, S).

```

1 for  $s \in S$  do
2   | Compute  $H_s \leftarrow (0, 1)$ -preserver of  $G$  with source  $s$ .
3 end
4 for  $s, t \in S$  do
5   | Solve SUBSET-RP( $H_s \cup H_t, \{s, t\}$ ) using Theorem 20 to get the result for  $(s, t)$ .
6 end

```

► **Lemma 21** (Correctness). *Algorithm 2 solves the SUBSET-RP(G, S) problem.*

Proof. $H_s \cup H_t$ is a $(0, 1)$ -preserver with respect to the source set $\{s, t\}$. By Lemma 17, $H_s \cup H_t$ is a 1-fault-tolerant distance-preserver for pairs in $\{s, t\} \times \{s, t\}$. Thus for any edge e , $\text{dist}(s, t, (H_s \cup H_t) \setminus e) = \text{dist}(s, t, G \setminus e)$. Thus, applying Theorem 20 on $H_s \cup H_t$ correctly computes $\text{dist}(s, t, G \setminus e)$ for all edges e . ◀

► **Lemma 22 (Running Time).** *Algorithm 2 runs in $O(|S|m) + \tilde{O}(|S|^2n)$ time.*



Proof. By Theorem 10, we can compute H_s in $O(m + n)$ time. Since H_s has $O(n)$ size, $\text{SUBSET-RP}(H_s \cup H_t, \{s, t\})$ can be solved in $\tilde{O}(n)$ time by Theorem 20. Therefore, the total time taken is $O(|S|m) + \tilde{O}(|S|^2n)$. ◀

References

- 1 Yehuda Afek, Anat Bremler-Barr, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by path concatenation: fast recovery of MPLS paths. *Distributed Computing*, 15(4):273–283, 2002. doi:10.1007/S00446-002-0080-6.
- 2 Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, Simon Krogmann, and Martin Schirneck. Approximate Distance Sensitivity Oracles in Subquadratic Space. In *Proceedings of the 55th Symposium on Theory of Computing (STOC)*, pages 1396–1409, 2023. doi:10.1145/3564246.3585251.
- 3 Greg Bodwin, Keerti Choudhary, Merav Parter, and Noa Shahar. New fault tolerant subset preservers. 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 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.15.
- 4 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 73:1–73:14, 2017. doi:10.4230/LIPICs.ICALP.2017.73.
- 5 Greg Bodwin and Merav Parter. Restorable shortest path tiebreaking for edge-faulty graphs. *J. ACM*, 70(5), October 2023. doi:10.1145/3603542.
- 6 Shiri Chechik and Sarel Cohen. Near optimal algorithms for the single source replacement paths problem. 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 2090–2109. SIAM, 2019. doi:10.1137/1.9781611975482.126.
- 7 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1 + \epsilon)$ -approximate f -sensitive distance oracles. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1479–1496. SIAM, 2017.
- 8 Dipan Dey and Manoj Gupta. Nearly optimal fault tolerant distance oracle. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 944–955. ACM, 2024. doi:10.1145/3618260.3649697.
- 9 Ran Duan and Hanlin Ren. Maintaining exact distances under multiple edge failures. 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 1093–1101. ACM, 2022. doi:10.1145/3519935.3520002.
- 10 Loukas Georgiadis and Robert Endre Tarjan. Dominators, directed bipolar orders, and independent spanning trees. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 375–386, 2012. doi:10.1007/978-3-642-31594-7_32.

- 11 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 127:1–127:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.127.
- 12 J. Hershberger and S. Suri. Vickrey prices and shortest paths: what is an edge worth? In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 252–259, 2001. doi:10.1109/SFCS.2001.959899.
- 13 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 345–354. ACM, 1987. doi:10.1145/28395.383347.
- 14 Merav Parter. Dual failure resilient BFS structure. *arXiv*, 2015. arXiv:1505.00692.
- 15 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013. doi:10.1007/978-3-642-40450-4_66.



Local Density and Its Distributed Approximation

Aleksander Bjørn Christiansen  

Technical University of Denmark, Lyngby, Denmark

Ivor van der Hoog  

Technical University of Denmark, Lyngby, Denmark

Eva Rotenberg  

Technical University of Denmark, Lyngby, Denmark

Abstract

The densest subgraph problem is a classic problem in combinatorial optimisation. Graphs with low maximum subgraph density are often called “uniformly sparse”, leading to algorithms parameterised by this density. However, in reality, the sparsity of a graph is not necessarily uniform. This calls for a formally well-defined, fine-grained notion of density.

Danisch, Chan, and Sozio propose a definition for *local density* that assigns to each vertex v a value $\rho^*(v)$. This local density is a generalisation of the maximum subgraph density of a graph. I.e., if $\rho(G)$ is the subgraph density of a finite graph G , then $\rho(G)$ equals the maximum local density $\rho^*(v)$ over vertices v in G . They present a Frank-Wolfe-based algorithm to approximate the local density of each vertex with no theoretical (asymptotic) guarantees.

We provide an extensive study of this local density measure. Just as with (global) maximum subgraph density, we show that there is a dual relation between the local out-degrees and the minimum out-degree orientations of the graph. We introduce the definition of the local out-degree $g^*(v)$ of a vertex v , and show it to be equal to the local density $\rho^*(v)$. We consider the local out-degree to be conceptually simpler, shorter to define, and easier to compute.

Using the local out-degree we show a previously unknown fact: that existing algorithms already dynamically approximate the local density for each vertex with polylogarithmic update time. Next, we provide the first distributed algorithms that compute the local density with provable guarantees: given any ε such that $\varepsilon^{-1} \in O(\text{poly } n)$, we show a deterministic distributed algorithm in the LOCAL model where, after $O(\varepsilon^{-2} \log^2 n)$ rounds, every vertex v outputs a $(1 + \varepsilon)$ -approximation of their local density $\rho^*(v)$. In CONGEST, we show a deterministic distributed algorithm that requires $\text{poly}(\log n, \varepsilon^{-1}) \cdot 2^{O(\sqrt{\log n})}$ rounds, which is sublinear in n .

As a corollary, we obtain the first deterministic algorithm running in a sublinear number of rounds for $(1 + \varepsilon)$ -approximate densest subgraph detection in the CONGEST model.

2012 ACM Subject Classification Theory of computation \rightarrow Graph algorithms analysis; Theory of computation \rightarrow Distributed algorithms

Keywords and phrases Distributed graph algorithms, graph density computation, graph density approximation, network analysis theory

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.25

Related Version *Full Version:* <https://arxiv.org/abs/2411.12694>

Funding This work was supported by the the VILLUM Foundation grant (VIL37507) “Efficient Recomputations for Changeful Problems”, the Independent Research Fund Denmark grant 2020-2023 (9131-00044B) “Dynamic Network Analysis”, and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987.



© Aleksander Bjørn Christiansen, Ivor van der Hoog, and Eva Rotenberg;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 25; pp. 25:1–25:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

Density or sparsity measures of graphs are widely studied and have many applications. Examples include the arboricity, the degeneracy, and the maximum subgraph density, all of which are asymptotically related within a factor of 2. Given a graph or subgraph H , its density, $\rho(H)$, is the average number of edges per vertex in H . The *maximum subgraph density* $\rho^{\max}(G)$ of a graph G is the maximum density $\rho(H)$ amongst all subgraphs $H \subseteq G$.

Computing maximum subgraph density has been studied both in the dynamic [10, 27], streaming [1, 4] and distributed [15, 28] setting. Often these measures are used to parameterise the sparsity of “uniformly sparse graphs” [2, 8, 25]. These measures are global measures in the sense that they measure the sparsity of the most dense part of the graph. In many cases the graph is not equally sparse (or dense) everywhere. Consider for example a lollipop graph: a large clique joined to a long path. The clique is a subgraph of high density, yet the vertices along the path sit in a part of the graph that is significantly less dense. Often, solutions for graph density related problems provide guarantees based on the most dense part of the graph. In some areas of computation more “local” solutions are desirable. Prior works of a global nature often completely disregard certain parts of the graphs, meaning that the output in sparser parts holds little to no information. We give three examples:

1) Many dynamic algorithms for estimating the subgraph density rely on modifying the solution locally. Algorithmic performance is expressed in terms of (global) graph sparsity, and thus fails to exploit the more fine-grained guarantee that local sparse areas yield.

2) In network analysis, one is often interested in determining dense subgraphs as these subgraphs can be interpreted, for instance, as communities within a social network. However, since many classical algorithms are tuned towards only detecting the densest subgraphs, these algorithms might fail to detect communities in sparser parts of the network [14, 26, 29].

3) Computing the maximum subgraph density is not very local, nor distributed. We consider the models LOCAL and CONGEST, and the lollipop graph. Here, almost instantly, the vertices of the clique realise they are part of a (very dense) clique. The vertices on the path may have to wait for diameter-many rounds before realising the maximum subgraph density of the graph. Distributed algorithms that wish to compute the *value* of the subgraph density are thus posed with a choice: either use $\Omega(D)$ rounds (where D is the diameter of the graph), or let every vertex output a value that is at most the maximum subgraph density.

1.1 Local density and results

We consider the definition of *local density* $\rho^*(v)$ by Danisch, Chan, and Sozio [14], defined at each node v of the graph (Definition 7). Our contributions can be split into four categories, which we present in four sections with corresponding titles.

A: Conceptual results for local density. Our primary contribution is an extensive overview of the theoretical properties of this local density measure. We show that, just as in the maximum-subgraph density problem, computing local density has a natural dual problem as we define the *local out-degree*. Consider a (fractional) orientation of the graph that is *locally fair*. i.e., for each directed edge (u, v) , the out-degree $g(u)$ is at most $g(v)$. We prove that for each vertex v , the out-degree of v has the same value over all locally fair orientations. We define this value $g^*(v)$ as the local out-degree of v . We prove that the local density of each vertex is the dual of its local out-degree and thereby $g^*(v) = \rho^*(v)$. This new definition for local out-degree is considerably shorter than the definition for local density. It allows us to show some previously unknown interesting properties of local density:

B: Results for dynamic algorithms. We prove that in an *approximately fair* orientation (a definition by Chekuri et al. [10]) the out-degree $g(v)$ of each vertex is a $(1 + \varepsilon)$ -approximation of $g^*(v) = \rho^*(v)$ (Theorem 12). This implies a previously unknown fact: that there exist dynamic polylogarithmic algorithms [10, 13, 12] where each vertex v maintains a $(1 + \varepsilon)$ -approximation of its local density $\rho^*(v)$ as by Danisch, Chan, and Sozio [14].

C: Results in LOCAL. We show that each node v can obtain a $(1 + \varepsilon)$ -approximation of $\rho^*(v)$ by surveying its $O(\varepsilon^{-2} \log^2 n)$ -hop neighbourhood. This induces a LOCAL algorithm where each vertex $v \in V$ computes a $(1 + \varepsilon)$ -approximation of $\rho^*(v)$ in $O(\varepsilon^{-2} \log^2 n)$ rounds.

Commentary on runtime: We observe that $\rho^{\max}(G)$ can be computed in LOCAL in $O(\varepsilon^{-1} \log n)$ rounds. In contrast, the stricter local density is computed in $O(\varepsilon^{-2} \log^2 n)$ rounds. This gap may be explained by considering the local density $\rho^*(v)$ for low-local-density vertices v in a graph that has high global density. The local density of v can be affected by a dense subgraph within a hop distance of $\Theta(\varepsilon^{-2} \log^2 n)$ (although it unclear if it can be affected *enough* to prohibit a $(1 + \varepsilon)$ -approximation of $\rho^*(v)$ in $o(\varepsilon^{-2} \log^2 n)$ time). The potential for a barrier of $O(\varepsilon^{-2} \log^2 n)$ rounds is also illustrated by existing dynamic algorithms [10, 28] that maintain η -fair orientations. In this scenario, these algorithms have a worst-case recourse of $\Omega(\varepsilon^{-2} \log^2 n)$. We consider it an interesting open problem to either improve our running time in LOCAL, or, show that $O(\varepsilon^{-2} \log^2 n)$ is tight.

D: Results in CONGEST. We show a significantly more involved algorithm in CONGEST, where after $O(\text{poly}\{\varepsilon^{-1}, \log n\} \cdot 2^{O(\sqrt{\log n})})$ rounds, each vertex v outputs a $(1 + \varepsilon)$ -approximation of $\rho^*(v)$. Since $\max_{v \in V} \rho^*(v) = \rho^{\max}(G)$, this is the first deterministic algorithm for $(1 + \varepsilon)$ -approximating of the global subgraph density $\rho^{\max}(G)$ in CONGEST, that runs in a number of rounds that is sublinear in the diameter of the graph.

In the main body, we focus on the value variant where we want each vertex v to output an approximation of $\rho^*(v)$. In the full version, we extend our analysis so that each vertex v can output a subgraph H with $v \in H$ where $\rho(H)$ approximates $\rho^*(v)$. See also Table 1.

■ **Table 1** Results in LOCAL (L) or CONGEST (C) where prior work for computing the global subgraph density is compared to our running time for the local subgraph density. D denotes the diameter. Orange running times are not deterministic and occur with high probability.

Model	Problem	Each v outputs ρ_v with	Rounds	Source
L	2.1	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$	$\Theta(D)$	[28]
	2.2	$\max_v \rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$	$O(\varepsilon^{-1} \log n)$	[28]
	3	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$	$O(\varepsilon^{-2} \log^2 n)$	Cor. 15
C	2.1	$\rho_v \in [(2 + \varepsilon)^{-1} \rho^{\max}(G), (2 + \varepsilon) \rho^{\max}(G)]$	$O(D \cdot \varepsilon^{-1} \log n)$	[15]
	2.1	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$	$O(\varepsilon^{-4} \log^4 n + D)$ whp.	[28]
	2.2	$\max_v \rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$	$O(\varepsilon^{-4} \log^4 n)$ whp.	[28]
	2.2	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$	$O(\text{poly}\{\log n, \varepsilon^{-1}\}) \cdot 2^{O(\sqrt{\log n})}$	Thm. 16
	3	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$	$O(\text{poly}\{\log n, \varepsilon^{-1}\}) \cdot 2^{O(\sqrt{\log n})}$	Thm. 16
	3	$\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$	$O(\text{poly}\{\log n, \varepsilon^{-1}\})$ whp.	Thm. 16

2 Preliminaries and related work

Let $G = (V, E)$ be an undirected weighted graph with n vertices and m edges. For any $v \in V$ and any integer k , we denote by $H_k(v)$ the k -hop neighborhood of v . For each edge $e \in E$ we denote by $g(e)$ the *weight* of e . Any edge with endpoints u, v may be denoted as \overline{uv} .

25:4 Local Density and Its Distributed Approximation

We can augment any weighted graph G with a (*fractional orientation*). An orientation \vec{G} assigns to each edge \overline{uv} two positive real values: $g(u \rightarrow v)$ and $g(v \rightarrow u)$ such that $g(\overline{uv}) = g(u \rightarrow v) + g(v \rightarrow u)$. These values may be interpreted as pointing a fraction of the edge \overline{uv} from u to v , and the other fraction from v to u . Given an orientation \vec{G} , we denote by $g(u) = \sum_{v \in V} g(u \rightarrow v)$ the *out-degree* of u (i.e., how much fractional edges point outwards from u in \vec{G}). Given these definitions, we can consider two graph measures of G : the maximum subgraph density and the minimum orientation of G .

Global graph measures. For any subgraph $H \subseteq G$, its *density* $\rho(H)$ is defined as $\rho(H) = \frac{1}{|V(H)|} \sum_{e \in E(H)} g(e)$. The *maximum subgraph density* $\rho^{\max}(G)$ is then the maximum over all $H \subseteq G$ of $\rho(H)$. A subgraph $H \subseteq G$ is *densest* whenever $\rho(H) = \rho^{\max}(G)$. For any orientation \vec{G} of G , its *maximum out-degree* $\Delta(\vec{G})$ is the maximum over all u of the out-degree $g(u)$. The *optimal out-degree* of G , denoted by $\Delta^{\min}(G)$, is subsequently the minimum over all \vec{G} of $\Delta(\vec{G})$. An orientation \vec{G} itself is *minimum* whenever $\Delta(\vec{G}) = \Delta^{\min}(G)$.

The density of G and the optimal out-degree are closely related. One way to illustrate this is through the following dual linear programs:

DS (Densest Subgraph)		FO (Fractional Orientation)
$\max \sum_{\overline{uv} \in E} g(\overline{uv}) \cdot y_{u,v}$	s.t.	$\min \rho$ s.t.
$x_u, x_v \geq y_{u,v} \quad \forall \overline{uv} \in E$		$g(u \rightarrow v) + g(v \rightarrow u) \geq g(\overline{uv}) \quad \forall \overline{uv} \in E$
$\sum_{v \in V} x_v \leq 1$		$\rho \geq \sum_{v \in V} g(u \rightarrow v) \quad \forall u \in V$
$x_v, y_{u,v} \geq 0 \quad \forall u, v \in V$		$g(u \rightarrow v), g(v \rightarrow u) \geq 0 \quad \forall u, v \in V$

Denote by R the optimal value of DS and by Δ the optimal value of FO. By duality, $R = \Delta$. Moreover, Charikar [9] relates these two linear programs to the densest subgraph problem:

► **Theorem 1** (Theorem 1 in [9]). *Let G be a unit weight graph. Denote by R the optimal solution of DS and by D the optimal solution of FO. Then $\rho^{\max}(G) = R = \Delta = \Delta^{\min}(G)$.*

We show that this can be generalised to when G is a weighted graph:

► **Lemma 2** (See the full version). *Let G be any weighted graph. Denote by R the optimal solution of DS and by D the optimal solution of FO. Then $\rho^{\max}(G) = R = D = \Delta^{\min}(G)$.*

2.1 Densest subgraph in dynamic algorithms

In a classical, non-distributed model of computation we can immediately formalise both the value variant of the (approximate) densest subgraph:

► **Problem 1.** *Given a graph G and an $\varepsilon > 0$, output $\rho' \in [(1+\varepsilon)^{-1}\rho^{\max}(G), (1+\varepsilon)\rho^{\max}(G)]$.*

Alternatively, in the Fractional Orientation (FO) problem the goal is to output a $(1 + \varepsilon)$ -approximation of $\Delta^{\min}(G)$. It turns out that FO is a more accessible problem to study. The LP formulations allow for a straightforward way to compute $\Delta^{\min}(G)$ and/or $\rho^{\max}(G)$. However, solving the LP requires information about the entire graph, and this information is expensive to collect. Sawlani and Wang [27] get around this difficulty by instead solving an approximate version of (FO). They work with a concept we call *local fairness*.

► **Definition 3.** Let \vec{G} be a fractional orientation of a graph G . We say that \vec{G} is locally fair whenever $g(u \rightarrow v) > 0$ implies $g(u) \leq g(v)$.

Chekuri et al. [10] extend this definition to η -fairness:

► **Definition 4.** Let \vec{G} be a fractional orientation of a graph G . We say that \vec{G} is η -fair (for $\eta > 0$) whenever $g(u \rightarrow v) > 0$ implies that $g(u) \leq (1 + \eta)g(v)$.

Related work in dynamic algorithms. Chekuri et al. [10] continue to focus on computing a $(1 + \varepsilon)$ -approximation of the Densest Subgraph problem. They show that, if G is a unit weight graph, there exists a $(1 + \varepsilon)$ -approximate solution to FO that is η -fair (for some smartly chosen η). They subsequently prove that an η -fair orientation allows you to find a $(1 + \varepsilon)$ -approximate densest subgraph. This allows them to dynamically maintain the value of the densest subgraph of G in $O(\varepsilon^{-6} \log^3 n \log \rho^{\max}(G))$ time per insertion or deletion of edges in G . By leveraging the η -fairness of the orientation, they can report a $(1 + \varepsilon)$ -approximate densest subgraph in time proportional to the size of the subgraph.

2.2 Approximate densest subgraph in LOCAL and CONGEST

We focus on the *value* variant of the problem, where each vertex outputs a value (as opposed to the *reporting* variant in the full version, where the goal is to report a densest subgraph).

► **Problem 2.** Given a graph G and an $\varepsilon > 0$, each vertex v outputs a value ρ_v and either:

- **Problem 2.1:** we require that $\forall v, \rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$, or
- **Problem 2.2:** we require that $\max_v \rho_v \in [(1 + \varepsilon)^{-1} \rho^{\max}(G), (1 + \varepsilon) \rho^{\max}(G)]$.

Related work. Problem 2.1 has a trivial $\Omega(D)$ lower bound, obtained by constructing a lollipop graph (where D denotes the diameter). In LOCAL, it is trivial to solve Problem 2.1 in $\Theta(D)$ time. Problem 2.2 was studied by Ghaffari and Su [20] who present a randomised $(1 + \varepsilon)$ -approximation in LOCAL that uses $O(\varepsilon^{-3} \log^4 n)$ rounds. Fischer et al. [16] present a deterministic $(1 + \varepsilon)$ -approximation in LOCAL that uses $2^{O(\log^2(\varepsilon^{-1} \log n))}$ rounds. Ghaffari et al. [19] improve this to $O(\varepsilon^{-9} \log^{15} n)$ rounds. The work by Harris [24] improves this to $\tilde{O}(\varepsilon^{-6} \log^6 n)$ rounds. Su and Vu [28] present the state-of-the-art in this area. They prove that for any graph G , there exists a vertex v such that for the k -hop neighbourhood $H_k(v)$ (with $k \in O(\varepsilon^{-1} \log n)$) the density $\rho^{\max}(H_k)$ is a $(1 + \varepsilon)$ -approximation of $\rho^{\max}(G)$. This immediately leads to a trivial LOCAL algorithm: each vertex u collects its k -hop neighbourhood $H_k(u)$ in $O(\varepsilon^{-1} \log n)$ rounds, solves the LP of Densest Subgraph in its own node, and reports the value $\rho^{\max}(H_k(u))$.

In CONGEST, the state-of-the-art deterministic algorithm for Problem 2.1 and 2.2 is by Das Sarma et al. [15] who present a $(2 + \varepsilon)$ -approximation in $O(D \cdot \varepsilon^{-1} \log n)$ rounds. The best randomised work is by Su and Vu [28] who present a randomised algorithm for Problem 2.2 that runs in $O(\varepsilon^{-4} \log^4 n)$ rounds w.h.p. See also Table 1 for an overview.

2.3 Local density

Danisch, Chan, and Sozio [14] introduce a more local measure which they call the *local density*. Its lengthy definition assigns to each vertex v a value. We note for the reader that we almost immediately define our local out-degree (Definition 8), and only use local out-degree in proofs. Hence, the reader is not required to have a thorough understanding of the following:

► **Definition 5** (Definition 2.2 in [14]). Let $G = (V, E)$ be a weighted graph where an edge e has weight $g(e)$. Let $B \subseteq V$. For any $X \subseteq V - B$, we define the quotient edges $\hat{E}_B(X)$ as all edges in G with one endpoint in X , and the other endpoint in X or B . We define:

- for $X \subseteq V - B$, the quotient subgraph density $\hat{\rho}_B(X) := \frac{1}{|X|} \sum_{e \in \hat{E}_B(X)} g(e)$.
- the maximum quotient density $\hat{\rho}_B(G) := \max_{X \subseteq V - B} \hat{\rho}_B(X)$.

► **Definition 6** (Definition 2.3 in [14]). Given a weighted undirected graph $G = (V, E)$, we define the diminishing-dense decomposition \mathcal{B} of G as the sequence $B_0 \subset B_1 \dots \subset B_\ell = V$:

We define $B_0 = \emptyset$. For $i \geq 1$ if $B_{i-1} = V$ then $\ell := i$. Otherwise:

$$S_i := \arg \max_{X \subseteq V - B_{i-1}} \hat{\rho}_{B_{i-1}}(X), \text{ and } B_i := B_{i-1} \cup S_i.$$

► **Definition 7** (Definition 2.3 in [14]). Given a weighted undirected graph $G = (V, E)$ and a diminishing-dense decomposition \mathcal{B} , each vertex $v \in V$ has one integer i where $v \in S_i$. We define the local density $\rho^*(v) := \hat{\rho}_{B_{i-1}}(S_i)$.

The benefit of local measures. Problem 2's variants have drawbacks in a distributed model of computation. Problem 2.1 has an $\Omega(d)$ lower bound (making it trivial in LOCAL). Problem 2.2 allows some vertices to output nonsense. The definition of local density alleviates these issues, as we may define an algorithmic problem which we consider to be more natural:

► **Problem 3.** Given (G, ε) , each vertex v outputs $\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$.

Related work. Danisch, Chan, and Sozio [14] introduce the local density (Definition 7). Intuitively, they define a partition $V_1 \dots V_k$ of V . For each i , they consider the set $X_i = \bigcup_{j=1}^i V_j$. They then define the graph H_i as the vertex-induced subgraph of X_i where for each edge in G between a vertex in X_i and a vertex not in X_i , they add a self loop. They define the *quotient density* of V_i as the density of H_i . The local density $\rho^*(v)$ of a vertex v is the quotient density of V_i for $v \in V_i$. Danisch, Chan, and Sozio [14] then define a quadratic program \mathbf{FO}^2 . The domain of this program is the space of all orientations of the graph G . The cost function is the sum over all vertices u , of the out-degree squared. Consider an orientation \vec{G} of G that optimises the cost function. Danisch, Chan, and Sozio show that for each vertex v its out-degree $g(v)$ equals the local density $\rho^*(v)$. As a consequence, over all optimal solutions to \mathbf{FO}^2 , the out-degree of each vertex is unique. They provide a Frank-Wolfe based algorithm to solve the quadratic program with no theoretical guarantees.

Chekuri, Harb, and Quanrud [22] study computing the local density by solving the quadratic program. They show [22, Theorem 3.4] some interesting properties of \mathbf{FO}^2 . Specifically, they show that the uniqueness property by Danish, Chan and Sozio is a special case of Fujishige's result [18] from 1980. They additionally show that the algorithm GREEDY++ by Boob, Gao, Peng, Sawlani, Tsourakakis, Wang, and Wang [5] (when applied to this quadratic program) converges to an orientation where the out-degree of each vertex v is a $(1 + \varepsilon)$ -approximation of $\rho^*(v)$. Chekuri, Harb, and Quanrud [23] subsequently show that the more general SUPER-GREEDY++ algorithm by Chekuri, Quanrud and Torres [11] also converges to $(1 + \varepsilon)$ -approximation of each $\rho^*(v)$.

Borradaile, Migler, and Wilfong [7] observe that there is a correlation between the fairness of an orientation and local density. An *integral orientation* is any orientation of the graph where for each edge (u, v) , $g(u \rightarrow v) \in \{0, 1\}$. They consider an *egalitarian orientation* which is defined as an integral orientation where “the total available out-degree is shared among

the vertices as equally as allowed by the topology of the graph”. An egalitarian orientation, and other equivalent notions of “fair integral orientations”, are formally defined through an integer flow problem on the graph [6, 17, 30]. From section 2 in [6] it follows that an integral orientation is egalitarian if and only if $g(u \rightarrow v) = 1$ implies that $g(u) \leq g(v) + 1$.

Using an egalitarian orientation, they create a decomposition of the graph that they call a *density decomposition*. The density decomposition by Borradaile, Migler, and Wilfong [7] is not equal to the local density. The analysis of Kopelowitz, Krauthgamer, Porat, and Solomon shows that if \vec{G} is an egalitarian orientation then $\max_v g(v) \leq \rho(G) + O(\log n)$. Their decomposition, compared to the one used for the local density, can thus be viewed as one that is coarser but similar in spirit. It can thereby be argued that Borradaile, Migler, and Wilfong [7] are the first to show a connection between the local density and fair orientations of a graph. However, we note that the integrality of their orientation prevents exact (or $(1 + \varepsilon)$ -approximate) computations of the local density.

3 Results and organisation

Now we are ready to formally state our contributions. Our primary contribution is that we show a dual definition to local density, which we call the local out-degree:

► **Definition 8.** *Given a graph $G = (V, E)$, we define the local out-degree as:*

$$g^*(u) := \text{the out-degree } g(u) \text{ in any locally fair fractional orientation of } G.$$

It is not immediately clear that the local out-degree is well-defined. We prove (Theorem 9) that each vertex in G has the same out-degree across all locally fair orientations of G (and thus, the set of all locally fair orientations of G assigns to each vertex a real value). We believe that the local out-degree is conceptually simpler than the local density. Through this definition, we are able to show various algorithms to approximate the local density.

3.A Conceptual results for local density

We prove in Section 4 that these local definitions generalise the global definition of subgraph density and out-degree, as they exhibit the same dual behaviour. We show several previously unknown properties of the local density, which we consider to be of independent interest:

► **Theorem 9.** *For any weighted graph G , $\forall v \in V$, $g^*(v)$ is well-defined and equals $\rho^*(v)$.*

► **Corollary 10.** *Given a weighted graph G , $\rho^{\max}(G) = \Delta^{\min}(G) = \max_v g^*(v)$.*

► **Corollary 11.** *For any graph G , there exists a fractional orientation \vec{G} that is locally fair.*

3.B Results for dynamic algorithms

We show in Section 5 that η -fair orientations imply approximations for our local measures:

► **Theorem 12.** *Let G be a weighted graph and \vec{G} be an η -fair fractional orientation for $\eta \leq \frac{\varepsilon^2}{128 \cdot \log n}$. Then $\forall v \in V$: $(1 + \varepsilon)^{-1} \rho^*(v) \leq g(v) \leq (1 + \varepsilon) \rho^*(v)$.*

This immediately implies the following Corollary by applying [10]:

► **Corollary 13.** *There exists an algorithm [10] that can fractionally orient a dynamic unit-weight graph G with n vertices subject to edge insertions and deletions with deterministic worst-case $O(\varepsilon^{-6} \log^4 n)$ update time such that for all $v \in V$:*

$$g(v) \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)].$$

3.C Results in LOCAL

The local density as a measure is not entirely local. However, we prove in Section 6 that far-away subgraphs affect the local density of a vertex v only marginally:

► **Theorem 14.** *Let G be a unit-weight graph. For any $\varepsilon > 0$ and vertex v , denote by $\rho^*(v)$ its local density and by $\rho_k^*(v)$ its local density in $H_k(v)$. Then $\rho_k^*(v) \in [(1+\varepsilon)^{-1}\rho^*(v), (1+\varepsilon)\rho^*(v)]$ for $k \in \Theta(\varepsilon^{-2} \log^2 n)$.*

This immediately implies a trivial algorithm for problem 2 in LOCAL (where each vertex v collects its k -hop neighbourhood $H_k(v)$ for $k \in \Theta(\varepsilon^{-2} \log^2 n)$ and then solves FO on $H_k(v)$):

► **Corollary 15.** *There exists an algorithm in LOCAL that given a unit graph G and $\varepsilon > 0$ computes in $O(\varepsilon^{-2} \log^2 n)$ rounds for all $v \in V$ a value $\rho_v \in [(1+\varepsilon)^{-1}\rho^*(v), (1+\varepsilon)\rho^*(v)]$.*

3.D Results in CONGEST

Finally in Section 7, we solve Problem 3 in CONGEST by computing an η -fair orientation. We use as a subroutine algorithm to compute *blocking flows* in an h -layered DAG [21]:

► **Theorem 16.** *Suppose one can compute a blocking flow in an n -node h -layered DAG in $\text{Blocking}(h, n)$ rounds. There exists an algorithm in CONGEST that given a unit-weight graph G and $\varepsilon > 0$ computes in $O(\varepsilon^{-3} \log^4 n \cdot (\varepsilon^{-2} \log^2 n + \text{Blocking}(\varepsilon^{-2} \log^2 n, n)))$ rounds an orientation \vec{G} such that for all $v \in V$: $g(v) \in [(1+\varepsilon)^{-1}\rho^*(v), (1+\varepsilon)\rho^*(v)]$.*

As a corollary, we obtain the first deterministic algorithm running in a sublinear number of rounds for $(1+\varepsilon)$ -approximate dense subgraph detection in the CONGEST model (Table 1).

4 Conceptual results for local density

Our primary contribution is the definition of local out-degree as a dual to local density.

► **Lemma 17.** *The local density is well-defined. That is, for any two locally fair orientations \vec{G} or \vec{G}' where a vertex u has out-degree $g(u)$ or $g'(u)$ respectively, $g(u) = g'(u)$.*

Proof. Suppose for the sake of contradiction that there exists two locally fair orientations (\vec{G}, \vec{G}') and a vertex $u \in V$ where $g(u) > g'(u)$. We define their *symmetric difference* graph S as a digraph where the vertices are V and there exists an edge \vec{ab} whenever $g(a \rightarrow b) > g'(a \rightarrow b)$. We may assume that S contains no directed cycles:

Indeed if S contains any directed cycle π we change \vec{G}' , where for all $\vec{ab} \in \pi$ we slightly increase $g'(a \rightarrow b)$ until S loses an edge. This operation does not change the out-degree of any vertex in \vec{G}' . So, we still have two locally fair orientations (\vec{G}, \vec{G}') with $g(u) > g'(u)$.

Since $g(u) > g'(u)$, the vertex u must have at least one out-edge in S and since S has no cycles, it follows that u must have some directed path π_v to a sink v in S . Since v is a sink in the symmetric difference graph it follows that $g(v) < g'(v)$.

However we now observe the following property of the path π_v :

- $\forall \vec{ab} \in \pi_v, g(a \rightarrow b) > g'(a \rightarrow b)$. Thus, $g(a \rightarrow b) > 0$ and so there exists a directed path from u to v in \vec{G} . Since \vec{G} is locally fair this implies that $g(u) \leq g(v)$.
- $\forall \vec{ab} \in \pi_v, g(a \rightarrow b) > g'(a \rightarrow b)$. Thus, $g'(a \rightarrow b) < g(ab)$ and so $g'(b \rightarrow a) > 0$. It follows that there exists a directed path from v to u in \vec{G}' . Local fairness implies $g'(v) \leq g'(u)$.

The 4 equations: $g(u) > g'(u)$, $g(v) < g'(v)$, $g(u) \leq g(v)$, and $g'(u) \geq g'(v)$ give a contradiction. ◀

Lemma 17 would imply that the local out-degree is well-defined, *if* the set of locally fair orientations is non-empty. Bera, Bhattacharya, Choudhari and Ghosh already aim to prove this in Section 4.1 of [3] (right above Equation 9). They claim that a locally fair orientation always exist by the following argument: They consider an arbitrary orientation \vec{G} that is not locally fair. They claim that for any pair (u, v) where $g(u) > g(v)$ and $g(u \rightarrow v) > 0$ it is possible to transfer some out-degree from $g(u)$ to $g(v)$. The existence of a locally fair orientation would follow, if it can be shown that this procedure converges to a locally fair orientation. Indeed, since the space of all orientations is a compact polytope, the limit of a converging sequence over this domain must lie within the space.

It is intuitive but not clear that this procedure indeed converges. Indeed, decreasing $g(u)$ and increasing $g(v)$ may cause some other edge (w, u) or (v, w) to start violating local fairness. One way to show convergence is to define a potential function and to show that such a transfer always decreases the potential. We define the potential function $\sum_{v \in V} g^2(v)$, thereby creating a quadratic program where the domain is the space of all orientations of G . We prove that any optimal solution to this quadratic program must be a locally fair orientation. Any quadratic function over a compact domain has an optimum and so the existence of a locally fair orientation follows.

► **Theorem 9.** *For any weighted graph G , $\forall v \in V$, $g^*(v)$ is well-defined and equals $\rho^*(v)$.*

Proof. We consider the following quadratic program \mathbf{FO}^2 from [14, Section 4] where we compute a fractional orientation of the graph G subject to a quadratic cost function:

$$\begin{array}{ll} \min \sum g(u)^2 & \text{s.t.} \\ g(u \rightarrow v) + g(v \rightarrow u) \geq g(\overline{uv}) & \forall \overline{uv} \in E \\ g(u) \geq \sum_{v \in V} g(u \rightarrow v) & \forall u \in V \\ g(u \rightarrow v), g(v \rightarrow u) \geq 0 & \forall u, v \in V \end{array}$$

Consider any optimal solution to the quadratic program. It must be that $g(u) = \sum_{v \in V} g(u \rightarrow v)$. Danisch, Chan, and Sozio [14, Corollary 4.4] prove that for any vertex u , the local density $\rho^*(u) = g(u)$.

We first note that any solution to the quadratic program is an orientation. Indeed, suppose for the sake of contradiction that there exists an edge $\overline{uv} \in E$ where $g(u \rightarrow v) + g(v \rightarrow u) > g(\overline{uv})$. We may now decrease either $g(u \rightarrow v)$ or $g(v \rightarrow u)$ to obtain another viable solution to the program. Consider decreasing $g(u \rightarrow v)$, then we may decrease $g(u)$ and maintain a viable and better solution to the program – a contradiction.

Secondly, we claim that the optimal solution to the quadratic program is a locally fair orientation. Suppose for the sake of contradiction that there exist u, v with $g(u) = g(v) + \delta'$ and $g(u \rightarrow v) = \delta$ for $\delta, \delta' > 0$. We can decrease $g(u \rightarrow v)$ to zero by increasing $g(v \rightarrow u)$ by $\Delta = \min\{\delta, \delta'\}$ and still maintain a solution to the program. This reduces the solution's value by $(g(u) - \Delta)^2 - (g(v) + \Delta)^2$. However, we now found a solution to the quadratic program with a lower value than the optimal solution – a contradiction.

Thus, the solution to \mathbf{FO}^2 gives a locally fair orientation \vec{G} where each vertex u has out-degree $g(u)$. The local density $g^*(u) = g(u)$ is by Lemma 17 well-defined. Danisch, Chan and Sozio [14, Corollary 4.4] show that $\rho^*(u) = g(u)$, which proves the theorem. ◀

Since the local density equals the local out-degree, we conclude from [14] that:

► **Corollary 10.** *Given a weighted graph G , $\rho^{\max}(G) = \Delta^{\min}(G) = \max_v g^*(v)$.*

25:10 Local Density and Its Distributed Approximation

Since a quadratic program over a convex domain always has a solution, we may also note the following interesting fact:

► **Corollary 11.** *For any graph G , there exists a fractional orientation \vec{G} that is locally fair.*

5 Results for dynamic algorithms

We use our definition of local out-degree to show that there already exist dynamic algorithms that approximate the local density of each vertex. Recall that an orientation \vec{G} is η -fair whenever for all $\overline{uv} \in E(\vec{G})$, $g(u \rightarrow v) > 0$ implies that $g(u) \leq (1 + \eta)g(v)$. We show that if we choose $\eta \leq \frac{\varepsilon^2}{128 \cdot \log n}$, then for any η -fair orientation, for all v , the out-degree $g(v)$ is a $(1 + \varepsilon)$ approximation of $g^*(v) = \rho^*(v)$. Moreover, we prove that the maximal local out-degree (i.e. $\max_{u \in V} g(u)$) is a $(1 + \varepsilon)$ approximation of $\Delta^{\min}(G) = \rho^{\max}(G)$; illustrating that approximating the local measures is a strictly more general problem. To this end, we prove the following helper lemma:

► **Lemma 18.** *Let $\eta \leq \frac{\varepsilon^2}{128 \cdot \log n}$ and $k \leq \log_{(1 + \frac{1}{16}\varepsilon)} n$. Then $(1 + \eta)^{-k} \geq (1 + 0.5\varepsilon)^{-1}$.*

Proof. Using $\log(1 + x) \geq x/2$ whenever $x < 1$, we obtain:

$$-\log_{1 + \frac{\varepsilon}{16}}(n) = -\frac{\log(n)}{\log 1 + \frac{\varepsilon}{16}} \geq -\frac{\log(n)}{\frac{\varepsilon}{32}} \geq -\frac{\varepsilon}{4} \cdot \frac{128 \log(n)}{\varepsilon^2}$$

$$(1 + \eta)^{-k} \geq \left(1 + \frac{\varepsilon^2}{128 \cdot c \cdot \log n}\right)^{-c \cdot \log_{1 + \frac{\varepsilon}{16}}(n)} \geq \left(1 + \frac{\varepsilon^2}{128 \cdot c \cdot \log n}\right)^{-\frac{\varepsilon}{4} \cdot \frac{128 \cdot c \cdot \log(n)}{\varepsilon^2}}$$

$$(1 + \eta)^{-k} \geq \text{EXP}\left[-\frac{\varepsilon}{4}\right] \geq \text{EXP}\left[-\log\left(1 + \frac{\varepsilon}{2}\right)\right] \geq \left(1 + \frac{\varepsilon}{2}\right)^{-1} \quad \blacktriangleleft$$

► **Theorem 12.** *Let G be a weighted graph and \vec{G} be an η -fair fractional orientation for $\eta \leq \frac{\varepsilon^2}{128 \cdot \log n}$. Then $\forall v \in V: (1 + \varepsilon)^{-1} \rho^*(v) \leq g(v) \leq (1 + \varepsilon) \rho^*(v)$.*

Proof. First, we show that for all vertices v , $g(v) \leq (1 + \varepsilon) \rho^*(v)$.

Suppose for the sake of contradiction that there exists a vertex u with $g(u) > (1 + \varepsilon) \rho^*(u)$. We fix $\rho^*(u) = g^*(u) = \gamma$ and work with γ throughout the remainder of this proof to show a contradiction. By Corollary 11, there exists at least one locally fair fractional orientation \vec{G}_x . By Corollary 10, every vertex v in this orientation has out-degree $g_x(v) = g^*(v) = \rho^*(v)$. And thus, the fractional orientation \vec{G}_x is not equal to \vec{G} .

Given \vec{G} and a locally fair fractional orientation \vec{G}_x , we do three steps:

- We partition the vertices of G to create two graphs G_1 and G_2 . The partition is based on the orientation \vec{G}_x as we set: $G^1 = G[v \in V \mid g_x(v) \leq \gamma]$ and by $G^2 = G[v \in V \mid g_x(v) > \gamma]$. For ease of notation, we write any edge with one endpoint $a \in V(G^1)$ and one endpoint $b \in V(G^2)$ as (a, b) and never as (b, a) .
- From G^1 , we create a family of nested subgraphs using \vec{G} . We define graphs $G_i^1 := G^1[v \in V(G^1) \mid g(v) \geq \frac{g(u)}{(1 + \eta)^i}]$. We denote by k the lowest integer such that $|V(G_{k+1}^1)| < (1 + \frac{\varepsilon}{16})|V(G_k^1)|$. We apply Lemma 18 to observe that $(1 + \eta)^{-k} \geq (1 + \varepsilon/2)^{-1}$.
- Finally, we use both orientations to create three claims that contradict one another.

The first claim. We denote by E_{\uparrow} the set of all edges $e = (a, b)$ with $a \in V(G_{k+1}^1)$ and $b \in V(G^2)$ and claim that:

$$\sum_{e \in E(G_{k+1}^1)} g(e) + \sum_{e \in E_{\uparrow}} g(e) \leq \sum_{v \in V(G_{k+1}^1)} g_x(v). \quad (1)$$

Indeed for $\overline{ab} \in E(G_{k+1}^1)$, both endpoints are in $V(G_{k+1})$. Because \vec{G}_x is locally fair, this implies that $g_x(a \rightarrow b) + g_x(b \rightarrow a) = g(\overline{ab})$. For all $\overline{ab} \in E_\uparrow$, $g_x(b) > \gamma \geq g_x(a)$ per definition of G^1 and G^2 . By local fairness, $g_x(a \rightarrow b) = g(\overline{ab})$ and the inequality follows.

The second claim. We secondly claim that:

$$\sum_{v \in V(G_k^1)} g(v) > \sum_{v \in V(G_{k+1}^1)} g_x(v). \quad (2)$$

This is because we can lower bound $\sum_{v \in V(G_k^1)} g(v)$ as follows:

$$\begin{aligned} \sum_{v \in V(G_k^1)} g(v) &\geq (1 + \eta)^{-k} \cdot g(u) \cdot |V(G_k^1)| > (1 + \eta)^{-k} \cdot g(u) \cdot |V(G_{k+1}^1)| \cdot (1 + \frac{\varepsilon}{16})^{-1} \\ &> (1 + \frac{\varepsilon}{2})^{-1} \cdot (1 + \varepsilon)\gamma \cdot |V(G_{k+1}^1)| \cdot (1 + \frac{\varepsilon}{16})^{-1} \geq \gamma \cdot |V(G_{k+1}^1)| \end{aligned}$$

The claim follows by noting that per definition of G^1 , for all $v \in V(G_{k+1}^1)$, $g_x(v) \leq \gamma$.

The third claim. Lastly, we claim that:

$$\sum_{v \in V(G_k^1)} g(v) \leq \sum_{e \in E(G_{k+1}^1)} g(e) + \sum_{e \in E_\uparrow} g(e) \quad (3)$$

Consider any $v \in V(G_k^1)$ and any vertex a with $g(v \rightarrow a) > 0$. Recall that \vec{G} is an η -fair orientation. Thus, if $a \in G^1$, then $\overline{va} \in E(G_{k+1}^1)$. If $a \in G^2$, then per definition $\overline{va} \in E_\uparrow$. Per definition of a fractional orientation $g(v \rightarrow a) \leq g(\overline{va})$ and so the claim follows.

A contradiction. Equation 1, 2 and 3 contradict each other. Thus, we have proven that for all vertices v , $g(v) \leq (1 + \varepsilon)\rho^*(v)$.

The full version finishes the proof by showing the other direction. ◀

If G is a unit-weight graph, Chekuri et al. [10] present a dynamic algorithm to maintain an η -fair orientation in a unit-weight graph with $\eta \in O(\varepsilon^{-2} \log n)$. Thus, by Theorem 12, we may now conclude that they approximate the local density (and/or the local out-degree):

► **Corollary 13.** *There exists an algorithm [10] that can fractionally orient a dynamic unit-weight graph G with n vertices subject to edge insertions and deletions with deterministic worst-case $O(\varepsilon^{-6} \log^4 n)$ update time such that for all $v \in V$:*

$$g(v) \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)].$$

6 Results in LOCAL

We prove that the local out-degree of each $v \in V$ is (largely) determined by its local neighbourhood. As a result, we immediately get an algorithm to solve Problem 3 in LOCAL.

► **Theorem 14.** *Let G be a unit-weight graph. For any $\varepsilon > 0$ and vertex v , denote by $\rho^*(v)$ its local density and by $\rho_k^*(v)$ its local density in $H_k(v)$. Then $\rho_k^*(v) \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$ for $k \in \Theta(\varepsilon^{-2} \log^2 n)$.*

Proof. To prove the theorem, we design a simple deletion-only algorithm to maintain an η -fair orientation. For $\eta = \frac{\varepsilon^2}{128 \log n}$, this algorithm has a recursive depth of $O(\varepsilon^{-2} \log^2 n)$.

■ **Algorithm 1** DECREASE($g(u \rightarrow v)$ by Δ – assuming that $\Delta \leq g(u \rightarrow v)$).

```

 $w^* \leftarrow \arg \max_{w \in V} \{g(w) \mid g(w \rightarrow u) > 0\}$ 
while  $\Delta > 0$  and  $g(w^*) > (1 + \eta)(g(u) - \Delta)$  do
    if  $\Delta > g(w^* \rightarrow u)$  then
        DECREASE( $g(w^* \rightarrow v)$  by  $g(w^* \rightarrow v)$ )
         $\Delta = \Delta - g(w^* \rightarrow u)$ 
    else
        DECREASE( $g(w^* \rightarrow v)$  by  $\Delta$ )
     $w^* \leftarrow \arg \max_{w \in V} \{g(w) \mid g(w \rightarrow u) > 0\}$ 
    
```

Specifically, we say that a directed edge \overline{uv} is bad whenever $g(u \rightarrow v) > 0$ and $g(u) > (1 + \eta)g(v)$. In an η -fair orientation no edge is bad. Whenever we delete an edge $\overline{x_1x_0}$, the out-degree $g(x_1)$ decreases by $g(x_1 \rightarrow x_0)$. For vertices x_2 with $g(x_2 \rightarrow x_1) > 0$, it may now be that $g(x_2) > (1 + \eta)g(x_1)$ (and thus the edge $\overline{x_2x_1}$ becomes bad). Note if there exists such a vertex x_2 , then it must hold for the vertex $x_2^* := \arg \max_{x_2 \in V} \{g(x_2) \mid g(x_2 \rightarrow x_1) > 0\}$. This leads to a recursive algorithm to decrease the out-degree of a vertex (Algorithm 1).

We claim that this algorithm as a recursive depth of $O(\log_{1+\eta} n)$. Indeed any sequence of recursive calls is a path in G . Denote the path belonging to the longest sequence of recursive calls by x_0, x_1, \dots, x_ℓ . Since Δ is always at most 1, it must hold for all i that: $g(x_i) > (1 + \eta)(g(x_{i-1}) - 1)$. Since a graph may have at most n^2 edges, $g(x_\ell) \leq n$ and it follows that the recursive depth is at most $\ell \in O(\log_{1+\eta} n)$. We now apply $\log(1 + x) \geq x/2$ and note that: $\ell \leq \frac{\log n}{\log(1+\eta)} \leq \frac{\log n}{\eta/2} \subseteq O(\frac{\log n}{64\varepsilon^2/\log n}) \subseteq O(\frac{\log^2 n}{\varepsilon^2})$.

Given this theoretical algorithm, we prove the lemma. Consider any fair orientation \vec{G} . Then by Theorem 9 for any vertex v , $g(v) = \rho^*(v)$. Choose some $k \in \Theta(\varepsilon^{-2} \log^2 n)$ sufficiently large and let $H_k(v)$ be the k -hop neighbourhood of v and E_k be all the edges in $H_{k+1}(v)$ that are not in H_k .

Choose $\eta = \frac{\varepsilon^2}{128 \log n}$. The orientation \vec{G} is per definition an η -fair orientation. We run on \vec{G} our deletion-only algorithm, deleting all edges in E_k . Since our algorithm has a recursive depth of $\ell < k$, we end up with an η -fair orientation of $H_k(v)$ where $g(v) = \rho^*(v)$. We apply Theorem 12 to conclude that $\rho^*(v) = g(v) \in [(1 + \varepsilon)^{-1} \rho_k^*(v), (1 + \varepsilon) \rho_k^*(v)]$ which concludes the theorem. ◀

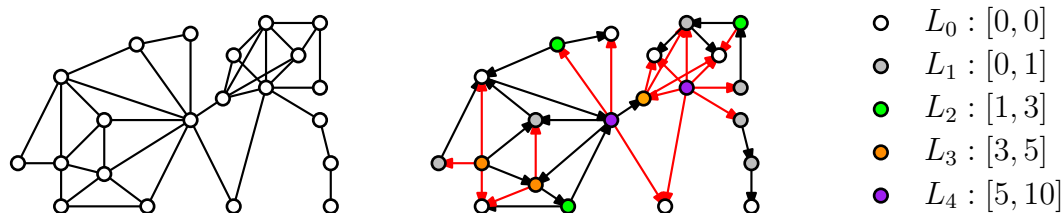
► **Corollary 15.** *There exists an algorithm in LOCAL that given a unit graph G and $\varepsilon > 0$ computes in $O(\varepsilon^{-2} \log^2 n)$ rounds for all $v \in V$ a value $\rho_v \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$.*

7 Results in CONGEST

We now describe an algorithm in CONGEST that for any unit-weight graph G , creates an η -fair orientation (with $\eta = \frac{\varepsilon^2}{128 \cdot \log n}$). Our algorithm uses as a subroutine a distributed algorithm to compute a *blocking flow* in an h -layered DAG in $O(\text{Blocking}(h, n))$ rounds.

► **Definition 19.** *An edge-capacitated DAG G is h -layered if the vertices can be embedded on a grid of height h , such that every directed edge \overline{uv} points downwards.*

► **Definition 20.** *For an edge-capacitated DAG G with sources S and terminals T , a flow f from S to T is blocking if every augmenting path of f contains at least one saturated edge.*



■ **Figure 1** Given a graph G , we arbitrarily orient G . This allows us to partition the vertices of G into levels L_1, \dots, L_6 based on their current out-degree. We say that an edge (u, v) is *violating* whenever $g(u \rightarrow v) > 0$, $u \in L_i$, $v \in L_j$ and $i > j + 1$. We show violating edges in red. Our algorithm iterates over an integer h from high to low, and tries to flip all violating edges from level L_h .

► **Lemma 21** (Lemma 7.2 and 9.1 in [21]). *There exists an algorithm which, given an n -node h -layer edge-capacitated DAG D with sources S and terminals T computes a blocking ST -flow in CONGEST in:*

- Blocking(h, n) = $\tilde{O}(h^4)$ rounds with high probability,
- Blocking(h, n) = $\tilde{O}(h^6 \cdot 2^{c\sqrt{\log n}})$ deterministic rounds for some constant c .

We compute an η -fair orientation by repeatedly constructing a DAG with $h \in O(\varepsilon^{-2} \log^2 n)$ and computing blocking flows. Theorem 12 implies in an η -fair orientation (for our choice of η) the out-degree of each vertex v is a $(1 + \varepsilon)$ -approximation $\rho^*(v)$.

The initialising step. Before we start our algorithm, we create a starting orientation where we set for every edge $e = \overline{uv}$, $g(u \rightarrow v) = \frac{1}{2}g(e)$ and $g(v \rightarrow u) = \frac{1}{2}g(e)$. This gives each vertex u some out-degree $g(u)$ which we partition:

► **Definition 22.** *Let each vertex u have out-degree $g(u)$. We define level i as:*

$$L_i := \left\{ u \in V \mid g(u) \in \left[\left(1 + \frac{\eta}{2}\right)^i, \left(1 + \frac{\eta}{2}\right)^{i+1} \right] \right\}.$$

A vertex $u \in L_i$ is at level i and ℓ' denotes the highest level that is not empty.

Whenever $g(u) = \left(1 + \frac{\eta}{2}\right)^i$, u may decide whether $u \in L_i$ or $u \in L_{i-1}$; whenever our algorithm increases $g(u)$ the vertex u defaults to the lowest possible level and vice versa.

► **Definition 23.** *Consider an edge \overline{uv} with $u \in L_i$ and $v \in L_j$. We say that:*

- (u, v) is an out-edge from u and an in-edge into v whenever $i > j$ and $g(u \rightarrow v) > 0$, and
- (u, v) is violating whenever $i > j + 1$ and $g(u \rightarrow v) > 0$

Note that the orientation is η -fair whenever there exist no violating edges.

► **Lemma 24.** *Let $\eta \leq \frac{\varepsilon^2}{128 \cdot \log n}$. For our orientation, let ℓ' be the highest level such that $L_{\ell'}$ is not empty then $\ell' \leq \varepsilon^{-2} \log^2 n$.*

Proof. The maximal out-degree of a vertex is n . Thus, $\ell' \leq \frac{\log n}{\log(1 + \frac{\eta}{2})}$. We now apply $\log(1 + x) \geq x/2$ and note that: $\ell' \leq \frac{\log n}{\log(1 + \frac{\eta}{2})} \leq \frac{\log n}{\eta/4} = \frac{\log n}{32\varepsilon^2/\log n} \leq \frac{\log^2 n}{\varepsilon^2}$. ◀

7.1 Algorithm overview

Denote $\ell = \lceil \varepsilon^{-2} \log^2 n \rceil$. Our algorithm runs on a “clock” denoted by $(h : m : s)$ where:

- Each HOUR h lasts $2\lceil \eta^{-1} \rceil + 2$ MINUTES,
- Each even MINUTE m lasts $4\lceil \log_{8/7} n \rceil + 2$ rounds,

25:14 Local Density and Its Distributed Approximation

- Each odd MINUTE m in HOUR h lasts $\ell - h + 1$ SECONDS,
- Each SECOND s lasts $\ell + \text{Blocking}(\ell, n)$ rounds.

Each vertex tracks the clock to know which actions of our algorithm it should execute (if any). Our clock is special, in the sense that hours tick downwards. Minutes and seconds tick upwards, starting from zero. Each vertex v in our graph keeps track of the current time, measured in the current HOUR h , MINUTE m and SECOND s . We maintain the following:

► **Invariant 1.** *During HOUR h , there are no violating out-edges from level k for all $k > h + 1$. At the start of each even MINUTE in HOUR h , there are no violating out-edges from level k for all $k > h$.*

This invariant implies that we compute an η -fair orientation when the clock reaches $(0 : 0 : 0)$. Going from HOUR h to HOUR $h - 1$ we maintain this invariant by flipping directed paths:

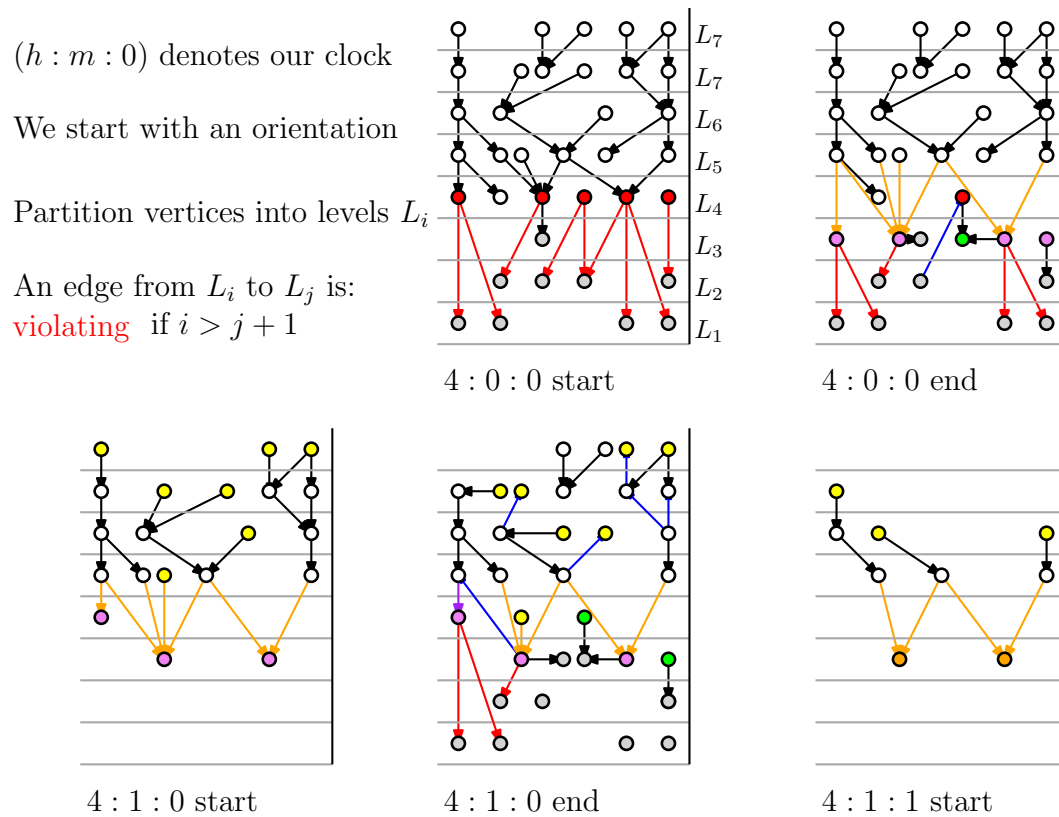
► **Definition 25.** *For any edge \overline{uv} with $g(u \rightarrow v) > 0$, we say that our algorithm is flipping \overline{uv} whenever it decreases $g(u \rightarrow v)$ (increasing $g(v \rightarrow u)$). Moreover, we say that \overline{uv} is flipped whenever our algorithm has decreased $g(u \rightarrow v)$ such that $g(u \rightarrow v) = 0$.*

Algorithm (see also Figure 2 and Algorithms 2 and 3 and 4 and 5). Each time frame has a purpose:

- Each HOUR h , the goal is to identify and “fix” all violating out-edges from vertices in L_h ; without introducing violating out-edges from vertices in a level L_k with $k > h$. We do this iteratively, using two different steps:
 - Each even MINUTE, we fix all violating out-edges from vertices in L_h , possibly creating violating out-edges from vertices in L_{h+1} to vertices in L_{h-1} .
 - Each odd MINUTE, we fix all violating out-edges from vertices in L_{h+1} to vertices in L_{h-1} . We do this in such a manner, that we create no new violating out-edges from vertices in L_k for $k > h$. However, we may create violating out-edges from level L_h .
- Each even MINUTE m' , we define the set $V_{m'} \subset L_h$ of vertices that have at least one violating out-edge. Over $4\lceil \log_{8/7} n \rceil + 2$ rounds, each $u \in V_{m'}$ flips violating out-edges \overline{uv} . Moreover:
 - The out-degree $g(u)$ is decreased such that u drops at most one level, and
 - The out-degree $g(v)$ such that v increases its level up to at most $h - 1$.

We prove that at the end of this MINUTE m' , there exist no more $u \in V_{m'}$ with a violating out-edge. Thus, for each vertex $u \in V_{m'}$, either u decreased one level, or all \overline{uv} that were violating now have that $g(u \rightarrow v) = 0$, or $v \in L_{h-1}$.
- In each odd MINUTE m , we inspect the vertices $T_m := \{u \in V_{m-1} \text{ that dropped a level}\}$. For each $u \in T_m$, for each edge \overline{wu} with $w \in L_{h+1}$ and $g(w \rightarrow u) > 0$ the edge (w, u) has become a violating in the previous MINUTE. We want to fix these edges, whilst guaranteeing that we create no violating in-edges from vertices in level $h + 2$ and above. We obtain this, by recording at the start of the MINUTE for every vertex u its level $l_m(u)$. We then invoke $\ell - h + 1$ SECONDS. Each SECOND s , we create a DAG D_s where the sinks are T_m . We increase for all $u \in T_m$ the out-degree $g(u)$ by flipping a directed path from a source in D_s to T_m . We construct our DAG in such a manner that this procedure does not create new violating in-edges, and that for all SECONDS s , D_{s+1} is a subgraph of D_s .
- At the start of SECOND s of each MINUTE m , we fix for every edge \overline{uv} the values $g_s(u \rightarrow v)$, and the levels $l_s(u)$ and $l_s(v)$ of u and v . We then define a graph $D_s = (V_s, E_s)$ as follows:
 - The edges E_s are all violating in-edges to vertices in T_m plus all \overline{uv} with: $l_s(u) = l_m(u) > h + 1$, $l_s(u) > l_s(v)$, and $g_s(u \rightarrow v) > 0$.
 - The vertices V_s include T_m plus all vertices in G with a directed path to T_m in E_s .

- The vertices $S_s \subset V_s$ are all sources in D_s (these are not in T_m). For each $v \in T_m$ we increase $g(v)$ by flipping a directed path from a vertex in S_s to v . We continue this until either $g(v) = (1 + \frac{\eta}{2})^{h+1}$, or, there exist no more edges $(u, v) \in E_s$ with $g(u \rightarrow v) > 0$. In both cases, v has no more violating in-edges. To find these directed paths, we create a flow problem on a graph D_s^* where the maximal path has length $\ell + 2$:
- For each $u \in S_s$, we define $\sigma(u) = g(u) - (1 + \frac{\eta}{2})^{l_m(u)-1}$ (the maximal amount we can decrease $g(u)$ such that it does not arrive in level $l_m(u) - 2$). We connect every $u \in S_s$ to a unique source s_u where the edge $\overline{s_u u}$ has capacity $\sigma(u)$.
- For each $v \in T_m$, we define $\delta(v) = (1 + \frac{\eta}{2})^{h+1} - g(v)$ (the maximal amount we can increase $g(v)$ such that it does not arrive in level $h + 1$). We connect every $v \in T_m$ to a unique sink t_v where the edge $\overline{v t_v}$ has capacity $\delta(v)$.
- Each other edge $\overline{uv} \in E_s$ has capacity $g(u \rightarrow v)$.



■ **Figure 2** $(4 : 0 : 0)$ – at the first MINUTE start in HOUR 4, we consider all violating out-edges \overline{uv} from level 4 (red). Per definition, these edges point to level 2 or lower.

$(4 : 0 : 0)$ – at the first MINUTE end, either u has dropped a level (pink), v increased their level to $h - 1$ (green) or the edge \overline{uv} is flipped (blue). We consider violating in-edges to pink vertices (orange)

$(4 : 1 : 0)$ – at the first SECOND start, we construct a DAG D_0 where the edges are the orange edges plus black edges. The vertex set are all u with a directed path to a pink vertex ($v \in T_1$). The yellow vertices are S_0 .

$(4 : 1 : 0)$ – at the first SECOND end, edges in the DAG may have flipped (blue), vertices in S_0 may have dropped a level or vertices in T_1 may have increased a level (making some edges no longer violating – purple).

$(4 : 1 : 1)$ – at the second SECOND start, we construct a DAG D_1 . Note that D_1 is a subgraph of D_0 .

7.2 Formal algorithm definition

We now formalise our algorithm top-down, starting with defining variables.

► **Definition 26.** *At the start of $(h, m', 0)$, where m' is even, we fix the set $V_{m'} := \{v \in L_h \mid v \text{ has at least one violating out-edge}\}$. At the start of $(h, m, 0)$ where m is odd, we fix:*

$$T_m := \{v \in V_{m-1} \mid v \text{ decreased by one level in the previous MINUTE and } v \text{ has at least one violating in-edge}\}.$$

Finally, we denote for any vertex u by $l_m(u)$ its level at the start of the MINUTE m .

► **Definition 27.** *At the start of $(h : m : s)$, where m is odd, we fix the following:*

- For any vertex u , we denote by $l_s(u)$ its level at the start of the SECOND.
- For any edge \overline{uv} denote by $g_s(u \rightarrow v)$ the out-degree from u to v at the start of the SECOND.
- We define the edge set E_s as all violating in-edges to vertices in T_m plus all \overline{uv} with: $l_s(u) = l_m(u) > h + 1$, $l_s(u) > l_s(v)$, and $g_s(u \rightarrow v) > 0$.
- V_s are all vertices with a directed path to a vertex in T_m .
- $D_s = (V_s, E_s)$ is a DAG where S_s are all the sources (per definition $S_s \cap T_m = \emptyset$).

► **Definition 28.** *At the start of (h, m, s) where m is odd, given T_m , S_s and D_s , we define the DAG D_s^* by connecting all $u \in S_s$ to a unique sink s_u and all $v \in T_m$ to a unique sink t_v .*

- For $u \in S_s$, the edge $\overline{s_u u}$ has capacity $\sigma(u) = g_s(u) - (1 + \frac{\eta}{2})^{l_m(u)-1}$.
- For $v \in T_m$, the edge $\overline{v t_v}$ has capacity $\delta(v) = (1 + \frac{\eta}{2})^{h+1} - g_s(v)$.
- Each other edge $\overline{uv} \in E_s$ has capacity $g_s(u \rightarrow v)$.

► **Observation 29.** *At the start of $(h : m : s)$ with m odd, if every vertex u is given $l_m(u)$, whether $u \in T_m$, and h , then we may compute all elements of Definitions 27 and 28 in ℓ rounds.*

■ **Algorithm 2** HOUR(h).

```

for  $m = 0$  to  $2\lceil\eta^{-1}\rceil + 2$  do
  if  $m$  is even then
    EVENMINUTE( $h, m$ )
  else
    ODDMINUTE( $h, m$ )
if  $h > 0$  then
  HOUR( $h - 1$ )

```

■ **Algorithm 3** EVENMINUTE(int h , int m).

```

 $V_m := \{v \in L_h \mid v \text{ has at least one violating out-edge}\}$ 
for  $t = 0$  to  $2\lceil\log_{8/7} n\rceil + 1$  do
  Let  $A_t \subset V_m$  be the set of vertices at level  $h$  with at least one violating out-edge
  Each  $a \in A_t$  determines the set  $E_t(a)$  of violating out-edges.
  Each  $a \in A_t$  computes  $\delta_t(a) = g(a) - (1 + \eta)^{h-1}$  and reports  $\delta_t(a)/|E_t(a)|$  across  $E_t(a)$ .
  /* next round: */
  Let  $B_t$  be the set of vertices that receive at least one violating in-edge from  $A_t$ .
  Each  $b \in B_t$  determines the set  $I_t(b)$  of vertices that reported a value to  $v$ .
  Each  $b \in B_t$  sorts the  $a \in I_t(b)$  by  $\delta(a)$ .
  Each  $b \in B_t$  greedily decreases  $g(a \rightarrow b)$  by at most  $\delta_t(a)$  for  $a \in I_t(b)$ ; until  $g(b) = (1 + \frac{\eta}{2})^h$ .

```

■ **Algorithm 4** ODDMINUTE(int h , int m).

$T_m := \{v \in V_{m-1} \mid v \text{ has at least one violating in-edge}\}$
for $s = \ell$ **down to** h **do**
 SECOND(h, m, s)

■ **Algorithm 5** SECOND(int h , int m , int s).

Compute the graph D_s in ℓ rounds.
 Compute the graph D_s^* by adding a shared source to S_s and a shared sink to T_m .
 $f = \text{COMPUTEBLOCKINGFLOW}(D_s^*)$
 Flip all edges in D_s with the corresponding flow in f .

7.3 Proving our algorithm's correctness

Per definition, our algorithm runs in $O(\eta^{-1} \log n \cdot \ell \cdot (\ell + \text{Blocking}(\ell, n))) = O(\varepsilon^{-3} \log^4 n \cdot (\varepsilon^{-2} \log^2 n + \text{Blocking}(\varepsilon^{-2} \log^2 n, n)))$ rounds. What remains is to show that we maintain Invariant 1, which implies that we compute an η -fair orientation.

We note that by the choice of our algorithm's variables, we have the following property:

► **Observation 30.** *For all times $(h : m : s)$ with m odd:*

- *Vertices at level $k > h$ only decrease their level and by at most one (because afterwards, $l_s(u) < l_m(u)$ and thus u has no out-edges in E_s),*
- *vertices at level $h - 1$ only increase their level (by at most 1), and*
- *vertices at level $k' < h - 1$ and level h do not change their level.*

We use this observation in the full version to show that we maintain Invariant 1 by induction. Trivially, Invariant 1 holds at the start of $(\ell : 0 : 0)$. We now assume that the invariant holds at $(h : 0 : 0)$, i.e. that there are no violating out-edges from level L_k with $k > h$. We prove that our algorithm ensures that at the start of $(h - 1 : 0 : 0)$ there are no violating out-edges from level L_k with $k \geq h$.

Moreover, we maintain the invariant that throughout HOUR $(h - 1)$ there are no violating out-edges from level $L_{k'}$ with $k' > h + 1$. We prove this in the following way:

- During $(h : m : s)$ for m even, our algorithm eliminates all violating edges going from L_h . We show that in each iteration, the number of violating edges from L_h drops by a factor $7/8$. The graph has at most n^2 edges. This implies that after the MINUTE, there are no more violating edges going from L_h . However, there may now be violating out-edges from vertices in L_{h+1} to vertices in $T_{m+1} \subseteq L_{h-1}$.
- During $(h : m : s)$ for m odd, our algorithm eliminates all violating edges going from L_{h+1} to T_m . We show that for all s , the DAG D_s is a subgraph of D_{s-1} where the height is one fewer. Since the height of D_s is at most $\ell - h + 1$, this implies that after the MINUTE m , the graph D_s is empty and there are no more violating in-edges to T_m .
- Each MINUTE, our algorithm alternates between having violating edges from L_h and L_{h+1} . We show that our algorithm can alternate at most η times before there are no violating edges from both L_h and L_{h+1} , which implies Invariant 1 at the start of $(h - 1 : 0 : 0)$.

Invariant 1 implies that we compute an η -fair orientation at time $(0 : 0 : 0)$, thus:

► **Theorem 16.** *Suppose one can compute a blocking flow in an n -node h -layered DAG in $\text{Blocking}(h, n)$ rounds. There exists an algorithm in CONGEST that given a unit-weight graph G and $\varepsilon > 0$ computes in $O(\varepsilon^{-3} \log^4 n \cdot (\varepsilon^{-2} \log^2 n + \text{Blocking}(\varepsilon^{-2} \log^2 n, n)))$ rounds an orientation \vec{G} such that for all $v \in V$: $g(v) \in [(1 + \varepsilon)^{-1} \rho^*(v), (1 + \varepsilon) \rho^*(v)]$.*

We plug in the runtime of $\text{Blocking}(h, n)$ of Lemma 21 by Haeupler, Hershkowitz, and Saranurak [21] to obtain the following runtime:

► **Corollary 31.** *There is an algorithm in CONGEST that given a unit-weight graph G and an $\varepsilon > 0$ computes an orientation \vec{G} such that $\forall v \in V$, the out-degree $g(v) \in [(1 + \varepsilon)^{-1}\rho^*(v), (1 + \varepsilon)\rho^*(v)]$ in:*

- $\tilde{O}(\varepsilon^{-11} \log^{12} n)$ rounds with high probability, or
- $\tilde{O}(\varepsilon^{-15} \log^{16} n \cdot 2^{O(\sqrt{\log n})})$ deterministic rounds.

Finally, we apply Corollary 10 which states that $\rho^{\max}(G) = \Delta^{\min}(G) = \max_v g^*(v)$ to conclude:

► **Corollary 32.** *There is a deterministic algorithm in CONGEST that given a unit-weight graph G and an $\varepsilon > 0$, that computes an orientation \vec{G} such that:*

$$\max_{v \in V} g(v) \in [(1 + \varepsilon)^{-1}\rho^{\max}(G), (1 + \varepsilon)\rho^{\max}(G)],$$

in a number of rounds that is sublinear in n .

This is the first deterministic sublinear algorithm that solves Problem 2.2.

References

- 1 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proc. VLDB Endow.*, 5(5):454–465, 2012. doi:10.14778/2140436.2140442.
- 2 Soheil Behnezhad, Mahsa Derakhshan, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching on uniformly sparse graphs. In *Algorithmic Game Theory: 12th International Symposium, SAGT 2019, Athens, Greece, September 30 – October 3, 2019, Proceedings*, pages 357–373, Berlin, Heidelberg, 2019. Springer-Verlag. doi:10.1007/978-3-030-30473-7_24.
- 3 Suman K. Bera, Sayan Bhattacharya, Jayesh Choudhari, and Prantar Ghosh. A new dynamic algorithm for densest subhypergraphs. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini, editors, *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1093–1103. ACM, 2022. doi:10.1145/3485447.3512158.
- 4 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182. ACM, 2015. doi:10.1145/2746539.2746592.
- 5 Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*, 2020.
- 6 Glencora Borradaile, Jennifer Iglesias, Theresa Migler, Antonio Ochoa, Gordon Wilfong, and Lisa Zhang. Egalitarian graph orientations. *Journal of Graph Algorithms and Applications*, 2017.
- 7 Glencora Borradaile, Theresa Migler, and Gordon Wilfong. Density decompositions of networks. In *Conference on Complex Networks (CompleNet)*. Springer, 2018.
- 8 Gerth Støltting Brodal and Rolf Fagerberg. Dynamic representations of sparse graphs. In *In Proc. 6th International Workshop on Algorithms and Data Structures (WADS)*, pages 342–351. Springer-Verlag, 1999. doi:10.1007/3-540-48447-7_34.
- 9 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization (APPROX)*. Springer, 2003.

- 10 Chandra Chekuri, Aleksander Bjørn Christiansen, Jacob Holm, Ivor van der Hoog, Kent Quanrud, Eva Rotenberg, and Chris Schwiegelshohn. Adaptive out-orientations with applications. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3062–3088. SIAM, 2024.
- 11 Chandra Chekuri, Kent Quanrud, and Manuel R Torres. Densest subgraph: Supermodularity, iterative peeling, and flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2022.
- 12 Aleksander Bjørn Grodt Christiansen, Krzysztof Nowicki, and Eva Rotenberg. Improved dynamic colouring of sparse graphs. 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 1201–1214. ACM, 2023. doi:10.1145/3564246.3585111.
- 13 Aleksander Bjørn Grodt Christiansen and Eva Rotenberg. Fully-Dynamic $\alpha + 2$ Arboricity Decompositions and Implicit Colouring. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.42.
- 14 Maximilien Danisch, T.-H. Hubert Chan, and Mauro Sozio. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web, WWW '17*, pages 233–242, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. doi:10.1145/3038912.3052619.
- 15 Atish Das Sarma, Ashwin Lall, Danupon Nanongkai, and Amitabh Trehan. Dense subgraphs on dynamic networks. In Marcos K. Aguilera, editor, *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, volume 7611 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2012. doi:10.1007/978-3-642-33651-5_11.
- 16 Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–191. IEEE, 2017. doi:10.1109/FOCS.2017.25.
- 17 András Frank and Kazuo Murota. Fair integral network flows. *Mathematics of Operations Research*, 2023.
- 18 Satoru Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, 5(2):186–196, 1980. doi:10.1287/MOOR.5.2.186.
- 19 Mohsen Ghaffari, David G Harris, and Fabian Kuhn. On derandomizing local distributed algorithms. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 662–673. IEEE, 2018. doi:10.1109/FOCS.2018.00069.
- 20 Mohsen Ghaffari and Hsin-Hao Su. Distributed degree splitting, edge coloring, and orientations. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2505–2523. SIAM, 2017. doi:10.1137/1.9781611974782.166.
- 21 Bernhard Haeupler, D Ellis Hershkowitz, and Thatchaphol Saranurak. Maximum length-constrained flows and disjoint paths: Distributed, deterministic, and fast. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1371–1383, 2023. doi:10.1145/3564246.3585202.
- 22 Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. Faster and scalable algorithms for densest subgraph and decomposition. *Advances in Neural Information Processing Systems*, 35:26966–26979, 2022.
- 23 Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. Convergence to lexicographically optimal base in a (contra) polymatroid and applications to densest subgraph and tree packing. In *European Symposium on Algorithms (ESA)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.56.


- 24 David G Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 700–724. IEEE, 2019. doi:10.1109/FOCS.2019.00048.
- 25 Krzysztof Onak, Baruch Schieber, Shay Solomon, and Nicole Wein. Fully dynamic mis in uniformly sparse graphs. *ACM Trans. Algorithms*, 16(2), March 2020. doi:10.1145/3378025.
- 26 Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. Locally densest subgraph discovery. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 965–974, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2783258.2783299.
- 27 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 181–193, 2020. doi:10.1145/3357713.3384327.
- 28 Hsin-Hao Su and Hoa T. Vu. Distributed dense subgraph detection and low outdegree orientation. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPICs*, pages 15:1–15:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.DISC.2020.15.
- 29 Nikolaž Tatti and Aristides Gionis. Density-friendly graph decomposition. In *Proceedings of the 24th International Conference on World Wide Web, WWW '15*, pages 1089–1099, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee. doi:10.1145/2736277.2741119.
- 30 Yalong Zhang, Rong-Hua Li, Qi Zhang, Hongchao Qin, and Guoren Wang. Efficient algorithms for density decomposition on large static and dynamic graphs. *Proceedings of the VLDB Endowment*, 2024.

Toward Better Depth Lower Bounds: Strong Composition of XOR and a Random Function

Nikolai Chukhin ✉

Neapolis University Pafos, Cyprus

JetBrains Research, Paphos, Cyprus

Alexander S. Kulikov ✉ 🏠 

JetBrains Research, Paphos, Cyprus

Ivan Mihajlin ✉

JetBrains Research, Paphos, Cyprus

Abstract

Proving formula depth lower bounds is a fundamental challenge in complexity theory, with the strongest known bound of $(3 - o(1)) \log n$ established by Håstad over 25 years ago. The Karchmer–Raz–Wigderson (KRW) conjecture offers a promising approach to advance these bounds and separate P from NC¹. It suggests that the depth complexity of a function composition $f \diamond g$ approximates the sum of the depth complexities of f and g .

The Karchmer–Wigderson (KW) relation framework translates formula depth into communication complexity, restating the KRW conjecture as $CC(KW_f \diamond KW_g) \approx CC(KW_f) + CC(KW_g)$. Prior work has confirmed the conjecture under various relaxations, often replacing one or both KW relations with the universal relation or constraining the communication game through strong composition.

In this paper, we examine the strong composition $KW_{\text{XOR}} \otimes KW_f$ of the parity function and a random Boolean function f . We prove that with probability $1 - o(1)$, any protocol solving this composition requires at least $n^{3-o(1)}$ leaves. This result establishes a depth lower bound of $(3 - o(1)) \log n$, matching Håstad’s bound, but is applicable to a broader class of inner functions, even when the outer function is simple. Though bounds for the strong composition do not translate directly to formula depth bounds, they usually help to analyze the standard composition (of the corresponding two functions) which is directly related to formula depth.

Our proof utilizes formal complexity measures. First, we apply Khrapchenko’s method to show that numerous instances of f remain unsolved after several communication steps. Subsequently, we transition to a different formal complexity measure to demonstrate that the remaining communication problem is at least as hard as $KW_{\text{OR}} \otimes KW_f$. This hybrid approach not only achieves the desired lower bound, but also introduces a novel technique for analyzing formula depth, potentially informing future research in complexity theory.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases complexity, formula complexity, lower bounds, Boolean functions, depth

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.26

1 Introduction

Proving formula depth lower bounds is an important and difficult challenge in complexity theory: the strongest known lower bound $(3 - o(1)) \log n$ proved by Håstad [6] (following a line of works starting from Subbotovskaya [17, 9, 16]) remains unbeaten for more than 25 years already (in 2014, Tal [18] improved lower order terms in this lower bound). One of the most actively studied approaches to this problem is the one suggested by Karchmer, Raz, and Wigderson [11]. They conjectured that the naive approach of computing a composition of two functions is close to optimal. Namely, for two Boolean functions $f: \{0, 1\}^m \rightarrow \{0, 1\}$ and $g: \{0, 1\}^n \rightarrow \{0, 1\}$, define their composition $f \diamond g: \{0, 1\}^{m \times n} \rightarrow \{0, 1\}$ as a function that first applies g to every row of the input matrix and then applies f to the resulting



© Nikolai Chukhin, Alexander S. Kulikov, and Ivan Mihajlin;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 26; pp. 26:1–26:15



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



26:2 Strong Composition of XOR and a Random Function

column vector. The KRW conjecture then states that $D(f \diamond g)$ is close to $D(f) + D(g)$, where $D(\cdot)$ denotes the minimum depth of a de Morgan formula computing the given function. Karchmer, Raz, and Wigderson [11] proved that if the conjecture is true, then $P \not\subseteq NC^1$, that is, there are functions in P that cannot be computed in logarithmic parallel time.

A convenient way of studying the KRW conjecture is through the framework of Karchmer–Wigderson relation [12]. It not only allows one to apply the tools from communication complexity, but also suggests various important special cases of the conjecture. For a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, the relation KW_f is defined as follows:

$$KW_f = \{(a, b, i) : a \in f^{-1}(1), b \in f^{-1}(0), i \in [n], a_i \neq b_i\}.$$

The communication complexity $CC(KW_f)$ of this relation is the minimum number of bits that Alice and Bob need to exchange to solve the following communication problem: Alice is given $a \in f^{-1}(1)$, Bob is given $b \in f^{-1}(0)$, and their goal is to find an index $i \in [n]$ such that $(a, b, i) \in KW_f$ (i.e., $a_i \neq b_i$). Karchmer and Wigderson [12] proved that, for any function f , the communication complexity of KW_f is equal to the depth complexity of f : $CC(KW_f) = D(f)$. Within this framework, the KRW conjecture is restated as follows: $CC(KW_f \diamond KW_g)$ is close to $CC(KW_f) + CC(KW_g)$ (where $KW_f \diamond KW_g$ is another name for $KW_{f \diamond g}$).

One natural way of relaxing the conjecture is to replace one or both of the two relations KW_f and KW_g by the universal relation, defined as follows:

$$U_n = \{(a, b, i) : a, b \in \{0, 1\}^n, a \neq b, i \in [n], a_i \neq b_i\}.$$

Using a universal relation instead of the Karchmer–Wigderson relation makes the corresponding communication game only harder, hence proving lower bounds for it is potentially easier and could lead to the resolution of the original conjecture. For this reason, such relaxations have been studied intensively.

Edmonds et al. [4] proved the KRW conjecture for the composition $U_m \diamond U_n$ of two universal relations using communication complexity methods. Håstad and Wigderson [7] improved it for a higher degree of composition using a different approach. Karchmer et al. [11] extended this result to monotone functions. Håstad [6] demonstrated the conjecture for the composition $f \diamond XOR_n$ of an arbitrary function $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with the parity function XOR_n . This was later reaffirmed by Dinur and Meir [3] through a communication complexity approach. Further advancements were made by Gavinsky et al. [5] who established the conjecture for the composition $f \diamond U_n$ of any non-constant function $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with the universal relation U_n . Mihajlin and Smal [15] proved the KRW conjecture for the composition of a universal relation with certain hard functions using XOR-composition. Subsequently, Wu [20] improved this result by extending it to the composition of a universal relation with a wider range of functions (though still not with the majority of them). de Rezende et al. [2] proved the conjecture in a semi-monotone setting for a wide range of functions g .

Another natural way of relaxing the initial conjecture is to constrain the communication game (instead of allowing for more inputs for the game). In the *strong composition* $KW_f \otimes KW_g$, Alice receives $X \in (f \diamond g)^{-1}(1)$ and Bob receives $Y \in (f \diamond g)^{-1}(0)$, and their objective is to identify a pair of indices (i, j) such that $X_{i,j} \neq Y_{i,j}$, similar to the regular composition. However, this time it must hold additionally that $g(X_i) \neq g(Y_i)$.

This way of relaxing the conjecture was considered in a number of previous papers and was formalized recently by Meir [14]. Håstad and Wigderson, in their proof of the lower bound for two universal relations, initially establish the result for what they call the extended

universal relation, a concept closely related to strong composition. Similarly, Karchmer et al. [11] demonstrate that, in the monotone setting, strong composition coincides with the standard composition. de Rezende et al. [2] utilized this notion, although without explicitly naming it. Meir [14] formalized the notion of strong composition in his proof of the relaxation of the KRW conjecture.

► **Theorem 1** (Meir, [14]). *There exists a constant $\gamma > 0.04$ such that for every non-constant function $f: \{0, 1\}^m \rightarrow \{0, 1\}$ and for all $n \in \mathbb{N}$, there exists a function $g: \{0, 1\}^n \rightarrow \{0, 1\}$ such that*

$$\text{CC}(\text{KW}_f \circledast \text{KW}_g) \geq \log \text{CC}(\text{KW}_f) - (1 - \gamma)m + n - O(\log(mn)).$$

1.1 Our Result

Håstad [6] proved the KRW conjecture for $\text{KW}_f \circledast \text{KW}_{\text{XOR}}$. However it is still an open question to prove the KRW conjecture for $\text{KW}_{\text{XOR}} \circledast \text{KW}_f$. In this paper, we study the strong composition $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$ of the parity function XOR_m with a random function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$. Since Alice and Bob receive an input of size $m \log m$, we estimate the size of $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$ in terms of $n = m \log m$. It is not difficult to see that the communication complexity of the corresponding game is at most $3 \log n$: KW_f can be solved in $\log m$ bits of communication, whereas KW_{XOR_m} can be solved in $2 \log m$ bits of communication, using the standard divide-and-conquer approach (Alice sends the parity of the first half, Bob then identifies the half in which the parity differs, thus, by utilizing 2 bits of communication, the input size is reduced by a factor of two). We prove that if the function f is well balanced and hard to approximate (which happens with probability $1 - o(1)$), then the bound $3 \log n$ is essentially optimal. Below, we state the result in terms of the protocol size (i.e., the number of leaves), rather than depth, since this gives a more general lower bound. In particular, it immediately implies a $(3 - o(1)) \log n$ depth lower bound.

► **Corollary 2.** *With probability $1 - o(1)$, for a random function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$, any protocol solving $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$ has at least $n^{3-o(1)}$ leaves, where $n = m \log m$.*

In turn, this result follows from the following general lower bound, given in terms of $\mathbb{L}_{\frac{3}{4}}$ that stands for the smallest size of a formula that agrees with f on a $\frac{3}{4}$ fraction of inputs.

► **Theorem 3.** *For any 0.49-balanced function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$, any protocol solving $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$ has at least $n^{2-o(1)} \cdot \mathbb{L}_{\frac{3}{4}}(f)$ leaves, where $n = m \log m$.*

In contrast to many results mentioned above and similarly to the bound by de Rezende et al. [2], our result works for a wide range of inner functions f , what brings us closer to resolving KRW, which makes a claim about the complexity of composing any pair of functions. Also, many of the previous techniques work well in the regime where the outer function is hard and give no strong lower bounds when the outer function is easy (as it is the case with the XOR function). For example, random restrictions (as one of the most successful methods for proving lower bounds) does not seem to give meaningful lower bounds for $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$, as under a random restriction this composition turns into a XOR of a small number of variables which is easy to compute. The lower bound by Meir (see Theorem 1) also gives strong lower bounds in the regime where the outer function is hard (and only gives a trivial lower bound of the form $o(\log n)$ for the function that we study).

To prove the lower bound, we exploit formal complexity measures. As in [4, 15], we consider two stages of a protocol solving $\text{KW}_{\text{XOR}_m} \circledast \text{KW}_f$. During the first stage, we track the progress using the classical measure by Khrapchenko [13] and ensure that even after many steps of the

protocol, there are still many instances of f that need to be solved. At the second stage, we switch to another formal complexity measure and show that the remaining communication problem is, roughly, not easier than $\text{KW}_{\text{OR}} \otimes \text{KW}_f$. We believe that this proof technique is interesting on its own, since it is not only easy to show that Khrapchenko's measure cannot give superquadratic size lower bounds, but it is also known that natural generalizations of this measure are also unable to give stronger than quadratic lower bounds [8]. For more details on Khrapchenko's measure and its limitations, see Sections 2.1 and 2.5.

2 Notation, Known Facts, and Technical Lemmas

Throughout the paper, \log denotes the binary logarithm whereas \ln denotes the natural logarithm. By n we usually denote the size of the input. All asymptotic estimates are given under an implicit assumption that n goes to infinity. By $[n]$, we denote the set $\{1, 2, \dots, n\}$. By \mathbb{R}_+ we denote the set $\{x \in \mathbb{R} : x > 0\}$. We utilize the following asymptotic estimates for binomial coefficients. For any constant $0 < \alpha < 1$,

$$\Omega(n^{-1/2})2^{h(\alpha)n} \leq \binom{n}{\alpha n} \leq 2^{h(\alpha)n}, \quad (1)$$

where $h(x) = -x \log x - (1-x) \log(1-x)$ denotes the binary entropy function.

For a string $x \in \{0, 1\}^n$, its i -th bit of x is denoted by x_i . For a matrix $X \in \{0, 1\}^{m \times n}$, by X_i we denote the i -th row of X and by $X_{i,j}$ we denote the bit of X in the intersection of the i -th row and the j -th column.

2.1 Graphs

For a rooted tree, the *depth* of its node is the number of edges on the path from the node to the root; the depth of the tree is the maximum depth of its nodes.

Let $G(V, E)$ be a graph and $\emptyset \neq A \subseteq V$ be its nonempty subset of nodes. By $G[A]$, we denote a subgraph of G induced by A . By $\text{avgdeg}(G, A)$, we denote the average degree of A :

$$\text{avgdeg}(G, A) = \frac{1}{|A|} \sum_{v \in A} \deg(v). \quad (2)$$

For a bipartite graph $G(A \sqcup B, E)$ with nonempty parts, let

$$\psi(G) = \text{avgdeg}(G, A) \cdot \text{avgdeg}(G, B). \quad (3)$$

Clearly, $\psi(G) \leq |A| \cdot |B|$. The lemma below shows that this graph measure is subadditive.

► **Lemma 4.** *Let $G(A \sqcup B, E)$ be a bipartite graph and $A = A_L \sqcup A_R$ be a partition of A into two parts. Let $G_L = G[A_L \sqcup B]$ and $G_R = G[A_R \sqcup B]$. Then,*

$$\psi(G) \leq \psi(G_L) + \psi(G_R).$$

Proof. Let E_L and E_R be the set of edges of G_L and G_R , respectively. Clearly, $E = E_L \sqcup E_R$. Then,

$$\begin{aligned} \psi(G) \leq \psi(G_L) + \psi(G_R) &\iff \frac{|E|^2}{(|A_L| + |A_R|)|B|} \leq \frac{|E_L|^2}{|A_L||B|} + \frac{|E_R|^2}{|A_R||B|} \\ &\iff \frac{|E_L|^2 + |E_R|^2 + 2|E_L||E_R|}{|A_L| + |A_R|} \leq \frac{|E_L|^2}{|A_L|} + \frac{|E_R|^2}{|A_R|} \\ &\iff 2|E_L||E_R||A_L||A_R| \leq |E_R|^2|A_L|^2 + |E_L|^2|A_R|^2 \\ &\iff 0 \leq (|E_R||A_L| - |E_L||A_R|)^2. \quad \blacktriangleleft \end{aligned}$$

The next lemma shows that if G contains a node of small enough degree, then deleting it not only does not drop ψ , but also does not drop too much the average degree of the parts.

► **Lemma 5.** *Let a node $a \in A$ of a bipartite graph $G(A \sqcup B, E)$ satisfy $\deg(G, a) \leq \text{avgdeg}(G, A)/2$ and let $A' = A \setminus \{a\}$ and $G'(A' \sqcup B, E') = G[A \setminus \{a\} \sqcup B]$. Then,*

$$\psi(G') \geq \psi(G), \tag{4}$$

$$\text{avgdeg}(G', A') \geq \text{avgdeg}(G, A), \tag{5}$$

Proof. The inequality $\text{avgdeg}(G', A') \geq \text{avgdeg}(G, A)$ holds since A' results from A by removing a node of degree less than the average degree.

To prove the inequality (4), let $d = \deg(G, a)$. Then, $|E'| = |E| - d$ and

$$\begin{aligned} \psi(G') \geq \psi(G) &\iff \frac{(|E| - d)^2}{(|A| - 1)|B|} \geq \frac{|E|^2}{|A||B|} \\ &\iff \frac{|E|^2 - 2|E|d + d^2}{(|A| - 1)|B|} \geq \frac{|E|^2}{|A||B|} \\ &\iff \frac{|E| - 2d}{|A| - 1} \geq \frac{|E|}{|A|} \\ &\iff |E||A| - 2d|A| \geq |E|(|A| - 1) \\ &\iff d \leq \frac{|E|}{2|A|} = \frac{\text{avgdeg}(G, A)}{2}. \end{aligned} \quad \blacktriangleleft$$

2.2 Boolean Functions

By \mathbb{B}_n , we denote the set of all Boolean functions on n variables. For two disjoint sets $A, B \subseteq \{0, 1\}^n$, the set $A \times B$ is called a *combinatorial rectangle*, and it is called *full* if A and B form a partition of $\{0, 1\}^n$. Clearly, there is a bijection between \mathbb{B}_n and full combinatorial rectangles. For $f \in \mathbb{B}_n$, by $R_f = f^{-1}(1) \times f^{-1}(0)$, we denote the corresponding full rectangle. We say that a Boolean function f is *balanced* if $|f^{-1}(0)| = |f^{-1}(1)|$.

In this paper, it will prove convenient to apply a function $g \in \mathbb{B}_m$ not only to Boolean vectors $x \in \{0, 1\}^m$, but also to matrices $X \in \{0, 1\}^{n \times m}$:

$$g(X) = (g(X_1), \dots, g(X_n)),$$

i.e., $g(X) \in \{0, 1\}^n$ results by applying g to every row of X . This allows to define a composition in a natural way. For $f \in \mathbb{B}_m$ and $g \in \mathbb{B}_n$, their *composition* $f \diamond g: \{0, 1\}^{m \times n} \rightarrow \{0, 1\}$ treats the input as an $m \times n$ matrix and first applies g to all its rows and then applies f to the resulting column-vector:

$$f \diamond g(X) = f(g(X)) = f(g(X_1), \dots, g(X_m)).$$

For a set of matrices $\mathcal{X} \subseteq \{0, 1\}^{m \times n}$, by i -th projection $\text{proj}_i \mathcal{X}$, we denote the set of all i -th rows among the matrices of \mathcal{X} :

$$\text{proj}_i \mathcal{X} = \{X_i: X \in \mathcal{X}\} = \{t \in \{0, 1\}^n: \exists X \in \mathcal{X}: t = X_i\}. \tag{6}$$

In the proof of the main result, we will be dealing with Boolean matrices of dimension $n \times \log n$. Let $\mathcal{X} \subseteq \{0, 1\}^{n \times \log n}$ be a set of such matrices. We say that \mathcal{X} is α -*bounded* if $|\text{proj}_i \mathcal{X}| \leq \alpha n$, for all $i \in [n]$. The i -th projection of \mathcal{X} is called *sparse* if $|\text{proj}_i \mathcal{X}| < \frac{3}{8}n$, and *dense* otherwise. The following lemma shows that if $|\mathcal{X}|$ is large and \mathcal{X} is α -bounded,

26:6 Strong Composition of XOR and a Random Function

then the number of sparse projections of \mathcal{X} is low. Later on, we will be applying this lemma for \mathcal{X} which is almost 0.5-bounded and whose size gradually decreases to argue that the number of sparse projections cannot grow too fast.

► **Lemma 6.** *Let $k \in \mathbb{N}$ and $\alpha \in (\frac{3}{8}, \frac{1}{2}]$. If $\mathcal{X} \subseteq \{0, 1\}^{n \times \log n}$ is α -bounded and $|\mathcal{X}| \geq \alpha^n \frac{n^n}{2^k}$, then the number of sparse projections of \mathcal{X} does not exceed $k \log^{-1} \frac{8\alpha}{3}$.*

Proof. Let $\beta_i \in [0, 1]$ be such that $|\text{proj}_i \mathcal{X}| = \beta_i \alpha n$. The i -th projection is sparse if and only if $\beta_i < \frac{3}{8\alpha}$. Let t be the number of sparse projections and assume, without loss of generality, that the first t projections are sparse. Then,

$$\begin{aligned} \alpha^n \frac{n^n}{2^k} \leq |\mathcal{X}| &\leq \prod_{i=1}^n |\text{proj}_i \mathcal{X}| = (\alpha n)^n \prod_{i=1}^n \beta_i \iff \\ \frac{1}{2^k} &\leq \prod_{i=1}^n \beta_i = \prod_{i=1}^t \beta_i \cdot \prod_{i=t+1}^n \beta_i < \left(\frac{3}{8\alpha}\right)^t \implies k \log^{-1} \frac{8\alpha}{3} \geq t. \quad \blacktriangleleft \end{aligned}$$

2.3 Boolean Formulas

The computational model studied in this paper is *de Morgan formulas*: it is a binary tree whose leaves are labeled by variables x_1, \dots, x_n and their negations whereas internal nodes (called gates) are labeled by \vee and \wedge (binary disjunction and conjunction, respectively). Such a formula *computes* a Boolean function $f(x_1, \dots, x_n) \in \mathbb{B}_n$. We also say that a formula F *separates* a rectangle $A \times B$, if $f(a) = 1$ and $f(b) = 0$, for all $(a, b) \in A \times B$. This way, if a formula F computes a function f , then it separates R_f .

For a formula F , the *size* $L(F)$ is defined as the number of leaves in F . This extends to Boolean functions: for $f \in \mathbb{B}_n$, by $L(f)$ we denote the smallest size of a formula computing f . Similarly, the *depth* $D(F)$ is the depth of the tree whereas $D(f)$ is the smallest depth of a formula computing f .

By $L_{\frac{3}{4}}(f)$, we denote the smallest size of a formula F that agrees with f on a $\frac{3}{4}$ fraction of inputs, i.e.,

$$\Pr_{x \in \{0,1\}^n} [F(x) = f(x)] \geq \frac{3}{4}.$$

We say that F *approximates* f .

It is known that formulas can be balanced: $D(f) = \Theta(\log L(f))$ (see references in [10, Section 6.1]): this is proved by showing that, for any formula F , there exists an equivalent formula F' with $L(F') \leq L(F)^{O(1)}$ and $D(F') \leq O(\log L(F))$. The following theorem further refines this: by allowing a larger constant in the depth upper bound, one can control the size of the resulting balanced formula.

► **Theorem 7** ([1]). *For any $k \geq 2$ and any formula F , there exists an equivalent formula F' satisfying $D(F') \leq 3 \ln 2 \cdot k \cdot \log L(F)$ and $L(F') \leq L(F)^\gamma$, where $\gamma = 1 + \frac{1}{1 + \log(k-1)}$.*

Using a counting argument, one can show that, with probability $1 - o(1)$, for a random Boolean function $f \in \mathbb{B}_n$, $L(f) = \Omega(2^n / \log n)$. To prove this, one compares the number of small size formulas with the number of Boolean functions $|\mathbb{B}_n| = 2^{2^n}$, using the following estimate (see [10, Lemma 1.23]). It ensures that the number of formulas of size at most $\frac{2^n}{100 \log n}$ is $o(|\mathbb{B}_n|)$:

$$(17n)^{\frac{2^n}{100 \log n}} = 2^{\frac{\log(17n)}{100 \log n} 2^n}.$$

► **Lemma 8.** For all large enough l , the number of Boolean formulas over n variables with at most l leaves is at most

$$(17n)^l. \tag{7}$$

Proof. The number of binary trees with l leaves is at most 4^l . For each such tree, there are at most $(4n)^l$ ways to convert it into a de Morgan formula: there are $2n$ input literals for the leaves and two operations for each internal gate. Consequently, the total number of formulas with at most l leaves is at most

$$l \cdot 4^l \cdot (4n)^l = l \cdot 16^l \cdot n^l \leq (17n)^l,$$

which is true for $l \geq 71$. ◀

We say that $f \in \mathbb{B}_n$ is α -balanced if

$$\alpha \cdot 2^n \leq |f^{-1}(0)|, |f^{-1}(1)| \leq (1 - \alpha) \cdot 2^n$$

i.e., $||f^{-1}(0)| - |f^{-1}(1)|| \leq (1 - 2\alpha)2^n$.

► **Lemma 9.** For all sufficiently large n and any constant $\frac{3}{8} < \alpha < \frac{1}{2}$, a random function $f \in \mathbb{B}_n$ is α -balanced and $L_{\frac{3}{4}}(f) = \Omega(\frac{2^n}{\log n})$, with probability $1 - o(1)$.

Proof. For a formula over n variables, the number of Boolean functions it approximates is at most (by the estimate (1))

$$\sum_{d=3 \cdot 2^{n/4}}^{2^n} \binom{2^n}{d} = \sum_{d=0}^{2^n/4} \binom{2^n}{d} \leq 2^n \binom{2^n}{2^n/4} \leq 2^n \cdot 2^{h(1/4)2^n}.$$

Combining this with (7), we get that the number of functions approximated by formulas of size $\beta \frac{2^n}{\log n}$ is at most

$$(17n)^{\beta \frac{2^n}{\log n}} 2^n 2^{h(1/4)2^n} = 2^{2^n(\beta \frac{\log(17n)}{\log n} + h(1/4)) + n}.$$

For any constant $0 < \beta < 1 - h(1/4)$, this is a $o(1)$ fraction of \mathbb{B}_n .

Now, the probability that a random $f \in \mathbb{B}_n$ is not α -balanced (i.e., $||f^{-1}(0)| - |f^{-1}(1)|| > (1 - 2\alpha) \cdot 2^n$) is at most

$$\frac{1}{2^{2^n}} \cdot 2 \cdot \sum_{i=0}^{\alpha \cdot 2^n - 1} \binom{2^n}{i} \leq \frac{1}{2^{2^n}} \cdot 2 \cdot \alpha \cdot 2^n \cdot \binom{2^n}{\alpha \cdot 2^n} \leq 2^{2^n(h(\alpha) - 1) + n + 1} = o(1).$$

Thus, with probability $1 - o(1)$, a random $f \in \mathbb{B}_n$ is α -balanced and hard to approximate. ◀

2.4 Karchmer–Wigderson Games

Karchmer and Wigderson [12] came up with the following characterization of Boolean formulas. For a Boolean function $f \in \mathbb{B}_n$, the *Karchmer–Wigderson game* KW_f is the following communication problem. Alice is given $a \in f^{-1}(1)$, whereas Bob is given $b \in f^{-1}(0)$, and their goal is to find an index $i \in [n]$ such that $a_i \neq b_i$. A *communication protocol* for KW_f is a rooted binary tree whose leaves are labeled with indices from $[n]$ and each internal node v is labeled either by a function $A_v: f^{-1}(1) \rightarrow \{0, 1\}$ or by a function $B_v: f^{-1}(0) \rightarrow \{0, 1\}$. For any pair $(a, b) \in f^{-1}(1) \times f^{-1}(0)$, one can reach a leaf of the protocol by traversing

a path from the root to a leaf to determine to which of the two children to proceed from a node v , one computes either $A_v(a)$ or $B_v(b)$. We say that a protocol *solves* KW_f , if for any $(a, b) \in f^{-1}(1) \times f^{-1}(0)$, one reaches a leaf $i \in [n]$ such that $a_i \neq b_i$. Similarly to formulas, we say that a protocol separates a combinatorial rectangle $A \times B$, if it works correctly for all pairs $(a, b) \in A \times B$.

Karchmer and Wigderson showed that formulas computing f and protocols solving KW_f can be transformed (even mechanically) into one another. In particular, the smallest number of leaves in the protocol solving KW_f is equal to $L(f)$, whereas the smallest depth of a protocol (also known as the communication complexity of KW_f , denoted by $\text{CC}(\text{KW}_f)$) is nothing else but $D(f)$. By $L(A \times B)$ for a combinatorial rectangle $A \times B$, we denote the minimum number of leaves in a protocol separating A and B .

With each node of a protocol solving KW_f , one can associate a combinatorial rectangle in a natural way. The root of the protocol corresponds to R_f . For the two children of Alice's node v with a rectangle $A \times B$, one associates two rectangles $A_0 \times B$ and $A_1 \times B$, where $A_i = \{a \in A : A_v(a) = i\}$. This way, Alice splits the current rectangle horizontally. Similarly, when Bob speaks, he splits the current rectangle vertically. Each leaf of a protocol solving KW_f is associated with a *monochromatic* rectangle, i.e., a rectangle $A \times B$ such that there exists $i \in [n]$ for which $a_i \neq b_i$ for all $(a, b) \in A \times B$.

For functions $f \in \mathbb{B}_m$ and $g \in \mathbb{B}_n$, the *strong composition* of KW_f and KW_g , denoted as $\text{KW}_f \otimes \text{KW}_g$, is the following communication problem: Alice and Bob receive inputs $X \in (f \diamond g)^{-1}(1)$ and $Y \in (f \diamond g)^{-1}(0)$, respectively, and need to find indices (i, j) such that $X_{i,j} \neq Y_{i,j}$ and $g(X_i) \neq g(Y_j)$. We say that a protocol *strongly separates* sets $\mathcal{X} \subseteq (f \diamond g)^{-1}(1)$ and $\mathcal{Y} \subseteq (f \diamond g)^{-1}(0)$, if it solves the strong composition $\text{KW}_f \otimes \text{KW}_g$ on inputs $\mathcal{X} \times \mathcal{Y}$.

2.5 Formal Complexity Measures

For $f \in \mathbb{B}_n$, define a bipartite graph $G_f(f^{-1}(1) \sqcup f^{-1}(0), E_f)$ as follows:

$$E_f = \{\{u, v\} : u \in f^{-1}(1), v \in f^{-1}(0), d_H(u, v) = 1\},$$

where d_H is the Hamming distance. Khrapchenko [13] proved that, for any $f \in \mathbb{B}_n$, $\psi(G_f) \leq L(f)$ (recall (3) for the definition of $\psi(G)$). This immediately gives a lower bound $L(\text{XOR}_n) \geq n^2$. Note the two useful properties of $\psi(G_f)$: on the one hand, it is a lower bound to $L(f)$, on the other hand, it is much easier to estimate than $L(f)$.

Paterson [19, Section 8.8] noted that Khrapchenko's approach can be cast as follows. A function $\mu: \mathbb{B}_n \rightarrow \mathbb{R}_+$ is called a *formal complexity measure* if it satisfies the following two properties:

1. normalization: $\mu(x_i), \mu(\bar{x}_i) \leq 1$, for all $i \in [n]$,
 2. subadditivity: $\mu(f \vee g) \leq \mu(f) + \mu(g)$ and $\mu(f \wedge g) \leq \mu(f) + \mu(g)$, for all $f, g \in \mathbb{B}_n$.
- Note that Khrapchenko's measure can be defined in this notation as $\phi(f) = \psi(G_f)$. Its subadditivity is shown in Lemma 4, whereas the normalization property can be easily seen.

It is not difficult to see that L itself is a formal complexity measure. Moreover, it turns out that it is the largest formal complexity measure.

► **Lemma 10** (Lemma 8.1 in [19]). *For any formal complexity measure $\mu: \mathbb{B}_n \rightarrow \mathbb{R}$ and any $f \in \mathbb{B}_n$, $\mu(f) \leq L(f)$.*

3 Proof of the Main Result

In this section, we prove the main result of the paper.

► **Theorem 3.** *For any 0.49-balanced function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$, any protocol solving $\text{KW}_{\text{XOR}_m} \otimes \text{KW}_f$ has at least $n^{2-o(1)} \cdot \mathsf{L}_{\frac{3}{4}}(f)$ leaves, where $n = m \log m$.*

► **Corollary 2.** *With probability $1 - o(1)$, for a random function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$, any protocol solving $\text{KW}_{\text{XOR}_m} \otimes \text{KW}_f$ has at least $n^{3-o(1)}$ leaves, where $n = m \log m$.*

Proof. Lemma 9 guarantees that for a random function $f: \{0, 1\}^{\log m} \rightarrow \{0, 1\}$, f is 0.49-balanced and $\mathsf{L}_{\frac{3}{4}}(f) = \Omega(m / \log \log m)$ with probability $1 - o(1)$. Plugging this into Theorem 3 gives the required lower bound. ◀

3.1 Proof Overview

We start by proving a lower bound on the size of any protocol solving $\text{KW}_{\text{XOR}_m} \otimes \text{KW}_f$ and having a logarithmic depth. Then, using balancing techniques (see Theorem 7), we generalize the size lower bound to all protocols.

Fix a set $\mathcal{Z} \subseteq \{0, 1\}^{\log m}$ such that $|\mathcal{Z}| = 0.98m$ and f is balanced on \mathcal{Z} : $|\mathcal{X}_0| = |\mathcal{Y}_0| = 0.49m$, where $\mathcal{X}_0 = f^{-1}(1) \cap \mathcal{Z}$ and $\mathcal{Y}_0 = f^{-1}(0) \cap \mathcal{Z}$.

We prove a lower bound for any protocol that strongly separates $\text{KW}_{\text{XOR}_m} \otimes \text{KW}_f$ on inputs $\mathcal{X}_T \times \mathcal{Y}_T$ (which are defined later). To this end, we associate, with nodes of the protocol, a graph similar to G_{XOR_m} and use Khrapchenko’s measure to track the progress of the protocol. A node of the graph is associated with all inputs X having the same vector $f(X)$. The reasoning is that, in a natural scenario, the protocol will first solve XOR_m , followed by solving f , implying that the protocol does not need to distinguish between X and X' in the initial rounds, if $f(X) = f(X')$. We connect two graph nodes by an edge if their vectors differ in exactly one coordinate.

We aim to ensure that each edge in the graph has a large projection: for any two nodes connected by an edge, the elements of blocks associated with them cover a substantial number of inputs for the function f . There will be no small protocol capable of solving the problem within these two blocks since f is hard to approximate. This is the rationale behind ensuring that all edges in the graph have large projections on both sides. To achieve this, we enforce that each block that is associated with a node shrinks by at most a factor of two at each step of the protocol. This process ensures that a significant number of edges in the graph will maintain large projections on both sides.

Once the Khrapchenko measure becomes sufficiently small, we can assert that XOR_m is nearly solved, and the protocol, in a sense, must now solve an instance of $\text{OR}_d \otimes f$. Using the fact that solving each edge independently is hard, we conclude that solving an $\text{OR}_d \otimes f$ over these edges should be as difficult as approximately $d \cdot \mathsf{L}_{\frac{3}{4}}(f)$.

3.2 Proof

Throughout this section, we assume that m is large enough and $f \in \mathbb{B}_{\log m}$ is a fixed function that is 0.49-balanced. Fix sets $\mathcal{X}_0 \subseteq f^{-1}(1)$, $\mathcal{Y}_0 \subseteq f^{-1}(0)$ of size $0.49m$ and let

$$\mathcal{X}_T = \{X \in \{0, 1\}^{m \times \log m} : (\text{XOR}_m \diamond f)(X) = 1 \wedge X_i \in \mathcal{X}_0 \sqcup \mathcal{Y}_0, \forall i \in [m]\}, \quad (8)$$

$$\mathcal{Y}_T = \{Y \in \{0, 1\}^{m \times \log m} : (\text{XOR}_m \diamond f)(Y) = 0 \wedge Y_i \in \mathcal{X}_0 \sqcup \mathcal{Y}_0, \forall i \in [m]\}. \quad (9)$$

Let $\alpha > 0$ be a constant and P be a protocol that strongly separates $\mathcal{X}_T \times \mathcal{Y}_T$ and has depth at most $\alpha \log m$. Recall that each node S of P is associated with a rectangle $\mathcal{X}_S \times \mathcal{Y}_S$. We build a subtree D of P having the same root and associate a graph G_N to every node N of D . The graphs G_N are built inductively from the graphs associated with the parents of N

26:10 Strong Composition of XOR and a Random Function

as explained below, but all these graphs are subsets of the m -dimensional hypercube: the set of nodes of each such graph is a subset of $\{0, 1\}^m$ and for each edge $\{u, v\}$ it holds that $d_H(u, v) = 1$.

For the root T of the protocol P , the graph G_T is simply G_{XOR_m} (which is nothing else but the m -dimensional hypercube): its set of nodes is $\{0, 1\}^m$, two nodes are joined by an edge with label i if they differ in the i -th coordinate.

For any node v of the graph G_S , we associate the following set of inputs called *block*:

$$\mathcal{B}_S(v) = \{X \in \{0, 1\}^{m \times \log m} : X \in \mathcal{X}_S \sqcup \mathcal{Y}_S \text{ and } f(X) = v\}.$$

We say that an edge $\{u, v\}$ with label i of G_S is *heavy* if the projection of both $\mathcal{B}_S(u)$ and $\mathcal{B}_S(v)$ onto the i th coordinate is dense, i.e.,

$$|\text{proj}_i \mathcal{B}_S(u)|, |\text{proj}_i \mathcal{B}_S(v)| \geq \frac{3}{8}m,$$

and *light* otherwise.

Since the nodes of the graph G_S form a subset of $\{0, 1\}^m$, we can naturally divide them into two parts, as their blocks correspond to subsets of either \mathcal{X}_S or \mathcal{Y}_S .

$$A_S = \{v \in V(G_S) \mid \text{XOR}_m(v) = 1\}$$

$$B_S = \{v \in V(G_S) \mid \text{XOR}_m(v) = 0\}$$

For a graph G_S , we define $d_A(G_S)$ as the average degree of the part A_S and $d_B(G_S)$ as the average degree of the part B_S . We say that a graph G_S is *special* if

$$\min\{d_A(G_S), d_B(G_S)\} \leq 12\alpha \log^2 m.$$

We will construct the tree D inductively. For a node S in the tree D , we either stop the process if G_S is special, or construct the two children of S from the protocol P and their graphs. We continue building D on these two children inductively. Hence, all graphs corresponding to internal nodes of the tree D are not special, while all graphs associated with leaves of D are special.

► **Definition 11.** A graph G_S , associated with a node S in the tree D , is adjusted if all its edges are heavy and

$$\deg(v) > \frac{d_A(G_S)}{2}, \forall v \in A_S \quad \text{and} \quad \deg(v) > \frac{d_B(G_S)}{2}, \forall v \in B_S. \quad (10)$$

We will ensure that all graphs G_S for any node S in the tree D are adjusted.

► **Lemma 12.** For each node v of the graph G_T (associated with the root T of the protocol P),

1. the degree of v is m ;
2. $|\text{proj}_i \mathcal{B}_T(v)| = 0.49m$, for all $i \in [m]$;
3. $|\mathcal{B}_T(v)| = \frac{(0.98m)^m}{2^m}$.

Proof. Nodes of G_T are m -dimensional binary vectors, hence $\deg(v) = m$.

To prove the second property, recall that f is balanced on $\mathcal{X}_0 \sqcup \mathcal{Y}_0$. If $v_i = 1$ (or $v_i = 0$), for some $i \in [m]$, the i -th projection can take any value from \mathcal{X}_0 (\mathcal{Y}_0 , respectively). Hence, $|\text{proj}_i \mathcal{B}_T(v)| = 0.49m$.

Finally, to prove the third property, note the G_T has 2^m nodes and for each vertex v the size of the block $\mathcal{B}_T(v)$ is at most $(0.49m)^m$. Therefore, since each input from $\mathcal{X}_T \sqcup \mathcal{Y}_T$ belongs to exactly one block that is associated with a node from G_T and $|\mathcal{X}_T \sqcup \mathcal{Y}_T| = |\mathcal{X}_0 \sqcup \mathcal{Y}_0|^m = (0.98m)^m$, $|\mathcal{B}_T(v)| = \frac{(0.98m)^m}{2^m}$. ◀

Lemma 12 ensures that the graph G_T is adjusted and not special, thus the root T has two children. Using the function \mathcal{B} , we show how to construct an intermediate graph H_N for some child of a node S in the tree D and then we apply some cleanup procedures for the graph H_N to construct a graph G_N . Recall that each step of P partitions the set of either Alice's or Bob's inputs into two parts. Let G_S be a graph for some node S of the protocol P that is associated with a rectangle $\mathcal{X}_S \times \mathcal{Y}_S$ and assume, without loss of generality, that it is Alice's turn. Therefore, graph G_S is not special, otherwise we will stop the building process of the subtree of S . Let S_L be the left child of S in the protocol P and S_R be the right child. We add the same children of the node S in the tree D . Then, we put v from B_S into both H_{S_L} and H_{S_R} (since the block $\mathcal{B}_S(v)$ has not changed). For each node $v \in A_S$ we decide in which of the two graphs we will put it. The block $\mathcal{B}_S(v)$ is also split into two: $\mathcal{B}_{S_L}(v)$ and $\mathcal{B}_{S_R}(v)$, corresponding to the two ways of the protocol. We assign v to the left graph H_{S_L} if $2 \cdot |\mathcal{B}_{S_L}(v)| \geq |\mathcal{B}_S(v)|$, and to the right graph H_{S_R} if $2 \cdot |\mathcal{B}_{S_R}(v)| > |\mathcal{B}_S(v)|$. An edge $\{u, v\}$ from the edges of G_S goes to H_{S_L} if and only if both u and v are assigned to H_{S_L} . The same rule applies for edges in H_{S_R} . This approach ensures that the size of each block $\mathcal{B}_S(v)$ shrinks by at most a factor of two when transitioning from a parent to a child in the tree D . Then, the graphs G_{S_L} and G_{S_R} will be built using graphs H_{S_L} and H_{S_R} , respectively.

The idea of the structure of the graph G_S arises from Khrapchenko's graph, so we will use the same measure:

$$\psi(G_S) = d_A(G_S) \cdot d_B(G_S).$$

Lemma 4 states that ψ is subadditive.

After obtaining the graph H_C for a node C of the tree D , we make our first cleanup by deleting all light edges: let H'_C be a graph resulting from H_C by removing all its light edges. The next lemma shows that this does not drop the measure ψ too much.

► **Lemma 13.**

$$\psi(H'_C) \geq \psi(H_C) \left(1 - \frac{1}{\log m}\right).$$

Proof. Let S be the parent of C in D . Since S is not a leaf, we have that $\min\{d_A(G_S), d_B(G_S)\} > 12\alpha \log^2 m$ and the degree of every node in G_S is at least half of the average degree of its part. Without loss of generality, assume that inputs were deleted from \mathcal{X}_S , and therefore $d_A(H_C) \geq \frac{d_A(G_S)}{2} > 6\alpha \log^2 m$.

An edge $\{u, v\}$ can become light because of only one of its endpoints, because the blocks on the other side remain unchanged. From Lemma 12, we know that the initial size of each block is $(0.49m)^m$, and after each step of the protocol, the size of a block shrinks by at most a factor of two. Hence, for any node v , the size of its block $\mathcal{B}_S(v)$ is at least $\frac{(0.49m)^m}{2^{\alpha \log m}}$, because the protocol depth is bounded by $\alpha \log m$. Hence, we can bound the number of light edges incident to v by $3\alpha \log m$ using Lemma 6 (since $\log^{-1}(8 \cdot 0.49/3) < 3$). Therefore,

$$d_A(H'_C) \geq d_A(H_C) - 3\alpha \log m.$$

Now, consider $d_B(H'_C)$. Let E_C be the set of edges in H_C , whereas A_C and B_C be its parts of nodes. Then,

$$d_B(H'_C) \geq \frac{E_C - |A_C| \cdot 3\alpha \log m}{|B_C|} = d_B(H_C) - 3\alpha \log m \frac{|A_C|}{|B_C|}.$$

26:12 Strong Composition of XOR and a Random Function

Hence,

$$\begin{aligned}
\psi(H'_C) &= d_A(H'_C)d_B(H'_C) \geq (d_A(H_C) - 3\alpha \log m) \left(d_B(H_C) - 3\alpha \log m \frac{|A_C|}{|B_C|} \right) \\
&\geq \psi(H_C) - 3\alpha d_B(H_C) \log m - \frac{3\alpha |A_C| d_A(H_C) \log m}{|B_C|} \\
&= \psi(H_C) \left(1 - \frac{3\alpha \log m}{d_A(H_C)} - \frac{3\alpha |A_C| \log m}{|B_C|} \right) \\
&= \psi(H_C) \left(1 - \frac{6\alpha \log m}{d_A(H_C)} \right) \\
&> \psi(H_C) \left(1 - \frac{6\alpha \log m}{6\alpha \log^2 m} \right) = \psi(H_C) \left(1 - \frac{1}{\log m} \right). \quad \blacktriangleleft
\end{aligned}$$

The next lemma shows how to construct an adjusted graph G_C , from the intermediate graph H'_C .

► **Lemma 14.** *There exists a subgraph G_C of the graph H'_C such that G_C is adjusted and $\psi(G_C) \geq \psi(H_C) \left(1 - \frac{1}{\log m} \right)$.*

Proof. To get G_C , we keep removing nodes from H'_C until it satisfies (10). If (10) is violated, there exists, without loss of generality, a node $v \in A_C$ such that $\deg(v) \leq \frac{d_A(G_C)}{2}$. Let $G'_C = G_C \setminus \{v\}$. Lemma 5 guarantees that this does not decrease the measure. This process is clearly finite. ◀

This way, we construct the graph G_C for the node C . If C is not special, we continue expanding the subtree rooted at C . Recall also that, for each internal node S of the tree D , whose children are S_L and S_R , the following holds:

$$\psi(G_S) \leq \psi(H_{S_L}) + \psi(H_{S_R}).$$

Hence, combining it with Lemma 14 we have:

$$\psi(G_S) \left(1 - \frac{1}{\log m} \right) \leq \psi(G_{S_L}) + \psi(G_{S_R}). \quad (11)$$

On the other hand, if S is special, we will use the following two lemmas to argue that strongly separating $\mathcal{X}_S \times \mathcal{Y}_S$ is still difficult.

► **Lemma 15.** *Let S be a node of the tree D such that it has a node $v \in G_S$ having d adjacent edges. Then, any protocol that strongly separates \mathcal{X}_S and \mathcal{Y}_S has at least $\Omega \left(d \cdot L_{\frac{3}{4}}(f) \right)$ leaves.*

Proof. Consider the subgraph of G_S induced by v and its neighbors u_1, \dots, u_d connected to v . Denote by l_i the label of the edge $\{v, u_i\}$. Define a measure ξ on subrectangles of $\mathcal{X}_S \times \mathcal{Y}_S$:

$$\xi(\mathcal{X} \times \mathcal{Y}) = \sum_{i=1}^d L(\text{proj}_{l_i} \mathcal{B} \times \text{proj}_{l_i} \mathcal{B}_i),$$

where $\mathcal{X} \subseteq \mathcal{X}_S$, $\mathcal{Y} \subseteq \mathcal{Y}_S$, $\mathcal{B} = \mathcal{X} \cap \mathcal{B}_S(v)$ and $\mathcal{B}_i = \mathcal{Y} \cap \mathcal{B}_S(u_i)$, for all $i \in [d]$. By KW($A \times B$), for any $A \cap B = \emptyset$, we denote a Karchmer-Wigderson communication game where Alice gets $a \in A$, Bob gets $b \in B$, and they need to find $i: a_i \neq b_i$. We prove that any protocol strongly separating $\mathcal{X}_S \times \mathcal{Y}_S$ requires at least $\xi(\mathcal{X}_S \times \mathcal{Y}_S)$ leaves.

It is easy to see that ξ is subadditive, being a sum of subadditive measures: if $\mathcal{X} = \mathcal{X}' \sqcup \mathcal{X}''$, then $\xi(\mathcal{X} \times \mathcal{Y}) \leq \xi(\mathcal{X}' \times \mathcal{Y}) + \xi(\mathcal{X}'' \times \mathcal{Y})$ and the same applies when we split \mathcal{Y} . Namely, let $\mathcal{Y} = \mathcal{Y}' \sqcup \mathcal{Y}''$, $\mathcal{B}_i = \mathcal{Y}' \cap \mathcal{B}_S(u_i)$, and $\mathcal{B}_i'' = \mathcal{Y}'' \cap \mathcal{B}_S(u_i)$. Then,

$$\begin{aligned} \xi(\mathcal{X} \times \mathcal{Y}' \sqcup \mathcal{Y}'') &= \sum_{i=1}^d \mathsf{L}(\text{proj}_{l_i} \mathcal{B} \times \text{proj}_{l_i} \mathcal{B}'_i \sqcup \mathcal{B}''_i) \\ &\leq \sum_{i=1}^d \mathsf{L}(\text{proj}_{l_i} \mathcal{B} \times \text{proj}_{l_i} \mathcal{B}'_i) + \sum_{i=1}^d \mathsf{L}(\text{proj}_{l_i} \mathcal{B} \times \text{proj}_{l_i} \mathcal{B}''_i) \\ &= \xi(\mathcal{X} \times \mathcal{Y}') + \xi(\mathcal{X} \times \mathcal{Y}''). \end{aligned}$$

Consider a protocol P' strongly separating $\mathcal{X}_S \times \mathcal{Y}_S$ and its leaf L associated with a rectangle of inputs $\mathcal{X}'_L \times \mathcal{Y}'_L$. We show that $\xi(\mathcal{X}'_L \times \mathcal{Y}'_L) \leq 1$. Since L is a leaf, there exists i, j such that for each $X \in \mathcal{X}'_L$ and $Y \in \mathcal{Y}'_L$:

$$X_{i,j} \neq Y_{i,j} \quad \text{and} \quad f(X_i) \neq f(Y_i).$$

Let k be such that $\mathcal{B}_k \neq \emptyset$ (if all \mathcal{B}_t are empty, then $\xi = 0$). Then, $\mathcal{B}(u_t) = \emptyset$, for all $t \neq k$, as otherwise there would be no i such that $f(X_i) \neq f(Y_i)$ for all $(X, Y) \in \mathcal{X}'_L \times \mathcal{Y}'_L$, since u_k differs from v in the position l_k , and u_t differs from v in the position l_t and $l_k \neq l_t$. Thus, if $\xi(\mathcal{X}'_L \times \mathcal{Y}'_L) > 1$, then $\mathsf{L}(\text{proj}_{l_k} \mathcal{B} \times \text{proj}_{l_k} \mathcal{B}_k) > 1$, which contradicts to the existence of a pair (i, j) .

Thus, ξ is normal (has the value at most 1 for any leaf of any protocol that strongly separates $\mathcal{X}_S \times \mathcal{Y}_S$) and subadditive. Hence, its value for the whole protocol P' is a lower bound on the size of P' . Thus, it remains to estimate ξ for P' .

Since all d edges are heavy, we have:

$$|\text{proj}_{l_i} \mathcal{B}_S(v)| + |\text{proj}_{l_i} \mathcal{B}_S(u_i)| \geq \frac{3}{4}m, \quad \forall i \in [d].$$

Hence,

$$\mathsf{L}(\text{proj}_{l_i} \mathcal{B}_S(v) \times \text{proj}_{l_i} \mathcal{B}_S(u_i)) = \Omega\left(\mathsf{L}_{\frac{3}{4}}(f)\right),$$

for all $i \in [d]$. Summing over all $i \in [d]$, gives the desired lower bound. \blacktriangleleft

► **Lemma 16.** *For a special node S of the tree D , the number of leaves in any protocol strongly separating $\mathcal{X}_S \times \mathcal{Y}_S$ is*

$$\Omega\left(\frac{\psi(G_S) \cdot \mathsf{L}_{\frac{3}{4}}(f)}{\log^2 m}\right).$$

Proof. Assume, without loss of generality, that

$$d_A(G_S) \geq d_B(G_S) \quad \text{and} \quad d_B(G_S) \leq 12\alpha \log^2 m.$$

Applying Lemma 15 to a node of degree at least $d_A(G_S)$, we get that the number of leaves is at least

$$\Omega\left(d_A(G_S) \cdot \mathsf{L}_{\frac{3}{4}}(f)\right) = \Omega\left(\frac{\psi(G_S)}{d_B(G_S)} \cdot \mathsf{L}_{\frac{3}{4}}(f)\right) = \Omega\left(\frac{\psi(G_S) \cdot \mathsf{L}_{\frac{3}{4}}(f)}{\log^2 m}\right). \quad \blacktriangleleft$$

At this point, everything is ready to lower bound the size of any protocol of logarithmic depth.

26:14 Strong Composition of XOR and a Random Function

► **Theorem 17.** *The size of the protocol P (strongly separating $\mathcal{X}_T \times \mathcal{Y}_T$) is*

$$\Omega\left(\frac{m^2 \cdot \mathbb{L}_{\frac{3}{4}}(f)}{\log^2 m} \left(1 - \frac{1}{\log m}\right)^{\alpha \log m}\right).$$

Proof. Lemma 16 states that the number of leaves needed to resolve any leaf S of the tree D is $\Omega\left(\psi(G_S) \cdot \mathbb{L}_{\frac{3}{4}}(f) / \log^2 m\right)$. Let \mathcal{S} be the set of all leaves of the tree D . Using estimate (11), we have:

$$\psi(G_T) \cdot \left(1 - \frac{1}{\log m}\right)^{\alpha \log m} \leq \sum_{S \in \mathcal{S}} \psi(G_S).$$

Since $\psi(G_T) = m^2$ (by Lemma 12), Then, the number of leaves in P is

$$\Omega\left(\sum_{S \in \mathcal{S}} \frac{\psi(G_S) \cdot \mathbb{L}_{\frac{3}{4}}(f)}{\log^2 m}\right) \geq \Omega\left(\frac{m^2 \cdot \mathbb{L}_{\frac{3}{4}}(f)}{\log^2 m} \left(1 - \frac{1}{\log m}\right)^{\alpha \log m}\right). \quad \blacktriangleleft$$

Recall that α is a constant. Assuming $m \geq 4$, we have $\log m \geq 2$, and thus $1 - \frac{1}{\log m} \geq e^{-\frac{2}{\log m}}$. Then,

$$\frac{m^2 \cdot \mathbb{L}_{\frac{3}{4}}(f)}{\log^2 m} \left(1 - \frac{1}{\log m}\right)^{\alpha \log m} \geq \frac{m^2 \cdot \mathbb{L}_{\frac{3}{4}}(f)}{\log^2 m} e^{-\frac{2}{\log m} \cdot \alpha \log m} \geq m^{2-\varepsilon} \cdot \mathbb{L}_{\frac{3}{4}}(f),$$

for any constant $\varepsilon > 0$ when m is sufficiently large. Hence, the number of leaves needed for a protocol P is $m^{2-o(1)} \cdot \mathbb{L}_{\frac{3}{4}}(f)$.

Finally, we get rid of the assumption that the depth of P is logarithmic and prove the main result.

Proof of Theorem 3. Let P be a protocol with $m^{2-\varepsilon} \cdot \mathbb{L}_{\frac{3}{4}}(f)$ leaves, for some $\varepsilon > 0$, solving $\text{KW}_{\text{XOR}_m} \otimes \text{KW}_f$. We transform it into a protocol P' with $(m^{(2-\varepsilon)} \cdot \mathbb{L}_{\frac{3}{4}})^\gamma$ leaves and depth bounded by $3(3-\varepsilon)k \ln 2 \cdot \log m$, by applying Theorem 7, where $\gamma = 1 + \frac{1}{1+\log(k-1)}$. (Theorem 7 is stated in terms of formulas, but it is not difficult to see that it works also for protocols for strong composition.)

Since $\varepsilon > 0$ and $\lim_{k \rightarrow \infty} \gamma = 1$, there exist k and $\varepsilon' > 0$ such that

$$\left(m^{2-\varepsilon} \cdot \mathbb{L}_{\frac{3}{4}}(f)\right)^\gamma \leq m^{2-\varepsilon'} \cdot \mathbb{L}_{\frac{3}{4}}(f),$$

since $\mathbb{L}_{\frac{3}{4}}(m) \leq m$. Hence, protocol P' has logarithmic depth and at most $m^{2-\varepsilon'} \cdot \mathbb{L}_{\frac{3}{4}}(f)$ leaves, which contradicts Theorem 17. Therefore, P has $\Omega\left(m^{2-o(1)} \cdot \mathbb{L}_{\frac{3}{4}}(f)\right) = \Omega\left(m^{2-o(1)} \cdot \mathbb{L}_{\frac{3}{4}}(f)\right)$ leaves. ◀

References

- 1 Maria Luisa Bonet and Samuel R. Buss. Size-depth tradeoffs for boolean fomulae. *Inf. Process. Lett.*, 49(3):151–155, 1994. doi:10.1016/0020-0190(94)90093-0.
- 2 Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, and Robert Robere. KRW composition theorems via lifting. *Comput. Complex.*, 33(1):4, 2024. doi:10.1007/s00037-024-00250-7.
- 3 Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Comput. Complex.*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.

- 4 Jeff Edmonds, Russell Impagliazzo, Steven Rudich, and Jirí Sgall. Communication complexity towards lower bounds on circuit depth. *Comput. Complex.*, 10(3):210–246, 2001. doi:10.1007/s00037-001-8195-x.
- 5 Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson. Toward better formula lower bounds: The composition of a function and a universal relation. *SIAM J. Comput.*, 46(1):114–131, 2017. doi:10.1137/15M1018319.
- 6 Johan Håstad. The shrinkage exponent of de morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.
- 7 Johan Håstad and Avi Wigderson. Composition of the universal relation. In Jin-Yi Cai, editor, *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 119–134. DIMACS/AMS, 1990. doi:10.1090/dimacs/013/07.
- 8 Pavel Hrubes, Stasys Jukna, Alexander S. Kulikov, and Pavel Pudlák. On convex complexity measures. *Theor. Comput. Sci.*, 411(16-18):1842–1854, 2010. doi:10.1016/j.tcs.2010.02.004.
- 9 Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993. doi:10.1002/rsa.3240040202.
- 10 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.
- 11 Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Comput. Complex.*, 5(3/4):191–204, 1995. doi:10.1007/BF01206317.
- 12 Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discret. Math.*, 3(2):255–265, 1990. doi:10.1137/0403021.
- 13 V. M. Khrapchenko. Method of determining lower bounds for the complexity of p-schemes. *Mathematical notes of the Academy of Sciences of the USSR*, 10(1):474–479, 1971. doi:10.1007/BF01747074.
- 14 Or Meir. Toward better depth lower bounds: A krw-like theorem for strong composition. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1056–1081. IEEE, 2023. doi:10.1109/FOCS57990.2023.00064.
- 15 Ivan Mihajlin and Alexander Smal. Toward better depth lower bounds: The XOR-KRW conjecture. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPICs*, pages 38:1–38:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.CCC.2021.38.
- 16 Mike Paterson and Uri Zwick. Shrinkage of de morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993. doi:10.1002/rsa.3240040203.
- 17 B. A. Subbotovskaya. Realization of linear functions by formulas using \vee , $\&$, $\bar{}$. *Dokl. Akad. Nauk SSSR*, 136(3):553–555, 1961. URL: <http://mi.mathnet.ru/dan24539>.
- 18 Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.65.
- 19 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL: <http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/>.
- 20 Hao Wu. An improved composition theorem of a universal relation and most functions via effective restriction. *CoRR*, abs/2310.07422, 2023. doi:10.48550/arXiv.2310.07422.

Local Equivalence of Stabilizer States: A Graphical Characterisation

Nathan Claudet  

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

Simon Perdrix  

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

Abstract

Stabilizer states form a ubiquitous family of quantum states that can be graphically represented through the graph state formalism. A fundamental property of graph states is that applying a local complementation – a well-known and extensively studied graph transformation – results in a graph that represents the same entanglement as the original. In other words, the corresponding graph states are LU-equivalent. This property served as the cornerstone for capturing non-trivial quantum properties in a simple graphical manner, in the study of quantum entanglement but also for developing protocols and models based on graph states and stabilizer states, such as measurement-based quantum computing, secret sharing, error correction, entanglement distribution... However, local complementation fails short to fully characterise entanglement: there exist pairs of graph states that are LU-equivalent but cannot be transformed one into the other using local complementations. Only few is known about the equivalence of graph states beyond local complementation. We introduce a generalisation of local complementation which graphically characterises the LU-equivalence of graph states. We use this characterisation to show the existence of a strict infinite hierarchy of equivalences of graph states. Our approach is based on minimal local sets, which are subsets of vertices that are known to cover any graph, and to be invariant under local complementation and even LU-equivalence. We use these structures to provide a type to each vertex of a graph, leading to a natural standard form in which the LU-equivalence can be exhibited and captured by means of generalised local complementation.

2012 ACM Subject Classification Theory of computation → Quantum information theory

Keywords and phrases Quantum computing, Graph theory, Entanglement, Local complementation

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.27

Related Version *Full Version:* <https://arxiv.org/abs/2409.20183> [9]

Funding This work is supported by the the *Plan France 2030* through the PEPR integrated project EPiQ ANR-22- PETQ-0007 and the HQI platform ANR-22-PNCQ-0002; and by the European projects NEASQC and HPCQS.

Acknowledgements The authors want to thank David Cattaneo and Mehdi Mhalla for fruitful discussions on previous versions of this paper.

1 Introduction

Stabilizer states, and in particular graph states, form a versatile family of entangled quantum states, which allow for easy and compact representations. They are used as entangled resource states in various quantum information applications, like measurement-based computation [29, 30, 5], error corrections [35, 34, 10, 33], quantum communication network routing [19, 26, 4, 7], and quantum secret sharing [25, 17], to cite a few. In all these applications, stabilizer states are used as multi-partite entangled resources, it is thus crucial to understand when two such states have the same entanglement. According to standard quantum information theory, two quantum states have the same entanglement if they can be transformed into each



© Nathan Claudet and Simon Perdrix;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 27; pp. 27:1–27:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



other using only local operations, where “local” refers to operations that can be applied to at most one qubit at a time. Indeed, intuitively, such local operations can only decrease the strength of the entanglement of a quantum state, thus if a state can be transformed into another and back to the original one using only local operations, the two states do have the same entanglement. Concretely, the most general case is the so-called SLOCC-equivalence (stochastic local operations and classical communications) that encompass the use of local unitaries and measurements. In the particular case of stabilizer states, it is enough to consider *LU-equivalence* (local unitaries), as two stabilizer states are SLOCC-equivalent iff there exists $U = U_1 \otimes \dots \otimes U_n$ that transforms one state into the other, where each U_i is a 1-qubit unitary transformation [21]. Therefore, in this paper, we refer to LU-equivalence when stating that two stabilizer states possess the same entanglement.

Stabilizer states can be graphically represented through the graph state formalism, which consists in representing quantum states using graphs where each vertex corresponds to a qubit. Graph states form actually a sub-family of stabilizer states but any stabilizer state can be transformed into a graph state using local Clifford unitaries in a straightforward way [21], as a consequence it is sufficient to consider graph states when dealing with entanglement equivalence of stabilizer states. Notice that understanding the entanglement structure of graph states and how it can be characterised in a graphical way is the most fascinating fundamental question of the graph state formalism [21, 13, 14, 18, 23, 36, 37, 11].

An interesting combinatorial property of entanglement has been proved in [21]: LU-equivalent graph states have the same cut-rank function¹. However, there exist pairs of graph states that have the same cut-rank function but which are not LU-equivalent: a counterexample involves two isomorphic Petersen graphs [16, 21].

A fundamental property of the graph state formalism is that a *local complementation* – a graph transformation that consists in complementing the neighbourhood of a given vertex – preserves entanglement: if a graph can be transformed into another by means of local complementations then the two corresponding graph states are LU-equivalent. More precisely the corresponding graph states are LC-equivalent (local Clifford). Van den Nest [13] proved that two graph states are LC-equivalent if and only if the corresponding graphs are related by a sequence of local complementations. LC-equivalence and LU-equivalence of graph states were conjectured to coincide [24], as they coincide for several families of graph states [14, 38]. However, a counterexample of order 27 has been discovered using computer assisted methods [23]. Since then, the existence of a graphical characterisation of LU-equivalence has remained an open question. Furthermore, families of graph states for which LC-equivalence and LU-equivalence coincide have been a subject of interest [33, 37].

Contributions. We introduce the *r-local complementation*, a graphical transformation parametrised by a positive integer r , which coincides with the usual local complementation when $r = 1$. We show that an r -local complementation can be performed on the corresponding graph states by means of local unitaries in $LC_r = \langle H, Z(\pi/2^r) \rangle$, the group generated by the local Clifford group augmented with the $Z(\pi/2^r)$ gate. We additionally show that, conversely, for any fixed r , the LC_r -equivalence of graph states is captured by r -local complementations: if two graph states are LC_r -equivalent, then there is a sequence made of (usual) local complementations and at most one r -local complementation, that transforms one graph into the other. In particular, the known examples of graph states that are LU-equivalent but not

¹ The cut-rank function over a graph $G = (V, E)$ is the function that maps a set $A \subseteq V$ to the rank (in \mathbb{F}_2) of the cut-matrix $\Gamma_A = ((\Gamma_A)_{ab} : a \in A, b \in V \setminus A)$ where $\Gamma_{ab} = 1$ if and only if $(a, b) \in E$.

LC-equivalent [23, 36], are actually LC₂-equivalent, thus they are captured graphically by the 2-local complementation. This leads to the natural question of the existence of pairs of graph states that are LC₃-equivalent but not LC₂-equivalent, and so on. To answer this question, we show that the r -local complementations form a strict hierarchy: for any positive r there exist pairs of LC _{$r+1$} -equivalent graph states that are not LC _{r} -equivalent. Finally, we show that any LU-equivalent graph states are LC _{r} -equivalent for some r , even if there exist some local unitaries, like $R_Z(\pi/3)$, that are not contained in LC _{r} , for any r . In other words, two graph states are LU-equivalent if and only if the corresponding graphs can be transformed into each other by a sequence of generalised local complementations, leading to a full graphical description of LU-equivalence for graph states. As an application, we prove a conjecture on a family of graph states for which LU-equivalence and LC-equivalence coincide.

Related work. Other graphical extensions of local complementation have been introduced in particular for weighted hypergraph states [36], a wider family of quantum states. The connection between generalised local complementation and local complementation of weighted hypergraph states is discussed in section 3. In [18, 39], the authors proved that when considering LU-equivalence of stabilizer states it suffices to consider semi-Clifford unitaries². Symmetries of stabilizers and graph states have been explored in [15], where a characterisation of local operations that have graph states as fixed points, is provided; the characterisation of the LU-equivalence of graph states was however left as an open question.

Structure of the paper. In Section 2, we recall some definitions and notations and provide a handful of tools to manipulate graph states. In Section 3, we define the generalisation of the local complementation and we show that r -local complementations can be implemented on the corresponding graph states by means of local unitaries in LC _{r} . Section 4 is dedicated to the converse result: the graphs corresponding to any pair of LC _{r} -equivalent graph states can be transformed into each other by means of r -local complementations. To do so, we show that any graph can be put in a standard form by means of (usual) local complementations. This standard form is based on the so-called minimal local sets, which are subsets of vertices known to be invariant under LU-equivalence. As any graph can be covered by minimal local sets, we associate to any vertex of a graph a type on which is based the standard form. We then prove that if two graph states in standard form are LC _{r} -equivalent, there exists an r -local complementation transforming one graph into the other. We also prove that if two graph states on n qubits are LU-equivalent they are LC _{$n/2$} -equivalent. Finally, in Section 5, we introduce a family of variants of Kneser graphs, and show, using our graphical characterisation of LC _{r} -equivalence, that for any $r > 1$ there exists pairs of graph states that are LC _{r} -equivalent but not LC _{$r-1$} -equivalent.

2 Preliminaries

Let us first give some notations and basic definitions. A graph³ G is a pair (V, E) , where V is a finite set of vertices, and E is a set of unordered pairs of distinct vertices called edges. We assume the set of vertices to be totally ordered and use the notation $V = \{u_1, \dots, u_n\}$ s.t. $u_i \prec u_j$ iff $i < j$. The number $n = |V|$ of vertices is called the order of the graph and

² A unitary U is semi-Clifford if there exists a Pauli operator $P \in X, Y, Z$ such that UPU^\dagger is also a Pauli operator.

³ We only consider simple (no selfloop) undirected graphs.

$|G| := |E|$ its size. We use the notation $u \sim_G v$ when the vertices u and v are connected in G , i.e. $(u, v) \in E$. Given a vertex u , $N_G(u) = \{v \in V \mid (u, v) \in E\}$ is the neighbourhood of u . The odd and the common neighbourhoods are two natural generalisations of the notion of neighbourhoods to sets of vertices: for any $D \subseteq V$, $Odd_G(D) = \Delta_{u \in D} N_G(u) = \{v \in V \mid |N_G(v) \cap D| = 1 \pmod{2}\}$ is the odd neighbourhood of D , where Δ denotes the symmetric difference on vertices. Informally, $Odd_G(D)$ is the set of vertices that are the neighbours of an odd number of vertices in D . The common neighbourhood of D is denoted $\Lambda_G^D = \bigcap_{u \in D} N_G(u) = \{v \in V \mid \forall u \in D, v \in N_G(u)\}$. A local complementation with respect to a given vertex u consists in complementing the subgraph induced by the neighbourhood of u , leading to the graph $G \star u = G \Delta K_{N_G(u)}$ where Δ denotes the symmetric difference on edges and K_A is the complete graph on the vertices of A . Local complementation is an involution, i.e. $G \star u \star u = G$. A pivoting with respect to two connected vertices u and v is the operation that maps G to $G \wedge uv := G \star u \star v \star u$. Notice that pivoting is symmetric ($G \wedge uv = G \wedge vu$) and is an involution ($G \wedge uv \wedge uv = G$). With a slight abuse of notation we identify multisets of vertices with their multiplicity function $V \rightarrow \mathbb{N}$.⁴ A (multi)set S of vertices is independent if there is no two vertices of S that are connected.

Graph states form a standard family of quantum states that can be represented using graphs (Ref. [20] is an excellent introduction to graph states). Given a graph G of order n , the corresponding **graph state** $|G\rangle$ is the n -qubit state:

$$|G\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{|G[x]|} |x\rangle$$

where $G[x]$ denotes the subgraph of G induced by $\{u_i \mid x_i = 1\} \subseteq V$.

The i^{th} qubit of $|G\rangle$ can be associated with the vertex u_i of G , so, with a slight abuse of notation, we refer to V as the set of qubits of $|G\rangle$. Given a 1-qubit operation R , like $X : |a\rangle \mapsto |1-a\rangle$ or $Z : |a\rangle \mapsto (-1)^a |a\rangle$, and a subset $D \subseteq V$ of qubits, $R_D := \bigotimes_{u \in D} R_u$ is the local operation which consists in applying R on each qubit of D .

The graph state $|G\rangle$ is the unique quantum state (up to a global phase) that, for every vertex $u \in V$, is a fixed point of the Pauli operator $X_u Z_{N_G(u)}$. More generally, $|G\rangle$ is an eigenvector of any Pauli operator $X_D Z_{Odd(D)}$ with $D \subseteq V$:

► Proposition 1. *Let $G = (V, E)$ be a graph. The corresponding graph state $|G\rangle$ is a fixed point of $\mathcal{P}_D^G = (-1)^{|G[D]|} X_D Z_{Odd(D)}$ for any $D \subseteq V$, where $|G[D]|$ is the number of edges of the subgraph of G induced by D .*

The proof of Proposition 1 can be found in the full version of this paper [9]. The fact that a graph state can be defined as the common eigenvector of Pauli operators witnesses that graph states form a subfamily of stabilizer states⁵. Conversely, any stabilizer state can be turned into a graph state by means of local Clifford unitaries⁶ in a straightforward way (see for instance [13]). Thus we will focus in the rest of this paper on graph states.

The action of measuring, in the standard basis, a qubit of a graph state leads, roughly speaking, to the graph state where the corresponding vertex has been removed. A diagonal measurement can also be used to remove a vertex when this vertex is isolated:

⁴ Hence we also identify sets of vertices with their indicator functions $V \rightarrow \{0, 1\}$.

⁵ A n -qubit state is a stabilizer state if it is the fixpoint of n independent commuting Pauli operators (or equivalently 2^n distinct commuting Pauli operators).

⁶ Local Clifford unitaries are tensor products of single-qubit Clifford gates, which map the Pauli group to itself under conjugation. Formally, a single-qubit gate U is Clifford if for any $P \in \mathcal{P} := \{\pm 1, \pm i\} \times \{I, X, Y, Z\}$, $UPU^\dagger \in \mathcal{P}$.

► **Proposition 2.** For any graph G and any vertex u , $\langle 0|_u |G\rangle = \frac{1}{\sqrt{2}}|G \setminus u\rangle$. Moreover, if u is an isolated vertex (i.e. $N_G(u) = \emptyset$), then $\langle +|_u |G\rangle = |G \setminus u\rangle$ where $\langle +| = \frac{\langle 0| + \langle 1|}{\sqrt{2}}$.

► **Remark 3.** A standard basis measurement consists in applying either $\langle 0|$ or $\langle 1|$ which correspond the classical outcomes 0 and 1 respectively. Whereas the action in the 0 case is directly described in Proposition 2, the action in the 1 case can be recovered thanks to Proposition 1: $\langle 1|_u |G\rangle = \langle 0|_u X_u |G\rangle = \langle 0|_u Z_{N_G(u)} |G\rangle = \frac{1}{\sqrt{2}}Z_{N_G(u)} |G \setminus u\rangle$. Thus it also corresponds to a vertex deletion up to some Z corrections on the neighbourhood of the measured qubit.

We are interested in the action of 1-qubit unitaries on graph states, in particular Hadamard $H : |a\rangle \mapsto \frac{|0\rangle + (-1)^a |1\rangle}{\sqrt{2}}$, and Z - and X -rotations respectively defined as follows:

$$Z(\alpha) := e^{i\frac{\alpha}{2}} \left(\cos\left(\frac{\alpha}{2}\right) I - i \sin\left(\frac{\alpha}{2}\right) Z \right) \quad X(\alpha) := HZ(\alpha)H = e^{i\frac{\alpha}{2}} \left(\cos\left(\frac{\alpha}{2}\right) I - i \sin\left(\frac{\alpha}{2}\right) X \right)$$

In particular, local complementations can be implemented by $\pi/2$ X - and Z -rotations:

► **Proposition 4** ([13]). For any graph $G = (V, E)$ and any $u \in V$, $|G \star u\rangle = L_u^G |G\rangle$ where $L_u^G := X_u(\frac{\pi}{2}) \otimes_{v \in N_G(u)} Z_v(-\frac{\pi}{2})$.

Similarly, pivoting can be implemented by means of Hadamard transformations (up to Pauli transformations):

► **Proposition 5** ([27, 12]). For any graph $G = (V, E)$ and any $(u, v) \in E$, $|G \wedge uv\rangle = H_u H_v Z_{\Lambda_G^{\{u,v\}}} |G\rangle$

Beyond $\frac{\pi}{2}$ rotations, notice that any local unitary can be implemented using Hadamard and Z -rotations with arbitrary angles. We consider for any $r \in \mathbb{N}$ the set LC_r of local unitaries generated by H and $Z(\frac{\pi}{2r})$. In particular LC_1 is the set of local Clifford operators, moreover, for any r , $LC_r \subseteq LC_{r+1}$. Notice that there exist local unitaries, e.g. $Z(\frac{\pi}{3})$, that are not contained in LC_r for any r . We say that two states are LU-equivalent, denoted $|\psi\rangle =_{LU} |\psi'\rangle$ when there exists a local unitary transformation U such that $|\psi\rangle = U|\psi'\rangle$. Similarly two states are LC_r -equivalent, denoted $|\psi\rangle =_{LC_r} |\psi'\rangle$ when there exists a local unitary $U \in LC_r$ s.t. $|\psi\rangle = U|\psi'\rangle$.

It is well-known that two graph states $|G\rangle$ and $|G'\rangle$ are LC_1 -equivalent if and only if there exists a sequence of local complementations that transforms G into G' [13]. We slightly refine this result showing that there exists a reasonably small sequence of local complementation transforming G into G' , that corresponds to a particular Clifford operator:

► **Proposition 6.** If $|G_2\rangle = C|G_1\rangle$ where C is a local Clifford operator up to a global phase, then there exists a sequence of (possibly repeating) vertices a_1, \dots, a_m such that $G_2 = G_1 \star a_1 \star a_2 \star \dots \star a_m$ with $m \leq \lfloor 3n/2 \rfloor$, and the local Clifford operator that implements these local complementations is C up to a Pauli operator.

The proof of Proposition 6 can be found in the full version of this paper [9]

► **Remark 7.** The induced Clifford operator is of the form $\mathcal{P}_D^G C$ for some subset D of vertices. As $X_u Z_{N_G(u)} = (L_u^G)^2$, there exists a sequence at most $\lfloor 7n/2 \rfloor$ local complementations that transforms G_1 into G_2 and induces exactly the local Clifford operator C .

Whereas it has been originally conjectured that if two graphs states are LU-equivalent then there exists a sequence of local complementations transforming one in another [24], a couple of counter examples of pairs of LC_2 - but not LC_1 -equivalent graph states have been pointed out [23, 36]. To further understand the action of local unitaries on graph states, we thus introduce in the next section a generalisation of local complementation to graphically capture the equivalence of graph states that are LU- but not LC_1 -equivalent.



■ **Figure 1** Example of non LU-equivalent graph states. With the notations of Section 5, the graph on the left is a $C_{4,3}$ graph, and the one on the right is a $C'_{4,3}$ graph. Notice that applying a local complementation on the upper part of the bipartition ($S = \{a, b, c, d\}$) leaves the graphs invariant as each pair of vertices on the other part (e.g. (e, f)), has two common neighbours in S . However a 2-local complementation over S is not valid in both graphs as S is not 2-incident. See Section 4.1 for a proof that these graph states are not LU-equivalent.

3 Generalising Local Complementation

3.1 Local complementation over independent sets

As a first step towards a generalisation of local complementation, we introduce a natural shortcut $G \star S := G \star a_1 \dots \star a_k$ to denote the graph obtained after a series of local complementations when $S = \{a_1, \dots, a_k\}$ is an independent set in G . The independence condition is important as local complementations do not commute in general when applied on connected vertices.

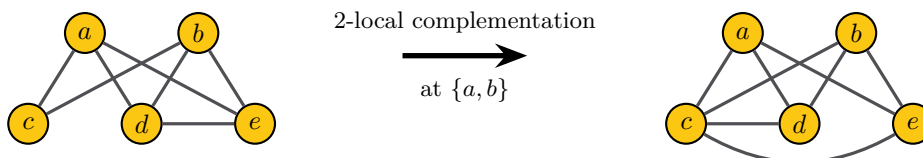
Notice that the action of a local complementation over S can be described directly as toggling edges which have an odd number of common neighbours in S :

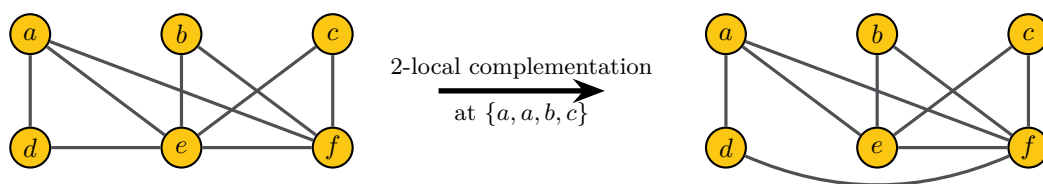
$$u \sim_{G \star S} v \Leftrightarrow (u \sim_G v \oplus |S \cap N_G(u) \cap N_G(v)| = 1 \pmod 2) \tag{1}$$

3.2 Towards a 2-local complementation

We introduce 2-local complementations as a refinement of *idempotent* local complementations, i.e. when $G \star S = G$. According to Equation (1), an idempotent local complementation occurs when each pair (u, v) of vertices has an even number of common neighbours in S , one can then consider a new graph transformation which consists in toggling an edge (u, v) when its number of common neighbours in S is equal to 2 modulo 4. However, such an action may not be implementable using local operations in the graph state formalism (see Figure 1). To guarantee an implementation by means of local unitary transformations on the corresponding graph states, we add the condition, called 2-incident, that any set of at most three vertices has an even number of common neighbours in S .

For instance in the following example, a 2-local complementation over $\{a, b\}$ performs the following transformation:





■ **Figure 2** Illustration on a 2-local complementation on the multiset $S = \{a, a, b, c\}$. S is 2-incident: indeed $S \bullet \Lambda_G^{\{d,e,f\}} = 2$, which is a multiple of $2^{2-1-0} = 2$. Similarly, $S \bullet \Lambda_G^{\{d,e\}} = S \bullet \Lambda_G^{\{d,f\}} = 2$ and $S \bullet \Lambda_G^{\{e,f\}} = 4$. Edges de and df are toggled as $S \bullet \Lambda_G^{\{d,e\}} = S \bullet \Lambda_G^{\{d,f\}} = 2 \pmod 4$, but not edge ef as $S \bullet \Lambda_G^{\{e,f\}} = 0 \pmod 4$.

In the LHS graph G , $S = \{a, b\}$ is an independent set. Moreover, a 1-local complementation over S is idempotent: $G \star S = G$, as a and b are twins. S is also 2-incident as any pairs and triplets of vertices have an even number of neighbours in S . Thus, a 2-local complementation over S is valid and consists in toggling the edges (c, d) , (c, e) , and (d, e) as each of them has a number of common neighbours in S equal to $2 \pmod 4$.

We also consider the case where S is actually a multiset, like in Figure 2 where a 2-local complementation over the multiset $\{a, a, b, c\}$ is performed. When S is a multiset, the number of common neighbours in S should be counted with their multiplicity.

Notice that when a 2-local complementation is invariant one can similarly refine the 2-local complementation into a 3-local complementation, leading to the general definition of generalised local complementation provided in the next section.

3.3 Defining generalised local complementation

We introduce a generalisation of local complementation, that we call r -local complementation for any positive integer r . This generalised local complementation denoted $G \star^r S$ is parametrised by a (multi)set S , that has to be independent and also r -incident, which is the following counting condition on the number of common neighbours in S of any set that does not intersect $\text{supp}(S)$, the support of S :

► **Definition 8** (r -Incidence). *Given a graph G , a multiset S of vertices is r -incident, if for any $k \in [0, r)$, and any $K \subseteq V \setminus \text{supp}(S)$ of size $k + 2$, $S \bullet \Lambda_G^K$ is a multiple of $2^{r-k-\delta(k)}$, where δ is the Kronecker delta⁷ and $S \bullet \Lambda_G^K$ is the number of vertices of S , counted with their multiplicity, that are neighbours to all vertices of K .⁸*

► **Definition 9** (r -Local Complementation). *Given a graph G and an r -incident independent multiset S , let $G \star^r S$ be the graph defined as*

$$u \sim_{G \star^r S} v \Leftrightarrow (u \sim_G v \oplus S \bullet \Lambda_G^{u,v} = 2^{r-1} \pmod{2^r})$$

A detailed example of a 2-local complementation is given in Figure 2.

We will also use r -local complementation parametrised with a set (rather than a multiset), in this case $S \bullet \Lambda_G^K = |S \cap \Lambda_G^K|$, and $S \bullet \Lambda_G^{u,v} = |S \cap N_G(u) \cap N_G(v)|$. Note that we recover the vanilla local complementation when $r = 1$.

⁷ $\delta(x) \in \{0, 1\}$ and $\delta(x) = 1 \Leftrightarrow x = 0$.

⁸ \bullet is the scalar product: $A \bullet B = \sum_{u \in V} A(u).B(u)$, so $S \bullet \Lambda_G^K = \sum_{u \in \Lambda_G^K} S(u)$.

We say that $G \star^r S$ is valid when S is an r -incident independent multiset in G . Moreover, generalising the notion of local equivalence on graphs, we say that two graphs G, G' are r -locally equivalent when there exists a sequence of r -local complementations transforming G into G' . In the case of the usual local complementation, we simply say that the graphs are locally equivalent.

Generalised local complementations satisfy a few basic properties (proven in the full version of this paper [9]), in particular, it is easy to double check that they are self inverse: $(G \star^r S) \star^r S = G$. Moreover, r -local complementations can be related to $(r + 1)$ and $(r - 1)$ -local complementations:

- **Proposition 10.** *If $G \star^r S$ is valid then:*
 - $G \star^{r+1} (2S)$ is valid and induces the same transformation: $G \star^{r+1} (2S) = G \star^r S$, where $2S$ is the multiset obtained from S by doubling the multiplicity of each vertex,
 - $G \star^{r-1} S$ is valid (when $r > 1$) and $G \star^{r-1} S = G$.

It implies in particular that two r -locally equivalent graphs are $(r + 1)$ -locally equivalent. An r -local complementation over S preserves the neighbourhood of the vertices in $\text{supp}(S)$:

- **Proposition 11.** *If $G \star^r S$ is valid, then for any $u \in \text{supp}(S)$, $N_{G \star^r S}(u) = N_G(u)$.*

Notice that r -local complementations can be composed:

- **Proposition 12.** *If $G \star^r S_1$ and $G \star^r S_2$ are valid and the disjoint union⁹ $S_1 \sqcup S_2$ is independent in G , then $G \star^r (S_1 \sqcup S_2) = (G \star^r S_1) \star^r S_2$.*

Finally, it is easy to see that the multiplicity in S can be upperbounded by 2^r : if $G \star^r S$ is valid, then $G \star^r S = G \star^r S'$, where, for any vertex u , $S'(u) = S(u) \bmod 2^r$.

3.4 Local Clifford operators and local complementation

It is well known that local complementations can be implemented on graph states by means of local Clifford operations [21, 20], we extend this result to r -local complementations that can be implemented using $\frac{\pi}{2^r}$ X- and Z-rotations:

- **Proposition 13.** *Given a graph $G = (V, E)$ and a r -incident independent multiset S of vertices,*

$$|G \star^r S\rangle = \bigotimes_{u \in V} X \left(\frac{S(u)\pi}{2^r} \right) \bigotimes_{v \in V} Z \left(-\frac{\pi}{2^r} \sum_{u \in N_G(v)} S(u) \right) |G\rangle$$

The proof of Proposition 13 can be found in the full version of this paper [9].

- **Remark 14.** The graph state formalism has been extended in various ways, notably through the introduction of *hypergraph states* [31] and even *weighted hypergraph states* [36]. Notice that the formalism of weighted hypergraph states provides a way to decompose a generalised local complementation over a multiset S of size k into k elementary transformations on weighted hypergraph states. Indeed, as shown in Proposition 13, a generalised local complementation can be implemented by means of k X-rotations (along with some Z-rotations), moreover the action of each X-rotation on weighted hypergraph states has been described in [36]. However, in such a decomposition, the intermediate weighted hypergraph states are not necessarily graph states.

⁹ for any vertex u , $(S_1 \sqcup S_2)(u) = S_1(u) + S_2(u)$.

In the next section, we prove the converse result: if a local unitary operation in LC_r transforms a graph state $|G\rangle$ into a graph state $|G'\rangle$ then there exists a sequence of r -local complementations transforming G into G' . More so, we show that r -local complementation actually captures completely the LU-equivalence of graph states. To prove these results, we make use of graphical tools, namely the so-called minimal local sets and a standard form for graphs.

4 Graphical characterisation of the action of local unitaries

4.1 Minimal local sets and Types

To characterise the action of local unitaries on graph states, we rely on properties that are invariant under local unitaries. A local set [22, 8] is one of them.

► **Definition 15.** *Given $G = (V, E)$, a local set L is a non-empty subset of V of the form $L = D \cup \text{Odd}_G(D)$ for some $D \subseteq V$ called a generator.*

Local sets of G are precisely the supports of the Pauli operators $\mathcal{P}_D^G = (-1)^{|G[D]|} X_D Z_{\text{Odd}(D)}$, with D a non-empty subset of qubits, that stabilises $|G\rangle$. Local sets are invariant under local complementation - hence their name: if L is a local set in a graph G , so is in $G \star u$, but possibly with a distinct generator [22]. Local sets are the same for graphs that have the same cut-rank function [8], thus graphs corresponding to LU-equivalent graph states have the same local sets.

A *minimal local set* is a local set that is minimal by inclusion. Minimal local sets have either 1 or 3 generators, and in the latter case, the minimal local sets are of even size. This was proved originally in the stabilizer formalism in [14], but an alternative graph-theoretic proof can be found in [8].

► **Proposition 16.** *Given a minimal local set L , only two cases can occur:*

- L has exactly one generator,
- L has exactly three (distinct) generators, of the form D_0 , D_1 , and $D_0 \Delta D_1$. This can only occur when $|L|$ is even.

In the first case, the minimal local set L is said to be of dimension 1; in the second, of dimension 2. The dimension of a minimal local set depends only on the cut-rank function, thus it is invariant by LU-equivalence. Just like (minimal) local sets, the dimension can be a useful tool to prove that two graph states are not LU-equivalent. For example, the two graphs of Figure 1 have the same minimal local sets, but they do not have the same dimension. Indeed, $\{a, b, c, h\}$ is a minimal local set of dimension 2 in the first graph, while it is a minimal local set of dimension 1 in the second one, proving that the two graph states are not LU-equivalent.

Minimal local sets provide crucial informations on local unitaries acting on graph states: it is known that if a local unitary U transforms $|G\rangle$ into $|G'\rangle$ and L is a minimal local set of dimension 2, then for any $v \in L$, U_v must be a Clifford operator [14]. Minimal local sets of dimension 1 are more permissive but also provide some constraints on U :

► **Lemma 17.** *Given two graphs G , G' and a local unitary U such that $|G\rangle = U|G'\rangle$, if L is a 1-dimensional minimal local set with generators respectively D in G and D' in G' , then*

$$\mathcal{P}_D^G U = U \mathcal{P}_{D'}^{G'}$$

27:10 Local Equivalence of Stabilizer States: A Graphical Characterisation

Proof. Following [14, 39], we consider the density matrix ρ_G^L (resp. $\rho_{G'}^L$) obtained by tracing out the qubits outside L . According to [20, 21]: $\rho_G^L = \frac{1}{2}(I + \mathcal{P}_D^G)$ and $\rho_{G'}^L = \frac{1}{2}(I + \mathcal{P}_{D'}^{G'})$. As $\rho_G^L = U\rho_{G'}^L U^\dagger$, we get $I + \mathcal{P}_D^G = I + U\mathcal{P}_{D'}^{G'}U^\dagger$, hence $\mathcal{P}_D^G U = U\mathcal{P}_{D'}^{G'}$. ◀

So, intuitively, every minimal local set L induces some constraints on a local unitary transformation U acting on the corresponding graph state, more precisely on the qubits of U that are in L . It has been recently shown that minimal local sets cover every vertex of a graph [8], which unlocks the use of minimal local sets in characterizing local unitaries acting on graph states, as it guarantees that minimal local sets impose constraints on every qubit of the local unitary:

► **Theorem 18** ([8]). *Each vertex of any graph is contained in at least one minimal local set.*

We abstract away the set of minimal local sets, which can be exponentially many in a graph (see [8]), as a simple labelling of the vertices that we call a *type*.

► **Definition 19.** *Given a graph G , a vertex u is of type*

- X if for any generator D of a minimal local set containing u , $u \in D \setminus \text{Odd}(D)$,
- Y if for any generator D of a minimal local set containing u , $u \in D \cap \text{Odd}(D)$,
- Z if for any generator D of a minimal local set containing u , $u \in \text{Odd}(D) \setminus D$,
- \perp otherwise.

The names X , Y and Z are chosen to match the Pauli operator at vertex u in \mathcal{P}_D^G . Notice that a vertex involved in a minimal local set of dimension 2 is necessarily of type \perp ¹⁰. We define $V_X^G \subseteq V$ (resp. V_Y^G, V_Z^G, V_\perp^G) as the set of vertices of type X (resp. Y, Z, \perp) in G . V_X^G, V_Y^G, V_Z^G and V_\perp^G form a partition of V : indeed, thanks to Theorem 18, every vertex has a type.

We show in the following a few properties of the vertex types. First notice that vertices of type Z represent at most half the vertices of a graph:

► **Lemma 20.** *For any graph G of order n , $|V_Z^G| \leq \lfloor n/2 \rfloor$.*

Proof. V_Z^G contains no minimal local set. Indeed, as the generator of a local set is non-empty, at least one vertex of a given minimal local set L is not of type Z . Conversely, any subset of the vertices of size $\lfloor n/2 \rfloor + 1$ contains at least one minimal local set [8]. ◀

Two LU-equivalent graph states share the same vertices of type \perp .

► **Lemma 21.** *Given two LU-equivalent graph states $|G_1\rangle$ and $|G_2\rangle$, $V_\perp^{G_1} = V_\perp^{G_2}$.*

Proof. Let U be a local unitary s.t. $|G_1\rangle = U|G_2\rangle$, and let $v \in V$ a vertex of type \perp in G_1 . If v is in a minimal local set of dimension 2 in G_1 , so is in G_2 as the dimension of minimal local sets is invariant under LU-equivalence. Otherwise, there exist two distinct minimal local sets L, L' , generated respectively by D_1 and by D'_1 in G_1 such that $\mathcal{P}_{D_1}^{G_1}$ and $\mathcal{P}_{D'_1}^{G_1}$ do not commute on qubit v .¹¹ Let D_2 (resp. D'_2) be the generator of L (resp. L') in G_2 . According to Lemma 17, $\mathcal{P}_{D_1}^{G_1} = U\mathcal{P}_{D_2}^{G_2}U^\dagger$ and $\mathcal{P}_{D'_1}^{G_1} = U\mathcal{P}_{D'_2}^{G_2}U^\dagger$, as a consequence, $\mathcal{P}_{D_2}^{G_2}$ and $\mathcal{P}_{D'_2}^{G_2}$ do not commute on qubit v , so v must be of type \perp in G_2 . ◀

¹⁰ If a vertex v has the same type with respect to two distinct generators D, D' of a minimal local set L , then v would not be in the local set generated by $D\Delta D'$ which contradicts the minimality of L .

¹¹ $V = \bigotimes_u V_u$ commutes with $W = \bigotimes_u W_u$ on qubit u_0 if V_{u_0} and W_{u_0} commute.

A vertex having the same type in two LU-equivalent graph states implies strong constraints on the local unitaries that relate the two corresponding graph states.

► **Lemma 22.** *If $|G_1\rangle =_{LU} |G_2\rangle$ i.e. $|G_2\rangle = U|G_1\rangle$, then $U = e^{i\phi} \bigotimes_{u \in V} U_u$ where:*

- U_u is a Clifford operator if u is of type \perp in both G_1 and G_2 ,
- $U_u = X(\theta_u)Z^{b_u}$ if u is of type X in both G_1 and G_2 ,
- $U_u = Z(\theta_u)X^{b_u}$ if u is of type Z in both G_1 and G_2 .

with $b_u \in \{0, 1\}$. Additionally, if $|G_1\rangle$ and $|G_2\rangle$ are LC_r -equivalent, there exists such a unitary U where every angle satisfies $\theta_u = 0 \pmod{\pi/2^r}$.

Proof. The first case, when u is of type \perp , is a variant of the minimal support condition [13] and was proved in [28] for the particular case of minimal local sets of dimension 2. If u is of type X , let L be a minimal local set such that $u \in L$. According to Lemma 17, $U_u X U_u^\dagger = e^{i\phi} X$. $U_u|+\rangle$ and $U_u|-\rangle$ are eigenvectors of $U_u X U_u^\dagger$ of eigenvalues respectively 1 and -1, implying $U_u X U_u^\dagger = \pm X$. If $U_u X U_u^\dagger = X$, then $U_u = e^{i\phi} X(\theta)$. If $U_u X U_u^\dagger = -X$, define $U' := U_u Z$. $U' X U'^\dagger = X$, so $U' = e^{i\phi} X(\theta)$, thus $U_u = e^{i\phi} X(\theta) Z$. The proof is similar when u is of type Z . If $|G_1\rangle$ and $|G_2\rangle$ are LC_r -equivalent, there exists $|G_2\rangle = U|G_1\rangle$ where $U = e^{i\phi} \bigotimes_{u \in V} U_u$, and each U_u is in LC_r . ◀

To fully characterise the unitaries that relate two graph states using Lemma 22, the type of every vertex needs to be the same. This is not the case in general, even for locally equivalent graphs¹². Thus, we introduce a standard form on graphs, such that two graphs in standard form corresponding to LU-equivalent graph states have the same types.

4.2 Standard Form

We define a standard form of graphs up to local complementation. A graph in standard form satisfies the following properties: all vertices are of type X , Z or \perp ; all the neighbours of a vertex of type X are of type Z (so in particular the vertices of type X form an independent set); and any vertex of type X is smaller than its neighbours according to the underlying total order \prec of the vertices. In other words:

► **Definition 23.** *A graph G is in said in standard form if*

- $V_Y^G = \emptyset$,
- $\forall u \in V_X^G$, any neighbour v of u is of type Z and satisfies $u \prec v$.

Note that standard form is not unique in general, in the sense that a class of local equivalence of graphs may contain several graphs in standard form. For example, any graph with only vertices of type \perp is in standard form, along with its entire orbit generated by local complementation. Conversely, each class of local equivalence contains at least a graph in standard form.

► **Proposition 24.** *For any graph G , there exists a locally equivalent G' in standard form.*

The proof of Proposition 24 is an algorithm that puts a graph in standard form by means of local complementation and can be found in the full version of this paper [9]. Standard form implies a similar structure in terms of types assuming LU-equivalence.

¹²A simple example involves the complete graph K_3 on 3 vertices and the line graph L_3 on three vertices. K_3 and L_3 are related by a single local complementation, however every vertex of K_3 is of type Y , while L_3 contains one vertex of type Z and two vertices of type X .

27:12 Local Equivalence of Stabilizer States: A Graphical Characterisation

► **Proposition 25.** *If G_1 and G_2 are both in standard form and $|G_1\rangle =_{LU} |G_2\rangle$, each vertex has the same type in G_1 and G_2 , and any vertex u of type X satisfies $N_{G_1}(u) = N_{G_2}(u)$.*

The proof of Proposition 25 can be found in the full version of this paper [9].

4.3 Graphical characterisation of local equivalence

Thanks to the standard form, one can accommodate the types of two LU-equivalent graph states, to simplify the local unitaries mapping one to the other:

► **Lemma 26.** *If G_1 and G_2 are both in standard form and $|G_1\rangle =_{LU} |G_2\rangle$, there exists G'_1 locally equivalent to G_1 in standard form such that $|G_2\rangle = \bigotimes_{u \in V_X^{G_1}} X(\alpha_u) \bigotimes_{v \in V_Z^{G_1}} Z(\beta_v) |G'_1\rangle$.*

The proof of Lemma 26 can be found in the full version of this paper [9]. Note that if $|G_1\rangle$ and $|G_2\rangle$ are LC_r -equivalent, we can choose the angles such that $\alpha_u, \beta_v = 0 \pmod{\pi/2^r}$. Additionally, G_1 and G'_1 are related by local complementation only on vertices of type \perp . They are strong constraints relating the angles of the X- and Z-rotations acting on different qubits:

- **Lemma 27.** *Given G_1, G_2 in standard form, if $|G_2\rangle = \bigotimes_{u \in V_X^{G_1}} X(\alpha_u) \bigotimes_{v \in V_Z^{G_1}} Z(\beta_v) |G_1\rangle$,*
- $\forall v \in V_Z^{G_1}, \beta_v = -\sum_{u \in N_{G_1}(v) \cap V_X^{G_1}} \alpha_u \pmod{2\pi}$,
- $\forall k \in \mathbb{N}, \forall K \subseteq V_Z^{G_1}$ of size $k+2$, $\sum_{u \in \Lambda_{G_1}^K \cap V_X^{G_1}} \alpha_u = 0 \pmod{\frac{\pi}{2^{k+\delta(k)}}}$.

The proof of Lemma 27 can be found in the full version of this paper [9]. The constraints coincide with r -incidence, so, when angles are multiples of $\pi/2^r$, this unitary transformation implements a r -local complementation.

► **Lemma 28.** *Given G_1, G_2 in standard form, if $|G_2\rangle = \bigotimes_{u \in V_X^{G_1}} X(\alpha_u) \bigotimes_{v \in V_Z^{G_1}} Z(\beta_v) |G_1\rangle$ and $\forall u \in V_X^{G_1}, \alpha_u = 0 \pmod{\pi/2^r}$, then $\bigotimes_{u \in V_X^{G_1}} X(\alpha_u) \bigotimes_{v \in V_Z^{G_1}} Z(\beta_v)$ implements an r -local complementation over the vertices of $V_X^{G_1}$.*

Proof. Let us construct an r -incident independent multiset S such that $G_2 = G_1 \star^r S$. We define S on the vertices of $V_X^{G_1}$ such that $\forall u \in V_X^{G_1}, \alpha_u = \frac{S(u)\pi}{2^r} \pmod{2\pi}$ with $S(u) \in [1, 2^r)$. Note that $\sum_{u \in \Lambda_{G_1}^K \cap V_X^{G_1}} \alpha_u = \frac{\pi}{2^r} S \bullet \Lambda_G^K$. Hence, by Lemma 27, For any $k \in [0, r)$, and any $K \subseteq V \setminus S$ of size $k+2$, $S \bullet \Lambda_G^K$ is a multiple of $2^{r-k-\delta(k)}$ meaning that S is r -incident. Also, by Lemma 27, $\beta_v = -\frac{\pi}{2^r} \sum_{u \in N_{G_1}(v)} S(u) \pmod{2\pi}$. Thus, $\bigotimes_{u \in V_X^{G_1}} X(\alpha_u) \bigotimes_{v \in V_Z^{G_1}} Z(\beta_v)$ implements an r -local complementation on S . ◀

We can now easily relate LC_r -equivalence to r -local complementations for graphs in standard form:

► **Lemma 29.** *If G_1 and G_2 are both in standard form and $|G_1\rangle =_{LC_r} |G_2\rangle$, then G_1 and G_2 are related by a sequence of local complementations on the vertices of type \perp along with a single r -local complementation over the vertices of type X .*

► **Remark 30.** The sequence of local complementations commutes with the r -local complementation, as vertices of type X and vertices of type \perp do not share edges by definition of the standard form.

Proof. By Lemma 26, there exists G'_1 locally equivalent to G_1 in standard form such that $|G_2\rangle = \bigotimes_{u \in V_X} X(\alpha_u) \bigotimes_{v \in V_Z} Z(\beta_v) |G'_1\rangle$ where V_X (resp. V_Z) denotes the set of vertices of type X (resp. Z) in G'_1 and G_2 , and $\forall u \in V$ of type X or Z, $\alpha_u, \beta_u = 0 \pmod{\pi/2^r}$. By Lemma 28, $\bigotimes_{u \in V_X} X(\alpha_u) \bigotimes_{v \in V_Z} Z(\beta_v)$ implements an r -local complementation over the vertices of type X. ◀

Notice in particular that when there is no vertex of type \perp , a single r -local complementation is required.

► **Corollary 31.** *If two \perp -free r -locally equivalent graphs G_1 and G_2 are both in standard form, they are related by a single r -local complementation on the vertices of type X.*

We are ready to prove that r -local equivalence coincides with LC_r -equivalence.

► **Theorem 32.** *The following properties are equivalent:*

1. $|G_1\rangle$ and $|G_2\rangle$ are LC_r -equivalent.
2. G_1 and G_2 are r -locally equivalent.
3. G_1 and G_2 are related by a sequence of local complementations along with a single r -local complementation.

Proof. We proceed by cyclic proof. (2 \Rightarrow 1): Follows from Proposition 13. (3 \Rightarrow 2): A local complementation is, in particular, an r -local complementation. (1 \Rightarrow 3): Follows from Proposition 24 along with Lemma 29. ◀

More than just LC_r -equivalence, r -local complementations can actually characterise the LU-equivalence of graph states.

► **Theorem 33.** *If $|G_1\rangle$ and $|G_2\rangle$ are LU-equivalent then G_1 and G_2 are $(\lfloor n/2 \rfloor - 1)$ -locally equivalent, where n is the order of the graphs.*

The proof of Theorem 33 can be found in the full version of this paper [9]. Remarkably, this result implies that for graph states, LU-equivalence reduces to equivalence up to operators in $\bigcup_{r \in \mathbb{N}} LC_r$, and even – as it has been hinted in [18] – to operators of the form $C_1 \bigotimes_{u \in V} Z(\alpha_u) C_2$ where C_1, C_2 are local Clifford operators and the angles are multiples of $\pi/2^r$ for some integer r . Notice that these local operators are actually those of the well-known Clifford hierarchy (see for example [1]).

4.4 Graph states whose class of LC and LU-equivalence coincide

Since the LU-LC conjecture was disproven in [38], classes of graph states whose class of LC_1 - and LU-equivalence coincide has been a subject of interest. We say that $LU \Leftrightarrow LC$ holds for a graph G if $|G\rangle =_{LU} |G'\rangle \Leftrightarrow |G\rangle =_{LC_1} |G'\rangle$ for any graph G' . It is known that $LU \Leftrightarrow LC$ holds for a graph G if either one of the following is true: **(1)** G is of order at most 8 [20, 6]; **(2)** G is a complete graph [14]; **(3)** every vertex of G is of type \perp (this is referred as the *minimal support condition*) [14]; **(4)** the graph obtained by removing all leaves (i.e. vertices of degree 1) vertices of G has only vertices of type \perp [38]; **(5)** G has no cycle of length 3 or 4 [38]; **(6)** the stabilizer of $|G\rangle$ has rank less than 6 [23]. A review of these necessary conditions, especially for **(6)**, can be found in [37]. Our results imply a new criterion for $LU \Leftrightarrow LC$ based on the standard form introduced in the last section. This criterion is actually a sufficient and necessary condition, meaning it can be used to prove both that $LU \Leftrightarrow LC$ holds for some graph, or the converse.

► **Proposition 34.** *Given a graph G , the following are equivalent:*

- $LU \Leftrightarrow LC$ holds for G .
- For **some** graph locally equivalent to G that is in standard form, any r -local complementation over the vertices of type X can be implemented by local complementations.
- For **any** graph locally equivalent to G that is in standard form, any r -local complementation over the vertices of type X can be implemented by local complementations.

The proof of Proposition 34 can be found in the full version of this paper [9]. Checking that r -local complementations can be implemented by local complementations is not easy in general, nonetheless it is convenient in many cases, for example when there is few vertices of type X or that they have low degree. Namely, this criterion is stronger than the minimal support condition, as graphs with only vertices of \perp have no vertex of type X . Furthermore, it has been left as an open question in [37] whether $LU \Leftrightarrow LC$ holds for some instances of repeater graph states. We use our new criterion to easily prove that this is the case. For this purpose, we prove that $LU \Leftrightarrow LC$ holds for a broader class of graph.

► **Proposition 35.** *$LU \Leftrightarrow LC$ holds for graphs where each vertex is either a leaf i.e. a vertex of degree 1, or is connected to a leaf.*

Proof. It is easy to prove (see the full version of this paper [9]) that such a graph G is in standard form (for some particular ordering of the vertices) and that the vertices of type X are exactly the leaves. Thus, any r -local complementation over the vertices of type X has no effect on G . According to Proposition 34, this implies that $LU \Leftrightarrow LC$ holds for G . ◀

Such graphs include some instances of *repeater graph states* [2]. A *complete-graph-based repeater graph state* is a graph state whose corresponding graph of order $2n$ is composed of a complete graph of order n , along with n leaves appended to each vertex. A *biclique-graph-based repeater graph state* is a graph state whose corresponding graph of order $4n$ is composed of a symmetric biclique (i.e. a symmetric bipartite complete graph) graph of order $2n$, along with $2n$ leaves appended to each vertex. Complete-graph-based repeater graph states are the all-photonic repeaters introduced in [3]. Biclique-graph-based repeater graph states are a variant introduced in [32] which is more efficient in terms of number of edges. It was left as an open question in [37] whether $LU \Leftrightarrow LC$ holds for complete-graph-based or biclique-graph-based repeater graph states (although it was proved for some variants). These graph states satisfy the condition of Proposition 35, hence we can answer this question by the positive:

► **Corollary 36.** *$LU \Leftrightarrow LC$ holds for complete-graph-based repeater graph states and biclique-graph-based repeater graph states.*

5 A hierarchy of generalised local equivalences

Two r -locally equivalent graphs are also $(r+1)$ -locally equivalent. This can be seen graphically as a direct consequence of Proposition 10, or using graph states through Theorem 32, as two LC_r -equivalent states are obviously LC_{r+1} -equivalent. The known counter examples to the LU - LC conjecture introduced in [23, 36] are actually LC_2 -equivalent graph states that are not LC_1 -equivalent, thus the corresponding graphs are 2-locally equivalent but not 1-locally equivalent. There was however no known examples of graph states that are LC_3 -equivalent but not LC_2 -equivalent, and more generally no known examples of graph states that are LC_r -equivalent but not LC_{r-1} -equivalent for $r > 2$. We introduce such examples in this section, by showing that for any r there exist pairs of graphs that are $(r+1)$ -locally equivalent

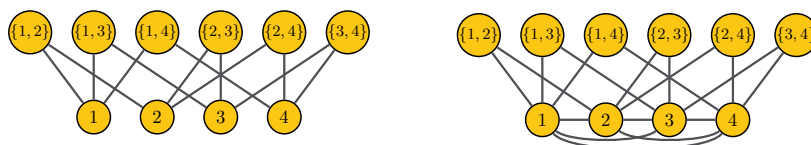
but not r -locally equivalent leading to an infinite hierarchy of generalised local equivalences. Our proof is constructive: for any r we exhibit pairs of graphs that are $(r + 1)$ -locally equivalent but not r -locally equivalent.

We introduce a family of bipartite graphs $C_{t,k}$ parametrized by two integers t and k . This family is a variant of the (bipartite) Kneser graphs and uniform subset graphs. The graph $C_{t,k}$ is bipartite: the first independent set of vertices corresponds to the integers from 1 to t , and the second independent set is composed of all the subsets of $[1, t]$ of size k . There is an edge between an integer $u \in [1, t]$ and a subset $A \subseteq [1, t]$ of size k if and only if $u \in A$:

► **Definition 37.** For any $k \geq 1$ and $t \geq k$ two integers, let $C_{t,k} = (V, E)$ be a bipartite graph defined as: $V = [1, t] \cup \binom{[1,t]}{k}$ and $E = \{(u, A) \in [1, t] \times \binom{[1,t]}{k} \mid u \in A\}$.

We introduce a second family of graphs $C'_{t,k}$, defined similarly to $C_{t,k}$, the independent set $[1, t]$ being replaced by a clique:

► **Definition 38.** For any $k \geq 1$ and $t \geq k$ two integers, let $C'_{t,k} = (V, E)$ be a graph defined as: $V = [1, t] \cup \binom{[1,t]}{k}$ and $E = \{(u, A) \in [1, t] \times \binom{[1,t]}{k} \mid u \in A\} \cup \{(u, v) \in [1, t] \times [1, t] \mid u \neq v\}$.



■ **Figure 3** (Left) The graph $C_{4,2}$. (Right) The graph $C'_{4,2}$.

Examples of such graphs with parameters $t = 4$ and $k = 2$ are given in Figure 3. For any odd $k \geq 3$, $t \geq k + 2$, $C_{t,k}$ and $C'_{t,k}$ are in standard form for any ordering such that, for any $u \in \binom{[1,t]}{k}$ and for any $v \in [1, t]$, $u \prec v$. More precisely the vertices in $\binom{[1,t]}{k}$ are of type X and the vertices in $[1, t]$ are of type Z. The proof can be found in the full version of this paper [9].

The following proposition gives a sufficient condition on t and k for the r -local equivalence of $C_{t,k}$ and $C'_{t,k}$.

► **Proposition 39.** For any $t \geq k \geq 1$, $C_{t,k}$ and $C'_{t,k}$ are r -locally equivalent if

$$\binom{t-2}{k-2} = 2^{r-1} \pmod{2^r} \quad \text{and} \quad \forall i \in [1, r-1], \quad \binom{t-i-2}{k-i-2} = 0 \pmod{2^{r-i}}$$

Proof. $C_{t,k}$ and $C'_{t,k}$ are related by a r -local complementation on the set $\binom{[1,t]}{k}$ (which can be seen as the multiset where each vertex of $\binom{[1,t]}{k}$ appears once). Let $K \subseteq [1, t]$ of size $k' + 2$. $\binom{[1,t]}{k} \bullet \Lambda_G^K = \left| \left\{ x \in \binom{[1,t]}{k} \mid K \subseteq x \right\} \right| = \binom{t-k'-2}{k-k'-2}$ is a multiple of $2^{r-k'-\delta(k')}$ by hypothesis. Thus, $\binom{[1,t]}{k}$ is r -incident. Besides, for any $u, v \in [1, t]$, $\binom{[1,t]}{k} \bullet \Lambda_G^{u,v} = \binom{t-2}{k-2} = 2^{r-1} \pmod{2^r}$ by hypothesis. Thus, $C_{t,k} \star^r \binom{[1,t]}{k} = C'_{t,k}$. ◀

The following proposition provides a sufficient condition on t and k for the non r -local equivalence of $C_{t,k}$ and $C'_{t,k}$.

► **Proposition 40.** For any odd $k \geq 3$, $t \geq k + 2$, $C_{t,k}$ and $C'_{t,k}$ are not r -locally equivalent if $\binom{t}{2}$ is odd and $\binom{k}{2} = 0 \pmod{2^r}$.

Proof. Let us suppose by contradiction that $C_{t,k}$ and $C'_{t,k}$ are r -locally equivalent. By Corollary 31, they are related by a single r -local complementation on the vertices in $\binom{[1,t]}{k}$. Let S be the multiset in $\binom{[1,t]}{k}$ such that $C_{t,k} \star^r S = C'_{t,k}$. For each $u, v \in [1, t]$, $S \bullet \Lambda_{C_{t,k}}^{u,v} = \sum_{x \in \Lambda_{C_{t,k}}^{u,v}} S(x) = 2^{r-1} \pmod{2^r}$. Summing over all pairs $u, v \in [1, t]$, and, as by hypothesis $\binom{t}{2}$ is odd, $\sum_{u,v \in [1,t]} S \bullet \Lambda_{C_{t,k}}^{u,v} = \binom{k}{2} \sum_{x \in \binom{[1,t]}{k}} S(x) = \binom{t}{2} 2^{r-1} \pmod{2^r} = 2^{r-1} \pmod{2^r}$. The first part of the equation is true because $\sum_{u,v \in [1,t]} S \bullet \Lambda_{C_{t,k}}^{u,v} = \sum_{u,v \in [1,t]} \sum_{x \in \Lambda_{C_{t,k}}^{u,v}} S(x) = \sum_{x \in \binom{[1,t]}{k}} \sum_{u,v \in x} S(x)$. This leads to a contradiction as $\binom{k}{2} = 0 \pmod{2^r}$ by hypothesis. ◀

It remains to find, for any r , parameters t and k such that the corresponding graphs are r -locally equivalent but not $(r-1)$ -locally equivalent. Fortunately, such parameters exist.

► **Theorem 41.** *For any $r \geq 2$, $C_{t,k}$ and $C'_{t,k}$ are r -locally equivalent but not $(r-1)$ -locally equivalent when $k = 2^r + 1$ and $t = 2^r + 2^{\lceil \log_2(r) \rceil + 1} - 1$. Thus, $|C_{t,k}\rangle$ and $|C'_{t,k}\rangle$ are LC_r -equivalent but not LC_{r-1} -equivalent.*

The proof of Theorem 41 can be found in the full version of this paper [9].

► **Remark 42.** For the case $r = 2$, we obtain the pair $(C_{7,5}, C'_{7,5})$, which is the 28-vertex counter-example to the LU-LC conjecture from [36]. In particular, this translates to an example of a pair of graph states that are LC_2 -equivalent but not LC_1 -equivalent.

Thus, while Ji et al. proved that there exist pairs of graph states that are LU-equivalent but not LC_1 -equivalent [23], we showed a finer result – the existence of a infinite strict hierarchy of graph states equivalence between LC_1 - and LU-equivalence. There exist LC_2 -equivalent graph states that are not LC_1 -equivalent, LC_3 -equivalent graph states that are not LC_2 -equivalent... On the other end, for any integer r , there exist LU-equivalent graph states that are not LC_r -equivalent.

6 Conclusion

In this paper, we have introduced of a graphical characterisation of LU-equivalence for graph states, and consequently for stabilizer states. To achieve this, we have introduced a generalisation of the local complementation, and leveraged the structures of minimal local sets in graphs. A key outcome of this characterisation is the establishment of a strict hierarchy of equivalences for graph states, which significantly advances our understanding of the gap between the LC- and LU-equivalence. Thanks to this graphical characterisation, we have also proven a conjecture regarding families of graphs states where LC- and LU-equivalence coincide.

This graphical characterisation has additional potential applications, such as the search of minimal examples of graph states that are LU-equivalent but not LC-equivalent. The smallest known examples are of order 27, and it is known there is no counter example of order less than 8. More generally one can wonder whether there are smaller examples of graphs that are LC_r equivalent but not LC_{r-1} as the ones we have introduced are of order exponential en r . In terms of complexity, determining whether two graph states are LC-equivalent can be done in polynomial time. The graphical characterization of LU-equivalence also offers new avenues for exploring the complexity of the LU-equivalence problem, as allows one to study this computational problem from a purely graphical standpoint.

References

- 1 Jonas T. Anderson. On groups in the qubit clifford hierarchy. *Quantum*, 8:1370, 2024. doi:10.22331/q-2024-06-13-1370.
- 2 Koji Azuma, Sophia E. Economou, David Elkouss, Paul Hilaire, Liang Jiang, Hoi-Kwong Lo, and Ilan Tzitrin. Quantum repeaters: From quantum networks to the quantum internet. *Reviews of Modern Physics*, 95(4):045006, 2023. doi:10.1103/RevModPhys.95.045006.
- 3 Koji Azuma, Kiyoshi Tamaki, and Hoi-Kwong Lo. All-photon quantum repeaters. *Nature communications*, 6(1):1–7, 2015. doi:10.1038/ncomms7787.
- 4 Sergey Bravyi, Yash Sharma, Mario Szegedy, and Ronald de Wolf. Generating k epr-pairs from an n -party resource state. *Quantum*, 8:1348, 2024. doi:10.22331/q-2024-05-14-1348.
- 5 Hans J. Briegel, David E. Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009. doi:10.1038/nphys1157.
- 6 Adán Cabello, Antonio J. López-Tarrida, Pilar Moreno, and José R. Portillo. Entanglement in eight-qubit graph states. *Physics Letters A*, 373(26):2219–2225, 2009. doi:10.1016/j.physleta.2009.04.055.
- 7 Maxime Cautrès, Nathan Claudet, Mehdi Mhalla, Simon Perdrix, Valentin Savin, and Stéphane Thomassé. Vertex-Minor Universal Graphs for Generating Entangled Quantum Subsystems. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:18, 2024. doi:10.4230/LIPIcs.ICALP.2024.36.
- 8 Nathan Claudet and Simon Perdrix. Covering a graph with minimal local sets. In *Proceedings of the 24th workshop on Graph Theory (WG 2024)*, 2024. doi:10.48550/arXiv.2402.10678.
- 9 Nathan Claudet and Simon Perdrix. Local equivalence of stabilizer states: a graphical characterisation. (long version with proofs), 2024. doi:10.48550/arXiv.2409.20183.
- 10 Andrew Cross, Graeme Smith, John A. Smolin, and Bei Zeng. Codeword stabilized quantum codes. In *2008 IEEE International Symposium on Information Theory*, pages 364–368. IEEE, 2008. doi:10.1109/TIT.2008.2008136.
- 11 Axel Dahlberg and Stephanie Wehner. Transforming graph states using single-qubit operations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2123):20170325, 2018. doi:10.1098/rsta.2017.0325.
- 12 Maarten Van den Nest and Bart De Moor. Edge-local equivalence of graphs, 2005. arXiv:math/0510246.
- 13 Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. Graphical description of the action of local clifford transformations on graph states. *Physical Review A*, 69(2), February 2004. doi:10.1103/physreva.69.022316.
- 14 Maarten Van den Nest, Jeroen Dehaene, and Bart De Moor. Local unitary versus local clifford equivalence of stabilizer states. *Physical Review A*, 71(6), June 2005. doi:10.1103/physreva.71.062323.
- 15 Matthias Englbrecht and Barbara Kraus. Symmetries and entanglement of stabilizer states. *Phys. Rev. A*, 101:062302, June 2020. doi:10.1103/PhysRevA.101.062302.
- 16 Dmitrii Germanovich Fon-Der-Flaass. *Local Complementations of Simple and Directed Graphs*, pages 15–34. Springer Netherlands, Dordrecht, 1996. doi:10.1007/978-94-009-1606-7_3.
- 17 Sylvain Gravier, Jérôme Javelle, Mehdi Mhalla, and Simon Perdrix. Quantum secret sharing with graph states. In *Mathematical and Engineering Methods in Computer Science: 8th International Doctoral Workshop, MEMICS 2012, Znojmo, Czech Republic, October 25-28, 2012, Revised Selected Papers 8*, pages 15–31. Springer, 2013. doi:10.1007/978-3-642-36046-6_3.
- 18 David Gross and Maarten Van den Nest. The LU-LC conjecture, diagonal local operations and quadratic forms over $\text{GF}(2)$. *Quantum Inf. Comput.*, 8(3):263–281, 2008. doi:10.26421/QIC8.3-4-3.
- 19 Frederik Hahn, Anna Pappa, and Jens Eisert. Quantum network routing and local complementation. *npj Quantum Information*, 5(1):1–7, 2019. doi:10.1038/s41534-019-0191-6.

- 20 Mayan Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, Maarten Van den Nest, and Hans J. Briegel. Entanglement in graph states and its applications. *Quantum computers, algorithms and chaos*, 162, March 2006. doi:10.3254/978-1-61499-018-5-115.
- 21 Mayan Hein, Jens Eisert, and Hans J. Briegel. Multiparty entanglement in graph states. *Physical Review A*, 69(6), June 2004. doi:10.1103/physreva.69.062311.
- 22 Peter Høyer, Mehdi Mhalla, and Simon Perdrix. Resources Required for Preparing Graph States. In *17th International Symposium on Algorithms and Computation (ISAAC 2006)*, volume 4288 of *Lecture Notes in Computer Science*, pages 638–649, December 2006. doi:10.1007/11940128_64.
- 23 Zhengfeng Ji, Jianxin Chen, Zhaohui Wei, and Mingsheng Ying. The LU-LC conjecture is false, 2007. arXiv:0709.1266.
- 24 Olaf Krueger and Reinhard F. Werner. Some open problems in quantum information theory, 2005. arXiv:quant-ph/0504166.
- 25 Damian Markham and Barry C. Sanders. Graph states for quantum secret sharing. *Physical Review A*, 78(4):042309, 2008. doi:10.1103/PhysRevA.78.042309.
- 26 Clément Meignant, Damian Markham, and Frédéric Grosshans. Distributing graph states over arbitrary quantum networks. *Physical Review A*, 100:052333, November 2019. doi:10.1103/PhysRevA.100.052333.
- 27 Mehdi Mhalla and Simon Perdrix. Graph states, pivot minor, and universality of (X,Z) -measurements. *Int. J. Unconv. Comput.*, (1-2):153–171. URL: <http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-153-171/>.
- 28 Eric M. Rains. Quantum codes of minimum distance two. *IEEE Trans. Inf. Theory*, 45(1):266–271, 1999. doi:10.1109/18.746807.
- 29 Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001. doi:10.1103/PhysRevLett.86.5188.
- 30 Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Physical review A*, 68(2):022312, 2003. doi:10.1103/PhysRevA.68.022312.
- 31 Matteo Rossi, Marcus Huber, Dagmar Bruß, and Chiara Macchiavello. Quantum hypergraph states. *New Journal of Physics*, 15(11):113022, 2013. doi:10.1088/1367-2630/15/11/113022.
- 32 Antonio Russo, Edwin Barnes, and Sophia E. Economou. Photonic graph state generation from quantum dots and color centers for quantum communications. *Physical Review B*, 98(8):085303, 2018. doi:10.1103/PhysRevB.98.085303.
- 33 Pradeep Sarvepalli and Robert Raussendorf. Local equivalence of surface code states. In *Theory of Quantum Computation, Communication, and Cryptography: 5th Conference, TQC 2010, Leeds, UK, April 13-15, 2010, Revised Selected Papers 5*, pages 47–62. Springer, 2011.
- 34 Dirk Schlingemann. Stabilizer codes can be realized as graph codes, 2001. arXiv:quant-ph/0111080.
- 35 Dirk Schlingemann and Reinhard F. Werner. Quantum error-correcting codes associated with graphs. *Physical Review A*, 65(1):012308, 2001. doi:10.1103/PhysRevA.65.012308.
- 36 Nikoloz Tsimakuridze and Otfried Gühne. Graph states and local unitary transformations beyond local clifford operations. *Journal of Physics A: Mathematical and Theoretical*, 50(19):195302, April 2017. doi:10.1088/1751-8121/aa67cd.
- 37 Ilan Tzitrin. Local equivalence of complete bipartite and repeater graph states. *Physical Review A*, 98(3):032305, 2018. doi:10.1103/PhysRevA.98.032305.
- 38 Bei Zeng, Hyeyoun Chung, Andrew W. Cross, and Isaac L. Chuang. Local unitary versus local clifford equivalence of stabilizer and graph states. *Physical Review A*, 75(3), March 2007. doi:10.1103/physreva.75.032325.
- 39 Bei Zeng, Andrew Cross, and Isaac L. Chuang. Transversality versus universality for additive quantum codes. *IEEE Transactions on Information Theory*, 57(9):6272–6284, 2011. doi:10.1109/TIT.2011.2161917.

Can You Link Up With Treewidth?

Radu Curticapean   

University of Regensburg, Germany
IT University of Copenhagen, Denmark

Simon Döring  

Max Planck Institute for Informatics, Saarbrücken, Germany
Saarland University (SIC), Saarbrücken, Germany

Daniel Neuen  

University of Regensburg, Germany
Max Planck Institute for Informatics, Saarbrücken, Germany

Jiaheng Wang   

University of Regensburg, Germany

Abstract

A central result by Marx [ToC '10] constructs k -vertex graphs H of maximum degree 3 such that $n^{o(k/\log k)}$ time algorithms for detecting colorful H -subgraphs would refute the Exponential-Time Hypothesis (ETH). This result is widely used to obtain almost-tight conditional lower bounds for parameterized problems under ETH.

Our first contribution is a new and fully self-contained proof of this result that further simplifies a recent work by Karthik et al. [SOSA 2024]. In our proof, we introduce a novel graph parameter of independent interest, the linkage capacity $\gamma(H)$, and show that detecting colorful H -subgraphs in time $n^{o(\gamma(H))}$ refutes ETH. Then, we use a simple construction of communication networks credited to Beneš to obtain k -vertex graphs of maximum degree 3 and linkage capacity $\Omega(k/\log k)$, avoiding arguments involving expander graphs, which were required in previous papers. We also show that every graph H of treewidth t has linkage capacity $\Omega(t/\log t)$, thus recovering a stronger result shown by Marx [ToC '10] with a simplified proof.

Additionally, we obtain new tight lower bounds on the complexity of subgraph detection for certain types of patterns by analyzing their linkage capacity: We prove that almost all k -vertex graphs of polynomial average degree $\Omega(k^\beta)$ for $\beta > 0$ have linkage capacity $\Theta(k)$, which implies tight lower bounds for finding such patterns H . As an application of these results, we also obtain tight lower bounds for counting small induced subgraphs having a fixed property Φ , improving bounds from, e.g., [Roth et al., FOCS 2020].

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Theory of computation \rightarrow Problems, reductions and completeness; Mathematics of computing \rightarrow Graph theory

Keywords and phrases subgraph isomorphism, constraint satisfaction problems, linkage capacity, exponential-time hypothesis, parameterized complexity, counting complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.28

Related Version *Full Version:* <https://arxiv.org/abs/2410.02606>

Funding The research is funded by the European Union (ERC, CountHom, 101077083). 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.



Acknowledgements The title was found with the help of a popular LLM. We thank Cornelius Brand for pointing out a connection to extension complexity.



© Radu Curticapean, Simon Döring, Daniel Neuen, and Jiaheng Wang;
licensed under Creative Commons License CC-BY 4.0
42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).
Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;
Article No. 28; pp. 28:1–28:24



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



1 Introduction

Over the past two decades, it has been shown that complexity assumptions about *exponential-time* problems imply far-reaching lower bounds for *polynomial-time* [13, 67, 68] and *parameterized* [26, 56] problems. Among the first such results, it was shown that the Exponential-Time Hypothesis (ETH) about the Boolean satisfiability problem SAT implies an $n^{\Omega(k)}$ -time lower bound for the seemingly unrelated parameterized problem CLIQUE of detecting k -cliques in n -vertex graphs [16, 17]. This lower bound solidifies the status of CLIQUE as a canonical hard problem in parameterized complexity.

Ideally, when reducing CLIQUE to some target problem, we would like to transfer the $n^{\Omega(k)}$ -time lower bound under ETH to the target problem. However, reductions from CLIQUE often require k gadgets to encode the vertices of a k -clique and $\Theta(k^2)$ additional gadgets to verify the edges between all pairs of encoded vertices. As each gadget typically increases the parameter by at least a constant amount, instances for CLIQUE are transformed into target instances with a parameter value of $\Theta(k^2)$ (see, e.g., [26, Section 13.6.3]). This in turn means that only $n^{o(\sqrt{\ell})}$ -time algorithms can be ruled out for a target problem with parameter ℓ .

Tighter lower bounds could be obtained if we could reduce from a subgraph problem similar to CLIQUE, but involving k -vertex patterns H with only $O(k)$ rather than $\Theta(k^2)$ edges. More specifically, for a fixed graph H , let COLSUB(H) be the problem of detecting H -subgraph copies in graphs G with vertex-colors from $V(H)$ such that every $v \in V(H)$ is mapped to a vertex of color v in G . (This problem can equivalently be interpreted as a *constraint satisfaction problem* with variables x_v for $v \in V(H)$ and arity-2 relations R_e for $e \in E(H)$. The domain of x_v is the set of v -colored vertices in G .) Many known parameterized reductions from CLIQUE can be modified to use COLSUB(H) as the reduction source, and a seminal result by Marx [59, Corollary 6.1] shows that COLSUB(H) is indeed hard under ETH for graphs H of maximum degree 3, albeit not with an entirely tight lower bound:

► **Theorem 1.1** ([50, 59]). *Assuming ETH, there exists a universal constant $\alpha > 0$ and an infinite sequence of graphs H_1, H_2, \dots such that, for all $k \in \mathbb{N}$, the graph H_k has k vertices and maximum degree 3, and COLSUB(H) does not admit an $O(n^{\alpha \cdot k / \log k})$ -time algorithm.*

This theorem has become a standard tool to prove almost-tight lower bounds along the lines of the above reduction scheme, and it has been applied to numerous parameterized problems from a diverse range of areas [1, 5, 8, 9, 10, 11, 12, 14, 18, 19, 20, 21, 22, 23, 25, 28, 34, 35, 36, 39, 42, 45, 49, 52, 57, 60, 62, 64].¹

1.1 Main Concept: Linkage Capacity

In this paper, we provide a new perspective on the seminal Theorem 1.1, which allows us to prove new results and to obtain a significantly simpler proof, even compared to a recent simplified version [50]. Our new perspective hinges upon a new graph parameter, the *linkage capacity* $\gamma(H)$ of a graph H . Roughly speaking, this parameter measures how well vertices of H can be connected by vertex-disjoint paths on specified endpoint pairs.

¹ Theorem 1.1 is actually a corollary of a more general result proved by Marx [59]: Assuming ETH, there exists a universal constant $\alpha > 0$ such that, for every fixed graph H with treewidth t , the problem COLSUB(H) cannot be solved in time $O(n^{\alpha \cdot t})$. We defer the discussion of this more general result to Section 1.3. For most of the applications cited above, the corollary stated in Theorem 1.1 suffices.

Known Lower Bound for Cliques

To explain our ideas, let us first sketch the classical $n^{\Omega(k)}$ -time lower bound for CLIQUE under ETH (see, e.g., [26, Theorem 14.21]). The proof is as follows: It is known that, assuming ETH, the 3-COLORING problem in n -vertex graphs G with maximum degree 4 cannot be solved in $2^{o(n)}$ time. If G can be transformed into an equivalent instance X of CLIQUE with approximately $3^{n/k}$ vertices, then an $n^{o(k)}$ -time algorithm for CLIQUE would imply a $2^{o(n)}$ -time algorithm for the 3-COLORING problem, contradicting ETH.

To transform G into X , the vertex set $V(G)$ is divided equitably into blocks V_1, \dots, V_k . The vertices of X correspond to the 3-colorings of these blocks, and two vertices in X are connected by an edge if their colorings are compatible, meaning they come from different blocks and together form a proper coloring. This way, the k -cliques K in this “compatibility graph” X correspond bijectively to valid 3-colorings of G : Indeed, the vertices of K provide a valid coloring for each block, and the presence of edges between all $u, v \in V(K)$ in X ensures that the union of these partial colorings is a valid coloring of the entire graph G .

From Cliques to General Subgraphs

To show hardness of COLSUB(H) with general k -vertex patterns H , we adapt the lower bound for CLIQUE. First, consider the favorable scenario that the vertices of an input graph G for 3-COLORING can be split equitably into blocks V_1, \dots, V_k , corresponding to the k vertices of H , such that the edges of G “respect” H : Every edge of G is contained within one block or between blocks V_i and V_j with $ij \in E(H)$. In this scenario, not all pairs of partial 3-colorings need to be checked for compatibility. Indeed, it suffices to check this only between blocks V_i and V_j with $ij \in E(H)$, since no other edges could lead to an incompatibility.

In general however, we cannot assume that an n -vertex graph G of maximum degree 4 can be split equitably such that its edges respect H . To address this, we “re-route” the edges in G along paths on new vertices (that are placed in the old blocks) and edges that *do* respect H . While this eventually yields a graph G' in which all edges indeed respect H , it may be possible that most edges are routed on paths of length $\Omega(k)$, thus increasing the block size from n/k back to n . Even if routing via short paths is possible, it may be possible that a few blocks are hit disproportionately often, leading to the same problem. Both issues would render a fast algorithm for COLSUB(H) useless for the purpose of obtaining a (too) fast algorithm for 3-COLORING.

Linkage Capacity

The crucial observation is that many patterns H enable a simultaneous “batch-rerouting” of batches with $\Omega(k)$ edges in G ; adding all paths for any such a batch to G increases each block size only by 1. Moreover, as also observed in [50, Theorem 4.2], it is sufficient to consider batches that are matchings, since G has maximum degree 4 and thus admits a 5-edge-coloring, i.e., a partition of its edges into 5 matchings.

The *linkage capacity* $\gamma(H)$ allows us to precisely quantify how well H supports batch-rerouting of matchings by vertex-disjoint paths. To define it, first let the *blowup* $H \otimes J_t$ for $t \in \mathbb{N}$ be H with every vertex copied to t clones that form a clique; this is essentially the maximal graph with block size t whose edges respect H . See also Figure 1. Second, call a set X in a graph H' *matching-linked* if, for every matching M with vertex-set X , there exist disjoint u - v -paths in H' realizing the edges $uv \in M$. Then the linkage capacity $\gamma(H)$ of a graph H is the largest $c > 0$ such that $H \otimes J_t$ contains a matching-linked set X of size $\lfloor ct \rfloor$; this is finite, and we even have $\gamma(H) \leq k$, since $|X| \leq |V(H \otimes J_t)| = kt$.

Following the reduction sketch from 3-COLORING given above, and using large matching-linked sets in blowups $H \otimes J_t$ to accommodate the vertices of a 3-COLORING instance G , we establish a conditional lower bound on the complexity of COLSUB(H) based on $\gamma(H)$. For later use, we also prove a lower bound for the counting version #COLSUB(H) under the counting exponential-time hypothesis #ETH.

► **Theorem 1.2.** *Assuming ETH, there exists a universal constant $\alpha > 0$ such that no fixed graph H admits an $O(n^{\alpha \cdot \gamma(H)})$ -time algorithm for COLSUB(H). The same holds for #COLSUB(H) under #ETH.*

It remains to determine when H has large linkage capacity. For example, if H itself admits a large matching-linked set, then this translates to its blowups, thus establishing large $\gamma(H)$. This is however only a sufficient criterion, even though most of our lower bounds are based on it. As we investigate in Section 6, the linkage capacity is related to certain fractional multicommodity flow problems whose relevance in the context of lower bounds for COLSUB(H) under ETH was already identified before [50, 59]. Linkage capacity however is a much more elementary and more applicable concept. In particular, the restriction to matchings allows us to connect it to known results on routing with specified terminal pairs in order to obtain lower bounds on $\gamma(H)$. This in turn allows us to prove new results under ETH without much technical effort.

1.2 Applications of Linkage Capacity

With Theorem 1.2 in hand, we show lower bounds on the complexity of the colorful H -subgraph problem via the linkage capacity $\gamma(H)$. For this, we enlist the help of communication network theory [54, 6], random graph theory [15], linear programming [37, 55], and classical results on connectivity via vertex-disjoint paths from graph theory [58, 66].

A Fully Self-Contained Proof of Theorem 1.1

Our first application of Theorem 1.2 is a significantly simplified and self-contained² proof of the seminal Theorem 1.1. The original proof of this theorem by Marx [59] uses highly nontrivial arguments regarding multicommodity flows as a black box [37]. Even a very recent simplification [50] still requires the construction of expander graphs and routing algorithms for such graphs, both of which are highly nontrivial [2, 55].

By approaching the problem through linkage capacity, we observe that expansion is *not* required to obtain Theorem 1.1. Instead, we can rely on a very simple construction of telecommunication networks, credited to a 1964 paper by Beneš [6], then employed at Bell Labs: A *Beneš network* contains $s = 2^\ell$ input and output vertices, and $k = O(s \log s)$ vertices in total. For every pairing of inputs to outputs, the network guarantees private data streams (i.e., vertex-disjoint paths) connecting each input to its specified output. Both the network construction and routing therein are elementary divide-and-conquer arguments that feature in undergraduate introduction courses to discrete mathematics [54]. A minuscule augmentation of this construction gives us k -vertex graphs of maximum degree 4 and linkage capacity $\Omega(k/\log k)$. Combined with Theorem 1.2, this gives a novel proof of Theorem 1.1.

² We give a self-contained proof starting from the known result that, under ETH, the 3-COLORING problem requires $2^{\Omega(n)}$ time on 4-regular graphs with n vertices. This can be shown easily from ETH together with the sparsification lemma.

We recently found that graphs with large matching-linked sets have been used in communication and extension complexity: A paper by Göös, Jain, and Watson [41, Section 3.3] briefly mentions “bounded-degree butterfly graphs” from an unpublished manuscript on pebble games by Nordström [61, Proposition 5.2] as an alternative to expanders; this alternative construction turns out to be precisely that of Beneš.

Tight Lower Bounds for Dense Graphs

Alon and Marx [4, Theorem 1.4] argue that the logarithmic slack in Theorem 1.1 cannot be overcome by current approaches – including ours. This holds even for patterns H of constant *average* rather than *maximum* degree. More modestly, one can ask for “just slightly” dense k -vertex patterns H such that $\text{COLSUB}(H)$ requires $n^{\Omega(k)}$ time under ETH.

Indeed, Alon and Marx [4, Theorem 1.5(2)] showed that, for every $\delta > 0$, certain specifically constructed patterns S with average degree $O(k^\delta)$ enjoy strong embeddability properties that entail $n^{\Omega(k)}$ -time lower bounds on the colorful S -subgraph problem [4, Theorem 1.8]. For some problems of interest however, e.g., for counting induced k -vertex patterns [24, 32, 64], one can only reduce from the colorful H -subgraph problem for *some* (say, adversarially chosen) dense pattern H , which may not necessarily be a graph S constructed by Alon and Marx. This imposes a bottleneck towards tight lower bounds for such problems.

One partial remedy lies in using large clique minors (see, e.g., [27, 64]). Kostochka [53] showed that *every* graph H of average degree d contains a K_q -minor with $q = \Omega(d/\sqrt{\log d})$. Given a K_q -minor in H , a straightforward reduction yields an $n^{\Omega(q)}$ -time lower bound on the colorful H -subgraph problem under ETH. This implies that *every* pattern H of linear average degree $\Omega(k)$ requires an exponent of $\Omega(k/\sqrt{\log k})$ for the colorful H -subgraph problem (see, e.g., [27, Corollary 2.1]). While this improves upon the lower bound from Theorem 1.1, a slack of $\Omega(\sqrt{\log k})$ remains.

Using linkage capacity, we eliminate this slack and obtain a tight lower bound for dense patterns: Combining two textbook results [30], we show that every pattern H of average degree d has linkage capacity $\Omega(d)$.³ Theorem 1.2 then immediately yields:

► **Theorem 1.3.** *Assuming ETH, there exists a universal constant $\alpha > 0$ such that no fixed graph H with average degree d admits an $O(n^{\alpha \cdot d})$ -time algorithm for $\text{COLSUB}(H)$. The same holds for $\#\text{COLSUB}(H)$ under $\#\text{ETH}$.*

This theorem covers the “worst case”, i.e., patterns H of fixed average degree d that are adversarially chosen so as to minimize $\gamma(H)$. In particular, for linear average degree, an $n^{\Omega(k)}$ bound under ETH follows. This implies new tight lower bounds for very general classes of induced pattern counting problems [24, 64] (see Section 7 for details).

In the “average case”, much lower density turns out to be sufficient for an $n^{\Omega(k)}$ bound. Indeed, known results on routing in random graphs [15] imply directly that *almost all* k -vertex graphs H with average degree $d \in \Omega(k^\beta)$ for constant $\beta > 0$ have linkage capacity $\Theta(k)$. Observe that the average degree is that of the specifically constructed patterns S by Alon and Marx [4]; we show that not only specific patterns, but *almost all* patterns of polynomial average degree have an $n^{\Omega(k)}$ bound for $\text{COLSUB}(H)$.

³ This lower bound is asymptotically tight, since worst-case examples like $K_{d,s-d}$ have linkage capacity at most $2d + 1$. Indeed, a linked set of $2d + 2$ vertices would imply a linkage with $d + 1$ paths in $K_{d,s-d}$. This would in particular imply a matching with $d + 1$ edges, which clearly does not exist in $K_{d,s-d}$.

More generally, we show that the linkage capacity of the Erdős-Rényi random graph $\mathcal{G}(k, p)$ for non-degenerate probabilities p is $\Omega(k/\rho)$, where $\rho = \log(k)/\log(kp)$ is the typical distance between vertices in $\mathcal{G}(k, p)$ [7, 51]. We obtain the following general lower bound:

► **Theorem 1.4.** *Assuming ETH, there exists a universal constant $\alpha > 0$ such that for every constant $\epsilon > 0$ and every $p \geq (1 + \epsilon) \log k/k$, the following holds: With high probability, for an Erdős-Rényi random graph $H \sim \mathcal{G}(k, p)$, the problem COLSUB(H) does not admit an $O(n^{\alpha k/p})$ -time algorithm. Here, $\rho = \log(k)/\log(kp)$ is the typical distance in $G(k, p)$. The same holds for #COLSUB(H) under #ETH.*

Note that ρ is the logarithm of k in the base of the average degree kp ; this captures the time needed to concurrently explore all k vertices in a process that branches into kp random vertices from each vertex. It is intuitively clear that the linkage capacity should be at most $O(k/\rho)$: Almost all vertex pairs u, v in a random graph require u - v -paths of length ρ , so we cannot connect more than k/ρ vertex pairs without exhausting k vertices. The bound from [15] shows that, with high probability, $\Omega(k/\rho)$ vertex pairs *can* be connected.

1.3 Linkage Capacity and Treewidth

Theorems 1.3 and 1.4 are based on lower bounds on the linkage capacity of graphs H in terms of the density of H . We show that the linkage capacity can also be lower-bounded as a function of the *treewidth* of H . As already indicated above, Theorem 1.1 is actually a corollary of a much more general theorem on large-treewidth graphs shown by Marx [59]: Assuming ETH, he proved the existence of a universal constant $\alpha > 0$ such that no fixed graph H with treewidth t admits an $O(n^{\alpha t/\log t})$ -time algorithm for COLSUB(H). To obtain Theorem 1.1 from this general theorem, it suffices to choose a k -vertex expander graph H of maximum degree 3, since such graphs are known to have treewidth $\Omega(k)$.

We recover this theorem by showing that the linkage capacity of a graph H is lower-bounded by its treewidth, up to the “same” logarithmic factor that is missing in the original result by Marx [59]. That is, we show the lower bound $\gamma(H) = \Omega(t/\log t)$ for every graph H of treewidth t . In this proof, we use the same approximate min-cut/max-flow theorem for multicommodity flows [37, 55] that also appears in [59] as black box. Together with Theorem 1.2, this indeed recovers the more general theorem of Marx [59] (also for the counting version #COLSUB(H)) with a more transparent proof. Complementing the lower bound, we use a simple argument about balanced separations in low-treewidth graphs to show an upper bound of $\gamma(H) = O(t)$. We stress that both bounds are asymptotically tight. In particular, it can be shown that k -vertex expander graphs H of maximum degree 3 have linkage capacity $\gamma(H) = \Theta(k/\log k)$ and treewidth $\Theta(k)$.

It is a major open question in parameterized complexity whether the logarithmic loss in Marx’s lower bound [59] can be avoided for all graphs H . Indeed, the following conjecture seems to be known (e.g., communicated to us by Daniel Lokshtanov), although we are not aware of an explicit reference.

► **Conjecture 1.5 (You Cannot Beat Treewidth).** *Assuming ETH, there exists a universal constant $\alpha > 0$ such that no fixed graph H with treewidth t admits an $O(n^{\alpha t})$ -time algorithm for COLSUB(H).*

We note that even removing the $1/\log k$ factor from Theorem 1.1 would constitute a significant breakthrough. Alon and Marx [4] showed that current approaches cannot be used to achieve this; this is also true for our techniques. Still, with Theorems 1.3 and 1.4, we extend the scope where tight bounds for COLSUB(H) are known.

2 Preliminaries

We write $\mathbb{N} = \{1, 2, 3, \dots\}$ for the natural numbers. For $n \in \mathbb{N}$, we write $[n] := \{1, 2, \dots, n\}$. All logarithms are natural unless specified otherwise.

2.1 Basic Definitions

We use standard graph notation [30]. Graphs are finite and undirected, and we write uv for edges between u and v . A *path* from u to v is a sequence $P = (u = w_0, w_1, \dots, w_\ell = v)$ of distinct vertices such that consecutive vertices are adjacent. Slightly abusing notation, we also interpret P as a path from v to u . For a graph G and $X \subseteq V(G)$, we write $G[X]$ for the subgraph induced by X and $G - X := G[V(G) \setminus X]$ for the result of deleting X from G .

A *colored graph* is a triple $G = (V, E, c)$ where $c: V(G) \rightarrow C$ is a not necessarily proper *coloring* of the vertices. We say G is *canonically colored* if c is the identity mapping and we write G^{can} for the canonically colored version of G .

Given a “pattern” graph H and “host” graph G , we write $\#\text{Sub}(H \rightarrow G)$ for the number of subgraphs of G that are isomorphic to H . If H and G are colored, only subgraphs preserving the coloring are counted. For a fixed graph H , the problem $\#\text{COLSUB}(H)$ takes as input a colored graph $G = (V, E, c)$ with $c: V(G) \rightarrow V(H)$, and asks to compute $\#\text{Sub}(H^{\text{can}} \rightarrow G)$, while its decision version $\text{COLSUB}(H)$ asks whether $\#\text{Sub}(H^{\text{can}} \rightarrow G) \geq 1$.

To analyze the complexity of $\text{COLSUB}(H)$, we rely on several tools.

► **Definition 2.1** (Blowup). *Given a graph H and an integer $t \geq 1$, the blowup graph $H \otimes J_t$ contains the vertices $v^{(i)}$ for all $v \in V(H)$ and $i \in [t]$, and edges*

$$\{u^{(i)}v^{(j)} \mid uv \in E(H), i, j \in [t]\} \cup \{u^{(i)}u^{(j)} \mid u \in V(H), i \neq j \in [t]\}.$$

► **Remark 2.2.** Marx [59] uses the notation $H^{(t)}$ instead of $H \otimes J_t$. We choose $H \otimes J_t$ since there is no exponential increase in size, but we are rather taking a tensor product of H and the $(t \times t)$ all-ones matrix (usually denoted by J_t) and then turn cloned vertices into cliques.

A *multigraph* M is a graph that allows parallel edges with the same endpoints, but no self-loops. The *degree* $\deg_M(v)$ of a vertex $v \in V(M)$ is the number of edges incident to v , taking multiplicities into account. The *average degree* of M is $d(M) := 2|E(M)|/|V(M)|$.

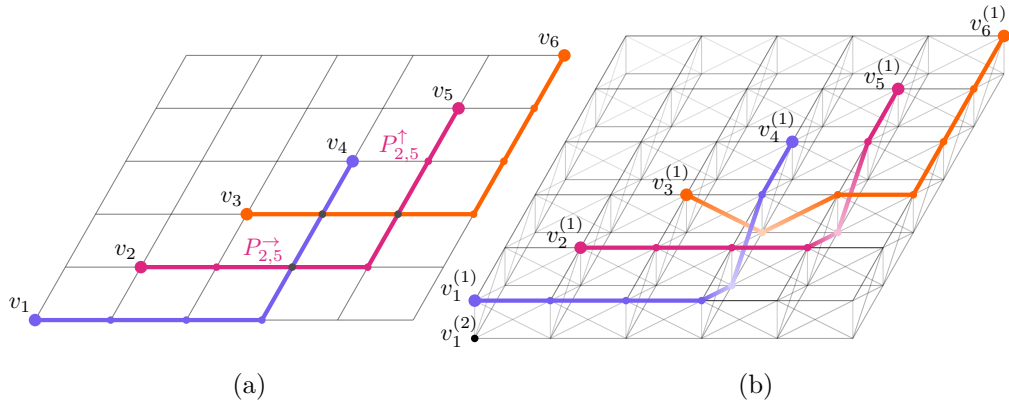
A *matching* in M is set $M' \subseteq E(M)$ of pairwise vertex-disjoint edges. Slightly abusing notation, we regularly interpret M' again as a graph with edge set M' and vertices for all endpoints in M' . A *q -edge coloring* of M is a partition of $E(M)$ into q matchings. The *edge-chromatic number* of M , denoted by $\chi'(M)$, is the minimum number q such that M admits a q -edge coloring. A theorem by Shannon [65] provides an upper bound on the edge-chromatic number in terms of the maximum degree, though a looser bound with factor 2 can be achieved by a straightforward greedy algorithm.

► **Theorem 2.3** ([65]). *Every multigraph M of maximum degree d has $\chi'(M) \leq \lfloor \frac{3}{2}d \rfloor$.*

2.2 Linkages

Our hardness proofs for $\text{COLSUB}(H)$ crucially rely on linkages in graphs.

► **Definition 2.4** (Linkage and congestion). *Given a graph H and a multigraph M with vertex set $X \subseteq V(H)$, an M -linkage in H is a collection of paths $Q = (P_{uv})_{uv \in E(M)}$ such that P_{uv} has endpoints u and v . For $r \in \mathbb{N}$, we say that Q is r -congested if, for all $w \in V(H)$, at most r paths $P_{uv} \in Q$ contain w . If $r = 1$, we call Q an uncongested M -linkage.*



■ **Figure 1** (a) The grid graph \mathbb{E}_6 . Thick paths depict a 2-congested M -linkage, where $M = \{v_1v_4, v_2v_5, v_3v_6\}$ is a matching on the diagonal vertices. (b) The blowup graph $\mathbb{E}_6 \otimes J_2$, and an ungested M -linkage obtained from the 2-congested M -linkage in \mathbb{E}_6 .

Observe that, if Q is an ungested M -linkage, then M is necessarily a matching. We note that we commonly work with ungested M -linkages. More precisely, we usually require ungested M -linkages in blowups of graphs H . Towards this end, it is often convenient to “project” M back to the base graph H . Let H be a graph and let M be a multigraph with $V(M) \subseteq V(H \otimes J_q)$. We define the H -projection of M to be the multigraph $\pi(M)$ with vertex set $V(\pi(M)) := \{v \mid v^{(i)} \in V(M)\}$ and edge multiset $E(\pi(M)) := \{\{vw \mid v^{(i)}w^{(j)} \in E(M), v \neq w\}\}$.

► **Lemma 2.5.** *Let H be a graph and let $q \in \mathbb{N}$. Also let M be a matching with $V(M) \subseteq V(H \otimes J_q)$. If there is a q -congested $\pi(M)$ -linkage in H , then there is an ungested M -linkage in $H \otimes J_{2q}$.*

Note that a blowup of order $2q$ rather than q is needed. This is needed since we do not allow self-loops in the projection, i.e., the projection of M ignores edges contained in the same block $\{v^{(i)} \mid i \in [q]\}$. For technical reasons, we decide to handle those edges separately at the cost of losing a factor of two.

Proof. Let Q be a q -congested $\pi(M)$ -linkage in H . We obtain an ungested M -linkage Q' in $H \otimes J_{2q}$ as follows. For a vertex $w \in V(H)$, let P_1, \dots, P_t (if any exists) be all paths in Q that contain w as an internal vertex. We have $t \leq q$ by definition. We replace w in P_i with the vertex $w^{(q+i)}$ from the blowup graph $H \otimes J_{2q}$. Also, all endpoints of the paths are replaced in the natural way, i.e., if P has endpoints u and v , and uv is the “projection” of $u^{(i)}v^{(j)}$ in M , then P gets endpoints $u^{(i)}$ and $v^{(j)}$. Finally, for each edge $v^{(i)}v^{(j)} \in E(M)$, we add the path $(v^{(i)}, v^{(j)})$ to Q' . By the definition of the blowup graph, the resulting collection Q' is an ungested M -linkage in $H \otimes J_{2q}$. ◀

The following example illustrates the notion of linkages and the interplay between congestion and blowups; see Figure 1.

► **Example 2.6 (Grid graph).** Write \mathbb{E}_ℓ for the grid graph on vertex set $[\ell] \times [\ell]$. For every matching M on the set of diagonal vertices $v_i = (i, i)$ for $i \in [\ell]$, we observe that \mathbb{E}_ℓ contains a 2-congested M -linkage $Q(M) = \{P_{uv}\}_{uv \in M}$. This 2-congested linkage induces an ungested M -linkage in $\mathbb{E}_\ell \otimes J_2$ via Lemma 2.5. See also Figure 1.

More specifically, given an edge from $u = (a, a)$ to $v = (b, b)$ for $a < b$, we define P_{uv} as the concatenation of the path P_{uv}^{\rightarrow} on vertices $u = (a, a), \dots, (b, a)$ and the path P_{uv}^{\uparrow} on vertices $(b, a), \dots, (b, b) = v$. The paths P_{uv}^{\rightarrow} for $uv \in M$ are vertex-disjoint (as distinct paths have distinct y -coordinates), and so are the paths P_{uv}^{\uparrow} for $uv \in M$ (having distinct x -coordinates), so $Q(M)$ is indeed a 2-congested M -linkage.

3 Lower Bounds from Linkage Capacity

The Exponential-Time Hypothesis ETH [43] postulates the existence of a constant $\alpha > 0$ such that no $O(2^{\alpha n})$ time algorithm can decide, on input a 3-CNF formula φ with n variables, whether φ admits a satisfying assignment. Its *a priori* weaker counting version #ETH postulates the same lower bound for *counting* the satisfying assignments of φ [29]. For both hypotheses, the sparsification lemma [44, 29] rules out such algorithms even under the additional condition that every variable in φ appears in at most C clauses, for some constant $C \in \mathbb{N}$. By a standard reduction, lower bounds follow for the problem 3-COLORING of deciding whether an input graph G admits a proper vertex-coloring with 3 colors where no adjacent vertices receive the same color; see for example [56, Theorem 3.2].

► **Theorem 3.1.** *Assuming ETH, there is a constant $\beta > 0$ such that 3-COLORING cannot be solved in time $O(2^{\beta n})$ for n -vertex input graphs G of maximum degree 4. The same holds for #3-COLORING under #ETH.*

This theorem is the foundation for the lower bounds shown in this paper.

3.1 Instances That Fit into Blowups

It is useful for us to generalize 3-COLORING slightly, by allowing edges to either enforce equality or disequality of their endpoint colors. Since “equality edges” can be contracted without changing the number of valid assignments, we obtain an immediate way to simulate edges in a 3-COLORING instance by paths.

► **Definition 3.2.** *Given a graph $G = (V, E)$ with a partition $E = E_{=} \cup E_{\neq}$, a proper 3-assignment is a function $a : V \rightarrow [3]$ such that $a(u) = a(v)$ for all $uv \in E_{=}$, while $a(u) \neq a(v)$ for all $uv \in E_{\neq}$. The problem 3-ASSIGNMENT asks to determine the existence of a proper assignment on input $(G, E_{=}, E_{\neq})$, while #3-ASSIGNMENT asks to count them.*

It is possible to convert instances G for 3-ASSIGNMENT into instances X for COLSUB(H). Moreover, if G fits into a moderately small blowup of H , then X is only of moderately exponential size. This can be shown by a simple “split-and-list” reduction that follows the sketch given in the introduction for CLIQUE.

► **Lemma 3.3.** *Let H be a fixed k -vertex graph with canonical vertex-coloring. Given a subgraph G of $H \otimes J_t$ for $t \in \mathbb{N}$, a colored graph X on $k \cdot 3^t$ vertices can be computed in $9^t \cdot \text{poly}(k, t)$ time such that $\#\text{Sub}(H \rightarrow X)$ equals the number of proper 3-assignments in G .*

Proof. Suppose $V(H) = [k]$ and consider the partition of $V(G)$ into $V_w = \{w^{(1)}, \dots, w^{(t)}\}$ for $w \in [k]$. Define X_w for $w \in [k]$ as the set of all proper 3-assignments to $G[V_w]$.

For $w, w' \in [k]$, we call two 3-assignments $a \in X_w$ and $a' \in X_{w'}$ *compatible* if their union is a proper 3-assignment of $G[V_w \cup V_{w'}]$. Let us define

$$A_X := \{(a_1, \dots, a_k) \in X_1 \times \dots \times X_k \mid \forall ww' \in E(H) : a_w \text{ and } a_{w'} \text{ are compatible}\}$$

$$A_G := \{a : V(G) \rightarrow [3] \mid a \text{ is proper 3-assignment of } G\}$$

28:10 Can You Link Up With Treewidth?

We observe that the map $a \mapsto (a_1, \dots, a_k)$ from A_G to A_X , where a_w is the restriction of a to V_w , is a bijection. Indeed, in the image of $a \in A_G$ under this map, a_w and $a_{w'}$ are compatible for all $w, w' \in E(H)$. Conversely, given $(a_1, \dots, a_k) \in A_X$, recall that every edge $uv \in E(G)$ satisfies $u \in V_w$ and $v \in V_{w'}$ for some $w, w' \in [k]$ with (a) $w = w'$, or (b) $ww' \in E(H)$. In case (a), since a_w is proper, the endpoints of uv receive a proper assignment under a . In case (b), because the union of a_w and $a_{w'}$ is a proper 3-assignment of $G[V_w \cup V_{w'}]$, the endpoints of uv receive a proper assignment under a . Thus $a \in A_G$.

Finally, the graph X is defined on vertices $\bigcup_{w \in [k]} X_w$, where each vertex in X_w is colored by $w \in [k]$. An edge is present between $a \in X_w$ and $b \in X_{w'}$ if and only if $ww' \in E(H)$ and a and b are compatible. The (colored) subgraphs S of X isomorphic to H correspond to tuples in A_X . Indeed, $V(S)$ corresponds to a tuple $(a_1, \dots, a_k) \in X_1 \times \dots \times X_k$, and the presence of edges of H in S implies that a_w and $a_{w'}$ are compatible for $ww' \in E(H)$. Since $|X_w| \leq 3^t$ for all $w \in [k]$, the graph X can be computed by brute-force in $9^t \cdot \text{poly}(k, t)$ time. ◀

3.2 The Linkage Capacity of a Graph

First, we need to define a term for vertex sets X in graphs that can be paired up arbitrarily via paths in H . This resembles Diestel's [30] notion of *linkedness*, see Section 5.1, which however requires this property for the entire graph H .

► **Definition 3.4** (Matching-linked set). *Given a graph H , we say that $X \subseteq V(H)$ is matching-linked if H contains an uncongested M -linkage for every matching M on vertex set X .*

► **Remark 3.5.** We stress that the condition in the definition is crucially required even if the edges of M are *not* contained in $E(H)$: Only the *endpoints* of M need to be contained in X .

A simple edge-coloring argument, also used in Lemma 3.11, shows that large matching-linked sets X in blowups $H \otimes J_t$ suffice to embed graphs G of maximum degree Δ into $H \otimes J_{2\Delta \cdot t}$. Thus, large matching-linked sets X in blowups of H are a useful “resource” attainable from H that allows us to use Lemma 3.3.

Not all such sets X however need to originate from matching-linked sets in H itself. Consider a set X in H that *just* fails to be matching-linked, as in Example 2.6, in the sense that X still admits M -linkages of congestion 2 in H . Such M -linkages then induce uncongested M -linkages in $H \otimes J_2$. As our goal is to embed a 3-COLORING instance G into a moderately large blowup of H , such a constant-factor loss would be acceptable. This flexibility is captured by the *linkage capacity*, which measures the maximum size of matching-linked sets in blowups of H relative to the blowup order.

► **Definition 3.6** (Linkage capacity). *The linkage capacity $\gamma(H)$ is the supremum over $c > 0$ such that $H \otimes J_t$ contains a matching-linked set X with $|X| = \lfloor ct \rfloor$ for all large enough $t \in \mathbb{N}$.*

Every graph H trivially satisfies $1 \leq \gamma(H) \leq |V(H)|$. We show below that large matching-linked sets in H lift into blowups, establishing high linkage capacity $\gamma(H)$ – but as mentioned above, even a matching-linked set in a small blowup of H suffices.

► **Lemma 3.7.** *Let H be a graph and suppose $H \otimes J_q$ for $q \in \mathbb{N}$ contains a matching-linked set X . Then $\gamma(H) \geq \frac{1}{3} \cdot |X|/q$.*

Proof. The proof rests on the following claim.

▷ **Claim 3.8.** Let H' be a graph and suppose $X' \subseteq V(H')$ is a matching-linked set. Then $X'_t := \{v^{(i)} \mid v \in X', 1 \leq i \leq t/3\}$ is matching-linked in $H' \otimes J_t$ for every $t \in \mathbb{N}$.

Proof. Let M be a matching on X'_t and let $M' := \pi(M)$ be the H' -projection of M . Observe that M' has maximum degree at most $t/3$, so $\chi'(M') \leq t/2$ by Theorem 2.3. Hence, the multigraph M can be partitioned into $r \leq t/2$ matchings M_1, \dots, M_r on X' . Since X' is a matching-linked set, for every M_i , there is an uncongested M_i -linkage Q_i in H' . So $Q := \bigcup_{i \in [r]} Q_i$ is a r -congested M' -linkage in H' , and there is an uncongested M -linkage in $H' \otimes J_t$ by Lemma 2.5. \triangleleft

Let $c < \frac{1}{3} \cdot |X|/q$. Then there is $t_0 \in \mathbb{N}$ and $c' < \frac{1}{3} \cdot |X|$ such that

$$c' \cdot \lfloor t/q \rfloor \geq c \cdot t \tag{1}$$

for all $t \geq t_0$. Also, there is some $\ell_0 \in \mathbb{N}$ such that

$$|X| \cdot \lfloor \ell/3 \rfloor \geq \lfloor c'\ell \rfloor \tag{2}$$

for all $\ell \geq \ell_0$. Now let $t \geq \max(t_0, \ell_0 \cdot q)$ and let $\ell := \lfloor t/q \rfloor \geq \ell_0$ and consider the graph $H' := H \otimes J_q$. By Claim 3.8 the graph $H' \otimes J_\ell$ contains a matching-linked set X'_ℓ with

$$|X'_\ell| \geq |X| \cdot \lfloor \ell/3 \rfloor \stackrel{(1)}{\geq} \lfloor c'\ell \rfloor \stackrel{(2)}{\geq} \lfloor ct \rfloor.$$

Since $(H \otimes J_q) \otimes J_\ell$ is a subgraph of $H \otimes J_t$, we conclude that $H \otimes J_t$ contains a matching-linked set of size $\lfloor ct \rfloor$, thus proving the lemma. \blacktriangleleft

As a concrete example, let us use Example 2.6 to bound the linkage capacity of grids.

► **Lemma 3.9.** *For the ℓ -by- ℓ grid graph \boxplus_ℓ , we have $\gamma(\boxplus_\ell) \geq \ell/6$.*

Proof. Suppose $V(\boxplus_\ell) = [\ell]^2$. By Example 2.6, the set $X := \{(i, i)^{(1)} \mid i \in [\ell]\}$ is matching-linked in the blowup $\boxplus_\ell \otimes J_2$. The lemma follows by invoking Lemma 3.7. \blacktriangleleft

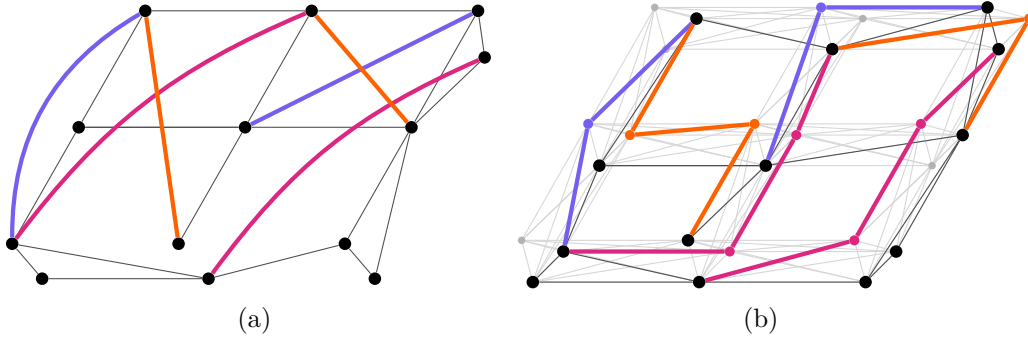
3.3 Fitting Instances into Blowups via Linkage Capacity

Having introduced linkage capacity and its key properties, we now use it to embed graphs G into blowups $H \otimes J_t$ with $t = O(n/\gamma(H))$. If we can show that $\gamma(H)$ is large, then ETH implies strong lower bounds for COLSUB(H) via Theorem 3.1 and Lemma 3.3.

A minor constructivity issue arises: Some techniques for lower-bounding $\gamma(H)$ do not necessarily yield efficient algorithms for finding linkages in blowups of H . Thus, it could *a priori* be possible for G to embed into $H \otimes J_t$, yet we cannot efficiently find an embedding. This concern is resolved by known algorithms for graphs of bounded neighborhood diversity [40, Theorem 3.7] (a self-contained argument is given in the full version):

► **Theorem 3.10.** *Let $f(k) = 3^{k^2+2}$. Given a k -vertex graph H and $t \geq 2$ as input, a matching-linked set X of maximum size in $H \otimes J_t$ can be found in $O(t^{f(k)})$ time. Given additionally a matching-linked set X in $H \otimes J_t$ and a matching M with vertex set X , an M -linkage in $H \otimes J_t$ can be found in $O(t^{f(k)})$ time.*

We can now turn to our main lemma. In the following, given graphs G and G' without loops or multi-edges, a G -linkage $Q = (P_{uv})_{uv \in E(G)}$ in G' is a *topological G -minor model* in G' if paths P_{uv} and $P_{u'v'}$ for $uv, u'v' \in E(G)$ in Q intersect only at endpoints. In particular, such intersections can occur only if uv and $u'v'$ share a common vertex. We refer to the subgraph of G' induced by Q as the *image* of Q .



■ **Figure 2** (a) A graph G that fails to be embedded into the blowup $\boxplus_3 \otimes J_2$ due to the colored edges, which are partitioned into three matchings. (b) An embedding of G into $\boxplus_3 \otimes J_3$ as a topological minor, where each colored edge gets routed via new vertices from the blowup.

▶ **Lemma 3.11.** *Let H be a fixed k -vertex graph and let $f(k) = 3k^{k+2}$. Then there is an $O(n^{f(k)})$ time algorithm that, given an instance G for 3-COLORING with n vertices and maximum degree 4, outputs an instance for 3-ASSIGNMENT with graph G' such that*

1. G' is the image of a topological G -minor in $H \otimes J_t$ for $t = 8\lceil n/\gamma(H) \rceil$, and
2. the proper 3-colorings of G correspond bijectively to the proper 3-assignments of G' .

Proof. Let $t' := \lceil n/\gamma(H) \cdot 15/14 \rceil$. Then $7t' \leq t$ whenever $t' \geq 14$. Definition 3.6 implies that, if t' is large enough then $H \otimes J_{t'}$ contains a matching-linked set X of size n .⁴ In $O(n^{f(k)})$ time, Theorem 3.10 finds such a set X . Fix $V(G) = X$ in the following.

The straightforward greedy algorithm yields a 7-edge-coloring $E(G) = M_1 \cup \dots \cup M_7$ in time $O(n)$. As X is matching-linked, the graph $H \otimes J_{t'}$ contains an uncongested M_i -linkage Q_i for every individual $i \in [7]$. Each linkage can be found $O(n^{f(k)})$ time via Theorem 3.10. These linkages together induce a topological G -minor model in $H \otimes J_{7t'}$ (see Figure 2): Consider $V(H \otimes J_{7t'})$ to be partitioned into 7 layers such that layer $i \in [7]$ contains the vertices $v^{(j)}$ with $v \in V(H)$ and $j \in (t' - 1)i + [t']$. By placing non-endpoint vertices from the linkages Q_1, \dots, Q_7 into different layers and keeping all endpoints in the first layer, we obtain a topological G -minor model Q in $H \otimes J_{7t'}$. Let us write G' for the image of Q .

We finalize the construction of the 3-ASSIGNMENT instance by specifying a partition of $E(G')$ into E_+ and E_- : For each path P_{uv} in Q , place one arbitrary edge into E_+ and all other edges into E_- . Then the proper 3-assignments to G' correspond to the proper 3-colorings of G , since contracting all equality edges in G' (which does not change the number of 3-assignments) yields an isomorphic copy of G on disequality edges. ◀

Combining the above, the proof of Theorem 1.2 is complete.

Proof of Theorem 1.2. By Theorem 3.1, ETH implies a constant $\beta > 0$ such that no $O(2^{\beta \cdot n})$ -time algorithm solves 3-COLORING on n -vertex graphs G of maximum degree 4. We set $\alpha = \beta/26$ and derive a contradiction from an $O(s^{\alpha \cdot \gamma})$ -time algorithm for COLSUB(H) on s -vertex input graphs, where H is any fixed graph with $\gamma = \gamma(H)$. Moreover, we only need to consider the case where $\gamma \geq 26/\beta$, because otherwise the theorem is trivial.

⁴ If t' is not large enough, then n is bounded by a function of H . We can then compute the number q of 3-colorings of G in constant time and output a dummy instance G' with q proper 3-assignments.

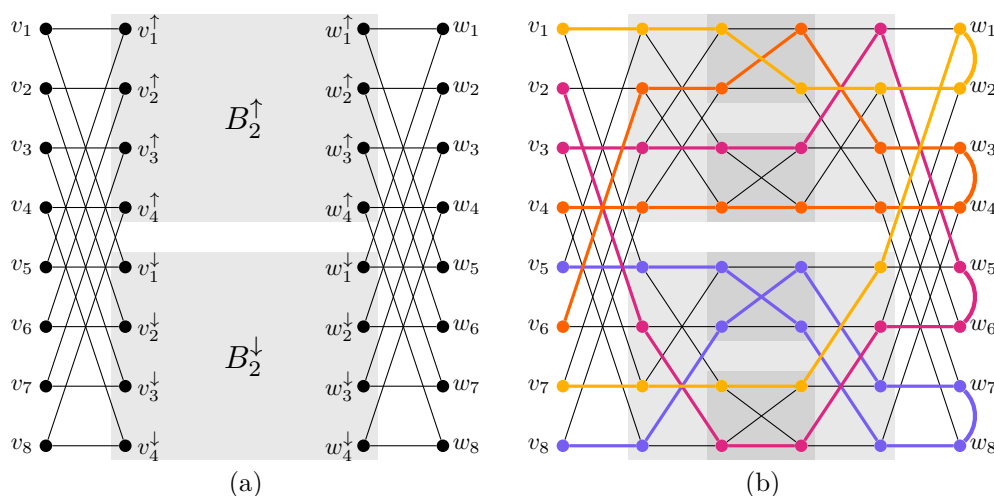


Figure 3 (a) Recursive construction of Beneš network B_3 with 8 inputs and 8 outputs from two copies of B_2 . (b) The augmented Beneš network \check{B}_3 is obtained by adding a matching to the outputs of B_3 , shown as curved edges. Thick paths indicate an M -linkage in \check{B}_3 for the matching $M = \{v_1v_7, v_2v_3, v_4v_6, v_5v_8\}$ on the input vertices.

In the following, let G be an instance for 3-COLORING of maximum degree 4. In time $O(n^{f(k)})$, Lemma 3.11 computes from G an equivalent instance for 3-ASSIGNMENT with a graph $G' \subseteq H \otimes J_t$ for $t = 8\lceil n/\gamma \rceil$. In time $9^t \cdot \text{poly}(k, t)$, Lemma 3.3 then yields a graph X with $|V(X)| \leq k \cdot 3^t$ such that 3-assignments in G' correspond to colorful H -copies in X . The overall running time to construct X is

$$O(n^{f(k)}) + 9^{8\lceil n/\gamma \rceil} \cdot \text{poly}(k, t) = O(2^{26n/\gamma}) = O(2^{\beta \cdot n}). \tag{3}$$

In the last step, we use the aforementioned assumption $\gamma \geq 26/\beta$. Then use the assumed $O(s^{\beta/26 \cdot \gamma})$ -time algorithm for COLSUB(H) on s -vertex input graphs. Its running time on the graph X constructed before, with $s \leq k \cdot 3^t$ vertices, is

$$O((k \cdot 3^t)^{\beta/26 \cdot \gamma}) = O(3^{8\lceil n/\gamma \rceil \cdot \beta/26 \cdot \gamma}) = O(3^{\beta/2 \cdot n}) = O(2^{\beta \cdot n}). \tag{4}$$

Combining Equations (3) and (4), we conclude that both constructing X and solving COLSUB(H) on X can be achieved in overall time $O(2^{\beta \cdot n})$. This contradicts Theorem 3.1. The proof for the counting version is analogous. \blacktriangleleft

4 Switching Networks

In this section, we consider a construction by Beneš [6] that yields k -vertex graphs with degree 4 and a linkage capacity of $\Omega(k/\log k)$. In particular, this allows us to complete the fully self-contained proof of Theorem 1.1.

The Beneš network B_ℓ for $\ell \in \mathbb{N}$ has 2^ℓ distinguished input and output vertices. In our terms, for every matching M between the inputs and outputs, the network B_ℓ admits an uncongested M -linkage. By “short-circuiting” the outputs, we obtain an *augmented Beneš network* \check{B}_ℓ , which allows routing paths from inputs back to inputs. In our terms, the inputs form a matching-linked set, since every matching M on the inputs admits an uncongested M -linkage in \check{B}_ℓ .

■ **Algorithm 1** Construct plain Beneš networks.

```

procedure BENES( $\ell$ ) returns  $B_\ell$  with  $s = 2^\ell$  in-/outputs
    if  $\ell = 1$  then return  $K_{2,2}$  with in-/outputs  $v_i, w_i$  for  $i \in [2]$ 
    else
         $B^\uparrow \leftarrow$  BENES( $\ell - 1$ ) with in-/outputs  $v_i^\uparrow, w_i^\uparrow$  for  $i \in [s/2]$ 
         $B^\downarrow \leftarrow$  BENES( $\ell - 1$ ) with in-/outputs  $v_i^\downarrow, w_i^\downarrow$  for  $i \in [s/2]$ 
         $B \leftarrow$  vertex-disjoint union of  $B^\uparrow$  and  $B^\downarrow$ 
        for  $i \in [s/2]$  do add to  $B$ 
            all four edges between  $\{v_i, v_{i+s/2}\}$  and  $\{v_i^\uparrow, v_i^\downarrow\}$ ,
            all four edges between  $\{w_i, w_{i+s/2}\}$  and  $\{w_i^\uparrow, w_i^\downarrow\}$ 
        return  $B$  with in-/outputs  $v_i, w_i$  for  $i \in [s]$ 
    
```

► **Definition 4.1** (Beneš networks). *The plain Beneš network B_ℓ for $\ell \in \mathbb{N}$ is the graph with distinguished inputs v_i and outputs w_i , for $i \in [s]$ with $s = 2^\ell$, returned by BENES(ℓ) in Algorithm 1. The augmented Beneš network \check{B}_ℓ is obtained from B_ℓ by adding an edge between outputs w_{2i-1} and w_{2i} , for each $i \in [s/2]$.*

A visualization can be found in Figure 3. Both B_ℓ and \check{B}_ℓ clearly have maximum degree 4. Let $T(s)$ for $s = 2^\ell$ count the vertices in the s -input Beneš network B_ℓ or \check{B}_ℓ . By construction, we have $T(s) = 2 \cdot T(s/2) + 2s$, and thus $T(s) = 2s \log_2 s$. Beneš networks are designed to admit uncongested linkages between the inputs and outputs [6]:

► **Theorem 4.2.** *For $\ell \in \mathbb{N}$, the set V of inputs in \check{B}_ℓ is matching-linked, with $|V| = s = 2^\ell$. Moreover, given as input $\ell \in \mathbb{N}$ and a matching M on V , an uncongested M -linkage in \check{B}_ℓ can be computed in $O(s \log s)$ time.*

With Lemma 3.7, we obtain:

► **Corollary 4.3.** *For $s = 2^\ell$, we have $\gamma(\check{B}_\ell) \geq s/3$.*

By combining Theorem 1.2 and Corollary 4.3, we can give an elementary proof of Theorem 1.1 (in a slightly modified form; see Remark 4.5).

► **Theorem 4.4.** *Assuming ETH, there exists a universal constant $\alpha > 0$ and an infinite sequence of graphs H_1, H_2, \dots such that, for all $k \in \mathbb{N}$, the graph H_k has k vertices and maximum degree 4, and COLSUB(H_k) does not admit an $O(n^{\alpha \cdot k / \log k})$ -time algorithm.*

Proof. For every $k \in \mathbb{N}$, pick $\ell \in \mathbb{N}$ maximal such that $|V(\check{B}_\ell)| \leq k$. Let H_k be obtained from \check{B}_ℓ by adding isolated vertices until the number of vertices is k . Since $|V(\check{B}_\ell)| = 2^{\ell+1}\ell$, we conclude that $2^{\ell+1}\ell \leq k < 2^{\ell+2}(\ell + 1)$ which implies that $k / \log_2 k < 2^{\ell+2}$. So

$$\gamma(H_k) \geq \gamma(\check{B}_\ell) \geq 2^\ell / 3 > \frac{1}{12} \cdot k / \log_2 k$$

by Corollary 4.3. Now the theorem follows from Theorem 1.2. ◀

► **Remark 4.5.** Observe that Theorem 1.1 provides a sequence of graphs of maximum degree 3 whereas Theorem 4.4 “only” guarantees maximum degree 4. However, the augmented Beneš networks \check{B}_ℓ can easily be modified to have maximum degree 3 by replacing each vertex with an edge, so \bowtie becomes $\bowtie \leftarrow$, and all other relevant properties remain the same.

For readers familiar with expander graphs, let us also remark that the Beneš network B_ℓ with $s = 2^\ell$ does *not* have constant expansion, as witnessed by its “upper half” U that contains the vertices of $B_{\ell-1}^+$ and all inputs and outputs with indices $i \in [s/2]$: We have $|U| = s \log_2 s$, but the $2s$ neighbors of U are all contained in the first two and last two columns of B_ℓ . This also holds for the augmented \check{B}_ℓ .

Universality of augmented Beneš networks

As an independent point of interest, let us remark that blowups of Beneš networks are universal for bounded-degree graphs with respect to topological minor containment: Mimicking the proof of Lemma 3.11, every n -vertex graph of maximum degree Δ can be found as a topological minor in the 2Δ -blowup of an augmented Beneš network with n inputs.

For comparison, every graph that contains every n -vertex graph of maximum degree Δ as a *subgraph* must necessarily have $\Omega(n^{2-2/\Delta})$ edges [3]. Under the more relaxed notion of universality via topological minor containment, Beneš networks show that universal graphs with only $O(\Delta^2 \cdot n \log n)$ vertices and edges are achievable.

► **Theorem 4.6.** *For every $n, \ell \in \mathbb{N}$, every graph G of maximum degree Δ and $n \leq 2^\ell$ vertices is a topological minor of $\check{B}_\ell \otimes J_{2\Delta-1}$. Moreover, on input G , a topological G -minor model in $\check{B}_\ell \otimes J_{2\Delta-1}$ can be computed in polynomial time.*

Proof. By Theorem 4.2, the inputs in \check{B}_ℓ form a matching-linked set X of size $s = 2^\ell \geq n$. We view $V(G) \subseteq X$ and decompose $E(G)$ into $2\Delta - 1$ matchings via the greedy edge-coloring algorithm. For each matching M , we use Theorem 4.2 to find an M -linkage Q in \check{B}_ℓ and place the internal vertices of Q in a private layer of $\check{B}_\ell \otimes J_{2\Delta-1}$, as in the proof of Lemma 3.11. The union of the linkages constructed this way is a topological G -minor model in $\check{B}_\ell \otimes J_{2\Delta-1}$. ◀

5 Patterns of Superlinear Density

We turn our attention to dense patterns, i.e., k -vertex patterns H of average degree $d(H) \in \omega(1)$. Unlike the sparse setting discussed earlier, a linkage capacity of $\Theta(k)$ is achievable in the dense case, which implies tight lower bounds for $\text{COLSUB}(H)$ under ETH.

5.1 Worst Case

We show that, for every graph H , the average degree $d(H) = 2|E(H)|/|V(H)|$ is a lower bound on the linkage capacity of H , up to a constant factor. First, we use Mader’s Theorem [58, Corollary 1] to extract a highly connected subgraph from H . A graph H is ℓ -connected if $|V(H)| > \ell$ and $H - X$ is connected for every set $X \subseteq V(H)$ with $|X| < \ell$.

► **Theorem 5.1** (see [30, Theorem 1.4.3]). *Every graph H with $d(H) \geq 4\ell$ contains a $(\ell + 1)$ -connected subgraph H' with $d(H') > d(H) - 2\ell$.*

Second, within the scope of this subsection only, we say that a graph H is ℓ -globally linked if $|V(H)| \geq 2\ell$ and each set $X \subseteq V(H)$ of size at most 2ℓ is matching-linked in H . (In graph theory, this notion is usually just called ℓ -linked – see, e.g., [30]. In our paper, we refer to it as ℓ -globally linked to distinguish it from our previous definitions of linkedness.) This definition implies in particular that H contains a matching-linked set X with $|X| \geq 2\ell$. Thomas and Wollan [66, Corollary 1.2] show that high connectivity implies high global linkedness.

► **Theorem 5.2** (see [30, Theorem 3.5.3]). *Let H be a graph and $\ell \in \mathbb{N}$. If H is 2ℓ -connected and $d(H) \geq 16\ell$, then H is ℓ -globally linked.*

28:16 Can You Link Up With Treewidth?

Together, these two theorems imply a lower bound on the linkage capacity that is linear in the average degree.

► **Lemma 5.3.** *For every graph H , we have $\gamma(H) \geq d(H)/48$.*

Proof. Theorem 5.1 yields a $\lceil d(H)/4 \rceil$ -connected subgraph H' of H with $d(H') > d(H)/2$. Theorem 5.2 shows that H' is $\lceil d(H)/32 \rceil$ -globally linked and thus contains a matching-linked set X of size at least $d(H)/16$. Then Lemma 3.7 shows that $\gamma(H) \geq \gamma(H') \geq d(H)/48$, where the first inequality uses that H' is subgraph of H . ◀

Now, Theorem 1.3 follows from Theorem 1.2 and Lemma 5.3.

5.2 Average Case

To show the hardness in the average case, we consider the linkage capacity of the Erdős-Rényi random graph. Let $\mathcal{G}(k, p)$ denote the distribution over k -vertex graphs where each edge is included independently with probability p . We need the following theorem adapted from [15], where “with high probability” refers to a probability tending to 1 for $k \rightarrow \infty$.

► **Theorem 5.4.** *Let $\varepsilon > 0$ be a constant. For all $p \geq (1 + \varepsilon) \log(k)/k$ the following holds: With high probability, for a random graph $H \sim \mathcal{G}(k, p)$, every matching M on $V(H)$ can be partitioned into $r = O(\log k / \log kp)$ matchings M_1, \dots, M_r such that H contains an uncongested M_i -linkage for all $i \in [r]$.*

The original theorem statement and proof in [15, Corollary 1.1] are concerned with the fixed-sized random graph model $G(k, m)$, and only deals with even k . But on the other hand, they give a stronger statement concerning the algorithmic efficiency of finding the desired partition, that it can be obtained with high probability by a random partition. A proof of the version stated here can be found in the full version.

The last theorem can be used to find large matching-linked sets inside a proper blowup of a random graph, which implies a high linkage capacity by Lemma 3.7.

► **Lemma 5.5.** *Let $\varepsilon > 0$ be a constant. For all $p \geq (1 + \varepsilon) \log(k)/k$, the linkage capacity of $H \sim \mathcal{G}(k, p)$ is at least $\Omega\left(\frac{k \log(kp)}{\log k}\right)$ with high probability.*

Proof. Let r be the bound specified in Theorem 5.4 and consider the blowup graph $H \otimes J_{2r}$. Let $X := \{v^{(1)} \mid v \in V(H)\}$. We show that X is matching-linked in $H \otimes J_{2r}$ with high probability, and the lemma then follows using Lemma 3.7.

Let M' be a matching on X . By the definition of X , its H -projection, $M := \pi(M')$, is also a matching on $V(H)$. We invoke Theorem 5.4 on the graph H with respect to the matching M to obtain a partition M_1, \dots, M_r , such that, for all $i \in [r]$ there is an uncongested M_i -linkage Q_i in H . Then $Q = \bigcup_{i \in [r]} Q_i$ is an r -congested M -linkage in H . So there is an uncongested M' -linkage in $H \otimes J_{2r}$ by Lemma 2.5. ◀

Now, Theorem 1.4 follows from Theorem 1.2 and Lemma 5.5.

6 Large-Treewidth Patterns and Concurrent Flows

In this section, we relate the linkage capacity of a graph to its treewidth. Towards this end, we first connect the linkage capacity to certain (fractional) multicommodity flows, and afterward rely on existing connections between such flows and treewidth [59, Section 3.1].

More specifically, given a graph H and $W \subseteq V(H)$, we consider the following multicommodity flow problem. For every pair $(u, v) \in W^2$, there is a distinct commodity uv that can be sent in arbitrary fractional amounts along different paths from u to v in H . The goal is to determine whether all pairs (u, v) can concurrently send an ϵ amount of uv to each other, while the total flow through every vertex $w \in V(H)$ is at most some globally fixed capacity C . Formally, this is captured by the following LP:

► **Definition 6.1.** *Let H be a graph. For $u, v \in V(H)$, write $\mathcal{P}_H(uv)$ for the set of paths from u to v in H ; the set $\mathcal{P}_H(uv)$ for $u = v$ contains only the path (u) . Given $W \subseteq V(H)$, the concurrent flow LP (for H and W) with vertex capacity $C > 0$ asks to*

$$\begin{aligned} & \text{maximize } \epsilon \\ & \text{subject to} \quad \sum_{p \in \mathcal{P}_H(uv)} x_p \geq \epsilon \quad \forall u, v \in W \\ & \quad \sum_{u, v \in W} \sum_{p \in \mathcal{P}_H(uv): w \in p} x_p \leq C \quad \forall w \in V(G) \\ & \quad x_p \geq 0 \quad \forall u, v \in W, p \in \mathcal{P}_H(uv). \end{aligned}$$

We write $\epsilon(H, W)$ to denote the optimal LP value for capacity $C = 1$.

While an optimal solution for $C = 1$ may assign fractional values to the variables x_p , every solution can be scaled to an integral solution, increasing the required capacity and the optimal LP value by the same factor. This integral solution then induces a congested model of the multigraph $K_{t,(q)}$ in H , where $t := |W|$ and $q \in \mathbb{N}$ is suitably chosen, and $K_{t,(q)}$ has t vertices and contains each possible (undirected) edge with multiplicity q .

► **Lemma 6.2.** *Let H be a graph and $W \subseteq V(H)$ be a set of size t . Then there is some $D \in \mathbb{N}$ such that $q := D \cdot \epsilon(H, W)$ is an integer and H contains a D -congested $K_{t,(q)}$ -linkage, where we set $V(K_{t,(q)}) = W$.*

Proof. Let D be the common denominator of the values for all x_p in a (rational) optimal solution of the concurrent flow LP for H and W with capacity $C = 1$. Then the LP with capacity D has an integral solution of value $q := D \cdot \epsilon(H, W)$. Now, consider the multiset Q which, for every distinct $u, v \in W$, contains every path $p \in \mathcal{P}_H(uv)$ with multiplicity x_p . Then Q is a D -congested $K_{t,(q)}$ -linkage where $V(K_{t,(q)}) = W$. ◀

Using this congested $K_{t,(q)}$ -linkage, we can establish lower bounds on the linkage capacity of H . Here, the following lemma is useful, since it allows us to route arbitrary multigraphs of bounded degree via short paths in this $K_{t,(q)}$.

► **Lemma 6.3.** *Let $q \in \mathbb{N}$ and let M be a multigraph with $V(M) = [t]$ and maximum degree at most qt . Then there is an M -linkage $Q = (P_{uv})_{uv \in E(M)}$ in K_t such that every edge $e \in E(K_t)$ appears in at most $18q$ paths in Q .*

We can conclude that a large value of the concurrent flow LP implies large linkage capacity.

► **Theorem 6.4.** *Let H be a graph and $W \subseteq V(H)$. Then $\gamma(H) \geq \epsilon(H, W) \cdot |W|^2/108$.*

Proof. Let $D \in \mathbb{N}$ be the integer obtained from Lemma 6.2 and set $D' := 18 \cdot D$. Let $q := D \cdot \epsilon(H, W)$ which, by Lemma 6.2, is an integer. Observe that $\epsilon(H, W) \leq 1/|W|$, so $D \geq q \cdot |W|$. Finally, let $s := q \cdot |W|$.

28:18 Can You Link Up With Treewidth?

Consider the graph $H \otimes J_{2D'}$ and let $X := \{w^{(i)} \mid w \in W, i \in [s]\}$. We show that X is matching-linked in $H \otimes J_{2D'}$. Let M be a matching on X . Let $\widehat{M} := \pi(M)$ be the H -projection of M . Observe that $\deg_{\widehat{M}}(w) \leq s = q \cdot |W|$ for all $w \in W$.

Lemma 6.3 finds an \widehat{M} -linkage $Q = (P_{uv})_{uv \in E(\widehat{M})}$ in $K_{t,(q)}$ (where $V(K_{t,(q)}) = W$) such that every edge of $K_{t,(q)}$ appears in at most 18 of those paths. Moreover, by Lemma 6.2, the graph H contains a D -congested $K_{t,(q)}$ -linkage $Q' = (P'_{uv})_{uv \in E(K_{t,(q)})}$. We construct a D' -congested \widehat{M} -linkage $\widehat{Q} = (\widehat{P}_{uv})_{uv \in E(\widehat{M})}$ in H as follows. For every $uv \in E(\widehat{M})$ we obtain \widehat{P}_{uv} from P_{uv} by substituting P'_e for every edge e appearing on P_{uv} . Clearly, \widehat{Q} is D' -congested since $D' = 18 \cdot D$. So there is an uncongested M -linkage in $H \otimes J_{2D'}$ by Lemma 2.5.

Overall, we get that X is matching-linked in $H \otimes J_{2D'}$. So

$$\gamma(G) \geq \frac{1}{3} \cdot \frac{|X|}{2 \cdot D'} = \frac{1}{3} \cdot \frac{s \cdot |W|}{36 \cdot D} = \frac{1}{108} \cdot \frac{q \cdot |W|^2}{D} = \frac{1}{108} \cdot \varepsilon(H, W) \cdot |W|^2$$

by Lemma 3.7. ◀

To bound the linkage capacity by the treewidth, we combine Theorem 6.4 with the following lemma that is (implicitly) shown by Marx [59].

► **Lemma 6.5** ([59]). *Let H be a graph of treewidth t . Then there is a set $W \subseteq V(H)$ such that $|W| = t$ and $\varepsilon(H, W) = \Omega(1/(t \log t))$.*

The basic idea to prove the lemma is to consider a set $W \subseteq V(H)$ of size t that does not admit balanced separators; large treewidth guarantees such a set (see [59, Lemma 3.2]). Then, using results from [37, 55], we obtain a bound on the optimal value of the dual LP, which gives $\varepsilon(H, W) = \Omega(1/(t \log t))$ (see [59, Proof of Lemma 3.6]).

► **Corollary 6.6.** *Let H be a graph of treewidth t . Then $\gamma(H) = \Omega(t/\log t)$.*

In particular, combining Theorem 1.2 and Corollary 6.6 allows us to recover the complexity lower bounds on $\text{COLSUB}(H)$ proved in [59].

We note without a proof that the bound in Corollary 6.6 is asymptotically optimal, since k -vertex 3-regular expander graphs have treewidth $\Theta(k)$ and linkage capacity $\Theta(k/\log k)$ (see also [4]). We complement Corollary 6.6 with the following upper bound on the linkage capacity.

► **Lemma 6.7.** *Let H be a graph of treewidth t . Then $\gamma(H) \leq 3(t+1)$.*

Proof. We first observe that $\text{tw}(H \otimes J_q) \leq q(t+1) - 1$ for every $q \in \mathbb{N}$. Indeed, given a tree decomposition (T, β) of H , we can obtain a tree decomposition for $H \otimes J_q$ by adding all copies of $v \in V(H)$ to all bags containing v .

Now, suppose towards a contradiction that $\gamma(H) > 3(t+1)$. Then there is some $q \in \mathbb{N}$ such that $H \otimes J_q$ contains a matching-linked set $X \subseteq V(H \otimes J_q)$ such that $|X| = 3q(t+1) + 3$. By [26, Lemma 7.20] there is a *balanced separation* for X , i.e., there are sets $A, B \subseteq V(H \otimes J_q)$ such that (1) $A \cup B = V(H \otimes J_q)$, (2) $|A \cap B| \leq q(t+1)$, (3) there is no edge between $A \setminus B$ and $B \setminus A$, and (4) $|X \cap A| \leq \frac{2}{3}|X|$ and $|X \cap B| \leq \frac{2}{3}|X|$. Hence, $|X \setminus A| \geq \frac{1}{3}|X| = q(t+1) + 1$ and $|X \setminus B| \geq \frac{1}{3}|X| = q(t+1) + 1$. Observe that the two sets $X \setminus A$ and $X \setminus B$ are disjoint (since $A \cup B = V(H \otimes J_q)$). So there is a matching M on the vertex set X (but not necessarily in $H \otimes J_q$) containing $q(t+1) + 1$ edges $M' \subseteq M$ with one endpoint in $X \setminus A$ and the other in $X \setminus B$. Now consider an uncongested M -linkage in $H \otimes J_q$ (which exists since X is matching-linked). Then every edge $e \in M'$ is realized by a path that needs to visit a vertex from $A \cap B$. However, this is a contradiction since $|M'| = q(t+1) + 1$ and $|A \cap B| \leq q(t+1)$. ◀

The upper bound is also asymptotically optimal by Lemma 3.9.

7 Implications for Counting Small Induced Subgraphs

We conclude with an application of our lower bounds for the complexity of counting induced k -vertex subgraphs. A k -vertex graph invariant Φ is an isomorphism-invariant map from k -vertex graphs H to the real numbers. We consider Φ to be fixed and wish to sum $\Phi(G[X])$ over all k -vertex subsets X of an input graph G to count, e.g., the planar or Hamiltonian induced k -vertex subgraphs of G . Formally, for a k -vertex graph invariant Φ , the problem $\#\text{INDSUB}(\Phi)$ takes as input a graph G , and asks to compute

$$\#\text{IndSub}(\Phi \rightarrow G) := \sum_{X \subseteq V(G)} \Phi(G[X]).$$

This problem was first studied in its parameterized version (where k is part of the input) by Jerrum and Meeks [46, 47, 48] and received significant attention in recent years [23, 24, 31, 32, 33, 38, 63, 64].

To determine the complexity of $\#\text{INDSUB}(\Phi)$, recent works usually analyze the *alternating enumerator* to build a generic reduction from $\#\text{COLSUB}(H)$. Formally, the *alternating enumerator* of a graph invariant Φ on a graph H is defined as⁵

$$\widehat{\Phi}(H) = (-1)^{|E(H)|} \sum_{S \subseteq E(H)} (-1)^{|S|} \Phi(H[S]),$$

where $H[S]$ has vertex set $V(H)$ and edge set S . Now, suppose H is a k -vertex graph with $\widehat{\Phi}(H) \neq 0$. Then the problem $\#\text{COLSUB}(H)$ can be reduced to $\#\text{INDSUB}(\Phi)$ in polynomial time (see, e.g., [24]). Hence, we also obtain new lower bounds for $\#\text{INDSUB}(\Phi)$ assuming $\widehat{\Phi}(H) \neq 0$ for suitable graphs H . In particular, we obtain the following result via Theorem 1.3, which improves over the corresponding lower bound in [24, Theorem 3.2(a)] (see also [31] and [32, Lemma 2.2]).

► **Theorem 7.1.** *There is a universal constant $\alpha_{\text{IND}} > 0$ and an integer $N_0 \geq 1$ such that for all numbers $k, \ell \geq 1$, the following holds: If Φ is a k -vertex graph invariant and there exists a graph H with $\widehat{\Phi}(H) \neq 0$ and $E(H) \geq k \cdot \ell \geq N_0$, then $\#\text{INDSUB}(\Phi)$ cannot be solved in time $O(n^{\alpha_{\text{IND}} \cdot \ell})$ unless ETH fails.*

As pointed out in Section 1.2, the weaker version, which only rules out an exponent of $\alpha_{\text{IND}} \cdot \ell / \sqrt{\log \ell}$, has been used to derive various lower bounds for specific types of invariants in [24, 32, 64]. All these lower bounds are improved by our new results. Let us give one concrete example, which improves over [24, Corollary 5.1]. For a k -vertex graph invariant Φ , we write $\text{supp}(\Phi)$ for the set of all graphs H with $V(H) = [k]$ and $\Phi(H) \neq 0$.

► **Corollary 7.2.** *For every $0 < \varepsilon < 1$ there are $N_0, \delta > 0$ such that the following holds. Let $k \geq N_0$ and let Φ be a k -vertex graph invariant with $1 \leq |\text{supp}(\Phi)| \leq (2 - \varepsilon)^{\binom{k}{2}}$. Then no algorithm solves $\#\text{INDSUB}(\Phi)$ in time $O(n^{\delta \cdot k})$ unless ETH fails.*

We stress that the exponent in the lower bound of Corollary 7.2 is asymptotically optimal.

For the other implications of Theorem 7.1, we refer the reader to the latest arXiv version of [24]. Indeed, following the first publication of this work, the latest version of [24] contains updated lower bounds for $\#\text{INDSUB}(\Phi)$ based on Theorem 7.1. However, let us stress that these improved lower bounds should (at least in part) be attributed to this work.

⁵ The precise formula is not relevant here, but we still give it for completeness.

References

- 1 Akanksha Agrawal, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Simultaneous feedback edge set: A parameterized perspective. *Algorithmica*, 83(2):753–774, 2021. doi:10.1007/S00453-020-00773-9.
- 2 Noga Alon. Explicit expanders of every degree and size. *Comb.*, 41(4):447–463, 2021. doi:10.1007/S00493-020-4429-X.
- 3 Noga Alon, Michael R. Capalbo, Yoshiharu Kohayakawa, Vojtech Rödl, Andrzej Rucinski, and Endre Szemerédi. Universality and tolerance. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 14–21. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892007.
- 4 Noga Alon and Dániel Marx. Sparse balanced partitions and the complexity of subgraph problems. *SIAM J. Discret. Math.*, 25(2):631–644, 2011. doi:10.1137/100812653.
- 5 Saeed A. Amiri, Stephan Kreutzer, Dániel Marx, and Roman Rabinovich. Routing with congestion in acyclic digraphs. *Inf. Process. Lett.*, 151, 2019. doi:10.1016/J.IPL.2019.105836.
- 6 Václav E. Beneš. Permutation groups, complexes, and rearrangeable connecting networks. *Bell System Tech. J.*, 43(4):1619–1640, 1964. doi:10.1002/j.1538-7305.1964.tb04102.x.
- 7 Béla Bollobás. The diameter of random graphs. *Trans. Amer. Math. Soc.*, 267(1):41–52, 1981. doi:10.2307/1998567.
- 8 Édouard Bonnet, Sergio Cabello, Bojan Mohar, and Hebert Pérez-Rosés. The inverse voronoi problem in graphs I: hardness. *Algorithmica*, 82(10):3018–3040, 2020. doi:10.1007/S00453-020-00716-4.
- 9 Edouard Bonnet, Panos Giannopoulos, and Michael Lampis. On the parameterized complexity of red-blue points separation. *J. Comput. Geom.*, 10(1):181–206, 2019. doi:10.20382/JOCG.V10I1A7.
- 10 Édouard Bonnet, Yoichi Iwata, Bart M. P. Jansen, and Lukasz Kowalik. Fine-grained complexity of k-OPT in bounded-degree graphs for solving TSP. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPICs*, pages 23:1–23:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ESA.2019.23.
- 11 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. *ACM Trans. Algorithms*, 16(4):42:1–42:23, 2020. doi:10.1145/3398684.
- 12 Édouard Bonnet and Florian Sikora. The graph motif problem parameterized by the structure of the input graph. *Discret. Appl. Math.*, 231:78–94, 2017. doi:10.1016/J.DAM.2016.11.016.
- 13 Karl Bringmann. Fine-grained complexity theory (tutorial). In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPICs*, pages 4:1–4:7. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.STACS.2019.4.
- 14 Karl Bringmann, László Kozma, Shay Moran, and N. S. Narayanaswamy. Hitting set for hypergraphs of low VC-dimension. 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 23:1–23:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.23.
- 15 Andrei Z. Broder, Alan M. Frieze, Stephen Suen, and Eli Upfal. An efficient algorithm for the vertex-disjoint paths problem in random graphs. In Éva Tardos, editor, *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 28-30 January 1996, Atlanta, Georgia, USA*, pages 261–268. ACM/SIAM, 1996. URL: <http://dl.acm.org/citation.cfm?id=313852.314072>.

- 16 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/J.IC.2005.05.001.
- 17 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/J.JCSS.2006.04.007.
- 18 Rajesh Chitnis, Andreas E. Feldmann, and Pasin Manurangsi. Parameterized approximation algorithms for bidirected Steiner network problems. *ACM Trans. Algorithms*, 17(2):12:1–12:68, 2021. doi:10.1145/3447584.
- 19 Rajesh Hemant Chitnis, Andreas E. Feldmann, Mohammad T. Hajiaghayi, and Dániel Marx. Tight bounds for planar strongly connected Steiner subgraph with fixed number of terminals (and extensions). *SIAM J. Comput.*, 49(2):318–364, 2020. doi:10.1137/18M122371X.
- 20 Vincent Cohen-Addad, Éric C. de Verdière, Dániel Marx, and Arnaud de Mesmay. Almost tight lower bounds for hard cutting problems in embedded graphs. *J. ACM*, 68(4):30:1–30:26, 2021. doi:10.1145/3450704.
- 21 Jason Crampton, Robert Crowston, Gregory Z. Gutin, Mark Jones, and Maadapuzhi S. Ramanujan. Fixed-parameter tractability of workflow satisfiability in the presence of seniority constraints. In Michael R. Fellows, Xuehou Tan, and Binhai Zhu, editors, *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, Third Joint International Conference, FAW-AAIM 2013, Dalian, China, June 26-28, 2013. Proceedings*, volume 7924 of *Lecture Notes in Computer Science*, pages 198–209. Springer, 2013. doi:10.1007/978-3-642-38756-2_21.
- 22 Radu Curticapean, Holger Dell, and Thore Husfeldt. Modular counting of subgraphs: Matchings, matching-splittable graphs, and paths. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPICs*, pages 34:1–34:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ESA.2021.34.
- 23 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. 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 210–223. ACM, 2017. doi:10.1145/3055399.3055502.
- 24 Radu Curticapean and Daniel Neuen. Counting small induced subgraphs: Hardness via Fourier analysis. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3677–3695. SIAM, 2025. doi:10.1137/1.9781611978322.122.
- 25 Radu Curticapean and Mingji Xia. Parameterizing the permanent: Genus, apices, minors, evaluation mod $2k$. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 994–1009. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.65.
- 26 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.
- 27 Mina Dalirrooyfard and Virginia Vassilevska Williams. Induced cycles and paths are harder than you think. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 531–542. IEEE, 2022. doi:10.1109/FOCS54457.2022.00057.
- 28 Argyrios Deligkas, Eduard Eiben, and Tiger-Lily Goldsmith. Parameterized complexity of hotelling-downs with party nominees. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 244–250. ijcai.org, 2022. doi:10.24963/IJCAI.2022/35.

- 29 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the tutte polynomial. *ACM Trans. Algorithms*, 10(4):21:1–21:32, 2014. doi:10.1145/2635812.
- 30 Reinhard Diestel. *Graph Theory*. Springer Berlin, 5 edition, 2017. doi:10.1007/978-3-662-53622-3.
- 31 Julian Dörfler, Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting induced subgraphs: An algebraic approach to $\#W[1]$ -hardness. *Algorithmica*, 84(2):379–404, 2022. doi:10.1007/S00453-021-00894-9.
- 32 Simon Döring, Dániel Marx, and Philip Wellnitz. Counting small induced subgraphs with edge-monotone properties. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1517–1525. ACM, 2024. doi:10.1145/3618260.3649644.
- 33 Simon Döring, Dániel Marx, and Philip Wellnitz. From graph properties to graph parameters: Tight bounds for counting on small subgraphs. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 3637–3676. SIAM, 2025. doi:10.1137/1.9781611978322.121.
- 34 Eduard Eiben, Gregory Z. Gutin, Philip R. Neary, Clément Rambdaud, Magnus Wahlström, and Anders Ye. Preference swaps for the stable matching problem. *Theor. Comput. Sci.*, 940(Part):222–230, 2023. doi:10.1016/J.TCS.2022.11.003.
- 35 Eduard Eiben, Dusan Knop, Fahad Panolan, and Ondrej Suchý. Complexity of the Steiner network problem with respect to the number of terminals. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPICs*, pages 25:1–25:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.STACS.2019.25.
- 36 David Eppstein and Daniel Lokshtanov. The parameterized complexity of finding point sets with hereditary properties. In Christophe Paul and Michal Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation, IPEC 2018, August 20-24, 2018, Helsinki, Finland*, volume 115 of *LIPICs*, pages 11:1–11:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.IPEC.2018.11.
- 37 Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008. doi:10.1137/05064299X.
- 38 Jacob Focke and Marc Roth. Counting small induced subgraphs with hereditary properties. *SIAM J. Comput.*, 53(2):189–220, 2024. doi:10.1137/22M1512211.
- 39 Fedor V. Fomin, Fahad Panolan, Maadapuzhi S. Ramanujan, and Saket Saurabh. On the optimality of pseudo-polynomial algorithms for integer programming. *Math. Program.*, 198(1):561–593, 2023. doi:10.1007/S10107-022-01783-X.
- 40 Robert Ganian. Using neighborhood diversity to solve hard problems. *CoRR*, abs/1201.3091, 2012. arXiv:1201.3091.
- 41 Mika Göös, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM J. Comput.*, 47(1):241–269, 2018. doi:10.1137/16M109884X.
- 42 Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013. doi:10.1007/S00453-012-9685-8.
- 43 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/JCSS.2000.1727.
- 44 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.


- 45 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.
- 46 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015. doi:10.1016/J.JCSS.2014.11.015.
- 47 Mark Jerrum and Kitty Meeks. Some hard families of parameterized counting problems. *ACM Trans. Comput. Theory*, 7(3):11:1–11:18, 2015. doi:10.1145/2786017.
- 48 Mark Jerrum and Kitty Meeks. The parameterised complexity of counting even and odd induced subgraphs. *Comb.*, 37(5):965–990, 2017. doi:10.1007/S00493-016-3338-5.
- 49 Mark Jones, Daniel Lokshtanov, Maadapuzhi S. Ramanujan, Saket Saurabh, and Ondrej Suchý. Parameterized complexity of directed Steiner tree on sparse graphs. *SIAM J. Discret. Math.*, 31(2):1294–1327, 2017. doi:10.1137/15M103618X.
- 50 Karthik C. Srikanta, Dániel Marx, Marcin Pilipczuk, and Uéverton S. Souza. Conditional lower bounds for sparse parameterized 2-CSP: A streamlined proof. In Merav Parter and Seth Pettie, editors, *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 383–395. SIAM, 2024. doi:10.1137/1.9781611977936.35.
- 51 Victor Klee and David Larman. Diameters of random graphs. *Canadian J. Math.*, 33(3):618–640, 1981. doi:10.4153/CJM-1981-050-1.
- 52 Dusan Knop, Simon Schierreich, and Ondrej Suchý. Balancing the spread of two opinions in sparse social networks (student abstract). In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12987–12988. AAAI Press, 2022. doi:10.1609/AAAI.V36I11.21630.
- 53 Alexandr V. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Comb.*, 4(4):307–316, 1984. doi:10.1007/BF02579141.
- 54 Eric Lehman, F. Thomson Leighton, and Albert R. Meyer. Beneš Network, june 30 2021. [Online; accessed 2024-09-26]. URL: <https://eng.libretexts.org/@go/page/48364>.
- 55 Frank T. Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. doi:10.1145/331524.331526.
- 56 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: <http://eatcs.org/beatcs/index.php/beatcs/article/view/92>.
- 57 Daniel Lokshtanov, Maadapuzhi S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. 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 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.
- 58 Wolfgang Mader. Existenz n -fach zusammenhängender Teilgraphen in Graphen genügend großer Kantendichte. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 37:86–97, 1972. doi:10.1007/BF02993903.
- 59 Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.
- 60 Jesper Nederlof and Céline M. F. Swennenhuis. On the fine-grained parameterized complexity of partial scheduling to minimize the makespan. *Algorithmica*, 84(8):2309–2334, 2022. doi:10.1007/S00453-022-00970-8.
- 61 Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Technical report, KTH Royal Institute of Technology, 2015.
- 62 Marcin Pilipczuk and Magnus Wahlström. Directed multicut is $W[1]$ -hard, even for four terminal pairs. *ACM Trans. Comput. Theory*, 10(3):13:1–13:18, 2018. doi:10.1145/3201775.

- 63 Marc Roth and Johannes Schmitt. Counting induced subgraphs: A topological approach to $\#W[1]$ -hardness. *Algorithmica*, 82(8):2267–2291, 2020. doi:10.1007/S00453-020-00676-9.
- 64 Marc Roth, Johannes Schmitt, and Philip Wellnitz. Counting small induced subgraphs satisfying monotone properties. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1356–1367. IEEE, 2020. doi:10.1109/FOCS46700.2020.00128.
- 65 Claude E. Shannon. A theorem on coloring the lines of a network. *J. Math. Physics*, 28(1-4):148–152, 1949. doi:10.1002/sapm1949281148.
- 66 Robin Thomas and Paul Wollan. An improved linear edge bound for graph linkages. *Eur. J. Comb.*, 26(3-4):309–324, 2005. doi:10.1016/J.EJC.2004.02.013.
- 67 Virginia V. Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPICs*, pages 17–29. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICS.IPEC.2015.17.
- 68 Virginia V. Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians – Rio de Janeiro 2018. Vol. IV. Invited lectures*, pages 3447–3487. World Sci. Publ., Hackensack, NJ, 2018. doi:10.1142/9789813272880_0188.

Noisy (Binary) Searching: Simple, Fast and Correct

Dariusz Dereniowski ✉ 

Faculty of Electronics, Telecommunications and Informatics,
Gdańsk University of Technology, Poland

Aleksander Łukasiewicz ✉ 

Institute of Computer Science, University of Wrocław, Poland
Computer Science Institute of Charles University, Prague, Czech Republic

Przemysław Uznański ✉ 

Institute of Computer Science, University of Wrocław, Poland

Abstract

This work considers the problem of the noisy binary search in a sorted array. The noise is modeled by a parameter p that dictates that a comparison can be incorrect with probability p , independently of other queries. We state two types of upper bounds on the number of queries: the worst-case and expected query complexity scenarios. The bounds improve the ones known to date, i.e., our algorithms require fewer queries. Additionally, they have simpler statements, and work for the full range of parameters. All query complexities for the expected query scenarios are tight up to lower order terms. For the problem where the target prior is uniform over all possible inputs, we provide an algorithm with expected complexity upperbounded by $(\log_2 n + \log_2 \delta^{-1} + 3)/I(p)$, where n is the domain size, $0 \leq p < 1/2$ is the noise ratio, and $\delta > 0$ is the failure probability, and $I(p)$ is the information gain function. As a side-effect, we close some correctness issues regarding previous work. Also, en route, we obtain new and improved query complexities for the search generalized to arbitrary graphs. This paper continues and improves the lines of research of Burnashev–Zigangirov [Prob. Per. Informatsii, 1974], Ben-Or and Hassidim [FOCS 2008], Gu and Xu [STOC 2023], and Emamjomeh-Zadeh et al. [STOC 2016], Dereniowski et al. [SOSA@SODA 2019].

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

Keywords and phrases Graph Algorithms, Noisy Binary Search, Query Complexity, Reliability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.29

Related Version *Full Version*: <https://arxiv.org/abs/2107.05753>

Funding *Dariusz Dereniowski*: Partially supported by National Science Centre (Poland) grant number 2018/31/B/ST6/00820.

Aleksander Łukasiewicz: Partially supported by the ERC-CZ project LL2406 of the Ministry of Education of Czech Republic.

Przemysław Uznański: Partially supported by National Science Centre (Poland) grant number 2018/31/B/ST6/00820.

1 Introduction

1.1 Problem statement

An adaptive search problem for a general *search domain* \mathcal{S} and an arbitrary *adversary* can be formulated as follows. The goal is to design an adaptive algorithm, also referred to as a *strategy*, that finds a *target* initially unknown to the algorithm. Adaptivity means that the subsequent actions of the algorithm depend on the answers already received. The process is divided into *steps*: in each step the algorithm performs a *query* and receives an *answer*. Each query-reply pair provides new information to the algorithm: it learns that some part of the search space $S \subseteq \mathcal{S}$ does not contain the target while its complement does. From both



© Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 29; pp. 29:1–29:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



theoretical and practical viewpoints, it is of interest to develop error-resilient algorithms for such a search process. This can be modeled, for example, by the presence of a probabilistic noise: each reply can be erroneous with some fixed probability $0 < p < \frac{1}{2}$, independently. The performance of a strategy is measured by the number of performed queries.

In this work, we focus on searching with probabilistic noise in two particular types of search domains. The first is a sorted array (which, in the absence of noise, would lead to the classical binary search problem). Formally, for a linear order $v_1 < \dots < v_n$ with an unknown position of the target $v^* = v_j$, each query selects an element v_i , and the algorithm learns from the reply whether $v^* < v_i$ or $v^* \geq v_i$.

The second search domain we consider is a simple, undirected graph. More precisely, for an input graph G and an unknown target vertex v^* , each query selects some vertex v . The answer either states that v is the target or provides a neighbor u of v , that lies on a shortest path from v to v^* .

Searching through a graph can be viewed as a certain generalization of the former setting, as searching a linear order resembles searching a graph that is a path. However, it is important to note that the two models are not directly comparable, as in a graph search on a path there are three possible replies to a query, whereas in a search through an array, the answers are binary.

1.2 Overview of our results

To complete the search process, we need to learn roughly $\log_2 n$ bits of information (the identifier of the target). We can extract approximately $I(p) = 1 - H(p)$ bits of information from each reply, where $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$ is the binary entropy function. Therefore we expect the optimal algorithm to use around $\frac{\log_2 n}{I(p)}$ queries. In an idealized scenario where there always exists a query that perfectly bisects the search space, regardless of the answer, one could achieve this bound. However, perfect bisection is typically not possible due to the discrete nature of the search space, which causes algorithms to lose some lower-order terms.

Our results are summarized in Table 1. We use the following naming convention: if we are interested in the worst-case analysis of the query complexity, we refer to it as the *worst case* setting throughout the paper. Conversely, if we analyze the expected number of queries made by an algorithm, we call this the *expected query complexity* setting. We note that in each setting the process is randomized due to answers of the adversary. Additionally, some of our algorithms use random bits as well.

The binary search algorithms referenced in the theorems below are detailed in Section 3: Algorithms 16 and 13 correspond to Theorems 1 and 3, respectively. Interestingly, both algorithms are essentially the same and differ only by the stopping condition.

► **Theorem 1.** *For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm that after the **expected** number of*

$$\frac{1}{I(p)} (\log_2 n + \log_2 \delta^{-1} + 3)$$

queries finds the target in any linear order correctly with probability at least $1 - \delta$, given that the target position is chosen uniformly at random.

Using a previously known reduction from adversarial target placement to the uniformly random choice of a target ([3], see Lemma 10), we automatically obtain the following result for the adversarial version of the problem.

► **Corollary 2.** For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm that after the **expected** number of

$$\frac{1}{I(p)} (\log_2 n + \mathcal{O}(\log \delta^{-1}))$$

queries finds the target in any linear order correctly with probability at least $1 - \delta$.

► **Theorem 3.** For any noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists a binary search algorithm for any linear order that after

$$\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}) \right)$$

queries returns the target correctly with probability at least $1 - \delta$.

For graph searching we obtain two analogous results (detailed in Section 4): Algorithms 23 and 25 correspond to Theorems 5 and 4, respectively.

► **Theorem 4.** For an arbitrary connected graph G , a noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists an graph searching algorithm that after the **expected** number of at most

$$\frac{1}{I(p)} (\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$$

queries returns the target correctly with probability at least $1 - \delta$.

► **Theorem 5.** For an arbitrary connected graph G , a noise parameter $0 < p < \frac{1}{2}$ and a confidence threshold $0 < \delta < 1$, there exists an graph searching algorithm that after

$$\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}) \right)$$

queries returns the target correctly with probability at least $1 - \delta$.

■ **Table 1** The summary of the results.

Setting	Binary search	Graph search
Worst case query complexity:	$\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}) \right)$ (Thm. 3)	$\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}) \right)$ (Thm. 5)
Expected query complexity:	$\frac{1}{I(p)} (\log_2 n + \mathcal{O}(\log \delta^{-1}))$ (Cor. 2)	$\frac{1}{I(p)} (\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$ (Thm. 4)

1.3 Comparison with previous works

1.3.1 Upper bounds for noisy binary search

Table 2 provides an overview of the known algorithms for noisy binary search that have complexity close to the optimal $\frac{\log_2 n}{I(p)}$. We provide a detailed comparison with our results below.

- Burnashev and Zigangirov [7] studied this problem from an information-theoretic perspective as early as 1974¹, in a setting where the location of the target element is chosen uniformly at random. We highlight that their query complexity is worse than ours by an additive term of $\log_2 \frac{1-p}{p}$, which tends to infinity when $p \rightarrow 0$. This behavior is rather unnatural, since when $p = 0$, the noise disappears, and the problem reduces to standard binary search.
- Feige et al. [15], in their seminal work, considered several problems in the noisy setting and, in particular, developed an asymptotically optimal algorithm for noisy binary search (with an adversarially placed target). However, their method intrinsically incurs a non-optimal constant in front of $\frac{\log_2 n}{I(p)}$.
- Later, Ben-Or and Hassidim [3], likely unaware of [7], developed algorithms with an expected query complexity of $\frac{1}{I(p)}(\mathcal{O}(\log n) + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$. However, we claim that their proofs contain two *serious issues*.

Firstly, in the proof of Lemma 2.6 in [3], they consider all the queries made by the algorithm throughout its execution and sort them by their positions. Then, the number of ' $<$ ' answers in a fixed interval of positions is claimed to follow a binomial distribution. Notice, however, that while the answers to the particular queries are independent random variables, the *positions* of the queries depend on the answers to the previous queries, and the act of forgetting the order introduces correlation. To further illustrate this point, we note the rightmost query in their algorithm is guaranteed to have a ' $<$ ' answer, because a ' \geq ' answer would have changed the weights maintained by the algorithm, causing the next query to be asked further to the right, which contradicts the assumption that this query is the rightmost.

Secondly, the final expected number of steps is bounded by the ratio of total information needed to identify the target and the expected information gain per step (without any additional comments). However, the expected value does not work in this manner directly – to make this approach effective, one needs to employ additional probabilistic tools. Our paper uses Wald's identity for this purpose, see [19] for an example of the usage of martingales and the Optional Stopping Theorem. Moreover, the choice of a particular probabilistic theorem and the way this tool is handled may incur additional lower-order terms, making it unclear what the final complexity would be.

1.3.2 Reductions in complexity for expected length setting

Ben-Or and Hassidim in their work [3] showed a general technique that can transform any of the aforementioned algorithms (regardless if they are in the worst case or expected complexity setting) into an algorithm with the expected query complexity that is better by roughly a multiplicative factor of $(1 - \delta)$ at the cost of additive lower order term of order $\frac{\mathcal{O}(\log \log n)}{I(p)}$. Very recently Gu and Xu [19] showed how to improve that reduction in order to obtain a better constant in front of $\log \delta^{-1}$. They plug in our algorithm for noisy binary search (Corollary 2) as a black-box in order to get the $(1 + o(1))((1 - \delta)(\frac{\log_2 n}{I(p)} + \frac{\mathcal{O}(\log \log n)}{I(p)}) + \frac{\log_2 1/\delta}{(1-2p) \log \frac{1-p}{p}})$ expected query complexity.

¹ Curiously, it appears that until recently, this work has been largely unknown to the algorithmic community, despite the fact that the paper in question has over 100 citations. There is no mention of [7] in the well-known survey by Pelc [27], nor in the subsequent works that we reference. We suspect that the main reason for this oversight is that, until very recently, the work was only available in Russian. For the English translation of the algorithm and the proof, see [34].

■ **Table 2** Upper bounds for noisy binary search.

<i>Setting</i>	<i>Query complexity</i>	<i>References and Notes</i>
Expected, uniform prior	$\frac{1}{I(p)}(\log_2 n + \log_2 \delta^{-1} + \log_2 \frac{1-p}{p})$	Burnashev and Zigangirov [7]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$	Ben-Or and Hassidim [3] (Correctness issues, see Sec.1.3.1).
	$\frac{1}{I(p)}(\log_2 n + \log_2 \delta^{-1} + 3)$	This work, Thm. 1.
Expected, adversarial target	$\frac{1}{I(p)}(\mathcal{O}(\log n) + \mathcal{O}(\log \delta^{-1}))$	Feige et al. [15]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$	Ben-Or and Hassidim [3] (Correctness issues, see Sec.1.3.1).
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log_2 \delta^{-1}))$	This work, Cor. 2.
Worst case	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 3.

1.3.3 Upper bounds for noisy graph search

Known algorithms for noisy graph search are summarized in Table 3. The problem for arbitrary graphs was first considered by Emamjomeh-Zadeh et al.[14]. Later on Dereniowski et al. [11] simplified that algorithm and obtained an improved dependence on $\log \delta^{-1}$. We make another progress in that direction: we further simplify the algorithms and the analysis while simultaneously improving the dependence on $\log \delta^{-1}$ even further.

■ **Table 3** Upper bounds for noisy graph search.

<i>Setting</i>	<i>Query complexity</i>	<i>References and Notes</i>
Expected	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\log \log n) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 4.
Worst case	$\frac{1}{I(p)}(\log_2 n + o(\log n) + \mathcal{O}(\log^2 \delta^{-1}))$	Emamjomeh-Zadeh et al., 2016 [14]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}} \cdot \log \frac{\log n}{\log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	Dereniowski et al., 2019 [11]
	$\frac{1}{I(p)}(\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1}))$	This work, Thm. 5.

1.3.4 Lower bounds

For the expected complexity of noisy binary search, Ben-Or and Hassidim [3] established the first lower bound of $(1 - \delta) \frac{\log_2 n - 10}{I(p)}$. Recently Gu and Xu [19] improved the lower bound to $(1 - o(1))((1 - \delta) \frac{\log_2 n}{I(p)} + \frac{\log_2 \frac{1}{\delta}}{(1-2p) \log \frac{1-p}{p}})$. However it works only for constant noise parameter p . They leave the question of improving the lower bound for an arbitrary p as an open problem.

Very recently, Gretta and Price [18] obtained a lower bound for the worst case setting of a more general problem (known as *Noisy Binary Search with Monotonic Probabilities*, which was introduced for the first time by [22]). For the worst case noisy binary search, their work implies a lower bound of the natural $\frac{\log_2 n}{I(p)}$. We note that we are not aware of any lower bounds for the lower order terms dependent on n in any of the considered settings.

1.4 Overview of the techniques

The core building block of our algorithms is Multiplicative Weights Update technique (MWU). This method has been employed in the past for noisy binary search and related problems [3, 5, 11, 22, 28]. The general outline of the method is as follows: we maintain weights that denote the "likelihood" of particular elements being the target and multiplicatively update them according to the answers to subsequent queries. After a certain number of steps, the algorithm returns the element with the highest weight, as it is the element we deem most likely to be the target.

The typical problem with this approach is that, at some point, we may encounter a situation where there is no good element to query, meaning that no query divides the search space close to the bisection. This usually occurs when a particular element becomes heavy, and subsequently querying this element yields less and less information. Previous works tried to different ways to resolve this issue, e.g. by ensuring that a good approximation of the target has been found and calling the algorithm recursively [3], introducing a phase with majority voting [14], etc.

We take a different approach by using a specifically tailored measure of progress of our algorithms, which differs from those used in previous works. For binary search, we define this measure as the total weight minus the weight of the target element. In the context of graph searching, the measure is slightly different – we use the total weight minus the weight of the heaviest vertex. This contrasts with the approach taken in prior studies, such as in [11], where the total weight itself was utilized. It is important to note that, in the case of graph search, the identity of the heaviest vertex may change throughout the search process, and at times, it may not even be the target vertex. However, our analysis guarantees the target will become the heaviest vertex by the end of the search, within the desired probability threshold.

This subtle change proves to be powerful and plays a vital role in all our proofs. We believe this is the key idea that enabled us to overcome the obstacles that the authors of the previous works might have faced.

Furthermore, in the case of binary search, when selecting which vertex to query, we employ a technique similar to that of [7]. Specifically, whenever we identify two elements that are closest to bisecting the search space, we randomly choose one of them with appropriate probability. Our analysis demonstrates that this effectively simulates the ideal subdivision of the search space and ensures the desired progress of our algorithm.

1.5 Other related work

There are many variants of the interactive query games, depending on the structure of queries and the way erroneous replies occur. The study of such games was initiated by Rényi [29] and Ulam [31]. A substantial amount of literature deals with a fixed number of errors for arbitrary membership queries or comparison queries; here we refer the reader to surveys [10, 27]. Among the most successful tools for tackling binary search with errors, is the idea of a volume [4, 28], which exploits the combinatorial structure of a possible distribution of errors. A natural approach of analyzing decision trees has been also successfully applied,

see e.g. [15]. See [5, 14] for examples of partitioning strategies into stages, where in each stage the majority of elements is eliminated and only few “problematic” ones remain. For a different reformulation (and asymptotically optimal results) of the noisy search see [22].

Although the adversarial and noisy models are most widely studied, some other ones are also considered. As an example, we mention the (linearly) bounded error model in which it is guaranteed that the number of errors is a r -fraction, $r < \frac{1}{2}$, of (any initial prefix of) the number of queries, see e.g. [1, 5, 12]. Interestingly, it might be the case that different models are so strongly related that a good bound for one of them provides also tight bounds for the other ones, see e.g. [11]. We refer the reader to distributional search where an arbitrary target distribution is known to the algorithm a priori [8, 9, 30]. A closely related theory of coding schemes for noisy communication is out of scope of this paper and we only point to some recent works [6, 16, 17, 20, 25].

The first steps towards generalizing binary search to graph-theoretic setting are works on searching in partially ordered data [2, 23, 24]. Specifically for the node search that we consider in this work, the first results are due to Onak and Parys for trees [26], where an optimal linear-time algorithm for error-less case was given.

2 Preliminaries

Whenever we refer to a *search space*, we mean either an (undirected and unweighted) graph or a linear order. Consequently, by an *element* of a search space, we refer to a vertex or an integer, respectively. In the following, let n denote the size of the search space, i.e., either the number of vertices in a graph or the number of integers in a linear order.

Throughout the search process, the strategies will maintain the weights $\omega(v)$ for the elements v of a search space V . For any $0 \leq c \leq 1$, v is *c-heavy* if $\omega(v)/\omega(V) \geq c$, where for any subset $U \subseteq V$ we write $\omega(U) = \sum_{u \in U} \omega(u)$. $\frac{1}{2}$ -heavy elements play a special role and we call them *heavy* for brevity. The weight of an element v at the end of step t is denoted by $\omega_t(v)$, with $\omega_0(v)$ being the initial value. The initial values are set uniformly by putting $\omega_0(v) = 1$ for each v in our algorithms.

Noisy binary search definition and model specifics

In the *noisy binary search* problem, we operate on a linear order $v_1 < \dots < v_n$. We are given an element v^* and the values $p \in [0, \frac{1}{2})$, $\delta \in (0, 1)$ as input. We are promised that there exists some $i \in [n]$ such that $v^* = v_i$. We call v^* the *target* element. We can learn about the search space by asking if $v^* < v_j$ for any $j \in [n]$, and receiving an answer that is correct with probability $1 - p$, independently for each query. The goal is to design an algorithm that finds the $i \in [n]$ such that $v^* = v_i$, and returns this index correctly with probability at least $1 - \delta$. We strive to minimize the number of queries performed in the process.

We adopt the following naming convention for query results. When we ask if $v^* \stackrel{?}{<} v_i$ and receive an affirmative answer (i.e., v^* is less than v_i), we call it a *yes-answer*. If the reply indicates v^* is greater than or equal to v_i (i.e., a negative answer), we call it a *no-answer*. An element v_j of a search space is considered *compatible* with the reply to a query $v^* \stackrel{?}{<} v_i$ if and only if:

- For a yes-answer (indicating $v^* < v_i$), $j < i$.
- For a no-answer (indicating $v^* \geq v_i$), $j \geq i$.

Noisy graph searching definition and model specifics

In the *noisy graph searching* problem we are given an unweighted, undirected, simple graph G and the values $p \in [0, \frac{1}{2}), \delta \in (0, 1)$. We know that one vertex v^* of G is marked as the *target*, but we don't know which one is it.

We can query the vertices of G , upon querying a vertex q we get one of two possible answers:

- $v^* = q$, i.e. the queried vertex is the target. We call it a *yes-answer*.
- $v^* \neq q$, but some neighbor u of q lies on a shortest path from q to v^* . We call it a *no-answer*. If there are multiple such neighbors (and hence shortest paths), then we can get an arbitrary one as an answer.

In fact, we assume for simplicity that each reply is given as a single vertex u . If $u = q$, then we interpret it as a yes-answer. If $u \neq q$, then u is a neighbor of q that lies on a shortest path from q to v^* . Again, we are interested in the noisy setting, therefore every reply is correct independently with probability $1 - p$. Observe that if the answer is *incorrect* then it can come in different flavors:

- if $q = v^*$, then an incorrect answer is any neighbor u of q ,
- if $q \neq v^*$, then an incorrect answer may be either q or any neighbor of q that does not lie on a shortest path from q to v^* .

Clearly, in both cases there might be several possible vertices that constitute an incorrect answer. Here we assume the strongest possible model where every time the choice among possible incorrect replies is made adversarially and independently for each query. The goal is, similarly as in noisy binary search, to design an algorithm that finds a target correctly with probability at least $1 - \delta$ and minimizes the number of queries. In fact, in this work we operate in a slightly weaker model of replies (as compared to [11, 13, 14, 26]) in which an algorithm receives less information in some cases. This is done in somewhat artificial way for purely technical reasons, i.e., to simplify several arguments during analysis. The only change to the model we have just described happens when we query a vertex that is heavy at the moment and a *no-answer* has been received. More specifically, if a *heavy* q is queried and a *no-answer* is given, the algorithm reads this reply as: the *target is not* q (ignoring the direction the target might be). Observe that this only makes our algorithms stronger, since they operate in a weaker replies model and any algorithmic guarantees for the above model carry over to the generic noisy graph search model.

Similarly to the case of noisy binary search, we say that a vertex v is *compatible* with the reply to the query q if and only if:

- $v = q$ in case of a yes-answer.
- The neighbor u given as a no-answer lies on a shortest path from q to v and q was not heavy.
- $v \neq q$ in case of a no-answer when q was heavy.

Common mathematical tools and definitions

We adopt the notation from [11] and denote $\varepsilon = \frac{1}{2} - p$ and $\Gamma = \frac{1-p}{p}$. These quantities appear frequently throughout the proofs, and this notation helps to make the presentation more concise.

The *information function*, denoted by $I(p)$, appears in all our running times. It is defined as follows $I(p) = 1 - H(p) = 1 + p \log_2 p + (1-p) \log_2 (1-p)$. In the analysis of our algorithms we frequently use the following quantitative fact about $I(p)$ and $\log_2 \Gamma$.

► **Proposition 6.** We have $I(p) = \Omega(\varepsilon^2)$ and $(\log_2 \Gamma)^2 p(1-p) = \mathcal{O}(\varepsilon^2)$.

Proof. We first observe that by Taylor's series expansion (which can be derived using elementary calculus, see e.g. [32]):

$$I(p) = \frac{1}{2 \ln 2} \sum_{n=1}^{\infty} \frac{(2\varepsilon)^{2n}}{n(2n-1)} \geq \frac{4\varepsilon^2}{2 \ln 2}.$$

To show the second bound we compute

$$\begin{aligned} p(1-p)(\log_2 \Gamma) &= (1/2 - \varepsilon)(1/2 + \varepsilon) \left(\log_2 \frac{1+2\varepsilon}{1-2\varepsilon} \right)^2 \\ &= (1 - 4\varepsilon^2) (\tanh^{-1} 2\varepsilon)^2 \frac{1}{(\ln 2)^2} \leq \frac{4}{(\ln 2)^2} \varepsilon^2 \end{aligned}$$

where the last step follows by observing that under the substitution $\varepsilon = 1/2 \cdot \tanh \gamma$ it reduces to $\gamma^2 \leq \sinh^2 \gamma$ and that inequality follows immediately from the Taylor expansion of $\sinh^2 \gamma$. ◀

Let $(X_m)_{m \in \mathbb{N}}$ be a sequence of i.i.d random variables. We say that a random variable T is a *stopping time* (with respect to $(X_m)_{m \in \mathbb{N}}$) if $\mathbb{1}_{\{T \leq m\}}$ is a function of X_1, X_2, \dots, X_m for every m .

We will use the following version of the Wald's identity.

► **Proposition 7 (Wald's Identity [33]).** Let $(X_m)_{m \in \mathbb{N}}$ be i.i.d with finite mean, and T be a stopping time with $\mathbb{E}[T] < \infty$. Then $\mathbb{E}[X_1 + \dots + X_T] = \mathbb{E}[X_1] \mathbb{E}[T]$.

Let us also recall a basic version of a Hoeffding bound, which we use in our calculations of query complexities in the worst case setting.

Multiplicative Weights Update Method

The core building block of our strategies for both binary and graph search is a standard Multiplicative Weights Update (MWU) technique. Below we formally define the version of MWU that we use in our algorithms (Algorithm 8).

► **Algorithm 8.** (MWU updates.)

In a step $t + 1$, for each element v of the search space do:

if v is compatible with the answer, then $\omega_{t+1}(v) \leftarrow \omega_t(v) \cdot 2(1-p)$,
if v is not compatible with the answer, then $\omega_{t+1}(v) \leftarrow \omega_t(v) \cdot 2p$.

Directly from the statement of our MWU method we obtain the following bound on the weight of the target. This bound applies to both binary and graph search, as the analysis is based solely on the number of erroneous replies and the fact that the target is always compatible with a correct answer.

► **Lemma 9.** If v^* is the target, then after τ queries, with probability at least $1 - \delta$ it holds

$$\omega_\tau(v^*) \geq \Gamma^{-\sqrt{2p(1-p)\tau \ln \delta^{-1}}} 2^{I(p)\tau}.$$

Proof. After τ queries with at most ℓ erroneous replies, the weight of the target satisfies:

$$\omega_\tau(v^*) \geq (2p)^\ell (2(1-p))^{\tau-\ell} = \Gamma^{p\tau-\ell} 2^{I(p)\tau}.$$

Denote $a = \sqrt{2p(1-p)\tau \ln \delta^{-1}}$. Then by Chernoff-Hoeffding bound [21], with probability at most δ there is $\ell - p\tau \geq a$. Thus, after τ queries, with probability at least $1 - \delta$ the weight of the target satisfies $\omega_\tau(v^*) \geq \Gamma^{-a} 2^{I(p)\tau}$. ◀

29:10 Noisy (Binary) Searching: Simple, Fast and Correct

Uniform prior for binary search

One can assume that the distribution of the target element in noisy binary search is *a priori* uniform by using a shifting trick described by Ben-Or and Hassidim [3]. We formally state it as a lemma below.

► **Lemma 10** (c.f. [3]). *Assume that the target element in noisy binary search problem was chosen adversarially. One can reduce that problem to the setting where the target element is chosen uniformly at random using $\mathcal{O}(\frac{\log \delta^{-1}}{I(p)})$ queries.*

3 Binary Search Algorithm

Each query performs the MWU updates using Algorithm 8. The element to be queried is selected as follows: let k be such that $\sum_{i=1}^{k-1} \omega(v_i) \leq \omega(V)/2$ and $\sum_{i=1}^k \omega(v_i) \geq \omega(V)/2$. Since the queries v_k and v_{k+1} are the closest possible to equi-division of the total weight, the algorithm chooses one of those with appropriate probability (cf. Algorithm 11.)

► **Algorithm 11.** (Query selection procedure.)

In step τ : let k be such that $\sum_{i=1}^{k-1} \omega_\tau(v_i) \leq \frac{\omega_\tau(V)}{2} \leq \sum_{i=1}^k \omega_\tau(v_i)$.

Then, query v_k with probability $\frac{1}{2\omega_\tau(v_k)}(\sum_{i=1}^k \omega_\tau(v_i) - \sum_{i=k+1}^n \omega_\tau(v_i))$, and otherwise query v_{k+1} .

In order to turn Algorithm 11 into a particular strategy, we will provide a stopping condition for each model. We start by determining the expected weight preservation during the search.

► **Lemma 12.** $\mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] \leq \omega_\tau(V \setminus \{v^*\})$.

Proof. We consider three cases, and show that this bound holds in each of those independently. Denote $A = \sum_{i=1}^{k-1} \omega_\tau(v_i)$, $B = \omega_\tau(v_k)$ and $C = \sum_{i=k+1}^n \omega_\tau(v_i)$. Denote the probability of querying v_k as $\alpha = \frac{A+B-C}{2B}$, and the probability of querying v_{k+1} as $\beta = \frac{C+B-A}{2B}$.

Case 1: $v^* = v_k$.

$$\begin{aligned} \mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= \alpha[2p^2C + 2(1-p)^2C + 2p(1-p)A + 2p(1-p)A] \\ &\quad + \beta[2p^2A + 2(1-p)^2A + 2p(1-p)C + 2p(1-p)C] \\ &= \alpha[(1+4\varepsilon^2)C + (1-4\varepsilon^2)A] + \beta[(1+4\varepsilon^2)A + (1-4\varepsilon^2)C]. \end{aligned}$$

Using the definition of α and β , we obtain

$$\begin{aligned} \mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= -4\varepsilon^2 \frac{(A-C)^2}{B} + (A+C) \\ &= -4\varepsilon^2 \frac{(A-C)^2}{B} + \omega_\tau(V \setminus \{v^*\}). \end{aligned}$$

Case 2: $v^* < v_k$. In this case,

$$\begin{aligned} \mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= (2p^2 + 2(1-p)^2)(A - \omega_\tau(v^*)) \\ &\quad + [\alpha 4p(1-p) + \beta(2p^2 + 2(1-p)^2)]B + 4p(1-p)C. \end{aligned}$$

Denote $p_1 = p^2 + (1-p)^2$ and $p_2 = 2p(1-p)$. Observe $p_1 + p_2 = 1$ and $p_1 \geq \frac{1}{2} \geq p_2$. Then,

$$\begin{aligned} \mathbb{E}[\omega_{\tau+1}(V \setminus \{v^*\}) \mid \omega_\tau(V \setminus \{v^*\})] &= 2p_1A + (C + B - A)p_1 + (A + B - C)p_2 + 2p_2C - 2p_1\omega_\tau(v^*) \\ &= A + B + C - 2p_1\omega_\tau(v^*) \\ &\leq \omega_\tau(V \setminus \{v^*\}). \end{aligned}$$

Case 3: $v^* > v_{k+1}$ is symmetric to case 2. ◀

3.1 Proof of Theorem 1 (The expected strategy length)

► **Algorithm 13.** (The expected strategy length for binary search.)

Initialization: $\omega_0(v_i) \leftarrow 1$ for each $v_i \in V$.

Execute Algorithm 11 until in some step τ it holds $\frac{\omega_\tau(v_j)}{\omega_\tau(V)} \geq 1 - \delta$ for some v_j .

Return v_j .

To show correctness of Algorithm 13, we need to observe that in the case of binary search, our MWU updates are, in fact Bayesian updates, that is, the normalized weights follow a posterior distribution conditioned on the replies seen so far. We state this as a lemma below.

► **Lemma 14.** *After any step τ of Algorithm 13 we have for every $v_i \in V$*

$$\Pr[v^* = v_i \mid \tau \text{ observed replies}] = \frac{\omega_\tau(v_i)}{\omega(V)}.$$

Proof. The proof is by induction on τ . The base case is trivial, because we assign the weights uniformly in the initialization step. The inductive step follows immediately from our definition of MWU updates (Algorithm 8) and the Bayes' rule. ◀

The correctness of Algorithm 13 follows directly from Lemma 14 and the stopping condition. To prove Theorem 1 it remains to analyze the expected length of the strategy.

► **Lemma 15.** *Algorithm 13 terminates after the expected number of at most $\frac{1}{1-p}(\log_2 n + \log_2 \frac{1}{\delta} + 3)$ steps.*

Proof. We measure the progress at any given step by a random variable $\zeta_t = \log_2 \omega_t(v^*)$. If the answer in step $t + 1$ is erroneous, then $\zeta_{t+1} = \zeta_t + 1 + \log_2 p$ and otherwise $\zeta_{t+1} = \zeta_t + 1 + \log_2(1 - p)$.

For the sake of bounding the number of steps of the algorithm, we consider it running indefinitely. Let Q be the smallest integer such that $\frac{\omega_Q(v^*)}{\omega_Q(V \setminus \{v^*\})} \geq \frac{1-\delta}{\delta}$, that is v^* is $(1 - \delta)$ -heavy in round Q . Obviously Q upper bounds the strategy length. By the definition, Q is a stopping time.

First, let us show that $\mathbb{E}[Q]$ is finite. To this end, let $\xi_t = \log_2 \frac{\omega_t(v^*)}{\omega_t(V \setminus \{v^*\})}$ and $X_t = \xi_t - \xi_{t-1}$ for $t \geq 1$. Observe that X_t 's are i.i.d and $\mathbb{E}[X_t] = p(-\log_2 \Gamma) + (1 - p) \log_2 \Gamma = (1 - 2p) \log_2 \Gamma > 0$. Therefore, using Lemma 29 for sequence X_t , with $\ell = -\log \Gamma$, $r = \log \Gamma$ and $T = \log_2 \frac{1-\delta}{\delta} - \log_2 \frac{\omega_0(v^*)}{\omega_0(V \setminus \{v^*\})}$, we indeed obtain $\mathbb{E}[Q] < \infty$.

Let Q_i for any positive integer i be smallest value such that $\omega_{Q_i}(v^*) \geq \frac{n}{\delta} \cdot 2^i$ (for completeness of notation, we define $Q_0 = 0$). Consider an event $\{Q > Q_i\}$. It means that in round Q_i the target v^* is not yet $(1 - \delta)$ -heavy. Hence, $\omega_{Q_i}(V \setminus \{v^*\}) > \frac{\delta}{1-\delta} \omega_{Q_i}(v^*) \geq \delta \omega_{Q_i}(v^*) \geq n \cdot 2^i$. But we know from Lemma 12 that $\mathbb{E}[\omega_{Q_i}(V \setminus \{v^*\})] \leq \mathbb{E}[\omega_0(V \setminus \{v^*\})] \leq n$. Using Markov's inequality we conclude that $\Pr[Q > Q_i] \leq \Pr[\omega_{Q_i}(V \setminus \{v^*\}) > n \cdot 2^i] \leq 2^{-i}$.

Additionally, since $\omega_{Q_{i-1}}(v^*) < \frac{n}{\delta} \cdot 2^i$, there is $\omega_{Q_i}(v^*) < 2 \frac{n}{\delta} \cdot 2^i$. We can then bound

$$\begin{aligned} \mathbb{E}[\zeta_Q] &< \sum_{i=1}^{\infty} \Pr[Q_{i-1} < Q \leq Q_i] \log_2(2 \cdot \frac{n}{\delta} 2^i) \\ &= \log_2(2 \cdot \frac{n}{\delta}) + \sum_{i=1}^{\infty} \Pr[Q_{i-1} < Q \leq Q_i] \cdot i \\ &= \log_2(2 \frac{n}{\delta}) + \sum_{i=1}^{\infty} \Pr[Q > Q_{i-1}] \end{aligned}$$

29:12 Noisy (Binary) Searching: Simple, Fast and Correct

$$\begin{aligned} &\leq 1 + \log_2 n + \log_2 \frac{1}{\delta} + \sum_{i=0}^{\infty} 2^{-i} \\ &\leq \log_2 n + \log_2 \frac{1}{\delta} + 3. \end{aligned}$$

Let $Y_t = \zeta_t - \zeta_{t-1}$. Obviously, Y_i 's are independent. We have already established that Q is a stopping time and that $\mathbb{E}[Q] < \infty$. This means we can employ the Wald's identity (Proposition 7) to obtain (using $\zeta_0 = 0$) $\mathbb{E}[\zeta_Q] = \mathbb{E}[Y_1 + \dots + Y_Q] = \mathbb{E}[Q]I(p)$. Therefore,

$$\log_2 n + \log_2 \frac{1}{\delta} + 3 \geq \mathbb{E}[\zeta_Q] = \mathbb{E}[Q]I(p). \quad \blacktriangleleft$$

3.2 Proof of Theorem 3 (Worst-case strategy length)

Take Q to be the smallest positive integer for which

$$I(p)Q > \log_2 n + \log_2 \frac{2}{\delta} + \sqrt{2p(1-p)Q \ln \frac{2}{\delta}} \log_2 \Gamma. \quad (1)$$

The Q gives our strategy length (see Algorithm 16). To prove Theorem 3 we bound the strategy length and the failure probability (see Lemma 17 below). The algorithm essentially remains the same except for the stop condition (cf. Algorithm 16).

► **Algorithm 16.** (Worst-case strategy length for binary search.)

Initialization: $\omega_0(v_i) \leftarrow 1$ for each element v_i .

Execute Algorithm 11 for exactly Q steps with Q as in (1).

Return the heaviest element.

► **Lemma 17.** For any $0 < \delta < 1$, Algorithm 16 finds the target correctly with probability at least $1 - \delta$ in $\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) \right)$ steps.

Proof. Firstly, observe that solving (1) for Q using Lemma 28 with parameters $a = I(p)$, $b = \log_2 n + \log_2 \frac{2}{\delta}$ and $c = 2p(1-p)(\log_2 \Gamma)^2 \ln \frac{2}{\delta}$ yields

$$Q = \frac{1}{I(p)} \left(\log_2 n + \log_2 \frac{2}{\delta} + \mathcal{O} \left(\ln \frac{2}{\delta} + \sqrt{\log_2 n + \log_2 \frac{2}{\delta}} \sqrt{\ln \frac{2}{\delta}} \right) \right)$$

where we have used, by Proposition 6, that $\frac{p(1-p)(\log_2 \Gamma)^2}{I(p)} = \mathcal{O}(1)$. The above equation can be simplified to

$$Q = \frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log \delta^{-1} \log n}) \right).$$

It remains to prove correctness. By Lemma 9, with probability at least $1 - \delta/2$ we have

$$\omega_Q(v^*) \geq \Gamma^{-\sqrt{2p(1-p)Q \ln 2/\delta}} 2^{I(p)Q}. \quad (2)$$

From Lemma 12 we also get $\mathbb{E}[\omega_Q(V \setminus \{v^*\})] \leq \mathbb{E}[\omega_0(V \setminus \{v^*\})] \leq n$. Therefore, by Markov's inequality with probability $1 - \delta/2$ we have

$$\omega_Q(V \setminus \{v^*\}) \leq \frac{2n}{\delta}. \quad (3)$$

It remains to observe that our definition of Q (Equation (1)) is equivalent to

$$\Gamma^{-\sqrt{2p(1-p)Q \ln 2/\delta} 2^{I(p)Q}} > \frac{2n}{\delta}.$$

Thus, by the union bound applied to (2) and (3), with probability at least $1 - \delta$ we get $\omega_Q(v^*) > \omega_Q(V \setminus \{v^*\})$. Then, v^* is the heaviest element and Algorithm 16 returns it. ◀

4 Graph Searching Algorithm

Denoting by $d(u, v)$ the graph distance between u and v , i.e., the length of the shortest path between these vertices, a *median* of the graph is a vertex

$$q = \arg \min_{v \in V} \sum_{u \in V} d(u, v) \cdot \omega(u).$$

For a query v and a reply u , let $C(v, u) = \{x \in V \mid u \text{ lies on some shortest path from } v \text{ to } x\}$ for $v \neq u$, and $C(v, v) = \{v\}$. We note a fundamental bisection property of a median:

► **Lemma 18** (cf. [14] Lemma 4). *If q is a median, then $\max_{u \in N(q)} \omega(C(q, u)) \leq \omega(V)/2$.*

Proof. Denote for brevity $\Phi(x) = \sum_{v \in V} d(x, v) \cdot \omega(v)$ for any $x \in V$. Suppose towards the contradiction that $\omega(C(q, u)) > \omega(V)/2$ for some $u \in N(q)$. Observe that $\Phi(u) \leq \Phi(q) - \omega(C(q, u)) + \omega(V \setminus C(q, u))$ since by moving from q to u we get closer to all vertices in $C(q, u)$. But $\Phi(q) - \omega(C(q, u)) + \omega(V \setminus C(q, u)) = \Phi(q) + \omega(V) - 2\omega(C(q, u)) < \Phi(q)$ by our assumption, hence $\Phi(u) < \Phi(q)$, which yields a contradiction. ◀

We now analyze how the weights behave when in each step a median is queried and the MWU updates are made. This analysis is common for both graph searching algorithms given later. Essentially we prove that, in an amortized way, the total weight (with the heaviest vertex excluded) remains the same in each step. In absence of heavy vertices we use Lemma 19. Lemmas 20 and 21 refer to an interval of queries to the same heavy vertex x . If the interval ends (cf. Lemma 20), then the desired weight drop can be claimed at its end. For this, informally speaking, the crucial fact is that x received many no-answers during this interval. If a strategy is at a step that is within such interval, then Lemma 21 is used to bound the total weight with the weight of x excluded. Hence, at any point of the strategy the weight behaves appropriately, as summarized in Lemma 22.

► **Lemma 19** (see also [11, 14]). *If in a step t there is no heavy vertex, then $\omega_{t+1}(V) \leq \omega_t(V)$.*

Proof. Let q be a query and u an answer in step t . Note that if there is no heavy vertex, then $C(q, u)$ is the set of vertices compatible with the reply. If $q \neq u$ then $\omega_t(C(q, u)) \leq \omega_t(V)/2$ by Lemma 18 and in case $q = u$ we have $C(q, u) = \{q\}$ and thus the same bound holds. Then in both cases, $\omega_{t+1}(V) = 2(1-p) \cdot \omega_t(C(q, u)) + 2p \cdot \omega_t(V \setminus C(q, u)) \leq \omega_t(V)$. ◀

► **Lemma 20** (see also [11]). *Consider an interval $I = \{\tau, \tau + 1, \dots, \tau + k - 1\}$ of k queries such that some x is heavy in each query in I and is not heavy after the last query in the sequence. Then $\omega_{\tau+k}(V) \leq \omega_\tau(V)$.*

Proof. First note that in each query in the interval I the queried vertex is x . Consider any two queries i and j in I such that they receive different replies. The contribution of these two queries is that together they scale down each weight multiplicatively by $2p \cdot 2(1-p) \leq 1$. Also, for a single no-answer in a query $i \in I$ we get $\omega_{i+1}(V) = 2p\omega_i(x) + 2(1-p)\omega_i(V \setminus \{x\}) \leq \omega_i(V)$

29:14 Noisy (Binary) Searching: Simple, Fast and Correct

because $\omega_i(x) \geq \omega_i(V \setminus \{x\})$ for the heavy vertex x . By assumption, the number of no-answers is at least the number of yes-answers in I . Thus, the overall weight drop is as claimed in the lemma. \blacktriangleleft

► **Lemma 21.** *Consider an interval $I = \{\tau, \tau + 1, \dots, \tau + k - 1\}$ of k queries such that some x is heavy in each query in I , and x remains heavy after the last query in I . Then*

$$\omega_{\tau+k}(V \setminus \{x\}) \leq \omega_{\tau}(V).$$

Proof. Recall that in each query in the interval I , the queried vertex is x . Assume that there were a yes-answers in I and b no-answers, with $a + b = k$. If $a \geq b$, then $\omega_{\tau+k}(V \setminus \{x\}) = (2p)^a (2(1-p))^b \omega_{\tau}(V \setminus \{x\}) \leq \omega_{\tau}(V \setminus \{x\}) \leq \omega_{\tau}(V)$. If $a < b$, then we bound as follows: $\omega_{\tau+k}(V \setminus \{x\}) \leq \omega_{\tau+k}(x) = (2p)^b (2(1-p))^a \omega_{\tau}(x) \leq \omega_{\tau}(x) \leq \omega_{\tau}(V)$. \blacktriangleleft

The bound in the next lemma immediately follows from Lemmas 19, 20 and 21. We say that an element v is *heaviest* if $\omega(v) \geq \omega(u)$ for each $u \in V$. For each step i , we denote by x_i a heaviest vertex at this step, breaking ties arbitrarily.

► **Lemma 22.** $\omega_{\tau}(V \setminus \{x_{\tau}\}) \leq \omega_0(V) = n$.

Proof. We consider the first τ queries and observe that they can be partitioned into a disjoint union of maximal intervals in which either there is a heavy vertex present (in the whole interval) or there is no heavy vertex (in the whole interval). We apply Lemma 19 for intervals with no heavy vertex and Lemmas 20, 21 otherwise (note that Lemma 21 can be applied only to the last interval. The latter happens only when there exists a heavy vertex after we perform all τ queries). \blacktriangleleft

4.1 Proof of Theorem 5 (Worst-case strategy length)

In this section we prove Theorem 5. Take Q to be the smallest positive integer for which

$$I(p)Q \geq \log_2 n + \sqrt{2p(1-p)Q \ln \delta^{-1}} \log_2 \Gamma. \quad (4)$$

The Q gives our strategy length (see Algorithm 23). To prove Theorem 5 we bound the strategy length and the failure probability (see Lemma 24 below).

► **Algorithm 23.** (Worst-case strategy length for graph search.)

Initialization: $\omega_0(v) = 1$ for each $v \in V$.

In each step: query the median and perform the MWU updates (Algorithm 8).

Stop condition: do exactly Q queries with Q defined by (4) and return the heaviest vertex.

► **Lemma 24.** *For any $0 < \delta < 1$, Algorithm 23 finds the target correctly with probability at least $1 - \delta$ in $\frac{1}{I(p)} \left(\log_2 n + \mathcal{O}(\log \delta^{-1}) + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) \right)$ steps.*

Proof. The proof is very similar to that of Lemma 17 (worst-case noisy binary search). It is actually simpler, thanks to the fact that Lemma 22 gives even stronger bound than Lemma 12.

Using Lemma 28 we solve (4) for Q . We bound the result further with Proposition 6:

$$Q = \frac{\log_2 n + \mathcal{O}(\sqrt{\log n \log \delta^{-1}}) + \mathcal{O}(\log \delta^{-1})}{I(p)}. \quad (5)$$

By Lemma 9 and the definition of Q in (4) it holds with probability $1 - \delta$

$$\log_2 \omega_Q(v^*) \geq -\sqrt{2p(1-p)Q \ln \delta^{-1}} \log_2 \Gamma + I(p)Q \geq \log_2 n \geq \log_2 \omega_Q(V \setminus \{x_Q\}),$$

where the last inequality is due to Lemma 22. Since the weights are non-negative at all times, the only way for this to happen is to have $v^* = x_Q$, that is the target being found correctly. \blacktriangleleft

4.2 Proof of Theorem 4 (The expected strategy length)

The solution for this case (given in Algorithm 25) paraphrases Algorithm 11 except for the proper adjustment of the confidence threshold.

► **Algorithm 25.** (The expected strategy length for graph search.)

Let $\delta' = c(\delta^2 \cdot (\log n + \log 1/\delta)^{-2})$ for small enough constant $c > 0$.

Initialization: $\omega_0(v) = 1$ for each $v \in V$.

In each step: query the median and perform the MWU updates (Algorithm 8).

Stop condition: if for any v in some step τ it holds $\frac{\omega_\tau(v)}{\omega_\tau(V)} \geq 1 - \delta'$, then return v .

► **Lemma 26.** Algorithm 25 stops after the expected number of at most $\frac{1}{I(p)}(\log_2 n + \log_2 \frac{1}{\delta'} + 1)$ steps.

Proof. We argue that within the promised expected number of steps, target v^* reaches the threshold weight. This clearly upperbounds the runtime. We measure the progress at any given moment by a random variable $\zeta_t = \log_2 \omega_t(v^*)$. Observe that if the reply is erroneous in a step $t + 1$, then $\zeta_{t+1} = \zeta_t + 1 + \log_2 p$, and if it is correct, then $\zeta_{t+1} = \zeta_t + 1 + \log_2(1 - p)$.

For the sake of bounding the number of steps of the algorithm, we assume it is simulated indefinitely. Let Q be the smallest integer such that $\zeta_Q \geq \log_2 n + \log_2 \frac{1 - \delta'}{\delta'}$.

By Lemma 22 we have that $\zeta_Q = \log_2 \omega_Q(v^*) \leq \log_2 \frac{\omega_Q(v^*)}{\omega(V \setminus \{x_Q\})/n}$, thus $\frac{\omega_Q(v^*)}{\omega(V \setminus \{x_Q\})} \geq \frac{1 - \delta'}{\delta'} > 1$ since w.l.o.g. $\delta' < 1/2$. But if for any t there is $\frac{\omega_t(v^*)}{\omega_t(V \setminus \{x_t\})} > 1$, then $x_t = v^*$, since $\omega_t(v^*) > \omega_t(V \setminus \{x_t\})$ implies $v^* \notin V \setminus \{x_t\}$. Thus we deduce that $x_Q = v^*$. Additionally, from $\frac{\omega_Q(v^*)}{\omega(V \setminus \{v^*\})} \geq \frac{1 - \delta'}{\delta'}$ we get that v^* is $(1 - \delta')$ -heavy, hence Q bounds the strategy length.

From $\zeta_{t+1} \in \{\zeta_t + 1 + \log_2 p, \zeta_t + 1 + \log_2(1 - p)\}$ and the minimality of Q we deduce $\zeta_Q \leq \log_2 n + \log_2 \frac{1 - \delta'}{\delta'} + 1 + \log_2(1 - p) \leq \log_2 n + \log_2 \frac{1}{\delta'} + 1$. In particular

$$\mathbb{E}[\zeta_Q] \leq \log_2 n + \log_2 \frac{1}{\delta'} + 1. \quad (6)$$

Let $X_t = \zeta_t - \zeta_{t-1}$ and observe that $\mathbb{E}[X_t] = p(1 + \log_2 p) + (1 - p)(1 + \log_2(1 - p)) = I(p) > 0$. Also, X_i 's are independent and Q is a stopping time. Finally, we have $\mathbb{E}[Q] < \infty$ from Lemma 29². Therefore, we meet all conditions of the Wald's identity (Proposition 7) and we get (since $\zeta_0 = 0$) $\mathbb{E}[\zeta_Q] = \mathbb{E}[X_1 + \dots + X_Q] = \mathbb{E}[Q]I(p)$. Thus, by (6) we have $1 + \log_2 \frac{1}{\delta'} + \log_2 n \geq \mathbb{E}[\zeta_Q] = \mathbb{E}[Q]I(p)$, from which the claim follows. \blacktriangleleft

► **Lemma 27.** Algorithm 25 finds the target correctly with probability at least $1 - \delta$.

² By plugging in $\ell = 1 + \log_2 p$, $r = 1 + \log_2(1 - p)$ and $T = \log_2 \frac{1 - \delta'}{\delta'} + \log_2 n$.

Proof. We first show correctness. Denote by $A \leq \frac{\log \frac{1-\delta'}{p}}{\log \frac{1-p}{p}} + 1$ the number of yes-answers required to go from a vertex being $1/2$ -heavy to being $(1-\delta')$ -heavy. For now assume that $A \geq 2$, we will deal with the other case later. For a non-target vertex u to be declared by the algorithm as the target, it has to observe a suffix of the strategy being a random walk on a 1-dimensional discrete grid $[0, \dots, A]$ and transition probabilities p for $i \rightarrow i+1$ and $1-p$ for $i \rightarrow i-1$. We consider a random walk starting at position $A/2$ and ending when reaching either 0 or A and call it a *subphase* (w.l.o.g. assume that A is even). Any execution of the algorithm can be partitioned into maximal in terms of containment, disjoint subphases. Each subphase starts when one particular heavy vertex v receives $A/2$ more yes-answers than no-answers within the interval in which v is heavy. Then, a subphase ends when either the algorithm declares v to be the target or v stops being heavy. By the standard analysis of the gamblers ruin problem, each subphase (where the heavy vertex is not the target) has failure probability $\delta'' = \frac{1}{1 + (\frac{1-p}{p})^{A/2}} \leq \frac{1}{1 + \sqrt{\frac{1-\delta'}{\delta'}}} = O(\sqrt{\delta'})$. Let us denote by a random variable D the number of subphases in the execution of the algorithm. Let F_i be the length of i -th subphase. By the standard analysis of the gamblers ruin problem,

$$\mathbb{E}[F_i] = \frac{A/2}{1-2p} - \frac{A}{1-2p} \frac{1}{1 + (\frac{1-p}{p})^{A/2}} \geq \frac{A/2}{1-2p} \left(1 - \frac{2}{1 + \sqrt{\frac{1-\delta'}{\delta'}}} \right) = \Omega\left(\frac{1}{\varepsilon^2}\right),$$

where the asymptotic holds since w.l.o.g. $\delta' < 1/3$, and also since if $\varepsilon < 1/3$, then $A = \Omega(1/\varepsilon)$, and otherwise $A \geq 2 = \Omega(1/\varepsilon)$. Let $F = F_1 + \dots + F_D$ be the total length of all subphases. Observe that D is a stopping time, hence we have $\mathbb{E}[F] = \mathbb{E}[D] \cdot \Omega(\frac{1}{\varepsilon^2})$ by Proposition 7. By Lemma 26, $\mathbb{E}[Q] = \mathcal{O}(\varepsilon^{-2}(\log n + \log \delta'^{-1}))$ holds for the strategy length Q . Since $F \leq Q$, $\mathbb{E}[D] = \mathcal{O}(\log n + \log 1/\delta') = \mathcal{O}(\log n + \log 1/\delta)$.

By application of the union bound, the error probability for the whole procedure is bounded by $\delta''\mathbb{E}[D] \leq \delta$ for appropriately chosen constant in the definition of δ' .

We now deal with case of $A \leq 1$. This requires $p < \delta'$, and $\varepsilon > 1/3$ (since if $\varepsilon < 1/3$, appropriate choice of constant in δ' enforces $A \geq 2$) and so the expected strategy length is $\mathbb{E}[Q] = \mathcal{O}(\log n + \log 1/\delta)$. By the union bound, algorithm receives a single erroneous response with probability at most $p\mathbb{E}[Q] \leq \delta'\mathbb{E}[Q] = \mathcal{O}(\delta^2/(\log n + \log 1/\delta)) \leq \delta$. ◀

References

- 1 Javed A. Aslam and Aditi Dhagat. Searching in the presence of linearly bounded errors (extended abstract). In *STOC*, pages 486–493, 1991. doi:10.1145/103418.103469.
- 2 Yosi Ben-Asher, Eitan Farchi, and Ilan Newman. Optimal search in trees. *SIAM J. Comput.*, 28(6):2090–2102, 1999. doi:10.1137/S009753979731858X.
- 3 Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *FOCS*, pages 221–230, 2008. doi:10.1109/FOCS.2008.58.
- 4 Elvyn R. Berlekamp. *Block Coding For The Binary Symmetric Channel With Noiseless, Delayless Feedback*, pages 61–88. Wiley & Sons, New York, 1968.
- 5 Ryan S. Borgstrom and S. Rao Kosaraju. Comparison-based search in the presence of errors. In *STOC*, pages 130–136, 1993. doi:10.1145/167088.167129.
- 6 Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. In *STOC*, pages 999–1010, 2016. doi:10.1145/2897518.2897563.
- 7 Marat Valievich Burnashev and Kamil'Shamil'evich Zigangirov. An interval estimation problem for controlled observations. *Problemy Peredachi Informatsii*, 10(3):51–61, 1974.

- 8 Yuval Dagan, Yuval Filmus, Ariel Gabizon, and Shay Moran. Twenty (simple) questions. In *STOC*, pages 9–21, 2017. doi:10.1145/3055399.3055422.
- 9 Yuval Dagan, Yuval Filmus, Daniel Kane, and Shay Moran. The entropy of lies: Playing twenty questions with a liar. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021*, volume 185 of *LIPICs*, pages 1:1–1:16, 2021. doi:10.4230/LIPICs.ITCS.2021.1.
- 10 Christian Deppe. *Coding with Feedback and Searching with Lies*, pages 27–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-32777-6_2.
- 11 Dariusz Dereniowski, Stefan Tiegel, Przemysław Uznański, and Daniel Wolleb-Graf. A framework for searching in graphs in the presence of errors. In *SOSA@SODA*, pages 4:1–4:17, 2019. doi:10.4230/OASIcs.SOSA.2019.4.
- 12 Aditi Dhagat, Péter Gács, and Peter Winkler. On playing “twenty questions” with a liar. In *SODA*, pages 16–22, 1992. URL: <http://dl.acm.org/citation.cfm?id=139404.139409>.
- 13 Ehsan Emamjomeh-Zadeh and David Kempe. A general framework for robust interactive learning. In *NIPS*, pages 7085–7094, 2017.
- 14 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *STOC*, pages 519–532, 2016. doi:10.1145/2897518.2897656.
- 15 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. doi:10.1137/S0097539791195877.
- 16 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Towards optimal deterministic coding for interactive communication. In *SODA*, pages 1922–1936, 2016. doi:10.1137/1.9781611974331.ch135.
- 17 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Trans. Information Theory*, 60(3):1899–1913, 2014. doi:10.1109/TIT.2013.2294186.
- 18 Lucas Gretta and Eric Price. Sharp Noisy Binary Search with Monotonic Probabilities. In *ICALP 2024*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.75.
- 19 Yuzhou Gu and Yinzhan Xu. Optimal bounds for noisy sorting. In *STOC*, pages 1502–1515, 2023. doi:10.1145/3564246.3585131.
- 20 Bernhard Haeupler. Interactive channel capacity revisited. In *FOCS*, pages 226–235, 2014. doi:10.1109/FOCS.2014.32.
- 21 Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.2307/2282952.
- 22 Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *SODA*, pages 881–890, 2007. URL: <http://dl.acm.org/citation.cfm?id=1283383.1283478>.
- 23 Eduardo Sany Laber, Ruy Luiz Milidiú, and Artur Alves Pessoa. On binary searching with nonuniform costs. *SIAM J. Comput.*, 31(4):1022–1047, 2002. doi:10.1137/S0097539700381991.
- 24 Tak Wah Lam and Fung Ling Yue. Optimal edge ranking of trees in linear time. *Algorithmica*, 30(1):12–33, 2001. doi:10.1007/s004530010076.
- 25 Debbie Leung, Ashwin Nayak, Ala Shayeghi, Dave Touchette, Penghui Yao, and Nengkun Yu. Capacity approaching coding for low noise interactive quantum communication. In *STOC*, pages 339–352, 2018. doi:10.1145/3188745.3188908.
- 26 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *FOCS*, pages 379–388, 2006. doi:10.1109/FOCS.2006.32.
- 27 Andrzej Pelc. Searching games with errors—fifty years of coping with liars. *Theoretical Computer Science*, 270(1):71–109, 2002. doi:10.1016/S0304-3975(01)00303-6.
- 28 Ronald L. Rivest, Albert R. Meyer, Daniel J. Kleitman, Karl Winkmann, and Joel Spencer. Coping with errors in binary search procedures. *J. Comput. Syst. Sci.*, 20(3):396–404, 1980. doi:10.1016/0022-0000(80)90014-8.
- 29 Alfréd Rényi. On a problem of information theory. *MTA Mat. Kut. Int. Kozl.*, 6B:505–516, 1961.

29:18 Noisy (Binary) Searching: Simple, Fast and Correct

- 30 Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. doi:10.1002/J.1538-7305.1948.TB01338.X.
- 31 Stanislaw M. Ulam. *Adventures of a Mathematician*. Scribner, New York, 1976.
- 32 Claude Leibovici (<https://math.stackexchange.com/users/82404/claude-leibovici>). The Taylor expansion of the binary entropy. Mathematics Stack Exchange. URL: <https://math.stackexchange.com/q/4502235> (version: 2022-07-29).
- 33 A. Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. URL: <https://www.jstor.org/stable/2235829>.
- 34 Ziao Wang, Nadim Ghaddar, and Lele Wang. Noisy sorting capacity. In *IEEE International Symposium on Information Theory, ISIT 2022*, pages 2541–2546. IEEE, 2022. doi:10.1109/ISIT50566.2022.9834370.

A Delegated Proofs

► **Lemma 28.** *The solution to $ax = b + \sqrt{cx}$ is of the form $x = \frac{1}{a} \left(b + \mathcal{O} \left(\frac{c}{a} + \sqrt{b} \cdot \sqrt{\frac{c}{a}} \right) \right)$.*

Proof. Solving the quadratic equation $a^2x^2 + b^2 - 2abx - cx = 0$, we get:

$$\Delta = (2ab + c)^2 - 4a^2b^2 = c(c + 4ab)$$

$$x = \frac{2ab + c + \sqrt{c^2 + 4abc}}{2a^2} = \frac{b}{a} + \mathcal{O} \left(\frac{c + \sqrt{abc}}{a^2} \right)$$

◀

► **Lemma 29.** *Let $(X_i)_{i \in \mathbb{N}_+}$ be a sequence of i.i.d random variables with $\Pr(X_i = \ell) = p$ and $\Pr(X_i = r) = (1 - p)$ for some $\ell, r \in \mathbb{R}$ and $0 < p < \frac{1}{2}$. Let us fix $T > 0$ and let $Q = \inf \left\{ m : \sum_{i=1}^m X_i \geq T \right\}$. If $\mathbb{E}[X_i] > 0$, then $\mathbb{E}[Q] < \infty$.*

Proof. Let $\zeta_m = \sum_{i=1}^m X_i$. If $Q > m$, then in particular $\zeta_m \leq T$, hence

$$\Pr(Q > m) \leq \Pr(\zeta_m \leq T) \tag{7}$$

for any $m \in \mathbb{N}_+$.

Let $\mu = \mathbb{E}[X_i]$. Obviously, $\mathbb{E}[\zeta_m] = m\mathbb{E}[X_i] = m\mu$. Now, let us define $N = \lceil \frac{T}{\mu} \rceil$. For any $m > N$ we have

$$\zeta_m \leq T \iff \zeta_m - m\mu \leq -(m\mu - T) \tag{8}$$

and $m\mu - T > 0$. Using Hoeffding bound [21] we get

$$\Pr(\zeta_m - m\mu \leq -(m\mu - T)) \leq \exp \left\{ \frac{-2(m\mu - T)^2}{m(\ell + r)^2} \right\} \leq \exp \left\{ \frac{-2m\mu^2}{(\ell + r)^2} + \frac{4\mu T}{(\ell + r)^2} \right\} = C\beta^m \tag{9}$$

with $C = e^{\frac{4\mu T}{(\ell + r)^2}}$ and $\beta = e^{\frac{-2\mu^2}{(\ell + r)^2}}$. Observe that $0 < \beta < 1$.

Putting together equations (7), (8) and (9) we get

$$\begin{aligned} \mathbb{E}[Q] &= \sum_{m=0}^{\infty} \Pr(Q > m) \\ &= \sum_{m \leq N} \Pr(Q > m) + \sum_{m > N} \Pr(Q > m) \leq \sum_{m \leq N} \Pr(Q > m) + C \sum_{m > N} \beta^m < \infty. \end{aligned} \quad \blacktriangleleft$$

Being Efficient in Time, Space, and Workload: a Self-Stabilizing Unison and Its Consequences

Stéphane Devismes ✉ 

Laboratoire MIS, Université de Picardie, 33 rue Saint Leu – 80039 Amiens cedex 1, France

David Ilcinkas ✉ 

Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

Colette Johnen ✉ 

Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

Frédéric Mazoit ✉ 

Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

Abstract

We present a self-stabilizing algorithm for the unison problem which is efficient in time, workload, and space in a weak model. Precisely, our algorithm is defined in the atomic-state model and works in anonymous asynchronous connected networks in which even local ports are unlabeled. It makes no assumption on the daemon and thus stabilizes under the weakest one: the distributed unfair daemon.

In an n -node network of diameter D and assuming the knowledge $B \geq 2D + 2$, our algorithm only requires $\Theta(\log(B))$ bits per node and is fully polynomial as it stabilizes in at most $2D + 2$ rounds and $O(\min(n^2B, n^3))$ moves. In particular, it is the first self-stabilizing unison for arbitrary asynchronous anonymous networks achieving an asymptotically optimal stabilization time in rounds using a bounded memory at each node.

Furthermore, we show that our solution can be used to efficiently simulate synchronous self-stabilizing algorithms in asynchronous environments. For example, this simulation allows us to design a new state-of-the-art algorithm solving both the leader election and the BFS (Breadth-First Search) spanning tree construction in any identified connected network which, to the best of our knowledge, beats all existing solutions in the literature.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Distributed algorithms; Theory of computation → Design and analysis of algorithms

Keywords and phrases Self-stabilization, unison, time complexity, synchronizer

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.30

Funding *David Ilcinkas, Colette Johnen, and Frédéric Mazoit:* This work was supported by the ANR project ENEDISC (ANR-24-CE48-7768). *Colette Johnen and Stéphane Devismes:* This work was supported by the ANR project SkyData (ANR-22-CE25-0008).

1 Introduction

1.1 Context

Self-stabilization is a general non-masking and lightweight fault tolerance paradigm [25, 3]. Precisely, a distributed system achieving this property inherently tolerates *any* finite number of transient faults.¹ Indeed, starting from an arbitrary configuration, which may be the result of such faults, a self-stabilizing system recovers within finite time, and without any external intervention, a so-called *legitimate configuration* from which it satisfies its specification.

¹ A *transient fault* occurs at an unpredictable time, but does not result in a permanent hardware damage. Moreover, as opposed to intermittent faults, the frequency of transient faults is considered to be low.

In this paper, we consider the most commonly used model in the self-stabilizing area: the *atomic-state* model [25, 3]. In this model, the state of each node is stored into registers and these registers can be directly read by neighboring nodes. Furthermore, in one atomic step, a node can read its state and that of its neighbors, perform some local computation, and update its state accordingly. In the atomic-state model, asynchrony is materialized by an adversary called *daemon* that can restrict the set of possible executions. We consider here the weakest (i.e., the most general) daemon: the *distributed unfair daemon*.

Self-stabilizing algorithms are mainly compared according to their *stabilization time*, i.e., the worst-case time to reach a legitimate configuration starting from an arbitrary one. In the atomic-state model, stabilization time can be evaluated in terms of *rounds* and *moves*. Rounds [13] capture the execution time according to the speed of the slowest nodes. Moves count the number of local state updates. So, the move complexity is rather a measure of work than a measure of time. It turns out that obtaining efficient stabilization times both in rounds and moves is a difficult task. Usually, techniques to design an algorithm achieving a stabilization time polynomial in moves make its round complexity inherently linear in n , the number of nodes (see, e.g., [2, 23, 19]). Conversely, achieving the asymptotic optimality in rounds, usually $O(D)$ where D is the network diameter, commonly makes the stabilization time exponential in moves (see, e.g., [22, 31]). Surprisingly, Cournier, Rovedakis, and Villain [14] manage to prove the first *fully polynomial* (i.e., with $Poly(n)$ move and $Poly(D)$ round complexities) *silent*² self-stabilizing algorithm. Their algorithm builds a BFS (Breadth-First Search) spanning tree in any rooted connected network and they prove that it stabilizes in $O(n^6)$ moves and $O(D^2)$ rounds using $\Theta(\log B + \log \Delta)$ bits per node, where B is an upper bound on D and Δ is the maximum degree of the network.

Up to now, fully polynomial self-stabilizing algorithms have only been proposed (see [14, 21]) for so called *static* problems [34], such as spanning tree constructions and leader election, which compute a fixed object in finite time. In this paper, we propose an algorithm for a fundamental *dynamic* (i.e., non static) problem: the *asynchronous unison* (*unison* for short). It consists in maintaining a local clock at each node. The domain of clocks can be bounded (like everyday clocks) or infinite. The liveness property of the problem requests each node to increment its own clock infinitely often. Furthermore, the safety property of the unison requires the difference between the clocks of any two neighbors to always be at most one increment. The usefulness of the unison comes from the fact that asynchrony often makes fault tolerance very difficult in distributed systems. The impossibility of achieving consensus in an asynchronous system in spite of at most one process crash [30] is a famous example illustrating this fact. Thus, fault tolerance, and in particular self-stabilization, often requires some kind of barrier synchronization, which the unison provides, to control the asynchronism of the system by making processes progress roughly at the same speed. Unison is thus a fundamental algorithmic tool that has numerous applications. Among others, it can be used to simulate synchronous systems in asynchronous environments [17], to free an asynchronous system from its fairness assumption (e.g., using the cross-over composition) [8], to facilitate the termination detection [9], to locally share resources [11], or to achieve infimum computations [10]. Thus, as expected, we also derive from our unison algorithm a *synchronizer* allowing us to obtain several new state-of-the-art self-stabilizing algorithms for various problems, including spanning tree problems and leader election.

² In the atomic-state model, a self-stabilizing algorithm is *silent* if all its executions terminate.

1.2 Related Work

Related Work on the Self-stabilizing Unison

The first self-stabilizing asynchronous unison for general graphs was proposed by Couvreur, Francez, and Gouda [15] in the link-register model (a locally-shared memory model without composite atomicity [27, 26]). However, no complexity analysis was given. Another solution, which stabilizes in $O(n)$ rounds, is proposed by Boulinier, Petit, and Villain [11] in the atomic-state model assuming a distributed unfair daemon. Its move complexity is shown in [24] to be in $O(Dn^3 + \alpha n^2)$, where α is a parameter of the algorithm that should satisfy $\alpha \geq L - 2$, where L is the length of the longest hole in the network. In his PhD thesis, Boulinier proposes a parametric solution that generalizes the solutions of both [15] and [11]. In particular, the time complexity analysis of this latter algorithm reveals an upper bound in $O(D \cdot n)$ rounds on the stabilization time of the atomic-state model version of the algorithm in [15]. Awerbuch, Kutten, Mansour, Patt-Shamir, and Varghese [4] propose a self-stabilizing unison that stabilizes in $O(D)$ rounds using an infinite state space. The move complexity of their solution is not analyzed. An asynchronous self-stabilizing unison algorithm is given in [23]. It stabilizes in $O(n)$ rounds and $O(\Delta \cdot n^2)$ moves using unbounded local memories. Emek and Keren [28] present in the stone age model a self-stabilizing unison that stabilizes in $O(B^3)$ rounds, where B is an upper bound on D known by all nodes. Their solution requires $\Theta(\log B)$ bits per node. Moreover, since node activations are required to be fair, the move complexity of their solution is unknown and may be unbounded.

Related Work on Simulations

Simulation is a useful tool to simplify the design of algorithms. In self-stabilization, simulation has been mainly investigated to emulate schedulers or to port solutions from a strong computational model to a weaker one. Awerbuch [7] introduced the concept of *synchronizer* in a non-self-stabilizing context. A synchronizer simulates a synchronous execution of an input algorithm into an asynchronous environment. The first two self-stabilizing synchronizers have been proposed in [4] for message-passing systems. Both solutions achieve a stabilization time in $O(D)$ rounds. The first solution is based on the previously mentioned unison, also proposed in the paper, that uses an infinite state space. To solve this latter issue, they then propose to mix it with the reset algorithm of [5] applied on links of a BFS spanning tree computed in $O(D)$ rounds. This reset algorithm is devoted, and so limits the approach, to locally checkable and locally correctable problems, and the BFS spanning tree construction uses a finite yet unbounded number of states per node and requires the presence of a distinguished node (a root). Again, the move complexity of their solutions is not analyzed. Awerbuch and Varghese [6] propose, still in the message-passing model, two synchronizers: the *rollback compiler* and the *resynchronizer*. The resynchronizer additionally requires the input algorithm to be locally checkable and assumes the knowledge of a common upper bound \mathcal{D} on the network diameter. Using the rollback, resp. the resynchronizer, method, a synchronous non-self-stabilizing algorithm can be turned into an asynchronous self-stabilizing algorithm that stabilizes in $O(T)$ rounds, resp. $O(T + \mathcal{D})$ rounds, using $\Omega(T \times S)$ space, resp. $\Theta(S)$ space, per node where T , resp. S , is the execution time, resp. the space complexity, of the input algorithm. Again, the move complexity of these synchronizers is not analyzed. Now, the straightforward atomic-state model version of the rollback compiler is shown to achieve exponential move complexities in [21]. Finally, the synchronizer proposed in [21] works in the atomic-state model and achieves round and space complexities similar to those of the rollback

compiler, but additionally offers polynomial move complexity. Hence, it allows to design fully polynomial self-stabilizing solutions for static problems, but still with an important memory requirement (using $\Omega(T \times S)$ space).

Simulation has been also investigated in self-stabilization to emulate other schedulers. For example, the *conflict manager* proposed in [32] allows to emulate an unfair locally central scheduler in fully asynchronous settings. Another example is fairness that can be enforced using a unison algorithm together with the *cross-over* composition [8].

Concerning now model simulations, Turau proposes in [35] a general procedure allowing to simulate any algorithm for the distance-two atomic-state model in the (classical) distance-one atomic-state model assuming that nodes have unique identifiers. Finally, simulation from the atomic-state model to the link-register one and from the link-register model to the message-passing one are discussed in [26].

1.3 Contributions

Fully Polynomial Self-stabilizing Unison

We propose a fully polynomial self-stabilizing bounded-memory unison in the atomic-state model assuming a distributed unfair daemon. It works in any anonymous network of arbitrary connected topology, and stabilizes in $O(D)$ rounds and $O(n^3)$ moves using $\Theta(\log B)$ bits per node, where $B \geq 2D + 2$ (see Table 1 below). To the best of our knowledge, our algorithm vastly improves on the literature as other self-stabilizing algorithms have at least one of the following drawbacks: an unbounded memory, an $\Omega(n)$ round complexity, a restriction on the daemon (synchronous, fair, ...). Note also that the computational model we use is at least as general as the *stone age* model of Emek and Wattenhofer [29]: it does not require any local port labeling at nodes, or knowing how many neighbors a node has.

Overall, our unison achieves outstanding performance in terms of time, workload, and space, which also makes it the first fully polynomial self-stabilizing algorithm for a *dynamic* problem.

Self-stabilizing Synchronizer

From our unison algorithm, we straightforwardly derive a self-stabilizing synchronizer that efficiently simulates synchronous executions of an input self-stabilizing algorithm in an asynchronous environment. More precisely, if the input algorithm Alg_I is silent, then the output algorithm $Sync(Alg_I)$ is silent as well and satisfies the same specification as Alg_I . The specification preservation property also holds for any algorithm, silent or not, solving a static problem. We analyze the complexity of this synchronizer and show that it mostly preserves the round and space complexities of the simulated algorithm (see Table 1 for details). This synchronizer is thus a powerful tool to ease the design of efficient *asynchronous* self-stabilizing algorithms. Indeed, for many tasks, the usual lower bound on the stabilization time in rounds is $\Omega(D)$. Now, thanks to our unison, one just has to focus on the design of a *synchronous* $O(D)$ -round self-stabilizing algorithm to finally obtain an asynchronous self-stabilizing solution asymptotically optimal in rounds, with a low overhead in space ($\Theta(\log B)$ bits per node) and a polynomial move complexity (i.e., a fully polynomial solution).

The transformer of [21] has similar round and move complexities. But this algorithm and ours are incomparable as they make different trade-offs. This paper prioritizes memory over generality, while the transformer of [21] makes the opposite choice by prioritizing generality over memory. More precisely, the transformer of [21] can simulate any synchronous algorithm (not necessarily self-stabilizing), by storing its whole execution. It thus has a much larger

space complexity than ours, which only stores two states of the simulated input algorithm. It turns out that the connections between our algorithm and the transformer of [21] are deeper than their move and round complexities. We further explain their similarities as well as their differences in Sections 3 and 4.5.

Implications of our Results

Using our synchronizer, one can easily obtain state-of-the-art (silent) self-stabilizing solutions for several fundamental distributed computing problems, e.g., BFS tree constructions, leader election, and clustering (see Table 1).

■ **Table 1** Complexities of the Unison, the Synchronizer, and some consequences.

	Moves	Rounds	Space
Unison	$O(\min(n^2B, n^3))$	$2D + 2$	$\lceil \log B \rceil + 2$
Synchronizer	$O(\min(n^2B, n^3) + nT)$	$5D + 3T$	$2M + \lceil \log B \rceil + 2$

Problem	Moves	Rounds	Space
BFS tree in rooted networks	$O(n^3)$	$O(D)$	$\Theta(\log B + \log \Delta)$
BFS tree in identified networks	$O(n^3)$	$O(D)$	$\Theta(\log N)$
Leader election	$O(n^3)$	$O(D)$	$\Theta(\log N)$
$O(\frac{n}{k})$ -clustering	$O(n^3)$	$O(D)$	$\Theta(\log k + \log N)$

T and M are the synchronous time and space complexities of the input algorithm, and B and N are input parameters satisfying $B \geq 2D + 2$ and $N \geq n$.

First, we obtain a new state-of-the-art asynchronous self-stabilizing algorithm for the BFS spanning tree construction in rooted and connected networks, by synchronizing the algorithm in [22] (which is a bounded-memory variant of the algorithm in [27]). This new algorithm converges in $O(n^3)$ moves and $O(D)$ rounds with $\Theta(\log B + \log \Delta)$ bits per node (the same round and space complexities as in [22]), where B is an upper bound on D and Δ is the maximum node degree. It improves both on the algorithm in [14], which only converges in $O(n^6)$ moves and $O(D^2)$ rounds, and on the algorithm in [21], which has similar complexities but uses $\Theta(B \cdot \log \Delta)$ bits per node.

In the following, we consider identified connected networks. In this setting, when nodes store identifiers, they usually know a bound k on the size of these identifiers. They thus know a bound $N = 2^k$ on n , and since N is a bound on D , we set $B = 2N + 2$.

In identified networks, a strategy to compute a BFS spanning tree is to compute a leader together with a BFS tree rooted at this leader. This is what the self-stabilizing algorithm in [33] actually does in a synchronous setting. Therefore, by synchronizing it, we obtain a new state-of-the-art asynchronous self-stabilizing algorithm for both the leader election and the BFS spanning tree construction in identified and connected networks. This new algorithm converges in $O(n^3)$ moves and $O(D)$ rounds with $\Theta(\log N)$ bits per node (i.e., the same round and space complexities as in [33]). To the best of our knowledge, no such efficient solutions exist until now in the literature. There are two incomparable asynchronous self-stabilizing algorithms that achieve an $O(D)$ round complexity [12, 1]. They operate in weaker models (resp. message-passing and link-register). However, their move complexity is not analyzed and the first one has a $\Theta(\log B \cdot \log N)$ space requirement (B being a known upper bound on D) while the second one uses an unbounded space.

Other memory-efficient fully polynomial self-stabilizing solutions can be easily obtained with our synchronizer, e.g., to compute the median or centers in anonymous trees by simulating algorithms proposed in [18]. Another application of our synchronizer is to remove fairness assumptions along with obtaining good complexities. For example, the silent self-stabilizing algorithm proposed in [16] computes a clustering of $O(\frac{n}{k})$ clusters in any rooted identified connected network. It assumes a distributed weakly fair daemon and its move complexity is unknown. With our synchronizer,³ we achieve a fully polynomial silent solution that stabilizes under the distributed unfair daemon and without the rooted network assumption, in $O(D)$ rounds, $O(n^3)$ moves, and using $\Theta(\log k + \log N)$ bits per node.

Note that, by using the compiler in [21], one can obtain similar time complexities for all the previous problems, but with a drastically higher space usage.

1.4 Roadmap

The rest of the paper is organized as follows. Section 2 is dedicated to the computational model and the basic definitions. We develop the links between the present paper and [21] in Section 3, and we present our algorithm in Section 4. We sketch its correctness and its time complexity in Section 5. In Section 6, the self-stabilizing synchronizer derived from our unison algorithm is presented and its complexity is also sketched. We conclude in Section 7.

2 Preliminaries

2.1 Networks

We model *distributed systems* as simple graphs, that is, pairs $G = (V, E)$ where V is a set of *nodes* and E is a set of *edges* representing communication links. We assume that communications are bidirectional. The set $N(p) = \{q \mid \{p, q\} \in E\}$ is the set of *neighbors* of p , with which p can communicate, and $N[p] = N(p) \cup \{p\}$ is the closed neighborhood of p . A *path* (from p_0 to p_l) of *length* l is a sequence $P = p_0 p_1 \cdots p_l$ of nodes such that consecutive nodes in P are neighbors. We assume that G is connected, meaning that any two nodes are connected by a path. We can thus define the *distance* $d(p, q)$ between two nodes p and q to be the minimum length of a path from p to q . The *diameter* D of G is then the maximum distance between nodes of G .

2.2 Computational Model: the Atomic-state Model

Our unison algorithm works in a variant of the *atomic-state model* in which each node holds locally shared registers, called *variables*, whose values constitute its *state*. The vector of all node states defines a *configuration* of the system.

An algorithm consists of a finite set of *rules* of the form *label* : *guard* \rightarrow *action*. In the variant that we consider, a *guard* is a boolean predicate on the state of the node and on the set of states of its neighbors. The *action* changes the state of the node. To shorten guards and increase readability, priorities between rules may be set. A rule whose guard is *true* is *enabled*, and can be executed. By extension, a node with at least one enabled rule is also *enabled*, and $Enabled(\gamma)$ contains the enabled nodes in a configuration γ . Note that

³ Also replacing the spanning tree construction used in [16] by the new BFS tree construction of the previous paragraph.

this model is quite weak. Indeed, in other variants, nodes may have, for example, distinct identifiers. In our case, the network is anonymous and since a node only accesses a set of states, it cannot even count how many neighbors it has.

An *execution* in this model is a maximal sequence of configurations $e = \gamma^0 \gamma^1 \dots \gamma^i \dots$ such that for each transition (called *step*) $\gamma^i \mapsto \gamma^{i+1}$, there is a nonempty subset \mathcal{X}^i of $\text{Enabled}(\gamma^i)$ whose nodes simultaneously and atomically execute one of their enabled rules, leading from γ^i to γ^{i+1} . We say that each node of \mathcal{X}^i executes a *move* during $\gamma^i \mapsto \gamma^{i+1}$. Note that e is either infinite, or ends at a *terminal* configuration γ^f where $\text{Enabled}(\gamma^f) = \emptyset$. An algorithm with no infinite executions is *terminating* or *silent*.

A *daemon* \mathcal{D} is a predicate over executions. An execution which satisfies \mathcal{D} is said to be *an execution under \mathcal{D}* . We consider the *synchronous daemon*, which is true if, at all steps, $\mathcal{X}^i := \text{Enabled}(\gamma^i)$, and the *fully asynchronous daemon*, also called *distributed unfair daemon* in the literature, which is always true. Note that under the distributed unfair daemon, a node may starve and may never be activated, unless it is the only enabled node.

In an execution, all the information in the states is not necessarily relevant for a problem. We thus use a *projection* to extract information (e.g., just an output boolean for the boolean consensus) from a node's state, and we canonically extend this projection to configurations and executions. A *specification* of a distributed problem is then a predicate over projected executions. A problem is *static* if its specification requires the projected executions to be constant, and it is *dynamic* otherwise.

An algorithm is *self-stabilizing* under a daemon \mathcal{D} if, for every network and input parameters, there exists a set of *legitimate* configurations such that (1) the algorithm *converges*, i.e., every execution under \mathcal{D} (starting from an arbitrary configuration) contains a legitimate configuration, and (2) the algorithm is *correct*, i.e., every execution under \mathcal{D} that starts from a legitimate configuration satisfies the specification.

We consider three complexity measures: *space*, *moves* which model the total workload, and *rounds* which model an analogous of the synchronous time by taking the speed of the slowest nodes into account. As done in the literature on the atomic-state model, the space complexity is the maximum space used by one node to store its own variables. As explained before, a move is the execution of a rule by a node. To define the round complexity of an execution $e = \gamma^0 \gamma^1 \dots$, we first need to define the notion of *neutralization*: a node p is *neutralized* in $\gamma^i \mapsto \gamma^{i+1}$, if p is enabled in γ^i and not in γ^{i+1} , but it does not apply any rule in $\gamma^i \mapsto \gamma^{i+1}$. Then, the rounds are inductively defined as follows. The first round of an execution $e = \gamma^0 \gamma^1 \dots$ is the minimal prefix e' such that every node that is enabled in γ^0 either executes a move or is neutralized during a step of e' . If e' is finite, then let e'' be the suffix of e that starts from the last configuration of e' ; the second round of e is the first round of e'' , and so on. For every $i > 0$, we denote by γ^{r^i} the last configuration of the i -th round of e , if it exists and is finite; we also conventionally let $\gamma^{r^0} = \gamma^0$. Consequently, $\gamma^{r^{i-1}}$ is also the first configuration of the i -th round of e . The *stabilization time* of a self-stabilizing algorithm is the maximum time (in moves or rounds) over every execution possible under the considered daemon (starting from any initial configuration) to reach (for the first time) a legitimate configuration.

3 A Glimpse of our Research Process

3.1 An Unbounded Unison Algorithm

We started this work on the bounded unison problem when we observed that an unbounded solution can easily be derived from [21]. This can be seen as follows. The algorithm given in [21] simulates a synchronous non self-stabilizing algorithm in an asynchronous

self-stabilizing setting. To do so, it uses a very natural idea. It stores, at each node, the whole execution of the algorithm so far as a list of states. Given its list and the lists of its neighbors, a given node can check for inconsistencies in the simulation and correct them.

Now if we implement this idea in an asynchronous algorithm which is not self-stabilizing, then the length of the lists satisfy the unison property. Indeed, to compute its $(i + 1)$ -th value, a node must wait for all its neighbors to have computed at least their i -th value.

Obviously, in a self-stabilizing setting, we cannot expect the length of the lists of the nodes to initially satisfy the unison property. It turns out that the error recovery mechanism in [21] not only solves the initial inconsistencies of the simulation, but also recovers the unison property.

If we simulate an algorithm “that does nothing”, we can compress the lists by only storing their lengths. We thus obtain a first (unbounded) unison algorithm, given below. Note that although we describe the whole algorithm, the reader does not need to fully understand it.

Each node p has a status $p.s \in \{E, C\}$ (Error/Correct) and a time $p.t \in \mathbb{N}$. Given these predicates,

$$\begin{aligned} \text{root}(p) &:= (p.s = E \wedge \neg(\exists q \in N(p), q.s = E \wedge q.t < p.t)) \vee \\ &\quad (p.s = C \wedge \exists q \in N(p), (q.t \geq p.t + 2)) \\ \text{activeRoot}(p) &:= \text{root}(p) \wedge (p.t > 0 \vee p.s = C) \\ \text{errProp}(p, i) &:= \exists q \in N(p), q.s = E \wedge q.t < i < p.t \\ \text{canClearE}(p) &:= p.s = E \wedge \forall q \in N(p), (|q.t - p.t| \leq 1 \wedge (q.t \leq p.t \vee q.s = C)) \\ \text{updatable}(p) &:= p.s = C \wedge (\forall q \in N(p), p.t \leq q.t \leq p.t + 1) \end{aligned}$$

the algorithm is defined by the following four rules

$$\begin{aligned} R_R &: \text{activeRoot}(p) \longrightarrow p.t := 0 ; p.s := E \\ R_P(i) &: \text{errProp}(p, i) \longrightarrow p.t := i ; p.s := E \\ R_C &: \text{canClearE}(p) \longrightarrow p.s := C \\ R_U &: \text{updatable}(p) \longrightarrow p.t := p.t + 1 \end{aligned}$$

in which R_R has the highest priority, and $R_P(i)$ has a higher priority than $R_P(i')$ for $i < i'$. The rules R_R , $R_P(i)$ and R_C are “error management” rules. Thus, once the algorithm has stabilized, the status of all nodes is C and only R_U is applicable.

This unbounded self-stabilizing unison algorithm is not really interesting by itself. Indeed, it converges in $2D + 2$ rounds in an asynchronous setting, but in this regard, the algorithm in [4] converges twice as fast, is simpler and operates in the message-passing model, which is more realistic. However, whereas nobody has been able to derive a bounded version of the algorithm in [4], we hoped that this could be done with this new algorithm.

In the following subsections, we present a first very natural attempt, which ultimately failed, and a more complex version, which we detail and prove in the next sections of the paper.

3.2 A Failed Bounded Unison Algorithm

The most natural strategy to turn an unbounded unison into a bounded one is simply to count modulo a large enough fixed bound B . To outline this change of paradigm from an ever-increasing time to a circular clock, we rename the variable $p.t$ into $p.c$ for any node p .

We thus modify the rule R_U as follows:

$$\begin{aligned} \text{updatable}(p) &:= p.s = C \wedge (\forall q \in N(p), q.c \in \{p.c, p.c + 1 \bmod B\}) \\ R_U : \text{updatable}(p) &\longrightarrow p.c := p.c + 1 \bmod B \end{aligned}$$

At first glance, we do not need to modify the other rules as their purpose is only to correct errors, but this intuition is wrong. Indeed, when two neighboring nodes p and q are such that $p.s = q.s = C$, $p.c = 0$ and $q.c = B - 1$, they satisfy the unison property, but p can apply the rule R_R , although there are no errors to correct. The problem comes from the term $\exists q \in N(p), (q.c \geq p.c + 2)$ in the *root* predicate which should detect out-of-sync neighbors. Hence, we must at least modify this predicate as follows:

$$\begin{aligned} \text{root}(p) &:= (p.s = E \wedge \neg(\exists q \in N(p), q.s = E \wedge q.c < p.c)) \vee \\ &\quad (p.s = C \wedge \exists q \in N(p), (q.c \geq p.c + 2) \wedge \neg(p.c = 0 \wedge q.c = B - 1)). \end{aligned}$$

Therefore, transforming the algorithm to implement this simple modulo- B idea is already not as straightforward as it may seem.

Moreover, even small modifications generally introduce new unforeseen behaviors, and the modified algorithm has no particular reasons to be efficient, or even correct. As a matter of fact, we failed to prove its correctness. To understand why, we must delve a bit into the proof scheme of [21].

An important observation is that rootless configurations (i.e., those without nodes satisfying the *root* predicate) satisfy the safety property of the unison. In [21], the correctness and the move complexity then follow from the key property that roots cannot be created, and that, in a “small” number of steps, at least one root disappears.

Sadly, this first attempt algorithm does not satisfy the “no root creations” property. To see this, consider a path $p - q - r$ and a configuration γ^a in which $p.c = q.c = B - 1$, $r.c = 3$, $p.s = q.s = C$ and $r.s = E$. In one step $\gamma^a \mapsto \gamma^b$,

- p applies the rule R_U and thus, in γ^b , $p.c = 0$ and $p.s = C$
- q applies the rule $R_P(4)$, and thus, in γ^b , $q.c = 4$.

Therefore, in γ^b , $p.s = C$ and p has a neighbor q such that $q.c \geq p.c + 2$ and $q.c \neq B - 1$. Thus, p is a root in γ^b , although it is not one in γ^a .

Note that the fact that roots can be created is not necessarily a problem. Indeed, if only a finite number of them appears, we recover the correctness of the algorithm. We actually believe that, for B large enough, any node can become a root only once per execution, and this would most likely imply that the move complexity remains polynomial. But n roots may appear sequentially, which would lead to an $\Omega(n)$ round complexity.

At this point, we cannot rule out that this algorithm is correct and has good properties. However, because of these problems, we took another approach.

3.3 Our Solution

In the end, our solution is obtained by a rather limited modification of the previous algorithm: we extend the range of the counters $p.c$ to the interval $[-B, B)$, but we restrict their range to $[-B, 0)$ when $p.s = E$.

Actually, this modification prevents *all* root creations. But, as with the previous attempt, we must be extra careful even with the smallest change, as proofs can easily break. We thus present the whole algorithm and its proofs in more details in the next sections, and further highlight the differences with [21] in Section 4.5.

4 A Unison Algorithm

4.1 Data Structures

Let $B \geq 2D + 2$ be an integer. Each node p maintains a single variable $p.v \in \{(C, x) \mid x \in [-B, B)\} \cup \{(E, x) \mid x \in [-B, 0)\}$. In the algorithm, $p.s$ and $p.c$, the *status* and the *clock* of p , respectively denote the left and right part of $p.v$. An assignment to $p.s$ or $p.c$ modifies the corresponding field of $p.v$.

We define the unison increment $a \oplus_B 1$ as $(B - 1) \oplus_B 1 = 0$ and $a \oplus_B 1 = a + 1$ if $a \in [-B, B - 2]$. Two clocks are *synchronized* if they are at most one increment apart. We then define $a \oplus_B b$ as the result of b iterations of $\oplus_B 1$ over a . Note that, as hinted in Section 3.2, we also use the usual addition and subtraction.

4.2 Some Predicates

Apart from its state, a node p has only access to the set $\{q.v \mid q \in N(p)\}$ of its neighbors' variables. A guard should thus not contain a direct reference to a neighbor q of p . This may look like a problem for we have already used such references. Nevertheless, these uses are legitimate as, for any predicate Pred , the semantics of $\exists(s, c) \in \{q.v \mid q \in N(p)\}, \text{Pred}(s, c)$ is precisely $\exists q \in N(p), \text{Pred}(q.s, q.c)$. We can similarly encode universal statements.

As a matter of fact, we use the following shortcuts to increase readability:

Shortcut1 $\exists q \in N(p), \text{Pred}(q.s, q.c) := \exists(s, c) \in \{q.v \mid q \in N(p)\}, \text{Pred}(s, c)$

Shortcut2 $\forall q \in N(p), \text{Pred}(q.s, q.c) := \forall(s, c) \in \{q.v \mid q \in N(p)\}, \text{Pred}(s, c)$

Below, we define the predicates used by our algorithm.

$$\begin{aligned} \text{root}(p) &:= (p.s = E \wedge \neg(\exists q \in N(p), q.s = E \wedge q.c < p.c)) \vee \\ &\quad (p.s = C \wedge \exists q \in N(p), (q.c \geq p.c + 2) \wedge \neg(p.c = 0 \wedge q.c = B - 1)) \\ \text{activeRoot}(p) &:= \text{root}(p) \wedge (p.c \neq -B \vee p.s = C) \\ \text{errorPropag}(p, i) &:= i < 0 \wedge \exists q \in N(p), q.s = E \wedge q.c < i < p.c \\ \text{canClearE}(p) &:= p.s = E \wedge \forall q \in N(p), (|q.c - p.c| \leq 1 \wedge (q.c \leq p.c \vee q.s = C)) \\ \text{updatable}(p) &:= p.s = C \wedge \forall q \in N(p), q.c \in \{p.c, p.c \oplus_B 1\} \end{aligned}$$

A node p is a *root* if $\text{root}(p)$. An *error rule* is either the rule R_R or a rule $R_P(i)$.

4.3 The Algorithm

A unison algorithm is rarely used alone. It is merely a tool to drive another algorithm. It thus makes sense that our algorithm depends on some properties which are external to the unison algorithm and its variables. Our algorithm uses a predicate P_{aux} which is not yet defined. As a matter of fact, its influence on the complexity analysis of the algorithm is very limited. To prove the correctness of the unison, we set $P_{\text{aux}} = \text{true}$, and we specialize P_{aux} differently in Section 6 when using our algorithm as a synchronizer.

$$\begin{aligned} R_R &: \text{activeRoot}(p) &\longrightarrow & p.c := -B ; p.s = E \\ R_P(i) &: \text{errorPropag}(p, i) &\longrightarrow & p.c := i ; p.s = E \\ R_C &: \text{canClearE}(p) &\longrightarrow & p.s := C \\ R_U &: \text{updatable}(p) \wedge P_{\text{aux}}(p) &\longrightarrow & p.c := p.c \oplus_B 1 \end{aligned}$$

The rule R_R has the highest priority, and $R_P(i)$ has a higher priority than $R_P(i')$ for $i < i'$.

4.4 An Overview of the Algorithm

Contrary to [4] which proceeds by only locally synchronizing out-of-sync clocks, i.e., the clocks of two neighboring nodes that differ by at least two increments, we organize the synchronizations in *error broadcasts*. Every node p involved in such a broadcast is *in error* and its status is $p.s = E$. Otherwise, it is *correct* and $p.s = C$.

If p is correct, in sync with its neighbors, and if its clock $p.c$ is a local minimum, then p can apply the rule R_U to increment its clock.

There is a *cliff* between r and one of its neighbors p if their clocks are out-of-sync and $p.c > r.c$. If r is correct and has a cliff with a neighbor, then r is said to be a *root* and should initiate an error broadcast by applying the rule R_R , which respectively sets $r.c$ to $-B$ and $r.s$ to E .

If there is a cliff between r and p , r is in error, and $r.c < -1$, then p should propagate the broadcast by applying the rule R_P which sets $p.c$ to $r.c + 1$ and $p.s$ to E . If p has several such neighbors r , it applies R_P according to the one with the minimum clock.

As a consequence, any node p in error with $p.c > -B$ should have at least one neighbor in error with a smaller clock. This way, the structure of an error broadcast is a *dag* (directed acyclic graph). We therefore extend the definition of root to include nodes in error with no “parents” in the broadcast dag.

Note that a node may decrease its clock multiple times using R_P , and in doing so may consecutively join several error dags or several parts of them. This way, nodes reduce the height of the error dags, which is a key element to achieve the $O(D)$ -round complexity. Furthermore, any node in error eventually has a clock smaller than $-B + D$ and all cliffs are eventually destroyed.

Finally, if p is in error, is not involved in any cliff (in which case an error must be propagated), and if all its neighbors with larger clocks are correct, then the broadcast from p is finished, and p can apply the rule R_C to switch back $p.s$ to C .

A key element to bound the move complexity is that a dag built during an error broadcast is cleaned from the larger clocks to the smaller, but nodes previously in the dag resume the “normal” increments (using the rule R_U) in the reverse order (i.e., from the smaller clocks to the larger). Indeed, a non-root node in an error broadcast is one increment ahead of its parents in the dag and so has to wait for their increment before being able to perform one itself. Hence, the first node in the dag that makes a R_U move after an error broadcast is its root.

4.5 Some Subtleties

Some statements and the corresponding proof arguments are very similar to the ones of [21] (rather its arXiv version [20]). However, the fact that the algorithm and its data structures are different imply that proofs are indeed different. As a matter of fact, we have tried but failed to unify both algorithms into a *natural* more general one.

Below, we outline subtleties which are specific to our algorithm.

- Since nodes in error are restricted to negative clocks, it is natural to expect that legitimate configurations require all clocks to be non-negative. This would suggest a $\Theta(B)$ round complexity, which is weaker than what we claim. But this intuition is false. For example, the configuration where all nodes are correct and all clocks are set to $-B$ is legitimate. This is one of the reasons for our $O(D)$ round complexity.
- In the unbounded unison algorithm above which we derive from [21], whenever two neighboring nodes p and q are such that $q.s = E$ and $p.c \geq q.c + 2$, the node p can always apply a rule $R_P(i)$. In our algorithm, this is not the case when $q.c = -1$. This could

introduce unexpected behaviors which could impact the complexities of our algorithm, or in the worst case, lead to deadlocks. We thus have to deal with this slight difference in the proofs.

- In [21], the proofs heavily rely on the fact that the counters increase when applying the rule R_U while they decrease when applying the rules R_R and $R_P(i)$. This monotony property is however not true in our setting. More generally, having two addition operators $+$ and \oplus_B requires special care throughout the proofs.
- Finally, to bound the memory, the maximum clock is $B - 1$, after which clocks go back to 0. Notice that to ensure the liveness property of the unison, we must have $B \geq 2D + 2$ (an example of deadlock is presented for $B = 2D + 1$ in Subsection 5.2).

5 Self-Stabilization and Complexity of the Unison Algorithm

As already mentioned, the unison algorithm corresponds to $P_{\text{aux}} = \text{true}$. However, since most proofs are valid regardless of the definition of P_{aux} , we only specify it when needed. We define the *legitimate* configurations as the configurations without roots. Let $e = \gamma^0 \gamma^1 \dots$ be an execution. We respectively denote by $p.s^i$ and $p.c^i$ the value of $p.s$ and $p.c$ in γ^i .

5.1 Convergence and Move Complexity of the Unison Algorithm

Although it is tedious, it is straightforward to prove, by case analysis, that roots cannot be created. Since roots are obstructions to legitimate configurations, it is natural to partition the steps of e into *segments* such that each step in which at least one root disappears is the last step of a segment. There are thus at most n segments with roots, which constitute the *stabilization phase*, and at most one root-less segment. We now show that the stabilization phase is finite by providing a (finite) bound on its move complexity.

In the following, s is any segment of the stabilization phase. The key fact is that in s , a node p in error cannot apply the rule R_U until the end of s . We prove this by induction on $p.c$. If $p.c = -B$, then p is a root, and the only rule that p can apply is R_C , which removes its root status. The base case thus follows. Now let p be in error with $p.c > -B$. If p does not move in s , then our claim holds. Otherwise let $\gamma^a \mapsto \gamma^b$ be the first step in which p moves. If p applies the rule R_R , then $p.c^b = -B$, and for the remainder of s , the claim holds by induction. Otherwise, p has a neighbor q such that $q.s^a = E$ and $q.c^a < p.c^a < 0$. By induction, $q.c$ cannot increase until the end of s . As long as $p.c > q.c$, p cannot apply the rule R_U and if, at some point, $p.c \leq q.c$, then p must have applied an error rule, thus decreasing its clock, at which point the claim holds by induction.

Since roots cannot be created, the number of R_R -moves is at most n . Moreover, since between two R_C -moves, there has to be at least one error move (R_R - or R_P -move), we have $\#R_C\text{-moves} \leq n + \#R_R\text{-moves} + \#R_P\text{-moves} \leq 2n + \#R_P\text{-moves}$. We thus only need to bound the number of R_U -moves and R_P -moves.

We now bound the number of R_U moves by a node in s . If a node q does not move between γ^a and γ^b in s with $a < b$, then a neighbor p can apply the rule R_U at most twice, to go from $q.c^a - 1$ to $q.c^a + 1$. More generally, if $p.c^b \geq p.c^a + 2 + i$ (we really mean the $+$ operator and not the \oplus_B operator), then every neighbor q of p must increase its clock by at least i between γ^a and γ^b . By induction on d , if $p.c^b = p.c^a + 2d + i$, then every node q at distance d from p increases its clock by at least i between γ^a and γ^b . Since roots cannot increase their clocks, this implies that $p.c^b \leq p.c^a + 2D$.

From this “linear” bound, we now derive a “circular” bound which takes into account the fact that the clock of a node may decrease while applying the rule R_U (from $B - 1$ to 0). In the worst case, p could apply the rule R_U $2D$ times to reach $p.c = B - 1$, then apply

R_U once so that $p.c = 0$, then reapply R_U $2D$ more times (recall that $B \geq 2D + 2$). To summarize, p may apply R_U at most $4D + 1$ times in s . This gives an $O(n^2D)$ bound on the number of R_U -moves done during the stabilization phase.

We now focus on the rule R_P in s . If a node p_0 applies a rule R_P in a step $\gamma^{j_1} \mapsto \gamma^{j_1+1}$ of s , it does so to “connect” to a neighbor p_1 which is already in error. Now p_1 may be in error in γ^{j_1} because it has applied a rule R_P in another step $\gamma^{j_2} \mapsto \gamma^{j_2+1}$ of s with $j_2 < j_1$, to connect to a neighbor p_2 , and so on. This defines a *causality chain* $p_0 \cdots p_l$ for some l . Since, according to the key fact, rules R_P and R_U do not alternate in s , a node cannot appear twice in the causality chain, thus $l < n$. Moreover, when considering a maximal causality chain, $p_l.c^{j_l}$ is either the value of $p_l.c$ at the beginning of s , or $-B$ if p_l has applied the rule R_R . The clock $p_0.c$ can thus take at most $n(n + 1)$ distinct values in s , which implies that the rule R_P is applied at most $O(n^2)$ times in s by a given node. This gives an overall $O(n^4)$ -bound on the number of rules R_P . Note that a more careful analysis gives an overall bound of $O(n^3)$ on the total number of R_P -moves.

We can also easily obtain a bound that involves B . Indeed, a node p has at most B R_P -moves and $4D + 1 = O(B)$ R_U -moves in s . This gives an $O(n^2B)$ -bound on the number of moves. To summarize, the stabilization phase terminates after at most $O(\min(n^3, n^2B))$ moves.

Note that any configuration γ with at least one root contains at least one enabled node. Indeed, if any two neighboring clocks are at most one increment apart, then any root is in error, and the rule R_C is enabled at any node p in error with $p.c$ maximum. Otherwise, there exist two neighbors p and q such that $p.c$ and $q.c$ are more than one increment apart. We choose them with $q.c < p.c$ and $q.c$ minimum. $q.c$ being minimum, we can show that either q is a root that is enabled for R_R , or p can apply the rule R_P because q is in error and satisfies $q.c \leq -B + D < -1$. Thus, the last configuration of the stabilization phase is legitimate.

Also, note that since roots cannot be created, being legitimate is a closed property, meaning that in a step $\gamma^a \mapsto \gamma^b$, if γ^a is legitimate, then so is γ^b .

5.2 Correctness of the Unison Algorithm

We now show that any legitimate configuration γ satisfies the safety property of the unison. First, γ cannot contain nodes in error, because any such node p with $p.c$ minimum would be a root. Moreover, if the clocks of two correct neighbors differ by more than one increment, then the node with the smaller clock is a root.

To prove the liveness property of the unison, we set $P_{\text{aux}} = \text{true}$ in this paragraph. In legitimate configurations, since neighboring clocks differ by at most one increment, any two clocks differ by at most D increments. And since $B \geq 2D + 2$, there exists $c \in [0, B)$ which is not the clock of any node. This implies that there exists at least one node p whose clock is not the increment of any other clock. Thus, p satisfies *updatable* and can apply R_U . This proves that at least one node applies R_U infinitely often, and thus so do all nodes. Observe that $B \geq 2D + 2$ is tight. Indeed, when $B = 2D + 1$, the configuration of the cycle p_0, p_1, \dots, p_{2D} in which all nodes are correct and $p_i.c = i$, is legitimate but is terminal.

5.3 Round Complexity of the Unison Algorithm

We claim that $\gamma^{T_{2D+2}}$ contains no roots and so is legitimate. Recall that for all $i \geq 1$, Round i is $\gamma^{r_{i-1}} \cdots \gamma^{r_i}$. We suppose that all γ^{r_i} with $i \leq 2D + 1$ contain roots otherwise our claim directly holds.

We now study the first $D + 1$ rounds. Let r be any root in $\gamma^{r_{D+1}}$. Since there are no root creations, r is already a root in γ^0 . By the end of the first round (using the rule R_R if needed), $r.c = -B$ and $r.s = E$. Now, since r is still a root in $\gamma^{r_{D+1}}$, it cannot make a move in the meantime, and its state does not change until $\gamma^{r_{D+1}}$. Furthermore, every neighbor p of r such that $p.c > -B + 1$ can apply the rule R_p . So, by the end of Round 2, and as long as r does not increment its clock, $p.c \leq -B + 1$. By induction on the distance $d(p, r)$ between p and r , we can prove that $p.c^{r_{D+1}} \leq -B + d(p, r)$ for every node p and every root r in $\gamma^{r_{D+1}}$.

We claim that $\gamma^{r_{D+1}}$ does not contain any *cliff*, i.e., a pair (q, p) of neighboring nodes whose clocks are out-of-sync and such that $q.c < p.c$. Suppose that (q, p) is a cliff in $\gamma^{r_{D+1}}$. The node q is in error as otherwise it would be a root not in error, which, as already mentioned, is impossible from γ^{r_1} . Moreover, we can prove by induction on $q.c$ that there is a root r in $\gamma^{r_{D+1}}$ such that $q.c \geq -B + d(q, r)$. Since $p.c \geq q.c + 2$, we have $p.c > -B + d(p, r)$, which contradicts the result of the previous paragraph.

We now consider the next $D + 1$ rounds. Since $\gamma^{r_{D+1}}$ contains no nodes which can apply R_R , and no cliffs, nodes can only apply the rules R_U or R_C . Furthermore, among nodes in error, those with the largest clock can apply the rule R_C , which implies that roots no longer exist by the end of Round $2D + 2$, and thus $\gamma^{r_{2D+2}}$ is legitimate.

6 A Synchronizer

Let us consider a variant of the atomic-state model which is at least as expressive as the model of our unison algorithm. This means that, in this model, we should be able to encode the shortcuts `Shortcut1` and `Shortcut2` (defined page 10).

In this model, let Alg_I be any silent algorithm which is self-stabilizing with a projection $proj$ for a static specification SP under the synchronous daemon. Using folklore ideas (see, e.g., [4] and [28]), we define in this section a synchronizer which uses our unison to transform Alg_I into an algorithm $Sync(Alg_I)$ which “simulates” synchronous executions of Alg_I in an asynchronous environment under a distributed unfair daemon.

6.1 The Synchronized Algorithm

On top of its unison variables, each node p stores two states of Alg_I , in the variables $p.old$ and $p.curr$. These variables ought to contain the last two states of p in a synchronous execution of Alg_I . When p applies the rule R_U , it also computes a next state of Alg_I . It does so by applying the function $\widehat{Alg_I}$ which selects $p.curr$ and, for each neighbor q , the variable $q.curr$ if $p.c = q.c$, and $q.old$ if $q.c = p.c \oplus_B 1$, and applies Alg_I on these values. We thus modify the rule R_U in the following way:

$$R_U : updatable(p) \wedge P_{aux}(p) \longrightarrow p.old := p.curr; p.curr := \widehat{Alg_I}(p); p.c := p.c \oplus_B 1.$$

The folklore algorithm corresponds to the case when $P_{aux}(p)$ is always *true*. In this case, the clocks of the unison constantly change. Thus, even if Alg_I is silent, its simulation is not. To obtain a silent simulation, we devise another strategy by defining $P_{aux}(p)$ as follows.

$$P_{aux}(p) = (\widehat{Alg_I}(p) \neq p.curr) \vee (\exists q \in N(p), q.c = p.c \oplus_B 1).$$

We define the legitimate configurations of $Sync(Alg_I)$ to be its terminal configurations. In the next sections, we sketch the proof that $Sync(Alg_I)$ is self-stabilizing for the same specification as Alg_I . As a matter of fact, our result is more general as the silent assumption is not necessary (we need a different definition for the legitimate configurations though).

6.2 Convergence and Move Complexity of the Synchronized Algorithm

In everyday life, we have a distinction between the value of a clock (modulo 24 hours) and the time. Both are obviously linked. We would like to make a similar distinction here. Let $e = \gamma^0 \gamma^1 \dots$ be an execution of legitimate configurations. Since γ^0 is legitimate, every two neighboring clocks differ by at most one increment.

Since $B \geq 2D + 2$, as already mentioned, at least one element of $[0, B)$ is the clock of no nodes in γ^0 . This implies that there is a node x such that $x.c^0 \oplus_B 1$ is not the clock of any node. We extend this local synchronization property by uniquely defining $time(p)^0 \in [-D, 0]$ by (1) $time(x)^0 = 0$, (2) if $p.c^0 = q.c^0$, then $time(p)^0 = time(q)^0$, and (3) if $p.c \oplus_B 1 = q.c$, then $time(p)^0 = time(q)^0 - 1$. Moreover, we also define $time(p)^{i+1} = time(p)^i + 1$ if p applies R_U in $\gamma^i \mapsto \gamma^{i+1}$, and $time(p)^{i+1} = time(p)^i$ otherwise.

For any $i, j \geq 0$ such that $time(p)^j = i$, we set $st_p^i := p.curr^j$. When st_p^i is defined for all p , let λ^i be the configuration in which the state of each node p is st_p^i . A careful analysis shows that, by definition of P_{aux} , λ^i exists as soon as some st_p^i does, and $\Lambda = \lambda^0 \lambda^1 \dots$ is precisely the synchronous execution of Alg_I from λ^0 .

Suppose that T is a bound on the number of rounds that Alg_I needs to reach silence. Thus, $\Lambda = \lambda^0 \dots \lambda^H$ for some $H \leq T$. In the simulation phase, a node makes at most D moves to have a non-negative time, and then at most T moves to finish the simulation. Together with the stabilization time of the unison, our simulated algorithm is also silent with an $O(\min(n^3, n^2B) + nD + nT) = O(\min(n^3, n^2B) + nT)$ move complexity.

6.3 Correctness of the Synchronized Algorithm

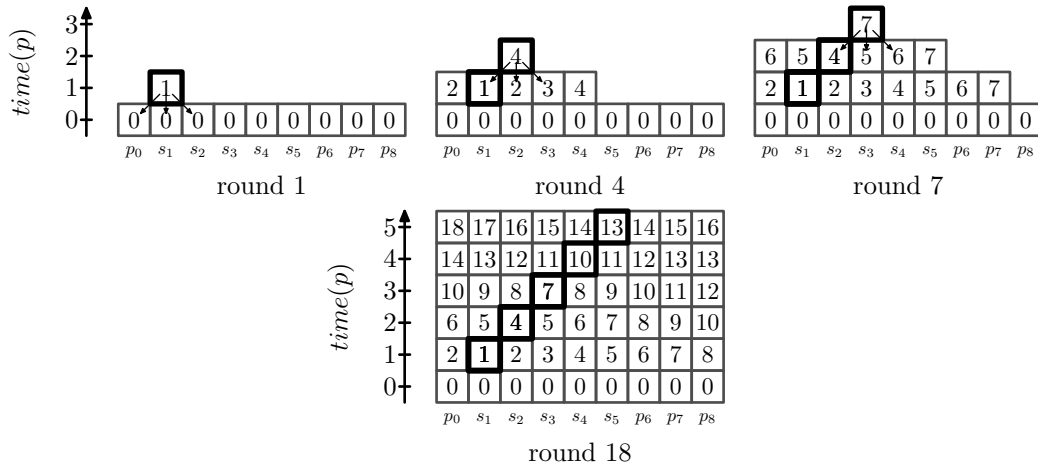
In $Sync(Alg_I)$, we define the *restriction* $rest(s)$ of the state s of any node p to be $p.curr$, and we canonically extend $rest$ to configurations and executions. Let us consider a legitimate configuration γ of $Sync(Alg_I)$. This configuration is terminal, and therefore there exists a unique execution e of $Sync(Alg_I)$ starting at γ (the one restricted to γ alone). Besides, since γ is terminal, its restriction is terminal too (for Alg_I). Therefore $rest(\gamma)$ is legitimate, and $proj(rest(e))$ satisfies the specification SP . Hence, the algorithm $Sync(Alg_I)$ also satisfies SP (for the projection $proj \circ rest$).

6.4 Round Complexity of the Synchronized Algorithm

The round complexity is analyzed by considering two stages: a first stage to have all times non-negative, and a second stage to have all times equal to H .

To give an intuition of our proof, as it is the more complex, we first consider the second stage. Figure 1 is an illustration of the following explanation. Suppose that all times are 0 in γ^0 , and only s_1 is such that $st_{s_1}^0 \neq st_{s_1}^1$. In the first round of the synchronous execution, s_1 applies R_U , and then, after each new round, nodes at distance 1 from s_1 , then 2, and so on will increase their time to 1. Now suppose that only $s_2 \in N[s_1]$ is such that $st_{s_2}^1 \neq st_{s_2}^2$. As soon as all nodes in $N[s_2]$ have a time of 1, s_2 applies R_U . This happens at Round 3 if $s_2 = s_1$ and at Round 4 otherwise. After this, after each new round, nodes at distance 1 from s_2 , then 2, and so on will increase their time to 2. If we consider some $s_3 \in N[s_2]$, and so on, then s_i increases its time to i at Round at most $3i - 2$, and all nodes do so at Round at most $3i + D - 2$. If nodes increase their time earlier, this only speed up the process.

Now, by definition of H , there is a node s_H whose state changes between λ_{H-1} and λ_H . If the states of all nodes in $N[p]$ were the same in λ_{H-2} and λ_{H-1} , then s_H would not have changed its state between λ_{H-1} and λ_H . There thus exists $s_{H-1} \in N[s_H]$ that changes its state between λ_{H-2} and λ_{H-1} . By repeating this process, we can prove that, unless $H = 0$,



■ **Figure 1** The intuition of the round complexity for the second stage.

Λ has a *starting sequence* that is a sequence $s_1 \cdots s_H$ verifying $st_{s_i}^{i-1} \neq st_{s_i}^i$ for $1 \leq i \leq H$, and $s_{i-1} \in N[s_i]$ for $1 < i \leq H$. We can then prove that, if all nodes have a positive time at Round X , then the algorithm becomes silent after at most $X + 3H + D - 2$ rounds.

Using similar ideas, we can prove that all times are non-negative after at most $X = 2D$ rounds. Taking into account the $3D + 2$ rounds of the stabilization phase, we obtain an overall $5D + 3T$ round complexity to reach the silence from any configuration.

7 Conclusion

We propose the first fully polynomial self-stabilizing unison algorithm for anonymous asynchronous bidirectional networks of arbitrary connected topology, and use it to obtain new state-of-the-art algorithms for various problems such as BFS constructions, leader election, and clustering.

A challenging perspective would be to generalize our approach to weaker models such as the message passing or the link-register models. We would also be curious to know the properties of the algorithm proposed in Section 3.2. Thirdly, although we could not do it, it would be nice to unify our result with that of [21] in a satisfactory manner. Finally, it would be interesting to know whether or not constant memory can be achieved by an asynchronous self-stabilizing unison for arbitrary topologies.

References

- 1 S. Aggarwal and S. Kutten. Time optimal self-stabilizing spanning tree algorithms. In *13th Foundations of Software Technology and Theoretical Computer Science, (TSTTCS'93)*, volume 761, pages 400–410, 1993. doi:10.1007/3-540-57529-4_72.
- 2 K. Altisen, A. Cournier, S. Devismes, A. Durand, and F. Petit. Self-stabilizing leader election in polynomial steps. *Information and Computation*, 254(3):330–366, 2017. doi:10.1016/j.ic.2016.09.002.
- 3 K. Altisen, S. Devismes, S. Dubois, and F. Petit. *Introduction to Distributed Self-Stabilizing Algorithms*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2019. doi:10.2200/S00908ED1V01Y201903DCT015.

- 4 B. Awerbuch, S. Kutten, Y. Mansour, B. Patt-Shamir, and G. Varghese. Time optimal self-stabilizing synchronization. In *25th Annual Symposium on Theory of Computing, (STOC'93)*, pages 652–661, 1993. doi:10.1145/167088.167256.
- 5 B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-stabilization by local checking and correction. In *32nd Annual Symposium of Foundations of Computer, Science (FOCS'91)*, pages 268–277, 1991. doi:10.1109/SFCS.1991.185378.
- 6 B. Awerbuch and G. Varghese. Distributed program checking: a paradigm for building self-stabilizing distributed protocols. In *32nd Annual Symposium on Foundations of Computer Science, (FOCS'91)*, pages 258–267, 1991. doi:10.1109/SFCS.1991.185377.
- 7 Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985. doi:10.1145/4221.4227.
- 8 J. Beauquier, M. Gradinariu, and C. Johnen. Cross-over composition - enforcement of fairness under unfair adversary. In *5th International Workshop on Self-Stabilizing Systems, (WSS'01)*, pages 19–34, 2001. doi:10.1007/3-540-45438-1_2.
- 9 L. Blin, C. Johnen, G. Le Boudier, and F. Petit. Silent anonymous snap-stabilizing termination detection. In *41st International Symposium on Reliable Distributed Systems, (SRDS'22)*, pages 156–165, 2022. doi:10.1109/SRDS55811.2022.00023.
- 10 C. Boulinier and F. Petit. Self-stabilizing wavelets and rho-hops coordination. In *22nd IEEE International Symposium on Parallel and Distributed Processing, (IPDPS'08)*, pages 1–8, 2008. doi:10.1109/IPDPS.2008.4536130.
- 11 C. Boulinier, F. Petit, and V. Villain. When graph theory helps self-stabilization. In *23rd Annual Symposium on Principles of Distributed Computing, (PODC'04)*, pages 150–159, 2004. doi:10.1145/1011767.1011790.
- 12 J. Burman and S. Kutten. Time optimal asynchronous self-stabilizing spanning tree. In *21st International Symposium on Distributed Computing, (DISC'07)*, volume 4731, pages 92–107, 2007. doi:10.1007/978-3-540-75142-7_10.
- 13 A. Cournier, A. K. Datta, F. Petit, and V. Villain. Snap-stabilizing PIF algorithm in arbitrary networks. In *22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 199–206, 2002. doi:10.1109/ICDCS.2002.1022257.
- 14 A. Cournier, S. Rovedakis, and V. Villain. The first fully polynomial stabilizing algorithm for BFS tree construction. *Information and Computation*, 265:26–56, 2019. doi:10.1016/j.ic.2019.01.005.
- 15 J.-M. Couvreur, N. Francez, and M. G. Gouda. Asynchronous unison (extended abstract). In *12th International Conference on Distributed Computing Systems, (ICDCS'92)*, pages 486–493, 1992. doi:10.1109/ICDCS.1992.235005.
- 16 A. K. Datta, S. Devismes, K. Heurtefeux, L. L. Larmore, and Y. Rivierre. Competitive self-stabilizing k -clustering. *Theoretical Computer Science*, 626:110–133, 2016. doi:10.1016/j.tcs.2016.02.010.
- 17 A. K. Datta, S. Devismes, and L. L. Larmore. A silent self-stabilizing algorithm for the generalized minimal k -dominating set problem. *Theoretical Computer Science*, 753:35–63, 2019. doi:10.1016/j.tcs.2018.06.040.
- 18 A. K. Datta and L. L. Larmore. Leader election and centers and medians in tree networks. In *15th International Symposium on Stabilization, Safety, and Security of Distributed Systems, (SSS'13)*, pages 113–132, 2013. doi:10.1007/978-3-319-03089-0_9.
- 19 S. Devismes, D. Ilcinkas, and C. Johnen. Optimized silent self-stabilizing scheme for tree-based constructions. *Algorithmica*, 84(1):85–123, 2022. doi:10.1007/s00453-021-00878-9.
- 20 S. Devismes, D. Ilcinkas, C. Johnen, and F. Mazoit. Making local algorithms efficiently self-stabilizing in arbitrary asynchronous environments. *CoRR*, abs/2307.06635, 2023. doi:10.48550/arXiv.2307.06635.
- 21 S. Devismes, D. Ilcinkas, C. Johnen, and F. Mazoit. Asynchronous self-stabilization made fast, simple, and energy-efficient. In *43rd Symposium on Principles of Distributed Computing, (PODC'24)*, pages 538–548, 2024. doi:10.1145/3662158.3662803.

- 22 S. Devismes and C. Johnen. Silent self-stabilizing BFS tree algorithms revisited. *Journal on Parallel Distributed Computing*, 97:11–23, 2016. doi:10.1016/j.jpdc.2016.06.003.
- 23 S. Devismes and C. Johnen. Self-stabilizing distributed cooperative reset. In *39th International Conference on Distributed Computing Systems, (ICDCS'19)*, pages 379–389, 2019. doi:10.1109/ICDCS.2019.00045.
- 24 S. Devismes and F. Petit. On efficiency of unison. In *4th Workshop on Theoretical Aspects of Dynamic Distributed Systems, (TADDS'12)*, pages 20–25, 2012. doi:10.1145/2414815.2414820.
- 25 E. W. Dijkstra. Self-stabilization in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974. doi:10.1145/361179.361202.
- 26 S. Dolev. *Self-Stabilization*. MIT Press, 2000.
- 27 S. Dolev, A. Israeli, and S. Moran. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Computing*, 7(1):3–16, 1993. doi:10.1007/BF02278851.
- 28 Y. Emek and E. Keren. A thin self-stabilizing asynchronous unison algorithm with applications to fault tolerant biological networks. In *40nd Symposium on Principles of Distributed Computing, (PODC'21)*, pages 93–102, 2021. doi:10.1145/3465084.3467922.
- 29 Y. Emek and R. Wattenhofer. Stone age distributed computing. In *32nd Symposium on Principles of Distributed Computing, (PODC'13)*, pages 137–146, 2013. doi:10.1145/2484239.2484244.
- 30 M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985. doi:10.1145/3149.214121.
- 31 C. Glacet, N. Hanusse, D. Ilcinkas, and C. Johnen. Disconnected components detection and rooted shortest-path tree maintenance in networks. *Journal of Parallel and Distributed Computing*, 132:299–309, 2019. doi:10.1016/j.jpdc.2019.05.006.
- 32 Maria Gradinariu and Sébastien Tixeuil. Conflict managers for self-stabilization without fairness assumption. In *27th IEEE International Conference on Distributed Computing Systems (ICDCS 2007), June 25-29, 2007, Toronto, Ontario, Canada*, page 46. IEEE Computer Society, 2007. doi:10.1109/ICDCS.2007.95.
- 33 A. Kravchik and S. Kutten. Time optimal synchronous self stabilizing spanning tree. In *27th International Symposium on Distributed Computing, (DISC'13)*, pages 91–105, 2013. doi:10.1007/978-3-642-41527-2_7.
- 34 S. Tixeuil. *Vers l'auto-stabilisation des systèmes à grande échelle*. Habilitation à diriger des recherches, Université Paris Sud - Paris XI, 2006. URL: https://tel.archives-ouvertes.fr/tel-00124848/file/hdr_final.pdf.
- 35 V. Turau. Efficient transformation of distance-2 self-stabilizing algorithms. *Journal of Parallel and Distributed Computing*, 72(4):603–612, 2012. doi:10.1016/j.jpdc.2011.12.008.

Efficient Approximation Schemes for Scheduling on a Stochastic Number of Machines

Leah Epstein ✉ 

Department of Mathematics, University of Haifa, Israel

Asaf Levin ✉ 

Faculty of Data and Decisions Science, The Technion, Haifa, Israel

Abstract

We study three two-stage optimization problems with a similar structure and different objectives. In the first stage of each problem, the goal is to assign input jobs of positive sizes to unsplitable bags. After this assignment is decided, the realization of the number of identical machines that will be available is revealed. Then, in the second stage, the bags are assigned to machines. The probability vector of the number of machines in the second stage is known to the algorithm as part of the input before making the decisions of the first stage. Thus, the vector of machine completion times is a random variable. The goal of the first problem is to minimize the expected value of the makespan of the second stage schedule, while the goal of the second problem is to maximize the expected value of the minimum completion time of the machines in the second stage solution. The goal of the third problem is to minimize the ℓ_p norm for a fixed $p > 1$, where the norm is applied on machines' completion times vectors. Each one of the first two problems admits a PTAS as Buchem et al. showed recently. Here we significantly improve all their results by designing an EPTAS for each one of these problems. We also design an EPTAS for ℓ_p norm minimization for any $p > 1$.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases Approximation algorithms, Approximation schemes, Two-stage stochastic optimization problems, Multiprocessor scheduling

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.31

Related Version *Full Version*: <https://arxiv.org/abs/2409.10155>

Funding *Asaf Levin*: Partially supported by ISF – Israel Science Foundation grant number 1467/22.

1 Introduction

We consider scheduling problems where the goal is to assign jobs non-preemptively to a set of identical machines. Unlike traditional scheduling problems [22], the number of identical machines available to the scheduler, denoted as k , is not a part of the input, but it is drawn from a known probability distribution on the set of integers $\{1, 2, \dots, m\}$ for an integer $m \geq 2$ that is a part of the input, and it is given together with the probabilities. As a first stage, the decision maker completes an initial set of decisions, namely it assigns the jobs to m bags, forming a partition of all input jobs. Later, once the realization of the number of machines becomes known, it packs the bags to the machines, where the packing of a bag to a machine means that all its jobs are scheduled together to this machine. That is, in this later step, every pair of jobs that were assigned to a common bag will be assigned to a common machine, and the decision of the first stage (that is, the partition of input jobs into m bags) is irrecoverable. This two-stage stochastic scheduling problem was recently introduced by Buchem et al. [9].

Formally, the input consists of a set of jobs $\mathcal{J} = \{1, 2, \dots, n\}$ where each job $j \in \mathcal{J}$ has a positive rational size p_j associated with it. We are given an integer $m \geq 2$ and let $\mathcal{B} = \{1, 2, \dots, m\}$ denote the set of bags. We are also given a probability measure q over \mathcal{B} , where q_k is a rational non-negative number denoting the probability that the number of



© Leah Epstein and Asaf Levin;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 31; pp. 31:1–31:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



identical machines in the resulting second stage instance is k . Here, we will use the property that $\sum_{k=1}^m q_k = 1$. In the first stage of our problem, the jobs are split into m bags by an algorithm. Namely, a feasible solution of the first stage is a function $\sigma_1 : \mathcal{J} \rightarrow \mathcal{B}$. In the second stage, after the value of k has been revealed, the bags are assigned to machines, such that all jobs of each bag are assigned together. Namely, the algorithm also computes assignment functions σ_2^k defined for every realization k in the support of q of the bags to the set of integers $\{1, 2, \dots, k\}$ denoting the indexes of machines in the instance of the second stage problem (corresponding to the realization of q that is equal to k). So if the realization of q is k , a job $j \in \mathcal{J}$ will be assigned to the machine of index $\sigma_2^{(k)}(\sigma_1(j))$. The first function σ_1 maps jobs to bags, and the second function which is based on the value of k , σ_2^k , maps bags to machines. The number of bags is m (and σ_1 is independent of k , so the partition into bags does not depend on k which is not known yet at the time of assignment into bags), and the number of machines is k , so for every realization of k the schedule of the jobs to the k machines is a feasible (non-preemptive) schedule for k identical machines.

We use the terminology of such scheduling problems and let W_i be the work (also called load) of machine i which is the total size of jobs assigned to machine i in a schedule. This is also the completion time of a unit speed machine processing continuously the set of jobs assigned to machine i starting at time 0 (in some order). The *makespan of the schedule in realization k* is the maximum work of a machine in the second stage solution in the realization k of q , and the *Santa Claus value of the schedule of realization k* is the minimum work of a machine (among the k machines) in the second stage solution in the realization k of q .

The expected value of the makespan is the expected value of the random variable of the makespan of the schedule in realization k . The expectation is computed based on all possible values of k . The first problem that we consider is the problem of minimizing the expected value of this random variable. We denote it as PR-MAKESPAN and refer to it as the *makespan minimization problem*. The expected value of the Santa Claus value is the expected value of the random variable of the Santa Claus value of the schedule in realization k . The second problem we consider is the problem of maximizing the expected value of this random variable. We denote it as PR-SANTACLUS and refer to it as the *Santa Claus maximization problem*. While the term makespan is used in the scheduling literature in this meaning the term Santa Claus is not traditional, and it is usually referred to as the minimum work of a machine in the schedule. Given the length of the resulting terminology for our settings, we prefer to use the non-traditional terminology of Santa Claus value. The expected value of the random variable is called cost or value for minimization problems, and it is called profit or value for maximization problems. In the ℓ_p norm minimization problem, the objective for k machines is $(\sum_{i=1}^k W_i^p)^{1/p}$. The cost of a solution of the third problem is the expected cost based on the random variable. This last problem is denoted by PR-NORM.

Since the stochastic nature of the problem results only from the different options for k (and once k is known, the problem is deterministic), we say that if the realization of q is k , then this is the scenario of index k or simply scenario k . We let OPT_k denote the objective function value of the second stage problem in scenario k for an optimal solution for the studied problem, and let $\text{OPT} = \sum_{k=1}^m q_k \cdot \text{OPT}_k$ denote the value of an optimal solution to our problem. An optimal solution needs to balance all scenarios, and we denote this optimal solution by OPT as well (that is, we use the same notation as the cost or the value). We stress that OPT_k is not necessarily the optimal objective function value for the input jobs and k machines, since the solution with a fixed set of bags may be inferior. We will assume that $\frac{1}{\varepsilon}$ is a positive integer such that $\frac{1}{\varepsilon} > 100$ (this can be done without loss of generality as one can first decrease ε to the minimum between its original value and $\frac{1}{100}$, and then it

is always possible to decrease the value of ε by a factor below 2 to satisfy the integrality condition on $\frac{1}{\varepsilon}$). The assumptions on ε will be used for all schemes (and in particular we will use $\varepsilon < \frac{1}{100}$). Let $p_{\max} = \max_{j \in \mathcal{J}} p_j$.

An overview of our results. Here, we study both minimization problems and a maximization problem. All types of algorithms defined here are required to have polynomial running times (and our algorithms will in fact have strongly polynomial running times). For a minimization problem, an \mathcal{R} -approximation algorithm is an algorithm that always finds a solution that is feasible and has a cost at most \mathcal{R} times the cost of an optimal solution. For a maximization problem, an \mathcal{R} -approximation algorithm is an algorithm that always finds a feasible solution of value at least $\frac{1}{\mathcal{R}}$ times the value of an optimal solution. The approximation ratio of a given algorithm is the infimum value of the parameter \mathcal{R} such that this algorithm is an \mathcal{R} -approximation algorithm.

A PTAS is a class of approximation algorithms such that for any $\varepsilon > 0$ the class has a $(1+\varepsilon)$ -approximation algorithm. An efficient polynomial time approximation scheme (EPTAS) is a stronger concept. This is a PTAS whose running time is of the form $h(\varepsilon) \cdot \text{poly}(N)$ where h is some (computable but not necessarily polynomial) function and $\text{poly}(N)$ is a polynomial function of the length N of the (binary) encoding of the input. We will justify the assumption $m \leq n$ later, and therefore the running time can be rewritten in this form when the polynomial is on m and n . A fully polynomial time approximation scheme (FPTAS) is an even stronger concept, defined like an EPTAS, but the function h must be a polynomial in $\frac{1}{\varepsilon}$. In this paper, we are interested in EPTAS's.

A PTAS may have time complexity of the form $n^{g(\varepsilon)}$, where g can be any function of $\frac{1}{\varepsilon}$ and even a power tower function. However, an EPTAS cannot have such a running time, which makes it more efficient. The concept of an EPTAS is related to fixed parameter tractable (FPT) algorithms [16].

In this paper, we focus on finding EPTAS's for the three objectives. We develop an EPTAS for the makespan minimization problem in Section 2. Then, in Section 3 we establish an EPTAS for the Santa Claus maximization problem PR-SANTACLAUS. Last, in the full version of this work we modify our scheme for PR-MAKESPAN in order to obtain an EPTAS for PR-NORM. Due to space limitations, some of the proofs of the first two results are given in the full version, and the last EPTAS for PR-NORM is presented in the full version as well. In the recent work [9], a PTAS was designed for PR-MAKESPAN, and another PTAS was designed for PR-SANTACLAUS. Those PTAS's are of forms that do not allow a simple modification of the PTAS into an EPTAS, and in particular for PR-SANTACLAUS a dynamic program over a state space whose size is exponential is used (a function of ε appears in the exponent of the input size), and for both schemes enumeration steps of such sizes are applied.

For all objectives, we assume that $n > m$ holds. If $n \leq m$, this means that every job can be assigned to its own bag. For each scenario, it is possible to apply an EPTAS for the corresponding problem (see for example [2], the details for previous work are discussed further below) and thereby get an EPTAS for the required problem.

The problems studied here generalize strongly NP-hard problems, and therefore one cannot expect to obtain an FPTAS (unless $P = NP$), and thus our results are the best possible. Specifically, the special case of each one of the three problems where the number of machines is known, i.e., the probability function has a single value q_m that is equal to 1, and other probabilities are equal to zero, is known to be NP-hard in the strong sense. The three objectives studied here are the three main objectives previously studied for scheduling on identical machines.

Related work. Stein and Zhong [33] introduced the scheduling problem with an unknown number of machines but in their model the number of machines is selected later by an adversary. The problem was mostly studied with respect to makespan minimization. In the deterministic variant [33], the goal is to assign a set of jobs with known properties to identical machines. However, only an upper bound m on the number of machines is given. Jobs have to be partitioned into m subsets, such that every subset will act as an inseparable bag. Then, the number of machines k (where $2 \leq k \leq m$) becomes known, and the schedule is created using the bags, without unpacking them, as in the problem that we study here, and after the number of machines is revealed, the bags are assigned to the machines, as in the problem that we study. Thus, this problem also has two steps or levels of computation, but the worst case out of all possible values of k is analyzed, where the comparison for each value of m is to an optimal offline solution for k identical machines and arbitrary bags. It is not hard to see that a constant approximation ratio (of at most 2) can be obtained using a round-robin approach even for jobs of arbitrary sizes (via pre-sorting). An improved upper bound of $\frac{5}{3} + \epsilon$ was shown using a much more careful assignment to bags [33].

A variant where machines have speeds (similar to uniformly related machines) was defined and studied by [18] with respect to makespan. In that version, the number of machines m is known, but not their speeds. The number of required bags is equal to the number of machines, but some machines may have speed zero, so the case of identical machines and the makespan objective is seen as binary speeds in this work. There are several input types of interest, which are arbitrary jobs, unit size jobs, sand (one large job that can be cut into any required parts), and pebbles (jobs that have arbitrary sizes, but they are relatively small) [18, 30]. Tight bounds were proved for the case of sand and makespan with and without speeds [33, 18], and for the Santa Claus objective without speeds [33]. All these values are strictly smaller than 1.6. For sand, since any partition of the input jobs is allowed, linear programming can be used to find the best partition. In the case with speeds for arbitrary sizes of jobs the algorithm of [18] has an approximation ratio of at most $2 - \frac{1}{m}$, while special cases allow smaller ratios [18, 30].

In [5], Balkanski et al. relate the problem of scheduling with an unknown number of machines and an unknown set of speeds to online algorithms with predictions. In online problems with predictions [31], the algorithm receives information on the input, where such information could have been computed via machine learning. This model takes into account both the situation where the input matches the prediction exactly and the case where it does not. It is required for the algorithm to have a relatively good performance for every input, but the performance has to be better if the input was predicted correctly. In many algorithms the performance improves as the input gets closer to the predicted input. The work of [5] provides results of this flavor.

Optimizing the worst scenario is pessimistic and does not allow one to take information learned from previous data into account, while the study of the expected value as in our stochastic problem allows us to prefer the typical scenarios in the bag assignment algorithm. See also [4, 8, 32, 17] for related work.

As we mentioned, the most relevant work to our work on the stochastic problem is [9], where two PTAS's are provided. It is also mentioned in that work that FPTAS's for the case where m is seen as a constant can be designed using standard methods. The problem is called *stochastic* due to the probability distribution on scenarios. However, since this distribution is simply a convex combination of the costs for different scenarios, the methods are related to those often used for deterministic algorithms and worst case analysis. The convex combination complicates the problem and it requires carefully tailored methods for

the algorithmic design. We follow the standard worst case studies of two-stage stochastic optimization problems, and we study the optimization problems of optimizing the expected value of the random variable.

All three objectives were studied initially for known numbers of identical machines, for which simple greedy approximation algorithms were presented first. Some time later PTAS's were designed. Finally EPTAS's were designed or it was observed that one of the known PTAS's is an EPTAS or can be converted into an EPTAS, which is the best possible result for each one of the three cases unless $P = NP$. We provide additional details for each one of the objectives. The makespan objective was introduced by Graham [22, 23], where greedy algorithms were studied (see also [13, 20]). Twenty years later a PTAS was designed [25]. Hochbaum [24] mentions that one of the approaches of assigning large jobs of rounded sizes (namely, solving an integer linear program in fixed dimension) gives an EPTAS, and attributes this approach to D. Shmoys (see [26, 11] for recent results). The Santa Claus objective (where the name was introduced much later [7]) was studied with respect to greedy heuristics [21, 15, 14]. A PTAS which is actually an EPTAS was designed by Woeginger [34]. The ℓ_p norm objective function was studied with respect to greedy heuristics [10, 12, 29], and a PTAS and an EPTAS were designed [1]. The same authors generalized their results for other objectives that satisfy natural conditions [2]. The ℓ_p norm of a vector is seen as a more suitable measure of fairness of a schedule compared to bottleneck measures [3, 6], and therefore we study this objective additionally to those studied in [9].

Our methods. We develop new techniques to address the uncertainty in the problems. We expect these techniques to be used in later work on approximating related variants.

We use rounding and discretization as in other work on scheduling, though we apply it not only on job sizes but also on bag sizes. The difficulty in relating new sizes to optimal solutions lies in the difference in objectives, since here the optimal value is not defined by a single schedule. However, we are still able to find such a relation using enumeration of sets of values, that is, via guessing. Since jobs are assigned to bags, we introduce configuration integer programs (IP's) on collections of bags, which can be seen as a generalization of previously known approaches. Our configuration IP's are based on the use of templates, where a template defines the contents of a bag, and configurations, where a configuration is an allocation of bags to one machine in a given scenario. We solve this IP using an algorithm for solving an IP in fixed dimension. All earlier steps of our schemes are highly motivated given the goal of creating such an IP and ensuring that it has a fixed dimension.

Machine numbers are split to intervals such that the most difficult interval for each guessed information is solved using an IP, while the others are solved using greedy approaches. The number of intervals for scenarios differs based on the specific problem.

While the sketched approach allows us to design an EPTAS for makespan minimization, the other objectives are harder to approximate. In order to tackle them, we develop an important tool called approximated histograms. Those are histograms of solution costs or values, and we use them in order to reduce the number of relevant scenarios to a constant. The condition for using the approximated histogram is a monotonicity assumption regarding the feasibility of solutions for different scenarios and the monotonicity of the costs for a given solution among scenarios in which it is feasible.

2 An EPTAS for the makespan minimization problem

We start with the makespan minimization problem, and continue to the other objectives in other sections. The first step will be to apply a discretization on bag sizes, and bound (from above) the set of relevant bag sizes as a function of OPT, where we enumerate possible values

of OPT and use rounding for this value as well. Our second step is to round the instance of jobs (which will be assigned to bags). We use standard guessing steps and rounding methods for these steps. Then, we are going to approximate the rounded instance by using templates of the assignment of jobs to bags, and configurations of assigning templates to machines in every realization of q . This will allow us to formulate an integer linear program that has a special structure, namely, it is a *two-stage stochastic integer linear program*. This special structure, as well as trivial bounds on the parameters of this formulation, provides us with an FPT algorithm to solve the problem. This algorithm has a running time dominated by a function of $\frac{1}{\varepsilon}$ times a polynomial in n , and therefore it is an EPTAS.

We will be able to reduce the number of variables such that we can apply Lenstra's algorithm [28, 27] on the integer linear program, and the running time for solving this program will be smaller than that of solving a two-stage stochastic integer linear program. This is the case also for the other two EPTAS's that we show, that is, we will apply Lenstra's algorithm for all objectives. The most difficult step in the current section is to reduce the number of different values of makespan for different scenarios. Without this step, the running time will not satisfy the requirements of an EPTAS. The reduction is based on linear grouping, which is typically used for bin packing [19] and not for scheduling. The other EPTAS's that we design require additional non-trivial steps, and we will discuss them later.

The optimal solution of this integer program is transformed into a set of decisions defining the functions σ_1 and $\sigma_2^{(k)} \forall k$. Next, we present the details of the scheme together with its analysis. In what follows we will present steps where in each of those steps the resulting approximation ratio of the scheme increases by at most a multiplicative factor of $1 + \Theta(\varepsilon)$. By scaling ε prior to applying the scheme, we get that the approximation ratio of the entire scheme will be at most $1 + \varepsilon$. We will use the next lemma for our proof.

► **Lemma 1.** *It holds that $OPT_k \in [p_{\max}, n \cdot p_{\max}]$ for any $k \leq m$. Thus, $OPT \in [p_{\max}, n \cdot p_{\max}]$.*

Guessing opt. Our first step is to guess the value of OPT within a multiplicative factor of $1 + \varepsilon$. That is, we would like to enumerate a polynomial number of candidate values, where the enumerated set will contain (at least one) value that is in the interval $[OPT, (1 + \varepsilon) \cdot OPT]$. We will show that for a guess in this interval we indeed find a solution of cost $(1 + \Theta(\varepsilon)) \cdot OPT$. The next lemma shows that it is possible to enumerate such a set of candidate values since $\log_{1+\varepsilon} n \leq \frac{n}{\varepsilon}$. Our algorithm performs this enumeration.

► **Lemma 2.** *There is a set consisting of $O(\log_{1+\varepsilon} n)$ values such that this set of values has at least one value in the interval $[OPT, (1 + \varepsilon)OPT]$.*

In what follows, with a slight abuse of notation, we let OPT be the value of this guessed candidate value (which is not smaller than OPT and it is larger by at most a factor of $1 + \varepsilon$ if the guess is correct). We will show that if there is a feasible solution to PR-MAKESPAN whose expected value of the cost is at most OPT, then we will construct a feasible solution with expected value of the cost being at most $(1 + \Theta(\varepsilon)) \cdot OPT$. By the above lemma, this means that the problem admits an EPTAS as desired.

Rounding bag sizes. Now, we provide a method to gain structure on the set of feasible solutions that we need to consider for finding a near optimal solution. Given a feasible solution, the size of bag i denoted as $P(i)$ is the total size of jobs assigned to this bag, that is, $P(i) = \sum_{j: \sigma_1(j)=i} p_j$.

► **Lemma 3.** *There exists a solution of cost at most $(1 + 3\varepsilon) \cdot OPT$ in which the size of every non-empty bag is in the interval $[\varepsilon \cdot OPT, (1 + \varepsilon) \cdot OPT]$.*

In what follows, we will increase the size of a bag to be the next value of the form $(\varepsilon + r\varepsilon^2) \cdot \text{OPT}$ for a non-negative integer r (except for empty bags for which the allowed size remains zero). Thus, we will not use precise sizes for bags but upper bounds on sizes, and we call them *allowed sizes*. We let $P'(i)$ be this increased (allowed) size of bag i , that is, $P'(i) = \min_{r: (\varepsilon + r\varepsilon^2)\text{OPT} \geq P(i)} (\varepsilon + r\varepsilon^2)\text{OPT}$. Since for every subset of bags, the total allowed size of the bags in the set is at most $1 + \varepsilon$ times the total size of the bags in the subset, we conclude that this rounding of the bag sizes will increase the cost of any solution by a multiplicative factor of at most $1 + \varepsilon$ (and therefore the expected value also increases by at most this factor). Thus, in what follows, we will consider only bag sizes that belong to the set $B = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \dots, \frac{1}{\varepsilon^2}\} \cup \{0\}$. We use this set by Lemma 3. We conclude that the following holds.

► **Corollary 4.** *If the guessed value (OPT) is at least the optimal expected value of the makespan, then there is a solution of expected value of the makespan of at most $(1 + \varepsilon)(1 + 3\varepsilon) \cdot \text{OPT}$ that uses only bags with allowed sizes in B .*

Note that the allowed size of a bag may be slightly larger than the actual total size of jobs assigned to the bag. Later, the allowed size acts as an upper bound (without a lower bound), and we take the allowed size into account in the calculation of machine completion times in some cases (instead of the size).

Rounding job sizes. We apply a similar rounding method for job sizes. Recall that by our guess, every job j has size at most OPT (see Lemma 1). Next, we apply the following modification to the jobs of sizes at most $\varepsilon^2\text{OPT}$. Whenever there is a pair of jobs, each of which has size at most $\varepsilon^2\text{OPT}$, we unite the two jobs (i.e., we delete the two jobs, adding a new job whose size is the sum of the two sizes of the two deleted jobs). This is equivalent to restricting our attention to solutions of the first stage where we add the constraint that the two (deleted) jobs must be assigned to a common bag. We repeat this process as long as there is such a pair. If there is an additional job of size at most $\varepsilon^2\text{OPT}$, we delete it from the instance, and in the resulting solution (after applying the steps below on the resulting instance) we add the deleted job to bag 1. This is done after the entire algorithm completes running, so it may increase the makespan of every scenario and therefore the expected value of the makespan, but we do not take it into account in the algorithm or in calculating allowed sizes of bags. This addition of the deleted job increases the makespan of every scenario by at most $\varepsilon^2\text{OPT}$, so the resulting expected value of the makespan will be increased by a multiplicative factor of at most $1 + \varepsilon^2$. We consider the instance without this possible deleted job, and we prove the following.

► **Lemma 5.** *The optimal expected value of the makespan of the instance resulting from the above modification of the job sizes among all solutions with allowed bag sizes in the following modified set $B' = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \dots, \frac{1}{\varepsilon^2} + 2\} \cup \{0\}$ is at most $(1 + 2\varepsilon)(1 + \varepsilon)(1 + 3\varepsilon) \cdot \text{OPT}$.*

We next round up the size of every job j to be the next value that is of the form $(1 + \varepsilon)^r \cdot \text{OPT}$ for an integer r . The size of every job cannot decrease and it may increase by a multiplicative factor of at most $1 + \varepsilon$. Job sizes are still larger $\varepsilon^2 \cdot \text{OPT}$ and the sizes do not exceed $(1 + \varepsilon) \cdot \text{OPT}$. Thus, the number of different job sizes is $O(\log_{1+\varepsilon} \frac{1+\varepsilon}{\varepsilon^2}) \leq \frac{2}{\varepsilon^3} - 1$. We increase the allowed size of each non-empty bag by another multiplicative factor of $1 + \varepsilon$, and round it up again to the next value of the form $\varepsilon + r\varepsilon^2$. Thus, the expected value of the makespan of a feasible solution increases by a multiplicative factor of at most $(1 + \varepsilon)^2$, where

one factor of $1 + \varepsilon$ is due to the rounding of jobs, and the second one is due to bringing allowed sizes back to the form $\varepsilon + r\varepsilon^2$. So by our guessing step, there is a feasible solution with expected value of the makespan of at most $(1 + \varepsilon)^3(1 + 2\varepsilon)(1 + 3\varepsilon)\text{OPT}$ that uses only bags with allowed size in $B'' = \{(\varepsilon + r\varepsilon^2)\text{OPT} : r = 0, 1, \dots, \frac{1+2\varepsilon}{\varepsilon^2} + 2\} \cup \{0\}$.

A bag is called *tight* if the set of jobs of this bag cannot use a bag of a smaller allowed size. Note that any solution where some bags are not tight can be converted into one where all bags are tight without increasing the cost. Note that the allowed size of a non-zero bag of allowed size $(\varepsilon + r\varepsilon^2)\text{OPT}$ can be written in the form $(r + \frac{1}{\varepsilon}) \cdot \varepsilon^2 \cdot \text{OPT}$, and since $\frac{1}{\varepsilon}$ is integral, the allowed size of each bag is an integer multiple of $\varepsilon^2 \cdot \text{OPT}$.

► **Lemma 6.** *For any tight bag it holds that the allowed size of the bag is at most $\frac{1}{\varepsilon}$ times its actual size (the total size of jobs assigned to the bag, where their rounded sizes are considered).*

Computing the maximum number of machines for which the assignment to bags does not matter. In our algorithm, we would like to focus on values of k for which the structure of bags is crucial. Now, we will detect values of k for which there always exists a good assignment of bags to machines (for reasonable sets of bags). Let P be the total size of all jobs (after the transformation applied on jobs for a fixed value of OPT). The total size (not the allowed size) of all bags is P , and we can compute the makespan based on the actual sizes of bags which are total sizes of jobs (after merging and rounding) assigned to the bags. The motivation is that for very small values of k the maximum bag size is so small compared to $\frac{P}{k}$ that bags can be seen as very small jobs, and one can apply a greedy algorithm [22] for assigning the bags, and still obtain a solution with a small makespan (based on bag sizes).

► **Lemma 7.** *If $k \leq \frac{\varepsilon \cdot P}{2 \cdot \text{OPT}}$, then any set of bags (such that every job is assigned to a bag) where every bag has an allowed size (and size) not exceeding 2OPT , leads to at least one schedule for k machines with makespan in $[\frac{P}{k}, (1 + \varepsilon) \cdot \frac{P}{k}]$. For other values of k , the makespan of an optimal schedule using tight bags (based on their allowed sizes) is at most $(3/\varepsilon^2) \cdot \text{OPT}$.*

We use the property that no bag has an allowed size above $2 \cdot \text{OPT}$. Our algorithm computes the maximum value of k for which $2\text{OPT} \leq \varepsilon \cdot \frac{P}{k}$ holds, and afterwards it excludes values of k for which $2\text{OPT} \leq \varepsilon \cdot \frac{P}{k}$ holds. We will have that the cost of the solution obtained for the scenario is at most $(1 + \varepsilon) \cdot \frac{P}{k}$, since bag sizes satisfy the condition for bags in the statement of Lemma 7, and therefore we can use the first part of this lemma. This is a valid approach as adding a scenario where $2\text{OPT} \leq \varepsilon \cdot \frac{P}{k}$ to the calculation of the approximation ratio will not increase the approximation ratio beyond the value $1 + \varepsilon$, and the running time for computing a good solution for such a value of k is polynomial in m, n . Thus, we assume that $2\text{OPT} > \varepsilon \cdot \frac{P}{k}$ holds for every k .

We consider the remaining scenarios. In what follows, we consider the assignment of jobs to bags and the assignment of the bags to machines in each scenario k subject to the condition that $2\text{OPT} > \varepsilon \cdot \frac{P}{k}$ and the optimal makespan of the scenario is at most $(3/\varepsilon^2) \cdot \text{OPT}$. In particular, it means that the number of bags of positive allowed sizes assigned to a machine in such a scenario is at most $3/\varepsilon^3$, since we consider allowed bag sizes not smaller than $\varepsilon \cdot \text{OPT}$ (if we exclude bags of allowed size zero).

Guessing an approximated histogram of the optimal makespan on all scenarios. Our next guessing step is motivated by linear grouping of the OPT_k values. To illustrate this step, consider a histogram corresponding to the costs of a fixed optimal solution satisfying all above structural claims, where the width of the bar corresponding to scenario k is q_k and its height is the value of the makespan in this scenario (that is, OPT_k). Observe that when k

Our next guessing step is motivated by linear grouping of the OPT_k values. To illustrate this step, consider a histogram corresponding to the costs of a fixed optimal solution satisfying all above structural claims, where the width of the bar corresponding to scenario k is q_k and its height is the value of the makespan in this scenario (that is, OPT_k). Observe that when k

is increased, the optimal makespan does not increase (without loss of generality, since the same assignment of bags can be used), so by plotting the bars in increasing order of k we get a monotone non-decreasing histogram, where the total area of the bars is the expected value of the makespan of the optimal solution and the total width of all the bars is 1 (since we assume that all scenarios are included in the histogram and every scenario κ satisfies the condition $2\text{OPT} > \varepsilon \cdot \frac{P}{\kappa}$ by scaling the probabilities of the remaining scenarios such that the total probability becomes 1). We sort the indexes of scenarios with (strictly) positive probabilities (q_k values) and we denote by K the resulting sorted list. For every $k \in K$, we compute the total width of the bars with indexes at most k (using the ordering in K , i.e., in increasing indexes), and denote this sum by Q_k . We also let $Q'_k = Q_k - q_k$. Thus, Q_k is the total probability of scenarios in $\{1, 2, \dots, k\}$, and Q'_k is the total probability of scenarios in $\{1, 2, \dots, k-1\}$. The set K does not contain scenarios with probability zero, but if $q_{k-1}, q_k > 0$, it holds that $Q_{k-1} = Q'_k$. According to the definitions, it holds that $Q_k - Q'_k = q_k$ for $k \in K$, and the interval $[Q'_k, Q_k)$ on the horizontal axis corresponds to scenario k .

The idea of the next parts is to get an overestimator for the histogram by extending the value of k to the right, and an underestimator by extending it to the left. The two areas below them will differ only by at most $O(\varepsilon \cdot \text{OPT})$, so the overestimator can also differ from the original histogram area by at most this amount.

Next, we compute an upper bound histogram W as follows. We start with selecting a sublist K' of K . The motivation is that schedules will be computed later only for scenarios in K' , and they will be copied to scenarios of some of the larger indexes. The set K' acts as a representative set, and we show that it is possible to restrict ourselves to such a set. Since we increase upper bounds on the makespan for some scenarios, the feasibility is not harmed.

An index $k \in K$ belongs to K' if there exists an integer ℓ such that $Q'_k \leq \varepsilon^3 \ell < Q_k$. The motivation is that we would like to consider points which are integral multiples of ε^3 and the set K' contains all values of $k \in K$ for which such special values belong to the interval $[Q'_k, Q_k)$. Values of makespan will be increased such that the makespan function will still be piecewise linear, but it will have a smaller number of image values.

For every $k \in K'$, the new upper bound histogram is defined as OPT_k for all points with horizontal value between Q'_k and up to $Q'_{k'}$ where $k' > k$ is the index just after k in the sublist K' or up to the last point where the histogram is defined (Q_t for the maximum value $t \in K$) if k is the largest element of K' . Since the original histogram was monotone non-increasing, the new histogram is pointwise not smaller than the original histogram. The possible modification is in the interval $[Q_k, Q'_{k'})$ if k is not the maximum value of K' , and in this case there may be a change for $k+1, \dots, k'-1$ (that is, if $k' \geq k+2$). If k is the largest element of K' but not of K , there may be a change for all $t \in K$ such that $t \geq k+1$. Thus, an upper bound on the total area below the histogram W is not smaller than the expected value of the makespan of the optimal solution.

We use the modified histogram W to obtain another bound. For that we see W as a step function whose values are the corresponding points in the upper edge of the histogram. We define a new histogram by letting it be $W(x + \varepsilon^3)$ for all $x \in [0, 1]$ (and zero for cases that W is undefined due to an argument above 1). In this way we delete a segment of length ε^3 from W and we shift the resulting histogram to the left. Since the makespan for every scenario never exceeds $(3/\varepsilon^2) \cdot \text{OPT}$, every point in W has a height of at most $\frac{3\text{OPT}}{\varepsilon^2}$ and we deleted a segment of width of ε^3 , the total area that we delete is at most $3\varepsilon\text{OPT}$. The resulting histogram is pointwise at most the original histogram (and thus also not larger than W). This property holds since every value OPT_k was extended to the right for an interval not longer than ε^3 . Thus, if we consider W instead of the original histogram, we increase the cost of the solution by a multiplicative factor not larger than $(1 + 3\varepsilon)$.

The guessing step that we use is motivated by this function W . We let W_k be the height of the histogram W in the bar corresponding to the scenario k . The relevant values of k are those in K' , and the histogram W only has values of the form OPT_k for $k \in K'$. We guess the histogram W . This means to guess the optimal makespan of $O(\frac{1}{\varepsilon^3})$ scenarios, where makespans can be one of at most $\frac{3}{\varepsilon^4}$ different values. This holds since all allowed bag sizes are integer multiples of $\varepsilon^2 \cdot \text{OPT}$, so makespans are also integer multiples of $\varepsilon^2 \cdot \text{OPT}$, and the makespan of each scenario is at most $\frac{3\text{OPT}}{\varepsilon^2}$. Thus, the number of possibilities of this guessing step is upper bounded by a function of $\frac{1}{\varepsilon}$ which is $O((\frac{1}{\varepsilon})^{O(1/\varepsilon^3)})$.

In what follows, we consider the iteration of the algorithm defined below when we use the value of the guess corresponding to the optimal solution. Algorithmically, we will try all possibilities, check the subset of those for which we can provide a feasible solution, and output the best feasible solution obtained in this exhaustive enumeration. The running time of the next algorithm is multiplied by the number of possibilities.

The template-configuration integer program. A *template* of a bag is a multiset of job sizes assigned to a common bag. We consider only multisets for which the total size of jobs is at most $(1 + 6\varepsilon) \cdot \text{OPT}$ since the largest allowed size of any bag is smaller. Since the number of distinct job sizes (in the rounded instance) is upper bounded by $\frac{2}{\varepsilon^3} - 1$ and there are at most $\frac{2}{\varepsilon^2}$ jobs assigned to a common bag (since the rounded size of each job is above $\varepsilon^2 \cdot \text{OPT}$), we conclude that the number of templates is at most $(\frac{2}{\varepsilon^3})^{(2/\varepsilon^2)}$, which is a constant depending only on $\frac{1}{\varepsilon}$. The reason is that each bag has $\frac{2}{\varepsilon^2}$ slots for jobs, such that each one may be empty or contain a job of one of the sizes. This calculation proves an upper bound on the number of possible templates, and we use a single template for every multiset of job sizes even when one multiset can be found in more than one way in the last calculation.

We are going to have a non-negative counter decision variable y_t for every template t , where this variable stands for the number of bags with template t . Let τ be the set of templates, and assume that a template is represented by a vector. The length of such a vector is the number of different (rounded) job sizes, and for every index ℓ , the ℓ -th component of the vector of the template (denoted by t_ℓ) is the number of jobs with size equal to the ℓ -th job size that are packed into a bag with this template. In order for such an assignment of the first stage to be feasible, we have the following constraints where n_ℓ denotes the number of jobs in the rounded instance whose size is the ℓ -th job size of the (rounded) instance.

$$\sum_{t \in \tau} y_t \leq m, \quad \sum_{t \in \tau} t_\ell \cdot y_t = n_\ell, \forall \ell.$$

The first constraint states that the number of bags is at most m . If this number is strictly smaller than m , then the remaining bags have allowed sizes of zero and they will not contain jobs. The second constraint, which is a family of constraints, states that the correct numbers of jobs of each size are assigned to templates, according to the number of copies of each template. Observe that the number of constraints in this family of constraints is a constant depending on ε (it is at most $\frac{2}{\varepsilon^3}$), and all coefficients in the constraint matrix are (non-negative) integers not larger than $\frac{2}{\varepsilon^2}$.

We augment K' with the minimum index in K if this index does not already belong to K' , in order to satisfy the condition that for every index of K there is some index of K' that is not larger. Consider a scenario $\kappa \in K'$ (satisfying the condition that $2\text{OPT} > \varepsilon \cdot \frac{P}{\kappa}$), and the optimal makespan of scenario κ , which is at most $\frac{3\text{OPT}}{\varepsilon^2}$. Recall that in scenario κ the number of non-empty bags assigned to each machine is at most $\frac{3}{\varepsilon^3}$. Define a configuration of a machine in scenario κ to be a multiset of templates such that the multiset has at most $\frac{3}{\varepsilon^3}$

templates (counting multiple copies of templates according to their multiplicities) and the total size of the templates encoded in the multiset is at most W_κ , which is the guess of a value of the histogram W . The number of configurations is at most $((\frac{2}{\varepsilon^3})^{(2/\varepsilon^2)})^{\frac{3}{\varepsilon^3}}$, and this is also an upper bound on the number of suitable configurations (whose total allowed size of bags does not exceed W_κ). Since our mathematical program will not limit the makespan of any scenario, we use an upper bound for it by not allowing configurations whose total allowed sizes exceed the planned makespan for the scenario. We let $C^{(\kappa)}$ denote the set of configurations for scenario κ , where $c \in C^{(\kappa)}$ is a vector of $|\tau|$ components where component c_t for $t \in \tau$ is the number of copies of template t assigned to configuration c . Components are non-negative integers not larger than $\frac{3}{\varepsilon^3}$. For scenario $\kappa \in K'$, we will have a family of non-negative decision variables $x_{c,\kappa}$ (for all $c \in C^{(\kappa)}$) counting the number of machines with this configuration.

For each such scenario κ , we have a set of constraints each of which involves only the template counters (acting as a family of global decision variables) and the family of the κ -th local family of decision variables, namely the ones corresponding to this scenario. The family of constraints for the scenario $\kappa \in K'$ are as follows.

$$\sum_{c \in C^{(\kappa)}} x_{c,\kappa} = \kappa, \quad \sum_{c \in C^{(\kappa)}} c_t \cdot x_{c,\kappa} - y_t = 0, \quad \forall t \in \tau.$$

Observe that the number of constraints in such a family of constraints is upper bounded by a constant depending only on ε (which is the number of possible templates plus 1) and the coefficients in the constraint matrix are again all integers of absolute value at most a constant depending only on ε (at most $\frac{3}{\varepsilon^3}$).

All decision variables are forced to be non-negative integers and we would like to solve the feasibility integer program defined above (with all constraints and decision variables). The right hand side vector consists of integers that are at most n (using $m \leq n$). Such an integer linear program is a two-stage stochastic IP. Using the property that the number of scenarios in K' is upper bounded by a function of $\frac{1}{\varepsilon}$, we conclude that the integer linear program has a fixed dimension. Thus, we use Lenstra's algorithm to solve it instead of an algorithm for two-stage stochastic IP.

We obtain a feasible solution (x, y) if such a solution exists. Based on such a feasible solution, we assign jobs to bags using the y variables. That is, for every $t \in \tau$, we schedule y_t bags using template t . By the constraint $\sum_{t \in \tau} y_t \leq m$ there are at most m bags, and the other bags will be empty. Next, for every bag and every size p of jobs in the rounded instance, if the template assigned to the bag has α jobs of this size, we will assign α jobs of this size to this bag. Doing this for all sizes and all bags is possible by the constraints $\sum_{t \in \tau} t_\ell \cdot y_t = n_\ell, \forall \ell$, and these constraints ensure that all jobs are assigned to bags. Note that the modification for jobs consisted of merging small jobs. When such a merged job is assigned, this means that a subset of jobs is assigned. Reverting jobs to their original sizes does not increase any of the costs, since no rounding steps decreased any sizes. Next consider the assignment of bags to machines in each scenario. Consider a scenario $\kappa' \in K$, and let κ be the largest index in K' such that $\kappa \leq \kappa'$. Assign $x_{c,\kappa}$ machines to configuration c (for all $c \in C^{(\kappa)}$). It is possible by the constraint $\sum_{c \in C^{(\kappa)}} x_{c,\kappa} = \kappa \leq \kappa'$. If a configuration c assigned to machine i is supposed to pack c_t copies of template t , we pick a subset of c_t bags whose assigned template is t and assign these bags to machine i . We do this for all templates and all machines. In this way we assign all bags by the constraints $\sum_{c \in C^{(\kappa)}} c_t \cdot x_{c,\kappa} - y_t = 0, \forall t \in \tau$. If $\kappa' > \kappa$, at least one machine will not receive any configuration and therefore it will not receive any bags or jobs, and we refer to such a machine as empty. Since the configurations we used in scenario κ have a total size of jobs of at most W_κ , we conclude the following.

► **Corollary 8.** *For every value of the guessed information for which the integer program has a feasible solution, there is a linear time algorithm that transform the feasible solution to the integer program into a solution to the rounded instance of PR-MAKESPAN of cost at most $\sum_{\kappa} q_{\kappa} \cdot W_{\kappa}$.*

Our scheme is established by noting that an optimal solution satisfying the assumptions of the guessed information provides us a multiset of templates all of which are considered in τ and a multiset of configurations (and all of them have total size of templates not larger than W) for which the corresponding counters satisfy the constraints of the integer program. Thus, we conclude our first main result.

► **Theorem 9.** *Problem PR-MAKESPAN admits an EPTAS.*

3 An EPTAS for the Santa Claus problem

In this section we apply additional ideas to obtain an EPTAS for the second problem. We present the details of the scheme together with its analysis. In what follows, and similarly to the scheme to PR-MAKESPAN, we will present steps, and in each of those steps the resulting approximation ratio of the scheme increases by at most a multiplicative factor of $1 + \Theta(\epsilon)$.

Preprocessing steps. We consider an optimal solution OPT, and analyze the information that one can guess in order to be able to approximate it. We assume without loss of generality that $\text{OPT}_k \leq \text{OPT}_{k'}$ for all $k > k'$, since a solution for scenario k can be used for scenario k' (by assigning the bags of $k - k'$ machines in an arbitrary way). Recall that we can assume that for every value of k in the support of q the instance has at least k non-zero sized jobs, since we assume $n > m$. We will use the next lemma for our proof.

► **Lemma 10.** *For any scenario k (such that $1 \leq k \leq m$) and an assignment of jobs to bags according to the solution OPT, there exists a job j^k for which it holds that $p_{j^k} \leq \text{OPT}_k \leq n \cdot p_{j^k}$.*

We would like to split the sequence of scenarios into four parts (where some of the parts may be empty). The suffix (with the largest numbers of machines) will contain scenarios for which the assignment of bags is unimportant since the gain from them in the objective function value OPT is small either because the probability is zero or because the number of machines is large. This suffix may be empty. There will also be a prefix (which could be empty) of machines for which the specific assignment to bags is unimportant because the number of machines is small and any reasonable assignment into bags allows a good schedule. For this prefix we will use different arguments from the ones we used for PR-MAKESPAN. For the remaining scenarios, the prefix will consist of scenarios for which the gain is also small, and a suffix of the most important scenarios.

The first step is to guess the maximum scenario k_{\max} for which $\text{OPT}_{k_{\max}} \geq \epsilon \cdot \text{OPT}$ holds, out of scenarios with strictly positive probabilities, that is, such that $q_{k_{\max}} > 0$ holds. This index is well-defined since there exists an index k for which $q_k > 0$ and $\text{OPT}_k \geq \text{OPT}$. There are at most m possible values for this guess. While OPT still denotes an optimal solution, we do not guess or use its value as a part of the algorithm. Let LB be equal to $\text{OPT}_{k_{\max}}$ rounded down to the next integer power of $1 + \epsilon$. For a fixed job $j^{k_{\max}}$ (see Lemma 10), the number of possible values for LB is $O(\frac{n}{\epsilon})$. Since the index $j^{k_{\max}}$ is also not known, the number of possible values for LB is $O(\frac{n^2}{\epsilon})$.

The proof of the next lemma is obtained by selecting a set of consecutive scenarios minimizing the total weighted profit out of $\frac{1}{\epsilon}$ disjoint sequences of scenarios of similar forms. The proof of the lemma requires the knowledge not only of OPT but of all values of the form

OPT_i . However, we use the lemma to obtain the information that given the correct value LB (this is a rounded value, so we will be able to guess it), there is a pair of values k, k' and a value ρ that specify the only properties of OPT that we use for our algorithm.

► **Lemma 11.** *There is a value of ρ which is an integer power of $\frac{1}{\varepsilon}$ that satisfies $\frac{1}{\varepsilon^2} \leq \rho \leq (\frac{1}{\varepsilon})^{1/\varepsilon+1}$ and there exist two indexes k, k' such that $0 \leq k' < k \leq k_{\max}$, where every non-zero index has to be a scenario in the support of q , such that the following conditions hold. $\text{OPT}_k \leq \rho \cdot LB$, if $k' \geq 1$, then $\text{OPT}_{k'} \geq \frac{\rho}{\varepsilon} \cdot LB, \sum_{\kappa=k'+1}^{k-1} q_{\kappa} \cdot \text{OPT}_{\kappa} \leq \varepsilon \cdot \text{OPT}$.*

The next guessing step. In this step we guess several additional values. As mentioned above, we guess the value of LB (which is a power of $1 + \varepsilon$ and it is equal to $\text{OPT}_{k_{\max}}$ within a multiplicative factor of $1 + \varepsilon$ since $\text{OPT}_{k_{\max}} \in [LB, LB \cdot (1 + \varepsilon))$), the value of ρ (by guessing ρ' from the proof of Lemma 11, that is, we guess the power of $\frac{1}{\varepsilon}$), and two indexes $k' < k \leq k_{\max}$. That is, we would like to enumerate a set of polynomial number of candidate values of four components vectors that contains at least one vector with first component value that is LB , with a second component whose value is ρ , and the two indexes $k' < k$ in the third and fourth components of that vector. The next lemma shows that it is possible to enumerate such a set of candidate values. Our algorithm performs this enumeration.

► **Lemma 12.** *There is a set consisting of $O((\frac{m-n}{\varepsilon})^2)$ vectors such that this set of vectors contains at least one vector with the required properties. Thus, the number of guesses including the guess of k_{\max} is at most $O((\frac{n}{\varepsilon})^2 \cdot m^3)$.*

In what follows, we let LB be the first component value of the guessed information, ρ be the second component value of the guessed information, and k, k' be the two guessed scenarios (where it is possible that $k' = 0$ is not an actual scenario). We will show that if there is a feasible solution to PR-SANTACLAUS for which the guessed information describes the solution and its expected value of the objective is OPT , then we will construct a feasible solution with expected value of the objective being at least $(1 - \Theta(\varepsilon)) \cdot \text{OPT}$. This means that the problem admits an EPTAS as desired.

We let $UB = LB \cdot \rho$. Furthermore, for every scenario κ with $k' < \kappa < k$, we set $q_{\kappa} = 0$. From now on we drop the assumption that the sum of all q values is 1 and instead of that we use the assumption that these are non-negative numbers with sum at most 1. This modification of the vector (q) decreases the expected value of the Santa Claus value of the optimal solution by at most $\varepsilon \cdot \text{OPT}$ (and it does not increase the objective function value of any feasible solution to our problem). The justification for this is as follows. From Lemma 11 it follows that there is a quadruple of integers $k, k', \rho, \log_{1+\varepsilon} LB$ (where each integer belongs to the set that we test for this integer) for which there is a solution of profit at least $\frac{\text{OPT}}{1+\varepsilon} - \varepsilon \cdot \text{OPT} \geq (1 - \varepsilon)^3 \cdot \text{OPT}$ with the first two properties stated in the lemma, and for every scenario κ such that $k' < \kappa < k$ the profit is at least zero.

Partitioning the input into two independent problems. Next, we use the above guessing step in order to partition the input into two independent inputs. First, a job $j \in \mathcal{J}$ is called huge if its size is strictly larger than UB , that is, if $p_j > UB$ (and otherwise it is non-huge).

We pack every huge job into its own bag and we let h denote the number of huge jobs (where we will show that $h \leq m - 1$ holds). Every huge job can cover one machine in every scenario $\kappa \geq k$ regardless of its exact size (because for all these scenarios we consider solutions for which OPT_{κ} is not larger than $\rho \cdot LB = UB$). On the other hand, the non-huge jobs (i.e., jobs of sizes at most UB) will be packed into bags of size at most $3 \cdot UB$. The packing of the non-huge jobs into bags will be optimized according to the scenarios of indexes at least k

(and at most k_{\max}), and we will ignore the scenarios of indexes smaller than k when we pack the non-huge jobs into bags. The resulting bags of the non-huge jobs together with the set of the huge jobs (which are bags consisting of a single job) will be packed into $\kappa \leq k'$ machines (in the corresponding scenario κ) based on existing EPTAS for the Santa Claus problem on identical machines (seeing bags as jobs). We will show that if indeed we use bags of sizes at most $3 \cdot UB$ when we pack the non-huge jobs into bags, then the scenarios of indexes at most k' can be ignored. We show that this holds for any collection of bags of this form, that is, where every huge job has its own bag and other bags have total sizes of jobs not exceeding $3 \cdot UB$. Thus, even though the bags are defined based on scenarios where the number of machines is at least k , still the scenarios with at most k' machines have good solutions. We will also show that it is possible to restrict ourselves to these types of collections of bags.

► **Lemma 13.** *If all guesses are correct, any partition into bags has at least one bag with a total size of jobs no larger than UB . In particular, there are at most $m - 1$ huge jobs.*

► **Lemma 14.** *An EPTAS for κ machines where $\kappa \leq k'$ which is applied on bags (instead of jobs) such that every huge job has its own bag and any additional bag has total size of jobs not exceeding $3 \cdot UB$ finds a solution of profit at least $\frac{OPT_\kappa}{(1-3\varepsilon)(1-\varepsilon)}$.*

Proof. Consider an optimal solution for κ machines and the original jobs. We show that this schedule can be modified into a schedule of the bags, such that the profit of the schedule is smaller by a factor not exceeding $1 - 3\varepsilon$. Thus, an optimal schedule of the bags is not worse, and by using an EPTAS the profit may decrease by another factor of $1 - \varepsilon$. The adaptation of the schedule is as follows. The huge jobs are assigned as before, since each of them has its own bag. All non-huge jobs are removed, and each machine receives bags until it is not possible to add another bag without exceeding the previous load or no unassigned bags remain. Given the property that the total size of bags is equal to the total size of jobs, it is either the case that the loads are equal to previous loads (in which case the value is unchanged) or there is at least one unassigned bag. In the latter case, no machine load decreased by more than an additive term of $3 \cdot UB$, and therefore the value is at least the previous one minus $3 \cdot UB$. To complete the assignment, all remaining bags are assigned arbitrarily. Since $OPT_\kappa \geq \frac{\rho \cdot LB}{\varepsilon} = \frac{UB}{\varepsilon}$ and the resulting value is at least $OPT_\kappa - 3 \cdot UB$, we find by $3 \cdot UB \leq 3\varepsilon \cdot OPT_\kappa$ that $OPT_\kappa - 3 \cdot UB \geq (1 - 3\varepsilon) \cdot OPT_\kappa$. ◀

► **Lemma 15.** *Consider the instance of our problem when we let $q_\kappa = 0$ for all $\kappa < k$ and for $\kappa > k_{\max}$. There exists a partition into bags where a profit at least OPT_κ can be obtained for any $\kappa \in [k, k_{\max}]$, the set of bags satisfies that every huge job is packed into its own bag, and each other bag consists of non-huge jobs of total size at most $2 \cdot UB$.*

Proof. The number of partitions into bags is finite for fixed m, n (it does not exceed m^n). Consider the set of partitions for which a solution of profit at least OPT_κ can be obtained for any $\kappa \in [k, k_{\max}]$. There is at least one such partition for the correct guess. For every partition it is possible to define a vector of m components, such that total sizes of bags appear in a non-increasing order. Consider the partition among the considered partitions for which the number of huge jobs that have their own bags is maximum, and out such partitions, one where the vector is lexicographically minimal. We claim that this partition satisfies the requirements.

Assume by contradiction that the number of huge jobs that do not have their own bags is not h . Consider a bag with a huge job that contains at least one additional job. This will be named the first bag. Consider a bag whose total size is at most UB , which must exist due to Lemma 13. This last bag does not have a huge job since the size of a huge job is above

UB , and we call it the second bag. Move all contents of the first bag into the second bag excluding one huge job. For any $\kappa \in [k, k_{\max}]$, since $\text{OPT}_\kappa \leq UB$, the assignment of bags to machines does not reduce the objective function value below OPT_κ . This holds because there is at most one machine whose total size was reduced (for every $\kappa \in [k, k_{\max}]$), but if such a machine exists, it still has a huge job whose size is above UB . Thus, the new partition is also one of the considered partitions. The new partition has a larger number of huge jobs assigned into their own bags, because the second bag was not such a bag and the first bag became such a bag. This contradicts the choice of assignment to bags. Thus, the partition into bags consists of h bags with huge jobs and $m - h \geq 1$ bags with non-huge jobs.

Next, consider only the components of the vector which do not correspond to bags of huge jobs. This vector is also minimal lexicographically out of vectors for partitions of the non-huge jobs into $m - h$ bags. Assume by contradiction that the first component is above $2 \cdot UB$. Consider the bag of the first component (called the first bag now) and a bag with a total size at most UB (called the second bag now). Move one job from the first bag to the second bag. The second bag now has a total size of at most $2 \cdot UB$ (since a non-huge job was moved). The first bag now has a smaller total size, but still larger than UB . The sorted vector is now smaller lexicographically, and it is still possible to obtain a solution of profit at least OPT_κ for any $\kappa \in [k, k_{\max}]$, similarly to the proof given here for huge jobs, which is a contradiction. \blacktriangleleft

In summary, we can focus on the scenarios interval $[k, k_{\max}]$, assume that huge jobs have their own bags, and remaining bags have total sizes not exceeding $2 \cdot UB$.

Modifying the input of scenarios with indexes at least k and at most k_{\max} . Motivated by the last partitioning of the original input into four parts, and the fact that only scenarios with indexes at least k and at most k_{\max} need to be considered, we apply the following transformation. First, for every $\kappa < k$ we let $q_\kappa = 0$, in the sense we can augment every solution to the remaining instance (with only a subset of scenarios) into an EPTAS for the original instance before this change to q . Furthermore, for every $\kappa > k_{\max}$ we let $q_\kappa = 0$ in the sense we can ignore the profit of such scenarios. Then, every huge job is packed into a separate bag. Such a bag suffices to cover one machine in every remaining scenario. Therefore, our second step in the transformation is the following one. We delete the huge jobs from \mathcal{J} , we decrease the index of each remaining scenario by h (in particular the new indexes in the support of q will be in the interval $[k - h, k_{\max} - h]$), and we enforce the condition that every bag size is at most $2 \cdot UB$.

As one can see, the transformations here are more complicated compared to those used in the previous section, and they have to be carefully designed and analyzed. However, now we are ready to apply methods that resemble the previous section. Using the fact that for every remaining scenario we have that OPT_κ is between LB and UB , and the fact that $\frac{UB}{LB} = \rho \leq (\frac{1}{\varepsilon})^{1/\varepsilon+1}$ we are able to apply the methods we have developed for the makespan minimization problem to obtain an EPTAS for PR-SANTACLUS, as we do next.

Rounding bag sizes. We next provide a method to gain structure on the set of feasible solutions that we need to consider for finding a near optimal solution. Given a feasible solution, the size of bag i denoted as $P(i)$ is the total size of jobs assigned to this bag, that is, $P(i) = \sum_{j: \sigma_1(j)=i} P_j$.

► **Lemma 16.** *There exists a solution of value at least $(1 - 3\varepsilon) \cdot \text{OPT}$ in which the size of every non-empty bag is in the interval $[\varepsilon \cdot LB, 2.5 \cdot UB]$.*

In what follows, we will decrease the size of a bag to be the next value of the form $(\varepsilon + r\varepsilon^2) \cdot LB$ for an integer $r \geq 0$ (except for empty bags for which the allowed size remains zero). Thus, we will not use precise sizes for bags but lower bounds on sizes, and we call them “allowed sizes” in what follows. For bags of allowed size zero the meaning remains that such a bag is empty. We let $P'(i)$ be this decreased (allowed) size of bag i , that is, $P'(i) = \max_{r: (\varepsilon + r\varepsilon^2) \cdot LB \leq P(i)} (\varepsilon + r\varepsilon^2) \cdot LB$. The allowed size is not smaller than $\varepsilon \cdot LB$ and it is smaller than the size by an additive term of at most $\varepsilon^2 \cdot LB$. Thus, for every subset of bags, the total allowed size of the bags in the set is at least $\frac{1}{1+\varepsilon}$ times the total size of the bags in the subset, we conclude that this rounding of the bag sizes will decrease the value of any solution by a multiplicative factor of at most $1 + \varepsilon$ (and therefore the expected value also decreases by at most this factor), and it will not increase the value of a solution for any scenario. Thus, in what follows, we will consider only bag sizes that belong to the set $B = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq 0, (\varepsilon + r\varepsilon^2) \cdot LB \leq 2.5 \cdot UB\} \cup \{0\}$. We use this set by Lemma 16. We conclude that the following holds.

► **Corollary 17.** *If the guessed information vector is satisfied by an optimal solution, then there is a solution of expected value of the Santa Claus value of at least $\frac{1-3\varepsilon}{1+\varepsilon} \cdot OPT$ that uses only bags with allowed sizes in B .*

Note that the allowed size of a bag may be slightly smaller than the actual total size of jobs assigned to the bag. Later, the allowed size acts as a lower bound (without an upper bound), and we sometimes take the allowed size into account in the calculation of machine completion times.

Rounding job sizes. We apply a similar rounding method for job sizes. Recall that by our transformation, every job j has size at most UB (since huge jobs were removed from the input). Next, we apply the following modification to the jobs of sizes at most $\varepsilon^2 \cdot LB$. While there is a pair of jobs of sizes at most $\varepsilon^2 \cdot LB$, we unite such a pair of jobs. If there is an additional job of size at most $\varepsilon^2 \cdot LB$, we delete it from the instance, and in the resulting solution (after applying the algorithm below on the resulting instance) we add the deleted job to an arbitrary bag. This deletion of the deleted job decreases the Santa Claus value of every scenario by at most $\varepsilon^2 \cdot LB$, so the resulting expected value of the Santa Claus value will be decreased by at most $\varepsilon^2 \cdot LB$. Adding the job back does not decrease the value. We consider the instance without this possibly deleted job, and we prove the following.

► **Lemma 18.** *The optimal expected value of the Santa Claus value of the instance resulting from the above modification of the job sizes among all solutions with allowed bag sizes in the following modified set $B' = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq -2, (\varepsilon + r\varepsilon^2) \cdot LB \leq 3 \cdot UB\} \cup \{0\}$ is at least $(1 - 2\varepsilon) \cdot \frac{1-3\varepsilon}{1+\varepsilon} \cdot OPT$.*

Next, we round down the size of every job j to be the next value that is of the form $(1 + \varepsilon)^r$ for an integer r . The size of every job cannot increase and it may decrease by a multiplicative factor of at most $1 + \varepsilon$. Job sizes are still larger than $\varepsilon^3 \cdot LB$ and not larger than $UB \leq (\frac{1}{\varepsilon})^{1/\varepsilon+1} \cdot LB$. Thus, the number of different job sizes is $O(\log_{1+\varepsilon}(\frac{1}{\varepsilon})^{1/\varepsilon+4}) \leq \frac{2}{\varepsilon^3} - 1$. We decrease the allowed size of each bag by another multiplicative factor of $1 + \varepsilon$. Bags sizes are rounded down again to the next value of the form $\varepsilon + r\varepsilon^2$, and since the smallest allowed size was $(\varepsilon - 2\varepsilon^2) \cdot LB$ and $\frac{\varepsilon - 2\varepsilon^2}{1+\varepsilon} \geq \varepsilon - 3\varepsilon^2$, the allowed bag sizes become $B'' = \{(\varepsilon + r\varepsilon^2) \cdot LB : r \in \mathbb{Z}, r \geq -3, (\varepsilon + r\varepsilon^2) \cdot LB \leq 3 \cdot UB\} \cup \{0\}$, and the expected value of the Santa Claus value of a feasible solution decreases by a multiplicative factor of at most $\frac{1-2\varepsilon}{1-3\varepsilon}$. By our guessing step and our transformation, there is a feasible solution with expected value of the Santa Claus value of at least $(1 - \varepsilon)(1 - 3\varepsilon)^2 \cdot OPT$.

A bag is called *tight* if the set of jobs of this bag cannot use a bag of a larger allowed size. Note that any solution where some bags are not tight can be converted into one where all bags are tight without decreasing the expected value of the objective.

► **Lemma 19.** *For any tight bag it holds that the allowed size of the bag is at least ϵ^2 times its actual size (the total size of jobs assigned to the bag, such that their rounded sizes are considered).*

The final steps of the EPTAS for Pr-SantaClaus. Our next step is to guess an approximated histogram of the optimal Santa Claus value in all scenarios. This step and the next step of formulating a template-configuration integer program of fixed dimension are similar to the ones we have established for PR-MAKESPAN. Using these additional steps we manage to prove our second main result.

► **Theorem 20.** *Problem PR-SANTACLUS admits an EPTAS.*

References

- 1 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, pages 493–500, 1997.
- 2 N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.
- 3 B. Awerbuch, Y. Azar, E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter. Load balancing in the l_p norm. In *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 383–391, 1995.
- 4 Y. Azar, L. Epstein, and R. van Stee. Resource augmentation in load balancing. *Journal of Scheduling*, 3(5):249–258, 2000.
- 5 E. Balkanski, T. Ou, C. Stein, and H.-T. Wei. Scheduling with speed predictions. In *Proc. of the 21st International Workshop on Approximation and Online Algorithms (WAOA'23)*, pages 74–89, 2023.
- 6 N. Bansal and K. R. Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM Journal on Computing*, 39(7):3311–3335, 2010. doi:10.1137/090772228.
- 7 N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proc. of the 38th Annual ACM Symposium on Theory of Computing (STOC'06)*, pages 31–40, 2006.
- 8 M. Brehob, E. Torng, and P. Uthaisombut. Applying extra-resource analysis to load balancing. *Journal of Scheduling*, 3(5):273–288, 2000.
- 9 M. Buchem, F. Eberle, H. K. Kasuya Rosado, K. Schewior, and A. Wiese. Scheduling on a stochastic number of machines. In *Proc. of the 27th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'24)*, pages 14:1–14:15, 2024.
- 10 A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM Journal on Computing*, 4(3):249–263, 1975. doi:10.1137/0204021.
- 11 L. Chen, K. Jansen, and G. Zhang. On the optimality of exact and approximation algorithms for scheduling problems. *Journal of Computer and System Sciences*, 96:1–32, 2018. doi:10.1016/J.JCSS.2018.03.005.
- 12 R. A. Cody and E. G. Coffman Jr. Record allocation for minimizing expected retrieval costs on drum-like storage devices. *Journal of the ACM*, 23(1):103–115, 1976. doi:10.1145/321921.321933.
- 13 E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978. doi:10.1137/0207001.

- 14 J. Csirik, H. Kellerer, and G. Woeginger. The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters*, 11(5):281–287, 1992. doi:10.1016/0167-6377(92)90004-M.
- 15 B. L. Deuermeier, D. K. Friesen, and M. A. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Discrete Mathematics*, 3(2):190–196, 1982.
- 16 R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, Berlin, 1999.
- 17 S. Dye, L. Stougie, and A. Tomagard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50(8):869–887, 2003.
- 18 F. Eberle, R. Hoeksma, N. Megow, L. Nölke, K. Schewior, and B. Simon. Speed-robust scheduling: sand, bricks, and rocks. *Mathematical Programming*, 197(2):1009–1048, 2023. doi:10.1007/S10107-022-01829-0.
- 19 W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
- 20 D. K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM Journal on Computing*, 13(1):170–181, 1984. doi:10.1137/0213013.
- 21 D. K. Friesen and B. L. Deuermeier. Analysis of greedy solutions for a replacement part sequencing problem. *Mathematics of Operations Research*, 6(1):74–87, 1981. doi:10.1287/MOOR.6.1.74.
- 22 R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.
- 23 R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.
- 24 D. S. Hochbaum. Various notions of approximations: Good, better, best and more. In D. S. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.
- 25 D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987. doi:10.1145/7531.7535.
- 26 K. Jansen, K.-M. Klein, and J. Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, 45(4):1371–1392, 2020. doi:10.1287/MOOR.2019.1036.
- 27 R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of the 15th annual ACM Symposium on Theory of Computing (STOC'83)*, pages 193–206, 1983.
- 28 H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/MOOR.8.4.538.
- 29 J. Y.-T. Leung and W.-D. Wei. Tighter bounds on a heuristic for a partition problem. *Information Processing Letters*, 56(1):51–57, 1995. doi:10.1016/0020-0190(95)00099-X.
- 30 J. Minařík and J. Sgall. Speed-robust scheduling revisited. In *Proc. of the 27th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'24)*, pages 8:1–8:20, 2024. doi:10.4230/LIPIcs.APPROX/RANDOM.2024.8.
- 31 M. Mitzenmacher and S. Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022. doi:10.1145/3528087.
- 32 K. Rustogi and V. A. Strusevich. Parallel machine scheduling: Impact of adding extra machines. *Operations Research*, 61(5):1243–1257, 2013. doi:10.1287/OPRE.2013.1208.
- 33 C. Stein and M. Zhong. Scheduling when you do not know the number of machines. *ACM Transactions on Algorithms*, 16(1):9:1–9:20, 2020. doi:10.1145/3340320.
- 34 G. J. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997. doi:10.1016/S0167-6377(96)00055-7.

A Faster Algorithm for Constrained Correlation Clustering

Nick Fischer  

INSAIT, Sofia University “St. Kliment Ohridski”, Bulgaria

Evangelos Kipouridis  

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Jonas Klausen  

BARC, University of Copenhagen, Denmark

Mikkel Thorup  

BARC, University of Copenhagen, Denmark

Abstract

In the Correlation Clustering problem we are given n nodes, and a preference for each pair of nodes indicating whether we prefer the two endpoints to be in the same cluster or not. The output is a clustering inducing the minimum number of violated preferences. In certain cases, however, the preference between some pairs may be too important to be violated. The constrained version of this problem specifies pairs of nodes that *must* be in the same cluster as well as pairs that *must not* be in the same cluster (hard constraints). The output clustering has to satisfy all hard constraints while minimizing the number of violated preferences.

Constrained Correlation Clustering is APX-Hard and has been approximated within a factor 3 by van Zuylen *et al.* [SODA '07]. Their algorithm is based on rounding an LP with $\Theta(n^3)$ constraints, resulting in an $\Omega(n^{3\omega})$ running time. In this work, using a more combinatorial approach, we show how to approximate this problem significantly faster at the cost of a slightly weaker approximation factor. In particular, our algorithm runs in $\tilde{O}(n^3)$ time (notice that the input size is $\Theta(n^2)$) and approximates Constrained Correlation Clustering within a factor 16.

To achieve our result we need properties guaranteed by a particular influential algorithm for (unconstrained) Correlation Clustering, the CC-PIVOT algorithm. This algorithm chooses a *pivot* node u , creates a cluster containing u and all its preferred nodes, and recursively solves the rest of the problem. It is known that selecting pivots at random gives a 3-approximation. As a byproduct of our work, we provide a derandomization of the CC-PIVOT algorithm that still achieves the 3-approximation; furthermore, we show that there exist instances where no ordering of the pivots can give a $(3 - \varepsilon)$ -approximation, for any constant ε .

Finally, we introduce a node-weighted version of Correlation Clustering, which can be approximated within factor 3 using our insights on Constrained Correlation Clustering. As the general weighted version of Correlation Clustering would require a major breakthrough to approximate within a factor $o(\log n)$, Node-Weighted Correlation Clustering may be a practical alternative.

2012 ACM Subject Classification Theory of computation \rightarrow Facility location and clustering

Keywords and phrases Clustering, Constrained Correlation Clustering, Approximation

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.32

Related Version *Full Version:* <https://arxiv.org/abs/2501.03154> [26]

Funding Jonas Klausen and Mikkel Thorup are part of BARC, Basic Algorithms Research Copenhagen, supported by VILLUM Foundation grants 16582 and 54451. 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).

Nick Fischer: 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. Parts of this work were done while the author was at Saarland University.

Acknowledgements We thank Lorenzo Beretta for his valuable suggestions on weighted sampling.



© Nick Fischer, Evangelos Kipouridis, Jonas Klausen, and Mikkel Thorup;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 32; pp. 32:1–32:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

Clustering is a fundamental task related to unsupervised learning, with many applications in machine learning and data mining. The goal of clustering is to partition a set of nodes into disjoint clusters, such that (ideally) all nodes within a cluster are similar, and nodes in different clusters are dissimilar. As no single definition best captures this abstract goal, a lot of different clustering objectives have been suggested.

Correlation Clustering is one of the most well studied such formulations for a multitude of reasons: Its definition is simple and natural, it does not need the number of clusters to be part of the input, and it has found success in many applications. Some few examples include automated labeling [1, 15], clustering ensembles [10], community detection [19, 36], disambiguation tasks [30], duplicate detection [5] and image segmentation [31, 40].

In Correlation Clustering we are given a graph $G = (V, E)$, and the output is a partition (clustering) $C = \{C_1, \dots, C_k\}$ of the vertex set V . We refer to the sets C_i of C as *clusters*. The goal is to minimize the number of edges between different clusters plus the number of non-edges inside of clusters. More formally, the goal is to minimize $|E \Delta E_C|$, the cardinality of the symmetric difference between E and E_C , where we define $E_C = \bigcup_{i=1}^k \binom{C_i}{2}$. In other words, the goal is to transform the input graph into a collection of cliques with the minimal number of edge insertions and deletions. An alternative description used by some authors is that we are given a *complete* graph where the edges are labeled either “+” (corresponding to the edges in our graph G) or “−” (corresponding to the non-edges in our graph G).

The problem is typically motivated as follows: Suppose that the input graph models the relationships between different entities which shall be grouped. An edge describes that we prefer its two endpoints to be clustered together, whereas a non-edge describes that we prefer them to be separated. In this formulation the cost of a correlation clustering is the number of violated preferences.

1.1 Previous Results

Correlation Clustering was initially introduced by Bansal, Blum, and Chawla [7], who proved that it is NP-Hard, and provided a deterministic constant-factor approximation, the constant being larger than 15,000. Subsequent improvements were based on rounding the natural LP: Charikar, Guruswami and Wirt gave a deterministic 4-approximation [17], Ailon, Charikar and Newman gave a randomized 2.5-approximation and proved that the problem is APX-Hard [3], while a deterministic 2.06-approximation was given by Chawla, Makarychev, Schramm and Yaroslavtsev [18]. The last result is near-optimal among algorithms rounding the natural LP, as its integrality gap is at least 2. In a breakthrough result by Cohen-Addad, Lee and Newman [23] a $(1.994 + \epsilon)$ -approximation using the Sherali-Adams relaxation broke the 2 barrier. It was later improved to $1.73 + \epsilon$ [22] by Cohen-Addad, Lee, Li and Newman, and even to 1.437 by Cao *et al.* [13]. There is also a combinatorial 1.847-approximation (Cohen-Addad *et al.* [24]).

Given the importance of Correlation Clustering, research does not only focus on improving its approximation factor. Another important goal is efficient running times without big sacrifices on the approximation factor. As the natural LP has $\Theta(n^3)$ constraints, using a state-of-the-art LP solver requires time $\Omega(n^{3\omega}) = \Omega(n^{7.113})$.

In order to achieve efficient running times, an algorithm thus has to avoid solving the LP using an all-purpose LP-solver, or the even more expensive Sherali-Adams relaxation; such algorithms are usually called *combinatorial* algorithms¹. Examples of such a direction can

¹ On a more informal note, combinatorial algorithms are often not only faster, but also provide deeper insights on a problem, compared to LP-based ones.

be seen in [3] where, along with their LP-based 2.5-approximation, the authors also design a combinatorial 3-approximation (the CC-PIVOT algorithm); despite its worse approximation, it enjoys the benefit of being faster. Similarly, much later than the 2.06-approximation [18], Veldt devised a faster combinatorial 6-approximation and a 4-approximation solving a less expensive LP [35].

Another important direction is the design of *deterministic* algorithms. For example, [3] posed as an open question the derandomization of CC-PIVOT. The question was (partially) answered affirmatively by [34]. Deterministic algorithms were also explicitly pursued in [35], and are a significant part of the technical contribution of [18].

Correlation Clustering has also been studied in different settings such as parameterized algorithms [28], sublinear and streaming algorithms [6, 12, 8, 9, 12, 16], massively parallel computation (MPC) algorithms [21, 14], and differentially private algorithms [11].

PIVOT. The CC-PIVOT algorithm [3] is a very influential algorithm for Correlation Clustering. It provides a 3-approximation and is arguably the simplest constant factor approximation algorithm for Correlation Clustering. It simply selects a node uniformly at random, and creates a cluster \mathcal{C} with this node and its neighbors in the (remaining) input graph. It then removes \mathcal{C} 's nodes and recurses on the remaining graph. Due to its simplicity, CC-PIVOT has inspired several other algorithms, such as algorithms for Correlation Clustering in the streaming model [6, 12, 8, 9, 12, 16] and algorithms for the more general Fitting Ultrametrics problem [2, 20].

One can define a meta-algorithm based on the above, where we do not necessarily pick the pivots uniformly at random. Throughout this paper, we use the term PIVOT algorithm to refer to an instantiation of the (Meta-)Algorithm 1². Obviously CC-PIVOT is an instantiation of PIVOT, where the pivots are selected uniformly at random.

■ **Algorithm 1** The PIVOT meta-algorithm. CC-PIVOT is an instantiation of PIVOT where pivots are selected uniformly at random.

```

procedure PIVOT( $G = (V, E)$ )
1   $C \leftarrow \emptyset$ 
2  while  $V \neq \emptyset$  do
3    Pick a pivot node  $u$ 
      /* an instantiation of PIVOT() only needs to specify how the
      pivot is selected in each iteration */
4    Add a cluster containing  $u$  and all its neighbors to  $C$ 
5    Remove  $u$ , its neighbors and all their incident edges from  $G$ 
6  return  $C$ 

```

The paper that introduced CC-PIVOT [3] posed as an open question the derandomization of the algorithm. The question was partially answered in the affirmative by [34]. Unfortunately there are two drawbacks with this algorithm. First, it requires solving the natural LP, which makes its running time equal to the pre-existing (better) 2.5-approximation. Second, this

² This is not to be confused with the more general pivoting paradigm for Correlation Clustering algorithms. In that design paradigm, the cluster we create for each pivot is not necessarily the full set of remaining nodes with which the pivot prefers to be clustered, but can be decided in any other way (e.g. randomly, based on a probability distribution related to an LP or more general hierarchies such as the Sherali-Adams hierarchy).

algorithm does not only derandomize the order in which pivots are selected, but also decides the cluster of each pivot based on an auxiliary graph (dictated by the LP) rather than based on the original graph. Therefore it is not an instantiation of PIVOT.

Weighted Correlation Clustering. In the weighted version of Correlation Clustering, we are also given a weight for each preference. The final cost is then the sum of weights of the violated preferences. An $O(\log n)$ -approximation for weighted Correlation Clustering is known by Demaine, Emanuel, Fiat and Immorlica [25]. In the same paper they show that the problem is equivalent to the Multicut problem, meaning that an $o(\log n)$ -approximation would require a major breakthrough. As efficiently approximating the general weighted version seems out of reach, research has focused on special cases for which constant-factor approximations are possible [32, 33].

Constrained Correlation Clustering. Constrained Correlation Clustering is an interesting variant of Correlation Clustering capturing the idea of critical pairs of nodes. To address these situations, Constrained Correlation Clustering introduces hard constraints in addition to the pairwise preferences. A clustering is valid if it satisfies all hard constraints, and the goal is to find a valid clustering of minimal cost. We can phrase Constrained Correlation Clustering as a weighted instance of Correlation Clustering: Simply give infinite weight to pairs associated with a hard constraint and weight 1 to all other pairs.

To the best of our knowledge, the only known solution to Constrained Correlation Clustering is given in the work of van Zuylen and Williamson who designed a deterministic 3-approximation [34]. The running time of this algorithm is $O(n^{3\omega})$, where $\omega < 2.3719$ is the matrix-multiplication exponent. Using the current best bound for ω , this is $\Omega(n^{7.113})$.

1.2 Our Contribution

Our main result is the following theorem. It improves the $\Omega(n^{7.113})$ running time of the state-of-the-art algorithm for Constrained Correlation Clustering while still providing a constant (but larger than 3) approximation factor³.

► **Theorem 1** (Constrained Correlation Clustering). *There is a deterministic algorithm for Constrained Correlation Clustering computing a 16-approximation in time $\tilde{O}(n^3)$.*

We first show how to obtain this result, but with a randomized algorithm that holds with high probability, instead of a deterministic one. In order to do so, we perform a (deterministic) preprocessing step and then use the CC-PIVOT algorithm. Of course CC-PIVOT alone, without the preprocessing step, would not output a clustering respecting the hard constraints. Its properties however (and more generally the properties of PIVOT algorithms) are crucial; we are not aware of any other algorithm that we could use instead and still satisfy all the hard constraints of Constrained Correlation Clustering after our preprocessing step.

To obtain our deterministic algorithm we derandomize the CC-PIVOT algorithm.

► **Theorem 2** (Deterministic PIVOT). *There are the following deterministic PIVOT algorithms for Correlation Clustering:*

- A combinatorial $(3 + \epsilon)$ -approximation, for any constant $\epsilon > 0$, in time $\tilde{O}(n^3)$.
- A non-combinatorial 3-approximation in time $\tilde{O}(n^5)$.

³ We write $\tilde{O}(T)$ to suppress polylogarithmic factors, i.e., $\tilde{O}(T) = T(\log T)^{O(1)}$.

We note that the final approximation of our algorithm for Constrained Correlation Clustering depends on the approximation of the applied PIVOT algorithm. If it was possible to select the order of the pivots in a way that guarantees a better approximation, this would immediately improve the approximation of our Constrained Correlation Clustering algorithm. For this reason, we study lower bounds for PIVOT; currently, we know of instances for which selecting the pivots at random doesn't give a better-than-3-approximation in expectation [3]; however, for these particular instances there *does* exist a way to choose the pivots that gives better approximations. Ideally, we want a lower bound applying for any order of the pivots (such as the lower bound for the generalized PIVOT solving the Ultrametric Violation Distance problem in [20]). We show that our algorithm is optimal, as there exist instances where no ordering of the pivots will yield a better-than-3-approximation.

► **Theorem 3 (PIVOT Lower Bound).** *There is no constant $\epsilon > 0$ for which there exists a PIVOT algorithm for Correlation Clustering with approximation factor $3 - \epsilon$.*

We also introduce the Node-Weighted Correlation Clustering problem, which is related to (but incomparable, due to their asymmetric assignment of weights) a family of problems introduced in [36]. As weighted Correlation Clustering is equivalent to Multicut, improving over the current $\Theta(\log n)$ -approximation seems out of reach. The advantage of our alternative type of weighted Correlation Clustering is that it is natural and approximable within a constant factor.

In Node-Weighted Correlation Clustering we assign weights to the nodes, rather than to pairs of nodes. Violating the preference between nodes u, v with weights ω_u and ω_v incurs cost $\omega_u \cdot \omega_v$. We provide three algorithms computing (almost-)3-approximations for Node-Weighted Correlation Clustering:

► **Theorem 4 (Node-Weighted Correlation Clustering, Deterministic).** *There are the following deterministic algorithms for Node-Weighted Correlation Clustering:*

- *A combinatorial $(3 + \epsilon)$ -approximation, for any constant $\epsilon > 0$, in time $\tilde{O}(n^3)$.*
- *A non-combinatorial 3-approximation in time $O(n^{7.116})$.*

► **Theorem 5 (Node-Weighted Correlation Clustering, Randomized).** *There is a randomized combinatorial algorithm for Node-Weighted Correlation Clustering computing an expected 3-approximation in time $O(n + m)$ with high probability $1 - 1/\text{poly}(n)$.*

1.3 Overview of Our Techniques

Constrained Correlation Clustering. We obtain a faster algorithm for Constrained Correlation Clustering by

1. modifying the input graph using a subroutine aware of the hard-constraints, and
2. applying a PIVOT algorithm on this modified graph.

In fact, no matter what PIVOT algorithm is used, the output clustering respects all hard constraints when the algorithm is applied on the modified graph.

To motivate this two-step procedure, we note that inputs exist where *no* PIVOT algorithm, if applied to the unmodified graph, would respect the hard constraints. One such example is the cycle on four vertices, with two vertex-disjoint edges made into hard constraints.

The solution of [34] is similar to ours, as it also modifies the graph before applying a Correlation Clustering algorithm. However, both their initial modification and the following Correlation Clustering algorithm require solving the standard LP, which is expensive ($\Omega(n^{7.113})$ time). In our case both steps are implemented with deterministic and combinatorial algorithms which brings the running time down to $\tilde{O}(n^3)$.

For the first step, our algorithm carefully modifies the input graph so that on one hand the optimal cost is not significantly changed, and on the other hand any PIVOT algorithm on the transformed graph returns a clustering that respects all hard constraints. For the second step, we use a deterministic combinatorial PIVOT algorithm.

Concerning the effect of modifying the graph, roughly speaking we get that the final approximation factor is $(2 + \sqrt{5}) \cdot \alpha + 3$, where α is the approximation factor of the PIVOT algorithm we use. Plugging in $\alpha = 3 + \epsilon$ from Theorem 2 we get the first combinatorial constant-factor approximation for Constrained Correlation Clustering in $\tilde{O}(n^3)$ time.

Node-Weighted Correlation Clustering. We generalize the deterministic combinatorial techniques from before to the Node-Weighted Correlation Clustering problem. In addition, we also provide a very efficient randomized algorithm for the problem. It relies on a weighted random sampling technique.

One way to view the algorithm is to reduce Node-Weighted Correlation Clustering to an instance of Constrained Correlation Clustering, with the caveat that the new instance's size depends on the weights (and can thus even be exponential). Each node u is replaced by a set of nodes of size related to u 's weight and these nodes have constraints forcing them to be in the same cluster.

We show that we can simulate a simple randomized PIVOT algorithm on that instance, where instead of sampling uniformly at random, we sample with probabilities proportional to the weights. Assuming polynomial weights, we can achieve this in linear time. To do so, we design an efficient data structure supporting such sampling and removal of elements.

It is easy to implement such a data structure using any balanced binary search tree, but the time for constructing it and applying all operations would be $O(n \log n)$. Using a non-trivial combination of the Alias Method [38, 37] and Rejection Sampling, we achieve a linear bound.

Due to space constraints the presentation of our algorithms for Node-Weighted Correlation Clustering is deferred to the full version of the paper [26].

Deterministic PIVOT algorithms. Our algorithms are based on a simple framework by van Zuylen and Williamson [34]. In this framework we assign a nonnegative “charge” to each pair of nodes. Using these charges, a PIVOT algorithm decides which pivot to choose next. The approximation factor depends on the total charge (as compared with the cost of an optimal clustering), and the minimum charge assigned to any bad triplet (an induced subgraph $K_{1,2}$).

The reason why these bad triplets play an important role is that for any bad triplet, any clustering needs to pay at least 1. To see this, let uvw be a bad triplet with uv being the only missing edge. For a clustering to pay 0, it must be the case that both uw and vw are together. However, this would imply that uv are also together although they prefer not to.

Our combinatorial $(3 + \epsilon)$ -approximation uses the multiplicative weights update method, which can be intuitively described as follows: We start with a tiny charge on all pairs. Then we repeatedly find a bad triplet uvw with currently minimal charge (more precisely: for which the sum of the charges of uv, vw, wu is minimal), and scale the involved charges by $1 + \epsilon$. One can prove that this eventually results in an almost-optimal distribution of charges, up to rescaling.

For this purpose it suffices to show that the total assigned charge is not large compared to the cost of the optimal correlation clustering. We do so by observing that our algorithm $(1 + \epsilon)$ -approximates the covering LP of Figure 1, which we refer to as the *charging LP*.

Our faster deterministic non-combinatorial algorithm solves the charging LP using an LP solver tailored to covering LPs [4, 39]. An improved solver for covering LPs would directly improve the running time of this algorithm.

■ **Figure 1** The primal and dual LP relaxations for Correlation Clustering, which we refer to as the *charging LP*. $T(G)$ is the set of all bad triplets in G .

$$\begin{aligned} \min \quad & \sum_{uv \in \binom{V}{2}} x_{uv} \\ \text{s.t.} \quad & x_{uv} + x_{vw} + x_{wu} \geq 1 \quad \forall uvw \in T(G), \\ & x_{uv} \geq 0 \quad \forall uv \in \binom{V}{2} \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{uvw \in T(G)} y_{uvw} \\ \text{s.t.} \quad & \sum_{w:uvw \in T(G)} y_{uvw} \leq 1 \quad \forall uv \in \binom{V}{2}, \\ & y_{uvw} \geq 0 \quad \forall uvw \in T(G) \end{aligned}$$

Lower Bound. Our lower bound is obtained by taking a complete graph K_n for some even number of vertices n , and removing a perfect matching. Each vertex in the resulting graph is adjacent to all but one other vertex and so *any* PIVOT algorithm will partition the vertices into a large cluster of $n - 1$ vertices and a singleton cluster. A non PIVOT algorithm, however, is free to create just a single cluster of size n , at much lower cost. The ratio between these solutions tends to 3 with increasing n .

We note that in [3] the authors proved that CC-PIVOT's analysis is tight. That is, its expected approximation factor is not better than 3. However, their lower bound construction (a complete graph K_n minus one edge) only works for CC-PIVOT, not for PIVOT algorithms in general.

1.4 Open Problems

We finally raise some open questions.

1. Can we improve the approximation factor of Constrained Correlation Clustering from 16 to 3 while keeping the running time at $\tilde{O}(n^3)$?
2. We measure the performance of a PIVOT algorithm by comparing it to the best correlation clustering obtained by *any* algorithm. But as Theorem 3 proves, there is no PIVOT algorithm with an approximation factor better than 3. If we instead compare the output to the best correlation clustering obtained by a *PIVOT algorithm*, can we get better guarantees (perhaps even an exact algorithm in polynomial time)?
3. In the Node-Weighted Correlation Clustering problem, we studied the natural objective of minimizing the total cost $\omega_v \cdot \omega_u$ of all violated preferences uv . Are there specific applications of this problem? Can we achieve similar for other cost functions such as $\omega_v + \omega_u$?

2 Preliminaries

We denote the set $\{1, \dots, n\}$ by $[n]$. We denote all subsets of size k of a set A by $\binom{A}{k}$. The symmetric difference between two sets A, B is denoted by $A \triangle B$. We write $\text{poly}(n) = n^{O(1)}$ and $\tilde{O}(n) = n(\log n)^{O(1)}$.

In this paper all graphs $G = (V, E)$ are undirected and unweighted. We typically set $n = |V|$ and $m = |E|$. For two disjoint subsets $U_1, U_2 \subseteq V$, we denote the set of edges with one endpoint in U_1 and the other in U_2 by $E(U_1, U_2)$. The subgraph of G induced by vertex-set U_1 is denoted by $G[U_1]$. For vertices u, v, w we often abbreviate the (unordered) set $\{u, v\}$ by uv and similarly write uvw for $\{u, v, w\}$. We say that uvw is a *bad triplet* in G if the induced subgraph $G[uvw]$ contains exactly two edges (i.e., is isomorphic to $K_{1,2}$). Let $T(G)$ denote the set of bad triplets in G . We say that the edge set E_C of a clustering $C = \{C_1, \dots, C_k\}$ of V is the set of pairs with both endpoints in the same set in C . More formally, $E_C = \bigcup_{i=1}^k \binom{C_i}{2}$.

We now formally define the problems of interest.

► **Definition 6** (Correlation Clustering). *Given a graph $G = (V, E)$, output a clustering $C = \{C_1, \dots, C_k\}$ of V with edge set E_C minimizing $|E \triangle E_C|$.*

An algorithm for Correlation Clustering is said to be a PIVOT algorithm if it is an instantiation of Algorithm 1 (Page 3). That is, an algorithm which, based on some criterion, picks an unclustered node u (the *pivot*), creates a cluster containing u and its unclustered neighbors in (V, E) , and repeats the process until all nodes are clustered. In particular, the algorithm may not modify the graph in other ways before choosing a pivot.

The constrained version of Correlation Clustering is defined as follows.

► **Definition 7** (Constrained Correlation Clustering). *Given a graph $G = (V, E)$, a set of friendly pairs $F \subseteq \binom{V}{2}$ and a set of hostile pairs $H \subseteq \binom{V}{2}$, compute a clustering $C = \{C_1, \dots, C_k\}$ of V with edge set E_C such that no pair $uv \in F$ has u, v in different clusters and no pair $uv \in H$ has u, v in the same cluster. The clustering C shall minimize $|E \triangle E_C|$.*

We also introduce Node-Weighted Correlation Clustering, a new related problem that may be of independent interest.

► **Definition 8** (Node-Weighted Correlation Clustering). *Given a graph $G = (V, E)$ and positive weights $\{\omega_u\}_{u \in V}$ on the nodes, compute a clustering $C = \{C_1, \dots, C_k\}$ of V with edge set E_C minimizing*

$$\sum_{uv \in E \triangle E_C} \omega_u \cdot \omega_v.$$

For simplicity, we assume that the weights are bounded by $\text{poly}(n)$, and thereby fit into a constant number of word RAM cells of size $w = \Theta(\log n)$. We remark that our randomized algorithm would be a polynomial (but not linear) time one if we allowed the weights to be of exponential size.

The Node-Weighted Correlation Clustering problem clearly generalizes Correlation Clustering since we pay $w(u) \cdot w(v)$ (instead of 1) for each pair uv violating a preference.

3 Combinatorial Algorithms for Constrained Correlation Clustering

Let us fix the following notation: A connected component in (V, F) is a *supernode*. The set of supernodes partitions V and is denoted by SN . Given a node u , we let $s(u)$ be the unique supernode containing u . Two supernodes U, W are *hostile* if there exists a hostile pair uw with $u \in U, w \in W$. Two supernodes U, W are *connected* if $|E(U, W)| \geq 1$. Two supernodes U, W are β -*connected* if $|E(U, W)| \geq \beta \cdot |U| \cdot |W|$.

The first step of our combinatorial approach is to transform the graph G into a more manageable form G' , see procedure TRANSFORM of Algorithm 2. The high-level idea is that in G' :

1. If uv is a friendly pair, then u and v are connected and have the same neighborhood.
2. If uv is a hostile pair, then u and v are not connected and have no common neighbor.
3. An $O(1)$ -approximation of the G' instance is also an $O(1)$ -approximation of the G instance.

As was already noticed in [34], Properties 1 and 2 imply that a PIVOT algorithm on G' gives a clustering satisfying the hard constraints. Along with Property 3 and our deterministic combinatorial PIVOT algorithm for Correlation Clustering in Theorem 2, we prove Theorem 1. Properties 1 and 2 (related to correctness) and the running time ($\tilde{O}(n^3)$) of our algorithm are relatively straightforward to prove. Due to space constraints, their proofs can be found in the full version of the paper [26]. In this section we instead focus on the most technically challenging part, the approximation guarantee.

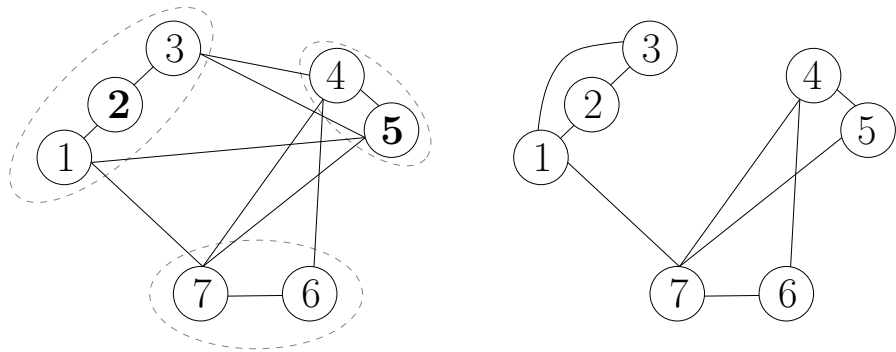
Our algorithm works as follows (see also Figure 2): If some supernode is hostile to itself, then it outputs that no clustering satisfies the hard constraints. Else, starting from the edge set E , it adds all edges within each supernode. Then it drops all edges between hostile supernodes. Subsequently, it repeatedly detects hostile supernodes that are connected with the same supernode, and drops one edge from each such connection. Finally, for each β -connected pair of supernodes, it connects all their nodes if $\beta > \frac{3-\sqrt{5}}{2}$, and disconnects them otherwise⁴.

From a high-level view, the first two modifications are directly related to the hard constraints: If u_1, u_2 are friendly and u_2, u_3 are friendly, then any valid clustering has u_1, u_3 in the same cluster, even if a preference discourages it. Similarly, if u_1, u_2 are friendly, u_3, u_4 are friendly, but u_1, u_3 are hostile, then any valid clustering has u_2, u_4 in different clusters, even if a preference discourages it. Our first two modifications simply make the preferences consistent with the hard constraints.

The third modification guarantees that hostile supernodes share no common neighbor. A PIVOT algorithm will thus never put their nodes in the same cluster, as the hostility constraints require. Concerning the cost, notice that if hostile supernodes U_1, U_2 are connected with supernode U_3 , then no valid clustering can put all three of them in the same cluster. Therefore we always need to pay either for the connections between U_1 and U_3 , or for the connections between U_2 and U_3 .

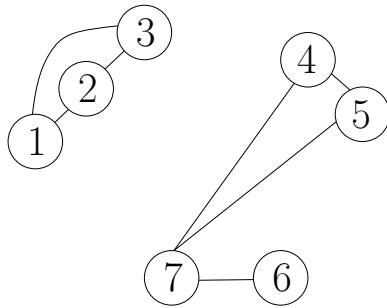
Finally, after the rounding step, for each pair of supernodes U_1, U_2 , the edge set $E(U_1, U_2)$ is either empty or the full set of size $|U_1| \cdot |U_2|$. This ensures that a PIVOT algorithm always puts all nodes of a supernode in the same cluster, thus also obeying the friendliness constraints. Concerning the cost of the rounded instance, a case analysis shows that it is always within a constant factor of the cost of the instance before rounding.

⁴ The constant $\frac{3-\sqrt{5}}{2}$ optimizes the approximation factor. The natural choice of 0.5 would still give a constant approximation factor, albeit slightly worse.

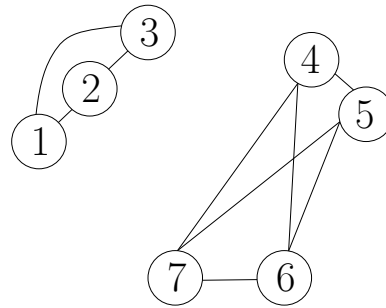


(a) The original graph. The set of friendly pairs is $F = \{\{1, 2\}, \{2, 3\}, \{4, 5\}, \{6, 7\}\}$, and the only hostile pair in H is $\{2, 5\}$.

(b) Line 3 introduces edge $\{1, 3\}$, and Line 4 disconnects the supernodes containing 2 and 5.



(c) Line 6 removes the pair of edges $\{1, 7\}$ and $\{4, 6\}$ because 1, 4 are in hostile supernodes while 6, 7 are in the same supernode.



(d) Line 9 introduces all edges connecting supernodes $\{4, 5\}$ and $\{6, 7\}$ because there were enough edges between them already.

■ **Figure 2** Illustrates an application of $\text{TRANSFORM}(G, F, H)$ (Algorithm 2). In the transformed graph, for any two supernodes U_1, U_2 , either all pairs with an endpoint in U_1 and an endpoint in U_2 share an edge, or none of them do. Furthermore, all pairs within a supernode are connected and no hostile supernodes are connected.

Formally, let E' be the edge set of the transformed graph G' , let E_3 be the edge set at Line 8 of Algorithm 2 (exactly before the rounding step), OPT be the edge set of an optimal clustering for E satisfying the hard constraints described by F and H , OPT' be the edge set of an optimal clustering for the preferences defined by E' , and E_C be the edge set of the clustering returned by our algorithm. Finally, let α be the approximation factor of the PIVOT algorithm used.

► **Lemma 9.** *Given an instance (V, E, F, H) of Constrained Correlation Clustering, if two nodes u_1, u_2 are in the same supernode, then they must be in the same cluster.*

Proof. The proof follows by “in the same cluster” being a transitive property.

More formally, u_1, u_2 are in the same connected component in (V, F) , as $s(u_1) = s(u_2)$. Thus, there exists a path from u_1 to u_2 . We claim that all nodes in a path must be in the same cluster. This is trivial if the path is of length 0 ($u_1 = u_2$) or of length 1 ($u_1 u_2 \in F$). Else, the path is $u_1, w_1, \dots, w_k, u_2$ for some $k \geq 1$. We inductively have that all of w_1, \dots, w_k, u_2 must be in the same cluster, and u_1 must be in the same cluster with w_1 because $u_1 w_1 \in F$. Therefore, all nodes in the path must be in the same cluster with w_1 . ◀

We now show that it is enough to bound the symmetric difference between E and E' .

■ **Algorithm 2** The procedure `CONSTRAINEDCLUSTER` is given a graph $G = (V, E)$ describing the preferences, a set of friendly pairs F and a set of hostile pairs H . It creates a new graph G' using the procedure `TRANSFORM` and uses any `PIVOT` algorithm on G' to return a clustering.

```

procedure TRANSFORM( $G = (V, E), F, H$ )
1  Compute the connected components of  $(V, F)$ 
   // Impossible iff some pair must both be and not be in the same
   // cluster.
2  if  $\exists U \in SN$  hostile to itself then return  $G' = (\emptyset, \emptyset)$ 
   // Connect nodes in the same supernode.
3   $E_1 \leftarrow E \cup \{uv \in \binom{V}{2} \mid s(u) = s(v)\}$ 
   // Disconnect pairs in hostile supernodes.
4   $E_2 \leftarrow E_1 \setminus \{uv \in \binom{V}{2} \mid s(u) \text{ and } s(v) \text{ are hostile}\}$ 
   // While hostile supernodes  $U_1, U_2$  are both connected with super-
   // node  $U_3$ , drop an edge between  $U_1, U_3$  and an edge between  $U_2, U_3$ 
5   $E_3 \leftarrow E_2$ 
6  while  $\exists U_1, U_2, U_3 \in \binom{SN}{3}$  such that  $U_1, U_2$  are hostile and
    $\exists u_1 \in U_1, u_2 \in U_2, u_3 \in U_3, u'_3 \in U_3$  such that  $u_1 u_3 \in E_3, u_2 u'_3 \in E_3$  do
7  |  $E_3 \leftarrow E_3 \setminus \{u_1 u_3, u_2 u'_3\}$ 
   // Round connections between pairs of supernodes
8   $E_4 \leftarrow E_3$ 
9  foreach  $\{U_1, U_2\} \in \binom{SN}{2}$  do
10 |  $E_{U_1, U_2} \leftarrow \{u_1 u_2 \mid u_1 \in U_1, u_2 \in U_2\}$ 
11 | if  $|E_{U_1, U_2} \cap E_4| > \frac{3-\sqrt{5}}{2} |U_1| \cdot |U_2|$  then  $E_4 \leftarrow E_4 \cup E_{U_1, U_2}$ 
12 | else  $E_4 \leftarrow E_4 \setminus E_{U_1, U_2}$ 
13 return  $G' = (V, E_4)$ 

procedure CONSTRAINEDCLUSTER( $G = (V, E), F, H$ )
14  $G' \leftarrow$  TRANSFORM( $G = (V, E), F, H$ )
15 if  $G' = (\emptyset, \emptyset)$  then return “Impossible”
16 return PIVOT( $G'$ )

```

► **Lemma 10.** *The cost of our clustering C is $|E \triangle E_C| \leq (\alpha + 1)|E \triangle E'| + \alpha|E \triangle \text{OPT}|$.*

Proof. The symmetric difference of sets satisfies the triangle inequality; we therefore have

$$|E \triangle E_C| \leq |E \triangle E'| + |E' \triangle E_C|.$$

C is an α -approximation for $G' = (V, E')$ and thus $|E' \triangle E_C| \leq \alpha|E' \triangle \text{OPT}'| \leq \alpha|E' \triangle \text{OPT}|$. Therefore:

$$|E \triangle E_C| \leq |E \triangle E'| + \alpha|E' \triangle \text{OPT}| \leq |E \triangle E'| + \alpha|E' \triangle E| + \alpha|E \triangle \text{OPT}|.$$

with the second inequality following by applying the triangle inequality again. ◀

In order to upper bound $|E \triangle E'|$ by the cost of the optimal clustering $|E \triangle \text{OPT}|$, we first need to lower bound the cost of the optimal clustering.

32:12 A Faster Algorithm for Constrained Correlation Clustering

► **Lemma 11.** *Let S be the set of all pairs of distinct supernodes U, W that are in the same cluster in OPT . Then $|E \triangle \text{OPT}| \geq \sum_{\{U, W\} \in S} |E(U, W) \triangle E_3(U, W)|$.*

Proof. The high-level idea is that when a node is connected to two hostile nodes, then any valid clustering needs to pay for at least one of these edges. Extending this fact to supernodes, we construct an edge set of size $\sum_{\{U, W\} \in S} |E(U, W) \triangle E_3(U, W)|$ such that the optimal clustering needs to pay for each edge in this set.

First, for any $\{U, W\} \in S$ it holds that $E(U, W) \triangle E_3(U, W) = E(U, W) \setminus E_3(U, W)$ because Line 3 (Algorithm 2) does not modify edges between pairs of distinct supernodes, and Lines 4 and 6 only remove edges.

Each edge of $E(U, W) \setminus E_3(U, W)$ is the result of applying Line 6, seeing as Line 4 only removes edges from hostile pairs of supernodes. Thus each edge $uw \in E(U, W) \setminus E_3(U, W)$ can be paired up with a unique edge $xy \in E$ which is removed together with uw . Without loss of generality it holds that $x \in U, y \in Z$ for some supernode Z different from U and W . Due to the way Line 6 chooses edges it must be the case that Z and W are hostile, hence $xy \in E \triangle \text{OPT}$.

Summing over all pairs of clustered supernodes gives the result stated in the lemma. ◀

We are now ready to bound $|E \triangle E'|$.

► **Lemma 12.** $|E \triangle E'| \leq (1 + \sqrt{5})|E \triangle \text{OPT}|$

Proof. To prove this, we first charge each pair of nodes in a way such that the total charge is at most $2|E \triangle \text{OPT}|$. Then we partition the pairs of nodes into 5 different sets, and show that the size of the intersection between $E \triangle E'$ and each of the 5 sets is at most $\frac{1+\sqrt{5}}{2}$ times the total charge given to the pairs in the given set.

The first three sets contain the pairs across non-hostile supernodes; out of them the first one is the most technically challenging, requiring a combination of Lemma 11 (related to Line 6 of Algorithm 2) and a direct analysis on $E \triangle \text{OPT}$, as neither of them would suffice on their own. The analysis of the second and third sets relate to the rounding in Line 9. The fourth set contains pairs across hostile supernodes, while the fifth set contains pairs within supernodes. Their analysis is directly based on the hard constraints.

Let us define our charging scheme: first, each pair of nodes is charged if the optimal clustering pays for it, i.e. if this pair is in $E \triangle \text{OPT}$. We further put a charge on the pairs $uw \in E \triangle E_3$ which connect supernodes that are clustered together in OPT . Notice that the number of such edges is a lower bound on $|E \triangle \text{OPT}|$ by Lemma 11. Therefore the total charge over all pairs of nodes is at most $2|E \triangle \text{OPT}|$ and no pair is charged twice.

Case 1. Consider two distinct supernodes U, W that are not hostile, which have more than $\frac{3-\sqrt{5}}{2}|U| \cdot |W|$ edges between them in E , and have at most $\frac{3-\sqrt{5}}{2}|U| \cdot |W|$ edges in E_3 . Then the rounding of Line 9 removes all edges between them. Therefore $|E(U, W) \triangle E'(U, W)| = |E(U, W)| \leq |U| \cdot |W|$. If OPT separates U and W , then the pairs are charged $|E(U, W)|$; else they are charged $|U| \cdot |W| - |E(U, W)|$ due to the part of the charging scheme related to $E \triangle \text{OPT}$. In the latter case, they are also charged $|E(U, W)| - |E_3(U, W)|$ due to the part of the charging scheme related to Lemma 11. Therefore they are charged at least

$$\begin{aligned} |U| \cdot |W| - |E(U, W)| + |E(U, W)| - |E_3(U, W)| &= |U| \cdot |W| - |E_3(U, W)| \\ &\geq |U| \cdot |W| - \frac{3-\sqrt{5}}{2}|U| \cdot |W|. \end{aligned}$$

Thus, in the worst case, these pairs contribute

$$\max \left\{ \frac{|E(U, W)|}{|E(U, W)|}, \frac{|E(U, W)|}{|U| \cdot |W| - \frac{3-\sqrt{5}}{2}|U| \cdot |W|} \right\} \leq \frac{1}{1 - \frac{3-\sqrt{5}}{2}} = \frac{1 + \sqrt{5}}{2}$$

times more in $|E \triangle E'|$ compared to their charge.

Case 2. Consider two distinct supernodes U, W that are not hostile, which have more than $\frac{3-\sqrt{5}}{2}|U| \cdot |W|$ edges between them in E , and more than $\frac{3-\sqrt{5}}{2}|U| \cdot |W|$ edges in E_3 . Then the rounding of Line 9 will include all $|U| \cdot |W|$ edges between them. Thus we have $|E(U, W) \triangle E'(U, W)| = |U| \cdot |W| - |E(U, W)| < (1 - \frac{3-\sqrt{5}}{2})|U| \cdot |W|$. If OPT separates U and W it pays for $|E(U, W)| > \frac{3-\sqrt{5}}{2}|U| \cdot |W|$ pairs. Otherwise it pays $|U| \cdot |W| - |E(U, W)|$. Thus, in the worst case, these pairs contribute $\frac{1 - \frac{3-\sqrt{5}}{2}}{\frac{3-\sqrt{5}}{2}} = \frac{1+\sqrt{5}}{2}$ times more in $|E \triangle E'|$ compared to their charge.

Case 3. If two distinct supernodes U, W are not hostile and have at most $\frac{3-\sqrt{5}}{2}|U| \cdot |W|$ edges between them in E , then they also have at most that many edges in E_3 as we only remove edges between such supernodes. There are thus no edges between them in E' , meaning that $|E(U, W) \triangle E'(U, W)| = |E(U, W)| \leq \frac{3-\sqrt{5}}{2}|U| \cdot |W|$. If OPT separates U, W it pays for $|E(U, W)|$ pairs related to the connection between U, W ; else it pays for $|U| \cdot |W| - |E(U, W)| \geq (1 - \frac{3-\sqrt{5}}{2})|U| \cdot |W| > \frac{3-\sqrt{5}}{2}|U| \cdot |W|$. Thus these pairs' contribution in $|E \triangle E'|$ is at most as much as their charge.

Case 4. Pairs uv with $s(u) \neq s(v)$ and $s(u)$ hostile with $s(v)$ are not present in E' . That is because by Line 4 no pair of hostile supernodes is connected; then Line 6 only removes edges, and Line 9 does not add any edge between $s(u)$ and $s(v)$ as they had $0 \leq \frac{3-\sqrt{5}}{2}|s(u)| \cdot |s(v)|$ edges between them. The edge uv is also not present in OPT as $s(u)$ and $s(v)$ are not in the same cluster because they are hostile. These pairs' contribution in $|E \triangle E'|$ is exactly equal to their charge.

Case 5. Pairs uv with $s(u) = s(v)$ are present in E' by Line 3 and the fact that all subsequent steps only modify edges whose endpoints are in different supernodes. The pair uv is also present in OPT, by Lemma 9. Therefore these pairs' contribution in $|E \triangle E'|$ is exactly equal to their charge.

In the worst case, the pairs of each of the five sets contribute at most $\frac{1+\sqrt{5}}{2}$ times more in $|E \triangle E'|$ compared to their charge, which proves our lemma. \blacktriangleleft

We are now ready to prove the main theorem.

Proof of Theorem 1. In Theorem 2 we established that there is a deterministic combinatorial PIVOT algorithm computing a Correlation Clustering with approximation factor $\alpha = 3 + \epsilon$ in time $\tilde{O}(n^3)$, for any constant $\epsilon > 0$. Using this algorithm in Algorithm 2 gives a valid clustering. By Lemmas 10 and 12, its approximation factor is bounded by $(\alpha + 1) \cdot (1 + \sqrt{5}) + \alpha$. This is less than 16 for $\epsilon = 0.01$. \blacktriangleleft

4 PIVOT Algorithms for Correlation Clustering

4.1 Lower Bound

First we prove Theorem 3 which states that there is no PIVOT algorithm for Correlation Clustering with approximation factor better than 3.

Proof of Theorem 3. Let $G = ([2n], E)$ for some integer n , where the edge set E contains all pairs of nodes except for pairs of the form $(2k + 1, 2k + 2)$. In other words, the edge set of G contains all edges except for a perfect matching.

Note that if we create a single cluster containing all nodes, then the cost is exactly n . On the other hand, let u be the first choice that a PIVOT algorithm makes. If u is even, let $v = u - 1$, otherwise let $v = u + 1$. By definition of G , v is the only node not adjacent

to u . Therefore, the algorithm creates two clusters – one containing all nodes except for v , and one containing only v . There are $2n - 2$ edges across the two clusters, and $n - 1$ missing edges in the big cluster, meaning that the cost is $3n - 3$.

Therefore, the approximation factor of any PIVOT algorithm is at least $(3n-3)/n = 3 - \frac{3}{n}$. This proves the theorem, as for any constant less than 3, there exists a sufficiently large n such that $3 - \frac{3}{n}$ is larger than that constant. ◀

4.2 Optimal Deterministic PIVOT: 3-Approximation

A *covering* LP is a linear program of the form $\min_x \{ cx \mid Ax \geq b \}$ where A, b, c , and x are restricted to vectors and matrices of non-negative entries. Covering LPs can be solved more efficiently than LPs in general and we rely on the following known machinery to prove Theorem 2:

► **Theorem 13** (Covering LPs, Combinatorial [29, 27]). *Any covering LP with at most N nonzero entries in the constraint matrix can be $(1 + \epsilon)$ -approximated by a combinatorial algorithm in time $\tilde{O}(N\epsilon^{-3})$.⁵*

► **Theorem 14** (Covering LPs, Non-Combinatorial [4, 39]). *Any covering LP with at most N nonzero entries in the constraint matrix can be $(1 + \epsilon)$ -approximated in time $\tilde{O}(N\epsilon^{-1})$.*

Of the two theorems, the time complexity of the algorithm promised by Theorem 14 is obviously better. However, the algorithm of Theorem 13 is remarkably simple in our setting and could thus prove to be faster in practice. Note that either theorem suffices to obtain a $(3 + \epsilon)$ -approximation for Correlation Clustering in $\tilde{O}(n^3)$ time, for constant $\epsilon > 0$.

For completeness, and in order to demonstrate how simple the algorithm from Theorem 13 is in our setting, we include the pseudocode as Algorithm 3. In [26] we formally prove that Algorithm 3 indeed has the properties promised by Theorem 13.

■ **Algorithm 3** The combinatorial algorithm to $(1 + O(\epsilon))$ -approximate the LP in Figure 1 (Page 7) using the multiplicative weights update method. The general method was given by Garg and Könemann [29] and later refined by Fleischer [27]. We here use the notation $m(x) = \min_{uvw \in T(G)} x_{uv} + x_{vw} + x_{wu}$.

```

procedure CHARGE( $G = (V, E)$ )
1  Initialize  $x_{uv}, x_{uv}^* \leftarrow 1$  for all  $uv \in \binom{V}{2}$ 
2  while  $\sum_{uv} x_{uv} < B := \binom{n}{2}(1 + \epsilon)^{1/\epsilon} / (1 + \epsilon)$  do
3      Find a bad triplet  $uvw$  minimizing  $x_{uv} + x_{vw} + x_{wu}$ 
4       $x_{uv} \leftarrow (1 + \epsilon) \cdot x_{uv}$ 
5       $x_{vw} \leftarrow (1 + \epsilon) \cdot x_{vw}$ 
6       $x_{wu} \leftarrow (1 + \epsilon) \cdot x_{wu}$ 
7      if  $(\sum_{uv} x_{uv}) / m(x) < (\sum_{uv} x_{uv}^*) / m(x^*)$  then
8          foreach  $uv \in \binom{V}{2}$  do  $x_{uv}^* \leftarrow x_{uv}$ 
9  return  $\{ x_{uv}^* / m(x^*) \}_{uv}$ 

```

⁵ The running time we state seems worse by a factor of ϵ^{-1} as compared to the theorems in [29, 27]. This is because the authors assume access to a machine model with exact arithmetic of numbers of size exponential in ϵ^{-1} . We can simulate this model using fixed-point arithmetic with a running time overhead of $\tilde{O}(\epsilon^{-1})$.

The solution found by Algorithm 3 is used together with the framework by van Zuylen and Williamson [34], see CLUSTER in Algorithm 4. CLUSTER is discussed further in the full version [26], where the following lemmas are proven.

■ **Algorithm 4** The PIVOT algorithm by van Zuylen and Williamson [34]. Given a graph G and a good charging $\{x_{uv}\}_{uv}$ (in the sense of Lemma 15), it computes a correlation clustering.

```

procedure CLUSTER( $G = (V, E), x = \{x_{uv}\}_{uv \in \binom{V}{2}}$ )
1   $C \leftarrow \emptyset$ 
2  while  $V \neq \emptyset$  do
3    Pick a pivot node  $u \in V$  minimizing
      
$$\frac{\sum_{vw:uvw \in T(G)} 1}{\sum_{vw:uvw \in T(G)} x_{vw}}$$

4    Add a cluster containing  $u$  and all its neighbors to  $C$ 
5    Remove  $u$ , its neighbors and all their incident edges from  $G$ 
6  return  $C$ 

```

► **Lemma 15** (Correctness of CLUSTER). *Assume that $x = \{x_{uv}\}_{uv}$ is a feasible solution to the LP in Figure 1. Then CLUSTER(G, x) computes a correlation clustering of cost $3 \sum_{uv} x_{uv}$. In particular, if x is an α -approximate solution to the LP (for some $\alpha \geq 1$), then CLUSTER(G, x) returns a 3α -approximate correlation clustering.*

► **Lemma 16** (Running Time of CLUSTER, [34]). *CLUSTER(G, x) runs in time $O(n^3)$.*

Given Theorems 13 and 14 we quickly prove Theorem 2.

Proof of Theorem 2. We compute a $(1 + \epsilon/3)$ -approximate solution x of the charging LP using Theorem 13 (that is, using the procedure CHARGE(G)). Plugging this solution x into CLUSTER(G, x) returns a $(3 + \epsilon)$ -approximate correlation clustering by Lemma 15. The total running time is bounded by $O(n^3)$ by Lemma 16 plus $\tilde{O}(n^3 \epsilon^{-3})$ by Theorem 13 (note that there are n^3 constraints, each affecting only a constant number of variables, hence the number of nonzeros in the constraint matrix is $N \leq O(n^3)$). For constant $\epsilon > 0$, this becomes $\tilde{O}(n^3)$.

To obtain a 3-approximation, we observe that any correlation clustering has cost less than $\binom{n}{2}$. Hence, we can run the previous algorithm with $\epsilon = 1/\binom{n}{2}$ and the $(3 + \epsilon)$ -approximate solution is guaranteed to also be 3-approximate. The running time would be bounded by $\tilde{O}(n^9)$. To improve upon this, we use the covering LP solver in Theorem 14 which runs in time $\tilde{O}(n^3 \epsilon^{-1})$. By again setting $\epsilon = 1/\binom{n}{2}$, the running time becomes $\tilde{O}(n^5)$. ◀

References


- 1 Rakesh Agrawal, Alan Halverson, Krishnamurthy Kenthapadi, Nina Mishra, and Panayiotis Tsaparas. Generating labels from clicks. In Ricardo Baeza-Yates, Paolo Boldi, Berthier A. Ribeiro-Neto, and Berkant Barla Cambazoglu, editors, *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, pages 172–181. ACM, 2009. doi:10.1145/1498759.1498824.

- 2 Nir Ailon and Moses Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. *SIAM J. Comput.*, 40(5):1275–1291, 2011. Announced at FOCS’05. doi:10.1137/100806886.
- 3 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. Announced in STOC 2005. doi:10.1145/1411509.1411513.
- 4 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering LP solvers – achieving width-independence and -convergence. *Math. Program.*, 175(1-2):307–353, 2019. doi:10.1007/s10107-018-1244-x.
- 5 Arvind Arasu, Christopher Ré, and Dan Suciu. Large-scale deduplication with constraints using dedupalog. In Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng, editors, *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 952–963. IEEE Computer Society, 2009. doi:10.1109/ICDE.2009.43.
- 6 Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. *CoRR*, abs/2109.14528, 2021. arXiv:2109.14528.
- 7 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.
- 8 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 720–731. IEEE, 2022. doi:10.1109/FOCS54457.2022.00074.
- 9 Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Single-pass streaming algorithms for correlation clustering. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 819–849. SIAM, 2023. doi:10.1137/1.9781611977554.CH33.
- 10 Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. *Knowl. Inf. Syst.*, 35(1):1–32, 2013. doi:10.1007/s10115-012-0522-9.
- 11 Mark Bun, Marek Eliás, and Janardhan Kulkarni. Differentially private correlation clustering. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1136–1146. PMLR, 2021. URL: <http://proceedings.mlr.press/v139/bun21a.html>.
- 12 Melanie Cambus, Fabian Kuhn, Etna Lindy, Shreyas Pai, and Jara Uitto. *A $(3+\epsilon)$ -Approximate Correlation Clustering Algorithm in Dynamic Streams*, pages 2861–2880. SIAM, 2024. doi:10.1137/1.9781611977912.101.
- 13 Nairen Cao, Vincent Cohen-Addad, Euiwoong Lee, Shi Li, Alantha Newman, and Lukas Vogl. Understanding the cluster linear program for correlation clustering. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1605–1616. ACM, 2024. doi:10.1145/3618260.3649749.
- 14 Nairen Cao, Shang-En Huang, and Hsin-Hao SU. *Breaking 3-Factor Approximation for Correlation Clustering in Polylogarithmic Rounds*, pages 4124–4154. SIAM, 2024. doi:10.1137/1.9781611977912.143.
- 15 Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 377–386. ACM, 2008. doi:10.1145/1367497.1367549.
- 16 Sayak Chakrabarty and Konstantin Makarychev. Single-pass pivot algorithm for correlation clustering. keep it simple! *CoRR*, abs/2305.13560, 2023. doi:10.48550/arXiv.2305.13560.
- 17 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005. Announced in FOCS 2003. doi:10.1016/j.jcss.2004.10.012.




- 18 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 219–228. ACM, 2015. doi:10.1145/2746539.2746604.
- 19 Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2213–2221, 2012. URL: <https://proceedings.neurips.cc/paper/2012/hash/1e6e0a04d20f50967c64dac2d639a577-Abstract.html>.
- 20 Vincent Cohen-Addad, Chenglin Fan, Euiwoong Lee, and Arnaud de Mesmay. Fitting metrics and ultrametrics with minimum disagreements. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 301–311. IEEE, 2022. doi:10.1109/FOCS54457.2022.00035.
- 21 Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078. PMLR, 2021. URL: <http://proceedings.mlr.press/v139/cohen-addad21b.html>.
- 22 Vincent Cohen-Addad, Euiwoong Lee, Shi Li, and Alantha Newman. Handling correlated rounding error via preclustering: A 1.73-approximation for correlation clustering. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1082–1104. IEEE, 2023. doi:10.1109/FOCS57990.2023.00065.
- 23 Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with sherali-adams. *CoRR*, abs/2207.10889, 2022. doi:10.48550/arXiv.2207.10889.
- 24 Vincent Cohen-Addad, David Rasmussen Lolck, Marcin Pilipczuk, Mikkel Thorup, Shuyi Yan, and Hanwen Zhang. Combinatorial correlation clustering. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1617–1628. ACM, 2024. doi:10.1145/3618260.3649712.
- 25 Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theor. Comput. Sci.*, 361(2-3):172–187, 2006. doi:10.1016/j.tcs.2006.05.008.
- 26 Nick Fischer, Evangelos Kipouridis, Jonas Klausen, and Mikkel Thorup. A faster algorithm for constrained correlation clustering, 2025. arXiv:2501.03154.
- 27 Lisa Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 1001–1010. SIAM, 2004. URL: <http://dl.acm.org/citation.cfm?id=982792.982942>.
- 28 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/j.jcss.2014.04.015.
- 29 Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS ’98*, page 300, USA, 1998. IEEE Computer Society.
- 30 Dmitri V. Kalashnikov, Zhaoqi Chen, Sharad Mehrotra, and Rabia Nuray-Turan. Web people search via connection analysis. *IEEE Trans. Knowl. Data Eng.*, 20(11):1550–1565, 2008. doi:10.1109/TKDE.2008.78.

- 31 Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang Dong Yoo. Higher-order correlation clustering for image segmentation. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1530–1538, 2011. URL: <https://proceedings.neurips.cc/paper/2011/hash/98d6f58ab0dafbb86b083a001561bb34-Abstract.html>.
- 32 Domenico Mandaglio, Andrea Tagarelli, and Francesco Gullo. Correlation clustering with global weight bounds. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and José Antonio Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part II*, volume 12976 of *Lecture Notes in Computer Science*, pages 499–515. Springer, 2021. doi:10.1007/978-3-030-86520-7_31.
- 33 Gregory J. Puleo and Olgica Milenkovic. Correlation clustering with constrained cluster sizes and extended weights bounds. *SIAM J. Optim.*, 25(3):1857–1872, 2015. doi:10.1137/140994198.
- 34 Anke van Zuylen and David P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Math. Oper. Res.*, 34(3):594–620, 2009. Announced in SODA 2007. doi:10.1287/moor.1090.0385.
- 35 Nate Veldt. Correlation clustering via strong triadic closure labeling: Fast approximation algorithms and practical lower bounds. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 22060–22083. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/veldt22a.html>.
- 36 Nate Veldt, David F. Gleich, and Anthony Wirth. A correlation clustering framework for community detection. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 439–448. ACM, 2018. doi:10.1145/3178876.3186110.
- 37 Michael D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Trans. Software Eng.*, 17(9):972–975, 1991. doi:10.1109/32.92917.
- 38 Alastair J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10:127–128(1), April 1974.
- 39 Di Wang, Satish Rao, and Michael W. Mahoney. Unified acceleration method for packing and covering problems via diameter reduction. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 50:1–50:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.50.
- 40 Julian Yarkony, Alexander T. Ihler, and Charless C. Fowlkes. Fast planar correlation clustering for image segmentation. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012 – 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, volume 7577 of *Lecture Notes in Computer Science*, pages 568–581. Springer, 2012. doi:10.1007/978-3-642-33783-3_41.

Metric Dimension and Geodetic Set Parameterized by Vertex Cover

Florent Foucaud   

Université Clermont Auvergne, CNRS,
Mines Saint-Étienne, Clermont Auvergne INP,
LIMOS, 63000 Clermont-Ferrand, France

Liana Khazaliya   

Technische Universität Wien, Austria

Fionn Mc Inerney   



Telefónica Scientific Research, Barcelona, Spain

Prafullkumar Tale   

Indian Institute of Science Education and Research
Pune, India

Esther Galby  

Department of Computer Science and Engineering,
Chalmers University of Technology and University of Gothenburg, Sweden

Shaohua Li  

School of Computer Science and Engineering,
Central South University, Changsha, China

Roohani Sharma  

University of Bergen, Norway

Abstract

For a graph G , a subset $S \subseteq V(G)$ is called a *resolving set* of G if, for any two vertices $u, v \in V(G)$, there exists a vertex $w \in S$ such that $d(w, u) \neq d(w, v)$. The METRIC DIMENSION problem takes as input a graph G on n vertices and a positive integer k , and asks whether there exists a resolving set of size at most k . In another *metric-based graph problem*, GEODETIC SET, the input is a graph G and an integer k , and the objective is to determine whether there exists a subset $S \subseteq V(G)$ of size at most k such that, for any vertex $u \in V(G)$, there are two vertices $s_1, s_2 \in S$ such that u lies on a shortest path from s_1 to s_2 .

These two classical problems are known to be intractable with respect to the natural parameter, i.e., the solution size, as well as most structural parameters, including the feedback vertex set number and pathwidth. We observe that both problems admit an FPT algorithm running in $2^{\mathcal{O}(\text{vc}^2)} \cdot n^{\mathcal{O}(1)}$ time, and a kernelization algorithm that outputs a kernel with $2^{\mathcal{O}(\text{vc})}$ vertices, where vc is the vertex cover number. We prove that unless the Exponential Time Hypothesis (ETH) fails, METRIC DIMENSION and GEODETIC SET, even on graphs of bounded diameter, do not admit

- an FPT algorithm running in $2^{\mathcal{O}(\text{vc}^2)} \cdot n^{\mathcal{O}(1)}$ time, nor
- a kernelization algorithm that does not increase the solution size and outputs a kernel with $2^{\mathcal{O}(\text{vc})}$ vertices.

We only know of one other problem in the literature that admits such a tight algorithmic lower bound with respect to vc . Similarly, the list of known problems with exponential lower bounds on the number of *vertices* in kernelized instances is very short.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, ETH-based Lower Bounds, Kernelization, Vertex Cover, Metric Dimension, Geodetic Set

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.33

Related Version *Full Version:* <https://arxiv.org/abs/2405.01344> [19]

Funding *Florent Foucaud:* ANR project GRALMECO (ANR-21-CE48-0004), French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25), International Research Center “Innovation Transportation and Production Systems” of the I-SITE CAP 20-25.

Liana Khazaliya: Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029]; Austrian Science Fund (FWF) [10.55776/Y1329]; European Union’s Horizon 2020 COFUND programme [LogiCS@TUWien, grant agreement No. 101034440].



© Florent Foucaud, Esther Galby, Liana Khazaliya, Shaohua Li, Fionn Mc Inerney, Roohani Sharma, and Prafullkumar Tale;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 33; pp. 33:1–33:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Shaohua Li: National Natural Science Foundation of China under Grant 62472449.

Fionn Mc Inerney: Smart Networks and Services Joint Undertaking (SNS JU) under the EU's Horizon Europe and innovation programme under Grant Agreement No. 101139067 (ELASTIC).

1 Introduction

In this article, we study two *metric-based* graph problems, one of which is defined through distances, while the other relies on shortest paths. Metric-based graph problems are ubiquitous in computer science; for example, the classical (SINGLE-SOURCE) SHORTEST PATH, (GRAPHIC) TRAVELING SALESPERSON or STEINER TREE problems fall into this category. Those are fundamental problems, often stemming from applications in network design, for which a considerable amount of algorithmic research has been done. Metric-based graph packing and covering problems, like DISTANCE DOMINATION [29] or SCATTERED SET [30], have recently gained a lot of attention. Their non-local nature leads to non-trivial algorithmic properties that differ from most graph problems with a more local nature. We focus here on the METRIC DIMENSION and GEODETIC SET problems, which arise from network monitoring and network design, respectively. These two problems have far-reaching applications, as exemplified by, e.g., the recent work [3] where it was shown that enumerating minimal solution sets for METRIC DIMENSION and GEODETIC SET in (general) graphs and split graphs, respectively, is equivalent to the enumeration of minimal transversals in hypergraphs, whose solvability in total-polynomial time is arguably the most important open problem in algorithmic enumeration. Formally, these two problems are defined as follows.

METRIC DIMENSION

Input: A graph G on n vertices and a positive integer k .

Question: Does there exist $S \subseteq V(G)$ such that $|S| \leq k$ and, for any pair of vertices $u, v \in V(G)$, there exists a vertex $w \in S$ with $d(w, u) \neq d(w, v)$?

GEODETIC SET

Input: A graph G on n vertices and a positive integer k .

Question: Does there exist $S \subseteq V(G)$ such that $|S| \leq k$ and, for any vertex $u \in V(G)$, there are two vertices $s_1, s_2 \in S$ such that u lies on a shortest path from s_1 to s_2 ?

METRIC DIMENSION dates back to the 1970s [26, 38], whereas GEODETIC SET was introduced in 1993 [25]. The non-local nature of these problems has since posed interesting algorithmic challenges. METRIC DIMENSION was first shown to be NP-complete in general graphs in Garey and Johnson's book [22], and this was later extended to many restricted graph classes (see "Related work" below). GEODETIC SET was proven to be NP-complete in [25], and later shown to be NP-hard on restricted graph classes as well.

As these two problems are NP-hard even in very restricted cases, it is natural to ask for ways to confront this hardness. In this direction, the parameterized complexity paradigm allows for a more refined analysis of a problem's complexity. In this setting, we associate each instance I of a problem with a parameter ℓ , and are interested in algorithms running in $f(\ell) \cdot |I|^{\mathcal{O}(1)}$ time for some computable function f . Parameterized problems that admit such an algorithm are called fixed-parameter tractable (FPT for short) with respect to the considered parameter. Under standard complexity assumptions, parameterized problems that are hard for the complexity class $W[1]$ or $W[2]$ do not admit such algorithms.

This approach, however, had limited success for these two problems. In the seminal paper [28], METRIC DIMENSION was proven to be $W[2]$ -hard parameterized by the solution size k , even on subcubic bipartite graphs. Similarly, GEODETIC SET is $W[2]$ -hard parameterized by the solution size [15, 31], even on chordal bipartite graphs. These initial hardness results drove the ensuing meticulous study of the problems under structural parameterizations: we present an overview in the “Related work” below. In this article, we focus on the *vertex cover number*, denoted by vc , of the input graph and prove the following positive results.

► **Theorem 1.** *METRIC DIMENSION and GEODETIC SET admit*

- FPT algorithms running in $2^{\mathcal{O}(vc^2)} \cdot n^{\mathcal{O}(1)}$ time, and
- kernelization algorithms that output kernels with $2^{\mathcal{O}(vc)}$ vertices.

The second set of results follows from simple reduction rules, and was also observed in [28] for METRIC DIMENSION. The first set of results builds on the second set by using a simple, but critical observation. For METRIC DIMENSION, this also improves upon the $2^{2^{\mathcal{O}(vc)}} \cdot n^{\mathcal{O}(1)}$ algorithm mentioned in [28]. However, our main technical contribution is in proving that these results are optimal assuming the Exponential Time Hypothesis (ETH).

► **Theorem 2.** *Unless the ETH fails, METRIC DIMENSION and GEODETIC SET do not admit*

- FPT algorithms running in $2^{\mathcal{O}(vc^2)} \cdot n^{\mathcal{O}(1)}$ time, nor
- kernelization algorithms that do not increase the solution size and output kernels with $2^{\mathcal{O}(vc)}$ vertices,

even on graphs of bounded diameter.

Both these statements constitute a rare set of results. Indeed, we know of only one other problem that admits a lower bound of the form $2^{\mathcal{O}(vc^2)} \cdot n^{\mathcal{O}(1)}$ and a matching upper bound [1], whereas such results parameterized by pathwidth are mentioned in [36, 37]. Very recently, the authors in [7] also proved a similar result with respect to solution size. Similarly, the list of known problems with exponential lower bounds on the number of *vertices* in kernelized instances is very short. To the best of our knowledge, the only known results of this kind (i.e., ETH-based lower bounds on the number of vertices in a kernel) are for EDGE CLIQUE COVER [13], BICLIQUE COVER [9], STEINER TREE [35], STRONG METRIC DIMENSION [20], B-NCTD⁺ [8], LOCATING DOMINATING SET [7], and TELEPHONE BROADCASTING [39]. For METRIC DIMENSION, the above also improves a result of [24], which states that METRIC DIMENSION parameterized by $k + vc$ does not admit a polynomial kernel unless the polynomial hierarchy collapses to its third level. Indeed, the result of [24] does not rule out a kernel of super-polynomial or sub-exponential size.

Recently, Foucaud et al. [20] proved that, unless the ETH fails, METRIC DIMENSION and GEODETIC SET on graphs of bounded diameter do not admit $2^{2^{\mathcal{O}(tw)}} \cdot n^{\mathcal{O}(1)}$ -time algorithms, thereby establishing one of the first such results for NP-complete problems. Note that $n \succ vc \succ fvs \succ tw$ and $n \succ vc \succ td \succ pw \succ tw$ in the parameter hierarchy, where n is the order, fvs is the feedback vertex set number, td is the treedepth, pw is the pathwidth, and tw is the treewidth of the graph. They further proved that their lower bound also holds for fvs and td in the case of METRIC DIMENSION, and for td in the case of GEODETIC SET [20]. A simple brute-force algorithm enumerating all possible candidates runs in $2^{\mathcal{O}(n)}$ time for both of these problems. Thus, the next natural question is whether such a lower bound for METRIC DIMENSION and GEODETIC SET can be extended to larger parameters, in particular vc . Our first results answer this question in the negative. Together with the lower bounds with respect to vc , this establishes the boundary between parameters yielding single-exponential and double-exponential running times for METRIC DIMENSION and GEODETIC SET.

Before moving forward, we highlight the parallels and differences between Foucaud et al. [20] and our work. Their aim was to establish double-exponential lower bounds for NP-complete problems, and to do so they focused on the restriction of the problems to graphs of bounded treewidth and diameter. Our objective is to closely examine one of the very few tractable results for METRIC DIMENSION and GEODETIC SET on general graphs by focusing on the vertex cover parameter. While we use some gadgets from [20], overall our reductions significantly differ from the corresponding reductions in that article. Note that we need to “control” the vertex cover number of the reduced graph, whereas the corresponding reductions by Foucaud et al. [20] only need to “control” the treewidth.

Related Work. We mention here results concerning structural parameterizations of METRIC DIMENSION and GEODETIC SET, and refer the reader to the full version of [20] for a more comprehensive overview of applications and related work regarding these two problems.

As previously mentioned, METRIC DIMENSION is $W[2]$ -hard parameterized by the solution size k , even in subcubic bipartite graphs [28]. Several other parameterizations have been studied for this problem, on which we elaborate next (see also [21, Figure 1]). It was proven that there is an XP algorithm parameterized by the feedback edge set number [18], and FPT algorithms parameterized by the max leaf number [17], the modular-width and the treelength plus the maximum degree [2], the treedepth and the clique-width plus the diameter [23], and the distance to cluster (co-cluster, respectively) [21]. Recently, an FPT algorithm parameterized by the treewidth in chordal graphs was given in [5]. On the negative side, METRIC DIMENSION is $W[1]$ -hard parameterized by the pathwidth even on graphs of constant degree [4], para-NP-hard parameterized by the pathwidth [33], and $W[1]$ -hard parameterized by the combined parameter feedback vertex set number plus pathwidth [21].

The parameterized complexity of GEODETIC SET was first addressed in [31], in which it was observed that the reduction from [15] implies that the problem is $W[2]$ -hard parameterized by the solution size (even for chordal bipartite graphs). This motivated the authors of [31] to investigate structural parameterizations of GEODETIC SET. They proved the problem to be $W[1]$ -hard for the combined parameters solution size, feedback vertex set number, and pathwidth, and FPT for the parameters treedepth, modular-width (more generally, clique-width plus diameter), and feedback edge set number [31]. The problem was also shown to be FPT on chordal graphs when parameterized by the treewidth [6].

2 Preliminaries

For an integer a , we let $[a] = \{1, \dots, a\}$.

Graph theory. We use standard graph-theoretic notation and refer the reader to [14] for any undefined notation. For an undirected graph G , the sets $V(G)$ and $E(G)$ denote its set of vertices and edges, respectively. Two vertices $u, v \in V(G)$ are *adjacent* or *neighbors* if $(u, v) \in E(G)$. The *open neighborhood* of a vertex $u \in V(G)$, denoted by $N(u) := N_G(u)$, is the set of vertices that are neighbors of u . The *closed neighborhood* of a vertex $u \in V(G)$ is denoted by $N[u] := N_G[u] := N_G(u) \cup \{u\}$. For any $X \subseteq V(G)$ and $u \in V(G)$, $N_X(u) = N_G(u) \cap X$. Any two vertices $u, v \in V(G)$ are *true twins* if $N[u] = N[v]$, and are *false twins* if $N(u) = N(v)$. For a subset S of $V(G)$, we say that the vertices in S are true (false, respectively) twins if, for any $u, v \in S$, u and v are true (false, respectively) twins. The *distance* between two vertices $u, v \in V(G)$ in G , denoted by $d(u, v) := d_G(u, v)$, is the length of a (u, v) -shortest path in G . For a subset S of $V(G)$, we define $N[S] = \bigcup_{v \in S} N[v]$ and $N(S) = N[S] \setminus S$. For

a graph G , a set $X \subseteq V(G)$ is said to be a *vertex cover* if $V(G) \setminus X$ is an independent set. We denote by $\text{vc}(G)$ the size of a minimum vertex cover in G . When G is clear from the context, we simply say vc . A vertex is *simplicial* if its neighborhood forms a clique. Observe that any simplicial vertex v does not belong to any shortest path between any pair x, y of vertices (both distinct from v). Hence, the following holds.

► **Observation 3** ([10]). *If a graph G contains a simplicial vertex v , then v belongs to any geodesic set of G . Specifically, every degree-1 vertex belongs to any geodesic set of G .*

Metric Dimension. A subset of vertices $S' \subseteq V(G)$ *resolves* a pair of vertices $u, v \in V(G)$ if there exists a vertex $w \in S'$ such that $d(w, u) \neq d(w, v)$. A vertex $u \in V(G)$ is *distinguished* by a subset of vertices $S' \subseteq V(G)$ if, for any $v \in V(G) \setminus \{u\}$, there exists a vertex $w \in S'$ such that $d(w, u) \neq d(w, v)$.

Parameterized Complexity. An instance of a parameterized problem Π comprises an input I , which is an input of the classical instance of the problem, and an integer ℓ , which is called the parameter. A problem Π is said to be *fixed-parameter tractable* or in FPT if given an instance (I, ℓ) of Π , we can decide whether or not (I, ℓ) is a YES-instance of Π in $f(\ell) \cdot |I|^{\mathcal{O}(1)}$ time, for some computable function f whose value depends only on ℓ .

A *kernelization* algorithm for Π is a polynomial-time algorithm that takes as input an instance (I, ℓ) of Π and returns an *equivalent* instance (I', ℓ') of Π , where $|I'|, \ell' \leq f(\ell)$, where f is a function that depends only on ℓ . If such an algorithm exists for Π , we say that Π admits a kernel of *size* $f(\ell)$. If f is a polynomial or exponential function of ℓ , we say that Π admits a polynomial or exponential kernel, respectively. If Π is a graph problem, then I contains a graph, say G , and I' contains a graph, say G' . In this case, we say that Π admits a kernel with $f(\ell)$ vertices if the number of vertices of G' is at most $f(\ell)$.

It is typical to describe a kernelization algorithm as a series of reduction rules. A *reduction rule* is a polynomial-time algorithm that takes as an input an instance of a problem and outputs another (usually reduced) instance. A reduction rule said to be *applicable* on an instance if the output instance is different from the input instance. A reduction rule is *safe* if the input instance is a YES-instance if and only if the output instance is a YES-instance.

The Exponential Time Hypothesis (ETH) roughly states that n -variable 3-SAT cannot be solved in $2^{o(n)}$ time. For more on parameterized complexity and related terminologies, we refer the reader to the recent book by Cygan et al. [12].

3-Partitioned-3-SAT. Our lower bound proofs consist of reductions from the 3-PARTITIONED-3-SAT problem. This version of 3-SAT was introduced in [32] and is defined as follows.

3-PARTITIONED-3-SAT

Input: A formula ψ in 3-CNF form, together with a partition of the set of its variables into three disjoint sets $X^\alpha, X^\beta, X^\gamma$, with $|X^\alpha| = |X^\beta| = |X^\gamma| = N$, and such that no clause contains more than one variable from each of X^α, X^β , and X^γ .

Question: Determine whether ψ is satisfiable.

Organization of the paper. We start with the results for METRIC DIMENSION, which are then followed by those for GEODETIC SET. For each, we first present the algorithms and then the reductions. Since space requirements prohibit us from presenting our reductions in

detail, we give an outline that discusses the main technical ideas behind our reductions for getting the lower-bound results for both METRIC DIMENSION and GEODETIC SET. For the complete formal proofs, we refer the reader to the full version of this paper [19].

3 Metric Dimension: Algorithms for Vertex Cover Parameterization

In this section, we prove Theorem 1 for METRIC DIMENSION. The kernelization algorithm exhaustively applies the following reduction rule.

▷ **Reduction Rule 1.** If there exist three vertices $u, v, x \in I$ such that u, v, x are false twins, then delete x and decrease k by one.

Proof that Reduction Rule 1 is safe. Since u, v, x are false twins, $N(u) = N(v) = N(x)$. This implies that, for any vertex $w \in V(G) \setminus \{u, v, x\}$, $d(w, v) = d(w, u) = d(w, x)$. Hence, any resolving set that excludes at least two vertices in $\{u, v, x\}$ cannot resolve all three pairs $\{u, v\}$, $\{u, x\}$, and $\{v, x\}$. As the vertices in $\{u, v, x\}$ are distance-wise indistinguishable from the remaining vertices, we can assume, without loss of generality, that any resolving set contains both u and x . Hence, any pair of vertices in $V(G) \setminus \{u, x\}$ that is resolved by x is also resolved by u . In other words, if S is a resolving set of G , then $S \setminus \{x\}$ is a resolving set of $G - \{x\}$. This implies the correctness of the forward direction. The correctness of the reverse direction trivially follows from the fact that we can add x into a resolving set of $G - \{x\}$ to obtain a resolving set of G . ◀

► **Lemma 4.** METRIC DIMENSION, parameterized by the vertex cover number vc , admits a polynomial-time kernelization algorithm that returns an instance with $2^{\mathcal{O}(\text{vc})}$ vertices.

Proof. Given a graph G , let $X \subseteq V(G)$ be a minimum vertex cover of G . If such a vertex cover is not given, then we can find a 2-factor approximate vertex cover in polynomial time. Let $I := V(G) \setminus X$. By the definition of a vertex cover, the vertices of I are pairwise non-adjacent.

The kernelization algorithm exhaustively applies Reduction Rule 1. Now, consider an instance on which Reduction Rule 1 is not applicable. If the budget is negative, then the algorithm returns a trivial NO-instance of constant size. Otherwise, for any $Y \subseteq X$, there are at most two vertices $u, v \in I$ such that $N(u) = N(v) = Y$. This implies that the number of vertices in the reduced instance is at most $|X| + 2 \cdot 2^{|X|} = 2^{\text{vc}+1} + \text{vc}$. ◀

Next, we present an XP-algorithm parameterized by the vertex cover number. This algorithm, along with the kernelization algorithm above, imply¹ that METRIC DIMENSION admits an algorithm running in $2^{\mathcal{O}(\text{vc}^2)} \cdot n^{\mathcal{O}(1)}$ time.

► **Lemma 5.** METRIC DIMENSION admits an algorithm running in $n^{\mathcal{O}(\text{vc})}$ time.

Proof. The algorithm starts by computing a minimum vertex cover X of G in $2^{\mathcal{O}(\text{vc})} \cdot n^{\mathcal{O}(1)}$ time using an FPT algorithm for the VERTEX COVER problem, for example the one in [11] or [27]. Let $I := V(G) \setminus X$. Then, in polynomial time, it computes a largest subset F of I such that, for every vertex u in F , $I \setminus F$ contains a false twin of u . By the arguments in the previous proof, if there are false twins in I , say u, v , then any resolving set contains at least one of them. Hence, it is safe to assume that any resolving set contains F . If $k - |F| < 0$,

¹ Note that the application of Reduction Rule 1 does not increase the vertex cover number.

then the algorithm returns No. Otherwise, it enumerates every subset of vertices of size at most $|X|$ in $X \cup (I \setminus F)$. If there exists a subset $A \subseteq X \cup (I \setminus F)$ such that $A \cup F$ is a resolving set of G of size at most k , then it returns $A \cup F$. Otherwise, it returns No.

In order to prove that the algorithm is correct, we prove that $X \cup F$ is a resolving set of G . It is easy to see that, for a pair of distinct vertices u, v , if $u \in X \cup F$ and $v \in V(G)$, then the pair is resolved by u . It remains to argue that every pair of distinct vertices in $(I \setminus F) \times (I \setminus F)$ is resolved by $X \cup F$. Note that, for any two vertices $u, v \in I \setminus F$, $N(u) \neq N(v)$ as otherwise u can be moved to F , contradicting the maximality of F . Hence, there is a vertex in X that is adjacent to u , but not adjacent to v , resolving the pair $\langle u, v \rangle$. This implies the correctness of the algorithm. The running time of the algorithm easily follows from its description. ◀

4 Metric Dimension: Lower Bounds Regarding Vertex Cover

In this section, we prove Theorem 2 for METRIC DIMENSION. The first integral part of our technique is to reduce from a variant of 3-SAT known as 3-PARTITIONED-3-SAT [32]. In this problem, the input is a 3-CNF formula ψ , together with a partition of the set of its variables into three disjoint sets $X^\alpha, X^\beta, X^\gamma$, with $|X^\alpha| = |X^\beta| = |X^\gamma| = N$, and such that no clause contains more than one variable from each of X^α, X^β , and X^γ . The objective is to determine whether ψ is satisfiable. Unless the ETH fails, 3-PARTITIONED-3-SAT does not admit an algorithm running in $2^{o(N)}$ time [32, Theorem 3]. Our key result is the following.

► **Theorem 6.** *There is an algorithm that, given an instance ψ of 3-PARTITIONED-3-SAT on N variables, runs in $2^{\mathcal{O}(\sqrt{N})}$ time, and constructs an equivalent instance (G, k) of METRIC DIMENSION such that $\text{vc}(G) + k = \mathcal{O}(\sqrt{N})$ (and $|V(G)| = 2^{\mathcal{O}(\sqrt{N})}$).*

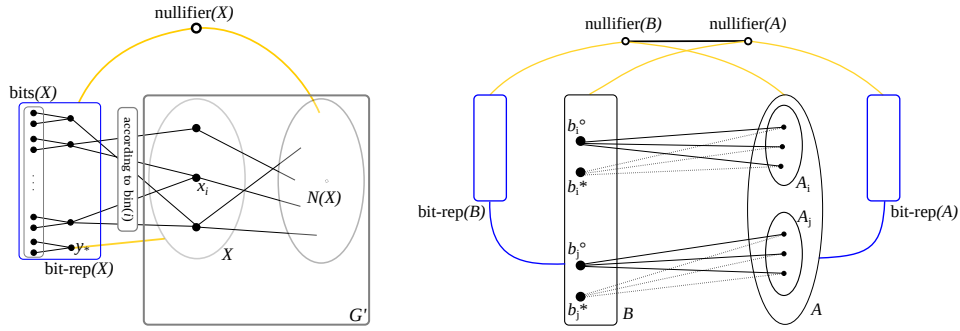
The above theorem, along with the arguments that are standard to prove the ETH-based lower bounds, immediately imply the following results.

► **Corollary 7.** *Unless the ETH fails, METRIC DIMENSION does not admit an algorithm running in $2^{o(\text{vc}^2)} \cdot n^{\mathcal{O}(1)}$ time.*

► **Corollary 8.** *Unless the ETH fails, METRIC DIMENSION does not admit a kernelization algorithm that does not increase the solution size k and outputs a kernel with $2^{o(k+\text{vc})}$ vertices.*

Proof. Toward a contradiction, assume that such a kernelization algorithm exists. Consider the following algorithm for 3-PARTITIONED-3-SAT. Given a 3-PARTITIONED-3-SAT formula on N variables, it uses Theorem 6 to obtain an equivalent instance of (G, k) such that $\text{vc}(G) + k = \mathcal{O}(\sqrt{N})$ and $|V(G)| = 2^{\mathcal{O}(\sqrt{N})}$. Then, it uses the assumed kernelization algorithm to construct an equivalent instance (H, k') such that H has $2^{o(\text{vc}(G)+k)}$ vertices and $k' \leq k$. Finally, it uses a brute-force algorithm, running in $|V(H)|^{\mathcal{O}(k')}$ time, to determine whether the reduced instance, or equivalently the input instance of 3-PARTITIONED-3-SAT, is a YES-instance. The correctness of the algorithm follows from the correctness of the respective algorithms and our assumption. The total running time of the algorithm is $2^{\mathcal{O}(\sqrt{N})} + (|V(G)| + k)^{\mathcal{O}(1)} + |V(H)|^{\mathcal{O}(k')} = 2^{\mathcal{O}(\sqrt{N})} + (2^{\mathcal{O}(\sqrt{N})})^{\mathcal{O}(1)} + (2^{o(\sqrt{N})})^{\mathcal{O}(\sqrt{N})} = 2^{o(N)}$. But this contradicts the ETH. ◀

The reduction, presented in Section 4.2, uses tools introduced in the next subsection.



■ **Figure 1 Set Identifying Gadget (left).** The blue box represents $\text{bit-rep}(X)$. The yellow lines represent that all possible edges exist between $\text{bit-rep}(X) \setminus \text{bits}(X)$ and $\text{nullifier}(X)$, $\text{nullifier}(X)$ and $N(X)$, and y_* and X . Note that G' is not necessarily restricted to the graph induced by the vertices in $X \cup N(X)$. **Vertex Selector Gadget (right).** For $X \in \{B, A\}$, the blue box represents $\text{bit-rep}(X)$, the blue link represents the connection with respect to the binary representation, and the yellow line represents that $\text{nullifier}(X)$ is adjacent to each vertex in $\text{bit-rep}(X) \setminus \text{bits}(X)$. Dotted lines highlight absent edges.

4.1 Preliminary Tools

4.1.1 Set Identifying Gadget

We redefine a gadget introduced in [20]. Suppose we are given a graph G' and a subset $X \subseteq V(G')$ of its vertices. Further, suppose that we want to add a vertex set X^+ to G' in order to obtain a new graph G such that (1) each vertex in $X \cup X^+$ will be distinguished by vertices in X^+ that must be in any resolving set S of G , and (2) no vertex in X^+ can resolve any pair of vertices in $V(G) \setminus (X \cup X^+)$ that are in the same distance class with respect to X .

The graph induced by the vertices of X^+ , along with the edges connecting X^+ to G' , is the Set Identifying Gadget for X [20]. Given a graph G' and a non-empty subset $X \subseteq V(G')$ of its vertices, to construct such a graph G , we add vertices and edges to G' as follows:

- The vertex set X^+ that we are aiming to add is the union of a set bit-rep and a special vertex denoted by $\text{nullifier}(X)$.
- Let $X = \{x_i \mid i \in [|X|]\}$ and set $q := \lceil \log(|X| + 2) \rceil + 1$. We select this value for q to (1) uniquely represent each integer in $[|X|]$ by its bit-representation in binary (note that we start from 1 and not 0), (2) ensure that the only vertex whose bit-representation contains all 1's is $\text{nullifier}(X)$, and (3) reserve one spot for an additional vertex y_* .
- For every $i \in [q]$, add three vertices y_i^a, y_i, y_i^b , and add the path (y_i^a, y_i, y_i^b) .
- Add three vertices y_*^a, y_*, y_*^b , and add the path (y_*^a, y_*, y_*^b) . Add all the edges to make $\{y_i \mid i \in [q]\} \cup \{y_*\}$ a clique. Make y_* adjacent to each vertex $v \in X$. Let $\text{bit-rep}(X) := \{y_i, y_i^a, y_i^b \mid i \in [q]\} \cup \{y_*, y_*^a, y_*^b\}$ and $\text{bits}(X) := \{y_i^a, y_i^b \mid i \in [q]\} \cup \{y_*^a, y_*^b\}$.
- For every integer $j \in [|X|]$, let $\text{bin}(j)$ denote the binary representation of j using q bits. Connect x_j with y_i if the i^{th} bit (going from left to right) in $\text{bin}(j)$ is 1.
- Add a vertex, denoted by $\text{nullifier}(X)$, and make it adjacent to every vertex in $\{y_i \mid i \in [q]\} \cup \{y_*\}$. One can think of $\text{nullifier}(X)$ as the only vertex whose bit-representation contains all 1's.
- For every vertex $u \in V(G) \setminus (X \cup X^+)$ such that u is adjacent to some vertex in X , add an edge between u and $\text{nullifier}(X)$. We add this vertex to ensure that vertices in $\text{bit-rep}(X)$ do not resolve any pairs of vertices in $V(G) \setminus (X \cup X^+)$ that are in the same distance class with respect to X .

This completes the construction of G . See Figure 1 for an illustration.

4.1.2 Gadget to Add Critical Pairs

Any resolving set needs to resolve *all* pairs of vertices in the input graph. As we will see, some pairs are harder to resolve than others.

Suppose that we need to have $m \in \mathbb{N}$ such “hard” pairs in a graph G . So, for each $i \in [m]$, we make a pair of vertices $\langle c_i^\circ, c_i^* \rangle$ *critical* as follows. Define $C := \{c_i^\circ, c_i^* \mid i \in [m]\}$. We then add $\text{bit-rep}(C)$ and $\text{nullifier}(C)$ as mentioned above (taking C as the set X), with the edges between $\{c_i^\circ, c_i^*\}$ and $\text{bit-rep}(C)$ defined by $\text{bin}(i)$, i.e., connect both c_i° and c_i^* with the j -th vertex of $\text{bit-rep}(C)$ if the j^{th} bit (going from left to right) in $\text{bin}(i)$ is 1. Hence, $\text{bit-rep}(C)$ can resolve any pair of the form $\langle c_i^\circ, c_\ell^\circ \rangle$, $\langle c_i^\circ, c_\ell^* \rangle$, or $\langle c_i^*, c_\ell^* \rangle$ as long as $i \neq \ell$. As before, $\text{bit-rep}(C)$ can also resolve all pairs with one vertex in $C \cup \text{bit-rep}(C) \cup \{\text{nullifier}(C)\}$, but no critical pair of vertices.

4.1.3 Vertex Selector Gadgets

Suppose that we are given a collection of sets A_1, A_2, \dots, A_q of vertices in a graph G , and we want to ensure that any resolving set of G includes at least one vertex from A_i for every $i \in [q]$. In the following, we construct a gadget that achieves a slightly weaker objective.

- Let $A = \bigcup_{i \in [q]} A_i$. Add a set identifying gadget for A as mentioned in Subsection 4.1.1.
- For every $i \in [q]$, add two vertices b_i° and b_i^* . Use the gadget mentioned in Subsection 4.1.2 to make all the pairs of the form $\langle b_i^\circ, b_i^* \rangle$ critical pairs (in the way it was introduced for $\langle c_i^\circ, c_i^* \rangle$).
- For every $a \in A_i$, add an edge (a, b_i°) . We highlight that we do not make a adjacent to b_i^* by a dotted line in Figure 1. Also, add the edges $(a, \text{nullifier}(B))$, $(b_i^\circ, \text{nullifier}(A))$, $(b_i^*, \text{nullifier}(A))$, and $(\text{nullifier}(A), \text{nullifier}(B))$.

This completes the construction. Note that the only vertices that can resolve a critical pair $\langle b_i^\circ, b_i^* \rangle$, apart from b_i° and b_i^* , are the vertices in A_i (see Figure 1, all other vertices are equidistant from both vertices of the pair). Hence, every resolving set contains at least one vertex in $\{b_i^\circ, b_i^*\} \cup A_i$.

4.2 Reduction

Consider an instance ψ of 3-PARTITIONED-3-SAT with $X^\alpha, X^\beta, X^\gamma$ the partition of the variable set, where each part contains N variables. By adding dummy variables in each of these sets, we can assume that \sqrt{N} is an integer. From ψ , we construct the graph G as follows. We describe the construction of the part of the graph G that corresponds to X^α , with the parts corresponding to X^β and X^γ being analogous. Rename the variables in X^α to $x_{i,j}^\alpha$ for $i, j \in [\sqrt{N}]$.

- We partition the variables of X^α into *buckets* $X_1^\alpha, X_2^\alpha, \dots, X_{\sqrt{N}}^\alpha$ such that each bucket contains \sqrt{N} variables. Let $X_i^\alpha = \{x_{i,j}^\alpha \mid j \in [\sqrt{N}]\}$ for all $i \in [\sqrt{N}]$.
- For every X_i^α , we construct a set A_i^α of $2^{\sqrt{N}}$ new vertices, $A_i^\alpha = \{a_{i,\ell}^\alpha \mid \ell \in [2^{\sqrt{N}}]\}$. Each vertex in A_i^α corresponds to a certain possible assignment of variables in X_i^α . Let A^α be the collection of all the vertices added in the above step, that is, $A^\alpha = \{a_{i,\ell}^\alpha \in A_i^\alpha \mid i \in [\sqrt{N}] \text{ and } \ell \in [2^{\sqrt{N}}]\}$. We add a set identifying gadget as mentioned in Subsection 4.1.1 in order to resolve every pair of vertices in A^α .
- For every X_i^α , we construct a pair $\langle b_i^{\alpha,\circ}, b_i^{\alpha,*} \rangle$ of vertices. Then, we add a gadget to make the pairs $\{\langle b_i^{\alpha,\circ}, b_i^{\alpha,*} \rangle \mid i \in [\sqrt{N}]\}$ critical as mentioned in Subsection 4.1.2. Let $B^\alpha = \{b_i^{\alpha,\circ}, b_i^{\alpha,*} \mid i \in [\sqrt{N}]\}$ be the collection of vertices in the critical pairs. We add edges in B^α to make it a clique.

33:10 Metric Dimension and Geodetic Set Parameterized by Vertex Cover

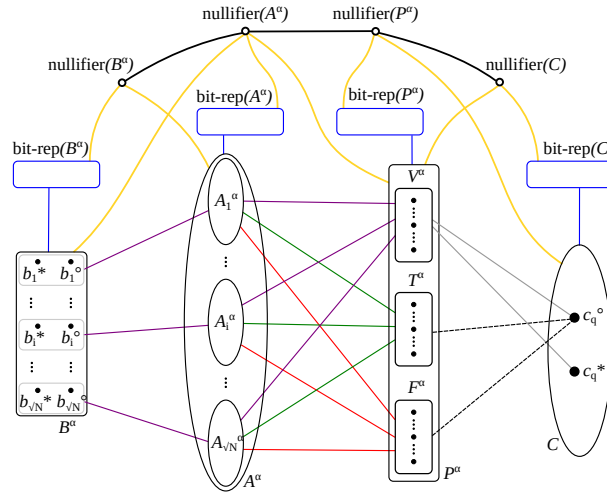
- We would like that any resolving set contains at least one vertex in A_i^α for every $i \in [\sqrt{N}]$, but instead we add the construction from Subsection 4.1.3 that achieves the slightly weaker objective as mentioned there. However, for every A_i^α , instead of adding two new vertices, we use $\langle b_i^{\alpha,\circ}, b_i^{\alpha,*} \rangle$ as the necessary critical pair. Formally, for every $i \in [\sqrt{N}]$, we make $b_i^{\alpha,\circ}$ adjacent to every vertex in A_i^α . We add edges to make $\text{nullifier}(B^\alpha)$ adjacent to every vertex in A^α , and $\text{nullifier}(A^\alpha)$ adjacent to every vertex in B^α . Recall that there is also the edge $(\text{nullifier}(B^\alpha), \text{nullifier}(A^\alpha))$.
- For every clause C_q in ψ , we have a pair of vertices $\langle c_q^\circ, c_q^* \rangle$. Let C be the collection of vertices in such pairs. We add *portals* that transmit information from vertices corresponding to assignments, i.e., vertices in A^α , to pairs corresponding to clauses. A portal is a clique on \sqrt{N} vertices in the graph G . We add three portals, the *truth portal* (denoted by T^α), *false portal* (denoted by F^α), and *validation portal* (denoted by V^α). Let $T^\alpha = \{t_1^\alpha, t_2^\alpha, \dots, t_{\sqrt{N}}^\alpha\}$, $F^\alpha = \{f_1^\alpha, f_2^\alpha, \dots, f_{\sqrt{N}}^\alpha\}$, and $V^\alpha = \{v_1^\alpha, v_2^\alpha, \dots, v_{\sqrt{N}}^\alpha\}$. Moreover, let $P^\alpha = V^\alpha \cup T^\alpha \cup F^\alpha$.
- We add a set identifying gadget for P^α as mentioned in Subsection 4.1.1. We add an edge between $\text{nullifier}(A^\alpha)$ and every vertex of P^α ; and the edge $(\text{nullifier}(P^\alpha), \text{nullifier}(A^\alpha))$. However, we note that we *do not* add edges between $\text{nullifier}(P^\alpha)$ and A^α , as can be seen in Figure 2. Lastly, we add edges in P^α to make it a clique.
- We add edges between A^α and the portals as follows. For $i \in [\sqrt{N}]$ and $\ell \in [2^{\sqrt{N}}]$, consider a vertex $a_{i,\ell}^\alpha$ in A_i^α . Recall that this vertex corresponds to an assignment $\pi : X_i^\alpha \mapsto \{\text{True}, \text{False}\}$, where X_i^α is the collection of variables $\{x_{i,j}^\alpha \mid j \in [\sqrt{N}]\}$. If $\pi(x_{i,j}^\alpha) = \text{True}$, then we add the edge $(a_{i,\ell}^\alpha, t_j^\alpha)$. Otherwise, $\pi(x_{i,j}^\alpha) = \text{False}$, and we add the edge $(a_{i,\ell}^\alpha, f_j^\alpha)$. We add the edge $(a_{i,\ell}^\alpha, v_i^\alpha)$ for every $\ell \in [2^{\sqrt{N}}]$.

Then, we repeat the above steps to construct $B^\beta, A^\beta, P^\beta, B^\gamma, A^\gamma, P^\gamma$. Now, we are ready to proceed through the final steps to complete the construction.

- For every clause C_q in ψ , as it has been already introduced above, we have a pair of vertices $\langle c_q^\circ, c_q^* \rangle$ and C is the collection of vertices in such pairs. Then, we add a gadget as was described in Subsection 4.1.2 to make each pair $\langle c_q^\circ, c_q^* \rangle$ a critical one.
- For each $\delta \in \{\alpha, \beta, \gamma\}$, we add an edge between $\text{nullifier}(P^\delta)$ and every vertex of C , and we add the edge $(\text{nullifier}(P^\delta), \text{nullifier}(C))$. Now, we add edges between C and the portals as follows for each $\delta \in \{\alpha, \beta, \gamma\}$. Consider a clause C_q in ψ and the corresponding critical pair $\langle c_q^\circ, c_q^* \rangle$ in C . As ψ is an instance of 3-PARTITIONED-3-SAT, there is at most one variable in X^δ that appears in C_q . If C_q does not contain a variable in X^δ , then we make c_q° and c_q^* adjacent to every vertex in V^δ , and they are not adjacent to any vertex in $T^\delta \cup F^\delta$. Otherwise, suppose that C_q contains the variable $x_{i,j}^\delta$ for some $i, j \in [\sqrt{N}]$. The first subscript decides the edges between $\langle c_q^\circ, c_q^* \rangle$ and the validation portal, whereas the second subscript decides the edges between $\langle c_q^\circ, c_q^* \rangle$ and either the truth portal or false portal in the following sense. We add all edges of the form $(v_{i'}^\delta, c_q^\circ)$ and $(v_{i'}^\delta, c_q^*)$ for every $i' \in [\sqrt{N}]$ such that $i' \neq i$. If $x_{i,j}^\delta$ appears as a positive literal in C_q , then we add the edge (t_j^δ, c_q°) . Otherwise, $x_{i,j}^\delta$ appears as a negative literal in C_q , and we add the edge (f_j^δ, c_q°) .

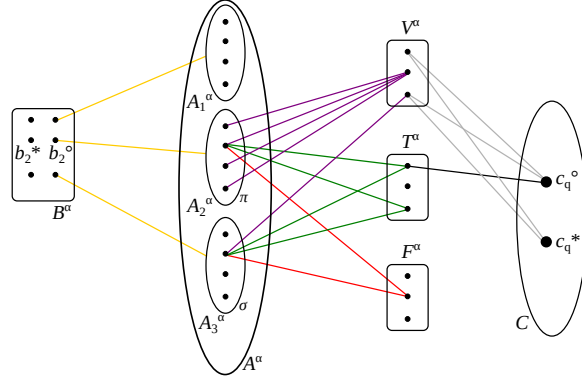
This concludes the construction of G . The reduction returns (G, k) as an instance of METRIC DIMENSION where

$$k = 3 \cdot \left(\sqrt{N} + (\lceil \log(|B^\alpha|/2 + 2) \rceil + 1) + (\lceil \log(|A^\alpha| + 2) \rceil + 1) + (\lceil \log(|P^\alpha| + 2) \rceil + 1) \right) + \lceil \log(|C|/2 + 2) \rceil + 1.$$



■ **Figure 2** Overview of the reduction. Sets in ellipses are independent sets and sets in rectangles are cliques. For $X \in \{B^\alpha, A^\alpha, P^\alpha, C\}$, the blue rectangle attached to it via the blue edge represents $\text{bit-rep}(X)$, and the yellow line between a vertex and $\text{bit-rep}(X)$ indicates that vertex is connected to every vertex in $\text{bit-rep}(X) \setminus \text{bits}(X)$. The remainder of the yellow lines represent that vertex is connected to every vertex in the set the edge goes to. Note the exception of $\text{nullifier}(P^\alpha)$ which is *not* adjacent to any vertex in A^α . Purple lines between two sets denote adjacencies with respect to indexing, i.e., $b_i^{\alpha,o}$ is adjacent only with all the vertices in A_i^α , and all the vertices in A_i^α are adjacent with v_i^α in validation portal V^α . Gray lines also indicate adjacencies with respect to indexing, but in a complementary way. If C_q contains a variable in B_i^α , then c_q^o and c_q^* are adjacent with all vertices $v_j^\alpha \in V^\alpha$ such that $j \neq i$. Green and red lines between the A^α and T^α and F^α roughly transfer, for each $a_{i,\ell}^\alpha \in A^\alpha$, the underlying assignment structure. If the j^{th} variable by $a_{i,\ell}^\alpha$ is assigned to **True**, then we add the green edge $(a_{i,\ell}^\alpha, t_j^\alpha)$, and otherwise the red edge $(a_{i,\ell}^\alpha, f_j^\alpha)$. Similarly, we add edges for each $c_i^\delta \in C$ depending on the assignment satisfying the variable from the part X^δ of a clause c_i , and in which block B_j^δ it lies, putting either an edge (c_i^δ, t_j^δ) or (c_i^δ, f_j^δ) accordingly ($\delta \in \{\alpha, \beta, \gamma\}$).

We give an informal description of the proof of correctness here. See Figure 3. Suppose $\sqrt{N} = 3$ and the vertices in the sets are indexed from top to bottom. For legibility, we omit some edges and only show 4 out of 8 vertices in each A_i^α for $i \in [3]$. We also omit bit-rep and nullifier for these sets. The vertex selection gadget and the budget k ensure that exactly one vertex in $\{b_i^{\alpha,o}, b_i^{\alpha,*}\} \cup A_i^\alpha$ is selected for every $i \in [3]$. If a resolving set contains a vertex from A_i^α , then it corresponds to selecting an assignment of variables in X_i^α . For example, the vertex $a_{2,2}^\alpha$ corresponds to the assignment $\pi : X_2^\alpha \mapsto \{\text{True}, \text{False}\}$. Suppose $X_2^\alpha = \{x_{2,1}^\alpha, x_{2,2}^\alpha, x_{2,3}^\alpha\}$, $\pi(x_{2,1}^\alpha) = \pi(x_{2,3}^\alpha) = \text{True}$, and $\pi(x_{2,2}^\alpha) = \text{False}$. Hence, $a_{2,2}^\alpha$ is adjacent to the first and third vertex in the truth portal T^α , whereas it is adjacent with the second vertex in the false portal F^α . Suppose the clause C_q contains the variable $x_{2,1}^\alpha$ as a positive literal. Note that c_q^o and c_q^* are at distance 2 and 3, respectively, from $a_{2,2}^\alpha$. Hence, the vertex $a_{2,2}^\alpha$, corresponding to the assignment π that satisfies clause C_q , resolves the critical pair $\langle c_q^o, c_q^* \rangle$. Now, suppose there is another assignment $\sigma : X_3^\alpha \mapsto \{\text{True}, \text{False}\}$ such that $\sigma(x_{3,1}^\alpha) = \sigma(x_{3,3}^\alpha) = \text{True}$ and $\sigma(x_{3,2}^\alpha) = \text{False}$. As ψ is an instance of 3-PARTITIONED-3-SAT and C_q contains a variable in $X_2^\alpha (\subseteq X^\alpha)$, C_q does not contain a variable in $X_3^\alpha (\subseteq X^\alpha)$. Hence, σ does not satisfy C_q . Let $a_{3,2}^\alpha$ be the vertex in X_3^α corresponding to σ . The connections via the validation portal V^α ensure that both c_q^o and c_q^* are at distance 2 from $a_{3,2}^\alpha$, and hence, $a_{3,2}^\alpha$ cannot resolve the critical pair $\langle c_q^o, c_q^* \rangle$. Hence, finding a resolving set in G corresponds to finding a satisfying assignment for ψ . These intuitions are formalized in the following subsection.



■ **Figure 3** An example to illustrate the reduction (bit-rep and nullifier are omitted for the sets).

4.3 Correctness of the Reduction

Suppose, given an instance ψ of 3-PARTITIONED-3-SAT, that the reduction of this subsection returns (G, k) as an instance of METRIC DIMENSION. We first prove the following lemma which will be helpful in proving the correctness of the reduction.

► **Lemma 9.** *For any resolving set S of G and for all $X \in \{C\} \cup \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$,*

1. S contains at least one vertex from each pair of false twins in $\text{bits}(X)$.
2. Vertices in $\text{bits}(X) \cap S$ resolve any non-critical pair of vertices $\langle u, v \rangle$ when $u \in X \cup X^+$ and $v \in V(G)$.
3. Vertices in $X^+ \cap S$ cannot resolve any critical pair of vertices $\langle b_i^{\delta', \circ}, b_i^{\delta', *}\rangle$ nor $\langle c_q^\circ, c_q^*\rangle$ for all $i \in [\sqrt{N}]$, $\delta' \in \{\alpha, \beta, \gamma\}$, and $q \in [m]$.

Proof.

1. Let G be a graph. For any false twins $u, v \in V(G)$ and any $w \in V(G) \setminus \{u, v\}$, $d(u, w) = d(v, w)$, and so, for any resolving set S of G , $S \cap \{u, v\} \neq \emptyset$. Hence, the statement follows for all $X \in \{C\} \cup \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$.
2. For all $X \in \{C\} \cup \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$, nullifier(X) is distinguished by $\text{bits}(X) \cap S$ as it is the only vertex in G at distance 2 from each vertex in $\text{bits}(X)$. We do a case analysis for the remaining non-critical pairs of vertices $\langle u, v \rangle$ assuming that nullifier(X) $\notin \{u, v\}$ (also, suppose that neither u nor v is in S , as otherwise, they are obviously distinguished):

Case i: $u, v \in X \cup X^+$.

Case i(a): $u, v \in X$ or $u, v \in \text{bit-rep}(X) \setminus \text{bits}(X)$. In the first case, let j be the bit in the binary representation of the subscript of u that is not equal to the j^{th} bit in the binary representation of the subscript of v (such a j exists since $\langle u, v \rangle$ is not a critical pair). In the second case, without loss of generality, let $u = y_i$ and $v = y_j$. By the first item of the statement of the lemma (1.), without loss of generality, $y_j^a \in S \cap \text{bits}(X)$. Then, in both cases, $d(y_j^a, u) \neq d(y_j^a, v)$.

Case i(b): $u \in X$ and $v \in \text{bit-rep}(X)$. Without loss of generality, $y_\star^a \in S \cap \text{bits}(X)$ (by 1.). Then, $d(y_\star^a, u) = 2$ and, for all $v \in \text{bits}(X) \setminus \{y_\star^b\}$, $d(y_\star^a, v) = 3$. Without loss of generality, let y_i be adjacent to u and let $y_i^a \in S \cap \text{bits}(X)$ (by 1.). Then, for $v = y_\star^b$, $3 = d(y_i^a, v) \neq d(y_i^a, u) = 2$. If $v \in \text{bit-rep}(X) \setminus \text{bits}(X)$, then, without loss of generality, $v = y_j$ and $y_j^a \in S \cap \text{bits}(X)$ (by 1.), and $1 = d(y_j^a, v) < d(y_j^a, u)$.

Case i(c): $u, v \in \text{bits}(X)$. Without loss of generality, $u = y_i^b$ and $y_i^a \in S$ (by 1.). Then, $2 = d(y_i^a, u) \neq d(y_i^a, v) = 3$.

Case i(d): $u \in \text{bits}(X)$ and $v \in \text{bit-rep}(X) \setminus \text{bits}(X)$. Without loss of generality, $v = y_i$ and $y_i^a \in S$ (by 1.). Then, $1 = d(y_i^a, v) < d(y_i^a, u)$.

Case ii: $u \in X \cup X^+$ and $v \in V(G) \setminus (X \cup X^+)$. For each $u \in X \cup X^+$, there exists $w \in \text{bits}(X) \cap S$ such that $d(u, w) \leq 2$, while, for each $v \in V(G) \setminus (X \cup X^+)$ and $w \in \text{bits}(X) \cap S$, we have $d(v, w) \geq 3$.

3. For all $X \in \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$, $u \in X^+$, $v \in \{c_q^\circ, c_q^*\}$, and $q \in [m]$, we have that $d(u, v) = d(u, \text{nullifier}(P^\delta)) + 1$. Further, for $X = C$ and all $u \in X^+$ and $q \in [m]$, either $d(u, c_q^\circ) = d(u, c_q^*) = 1$, $d(u, c_q^\circ) = d(u, c_q^*) = 2$, or $d(u, c_q^\circ) = d(u, c_q^*) = 3$ by the construction in Subsection 4.1.2 and since $\text{bit-rep}(X) \setminus \text{bits}(X)$ is a clique. Hence, for all $X \in \{C\} \cup \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$, vertices in $X^+ \cap S$ cannot resolve a pair of vertices $\langle c_q^\circ, c_q^* \rangle$ for any $q \in [m]$.

For all $\delta \in \{\alpha, \beta, \gamma\}$, if $v \in B^{\delta'}$, then, for all $X \in \{C\} \cup \{B^{\delta'}, A^{\delta'}, P^{\delta'} \mid \delta' \in \{\alpha, \beta, \gamma\}\}$ such that $\delta \neq \delta'$, and $u \in X^+$, we have that $d(u, v) = d(u, \text{nullifier}(A^\delta)) + 1$. Similarly, for all $\delta \in \{\alpha, \beta, \gamma\}$, if $v \in B^\delta$, then, for all $X \in \{A^\delta, P^\delta\}$ and $u \in X^+$, we have that $d(u, v) = d(u, \text{nullifier}(A^\delta)) + 1$. Lastly, for each $\langle b_i^{\delta, \circ}, b_i^{\delta, *}, \delta \in \{\alpha, \beta, \gamma\}$, and $i \in [\sqrt{N}]$, if $X = B^\delta$, then, for all $u \in X^+$, either $d(u, b_i^{\delta, \circ}) = d(u, b_i^{\delta, *}) = 1$, $d(u, b_i^{\delta, \circ}) = d(u, b_i^{\delta, *}) = 2$, or $d(u, b_i^{\delta, \circ}) = d(u, b_i^{\delta, *}) = 3$ by the construction in Subsection 4.1.2 and since $\text{bit-rep}(X) \setminus \text{bits}(X)$ is a clique. ◀

► **Lemma 10.** *If ψ is a satisfiable 3-PARTITIONED-3-SAT formula, then G admits a resolving set of size k .*

► **Lemma 11.** *If G admits a resolving set of size k , then ψ is a satisfiable 3-PARTITIONED-3-SAT formula.*

Proof of Theorem 6. In Subsection 4.2, we presented a reduction that takes an instance ψ of 3-PARTITIONED-3-SAT and returns an equivalent instance (G, k) of METRIC DIMENSION (by Lemmas 10 and 11) in $2^{\mathcal{O}(\sqrt{N})}$ time, where

$$k = 3 \cdot \left(\sqrt{N} + (\lceil \log(|B^\alpha|/2 + 2) \rceil + 1) + (\lceil \log(|A^\alpha| + 2) \rceil + 1) + (\lceil \log(|P^\alpha| + 2) \rceil + 1) \right) + (\lceil \log(|C|/2 + 2) \rceil + 1) = \mathcal{O}(\sqrt{N}).$$

Note that $V(G) = 2^{\mathcal{O}(\sqrt{N})}$. Further, note that taking all the vertices in B^δ and P^δ for all $\delta \in \{\alpha, \beta, \gamma\}$, and $X^+ \setminus \text{bits}(X)$ for all $X \in \{C\} \cup \{B^\delta, A^\delta, P^\delta \mid \delta \in \{\alpha, \beta, \gamma\}\}$, results in a vertex cover of G . Hence,

$$\text{vc}(G) \leq 3 \cdot ((\lceil \log(|B^\alpha|/2 + 2) \rceil + 2) + (\lceil \log(|A^\alpha| + 2) \rceil + 2) + (\lceil \log(|P^\alpha| + 2) \rceil + 2)) + 3 \cdot (|B^\alpha| + |P^\alpha|) + (\lceil \log(|C|/2 + 2) \rceil + 2) = \mathcal{O}(\sqrt{N}).$$

Thus, $\text{vc}(G) + k = \mathcal{O}(\sqrt{N})$. ◀

5 Geodetic Set: Algorithms for Vertex Cover Parameterization

To prove Theorem 1 for GEODETIC SET, we start with the following fact about false twins.

► **Lemma 12.** *If a graph G contains a set T of false twins that are not true twins and not simplicial, then any minimum-size geodetic set contains at most four vertices of T .*

Proof. Let $T = \{t_1, \dots, t_h\}$ be a set of false twins in a graph G , that are not true twins and not simplicial. Thus, T forms an independent set, and there are two non-adjacent vertices x, y in the neighborhood of the vertices in T . Toward a contradiction, assume that $h \geq 5$

33:14 Metric Dimension and Geodetic Set Parameterized by Vertex Cover

and G has a minimum-size geodetic set S that contains at least five vertices of T ; without loss of generality, assume $\{t_1, \dots, t_5\} \subseteq S$. We claim that $S' = (S \setminus \{t_1, t_2, t_3\}) \cup \{x, y\}$ is still a geodetic set, contradicting the choice of S as a minimum-size geodetic set of G .

To see this, notice that any vertex from $V(G) \setminus T$ that is covered by some pair of vertices in $T \cap S$ is also covered by t_4 and t_5 . Similarly, any vertex from $V(G) \setminus T$ covered by some pair $\langle t_i, z \rangle$ in $(S \cap T) \times (S \setminus T)$, is still covered by t_4 and z . Moreover, x and y cover all vertices of T , since they are at distance 2 from each other and all vertices in T are their common neighbors. Thus, S' is a geodetic set, as claimed. \blacktriangleleft

► **Lemma 13.** *GEODETIC SET, parameterized by the vertex cover number vc , admits a polynomial-time kernelization algorithm that returns an instance with $2^{\mathcal{O}(\text{vc})}$ vertices.*

Proof. Given a graph G , let $X \subseteq V(G)$ be a minimum-size vertex cover of G . If this vertex cover is not given, then we can find a 2-factor approximate vertex cover in polynomial time. Let $I := V(G) \setminus X$; I forms an independent set. The kernelization algorithm exhaustively applies the following reduction rules in a sequential manner.

▷ **Reduction Rule 2.** If there exist three simplicial vertices in G that are false twins or true twins, then delete one of them from G and decrease k by one.

▷ **Reduction Rule 3.** If there exist six vertices in G that are false twins but are not true twins nor simplicial, then delete one of them from G .

To see that Reduction Rule 2 is correct, assume that G contains three simplicial vertices u, v, w that are twins (false or true). We show that G has a geodetic set of size k if and only if the reduced graph G' , obtained from G by deleting u , has a geodetic set of size $k - 1$. For the forward direction, let S be a geodetic set of G of size k . By Observation 3, S contains each of u, v, w . Now, let $S' = S \setminus \{u\}$. This set of size $k - 1$ is a geodetic set of G' . Indeed, any vertex of G' that was covered in G by u and some other vertex z of S , is also covered by v and z in G' . Conversely, if G' has a geodetic set S'' of size $k - 1$, then it is clear that $S'' \cup \{u\}$ is a geodetic set of size k in G .

For Reduction Rule 3, assume that G contains six false twins (that are not true twins nor simplicial) as the set $T = \{t_1, \dots, t_6\}$, and let G' be the reduced graph obtained from G by deleting t_1 . We show that G has a geodetic set of size k if and only if G' has a geodetic set of size k . For the forward direction, let S be a minimum-size geodetic set of size (at most) k of G . By Lemma 12, S contains at most four vertices from T ; without loss of generality, t_1 and t_2 do not belong to S . Since the distances among all pairs of vertices in G' are the same as in G , S is still a geodetic set of G' . Conversely, let S' be a minimum-size geodetic set of G' of size (at most) k . Again, by Lemma 12, we may assume that one vertex among t_2, \dots, t_6 is not in S' , say, without loss of generality, that it is t_2 . Note that S' covers (in G) all vertices of G' . Thus, t_2 is covered by two vertices x, y of S' . But then, t_1 is also covered by x and y , since we can replace t_2 by t_1 in any shortest path between x and y . Hence, $S' \cup \{t_1\}$ is also a geodetic set of G .

Now, consider an instance on which the reduction rules cannot be applied. If $k < 0$, then we return a trivial NO-instance (for example, a single-vertex graph). Otherwise, notice that any set of false twins in I contains at most five vertices. Hence, G has at most $|X| + 5 \cdot 2^{|X|} = 2^{\mathcal{O}(\text{vc})}$ vertices. \blacktriangleleft

Next, we present an XP-algorithm parameterized by the vertex cover number. Together with Lemma 13, they imply Theorem 1 for GEODETIC SET.

► **Lemma 14.** *GEODETIC SET admits an algorithm running in $n^{\mathcal{O}(\text{vc})}$ time.*

Proof. The algorithm starts by computing a minimum vertex cover X of G in $2^{\mathcal{O}(\text{vc})} \cdot n^{\mathcal{O}(1)}$ time using an FPT algorithm for the VERTEX COVER problem, for example the one in [11] or [27]. Let $I := V(G) \setminus X$.

In polynomial time, we compute the set S of simplicial vertices of G . By Observation 3, any geodetic set of G contains all simplicial vertices of G . Note that $X \cup S$ is a geodetic set of G . Indeed, any vertex v from I that is not simplicial has two non-adjacent neighbors x, y in X , and thus, v is covered by x and y (which are at distance 2 from each other).

Hence, to enumerate all possible minimum-size geodetic sets, it suffices to enumerate all subsets S' of vertices of size at most $|X|$ in $(X \cup I) \setminus S$, and check whether $S \cup S'$ is a geodetic set. If one such set is indeed a geodetic set and has size at most k , we return YES. Otherwise, we return NO. The statement follows. ◀

6 Geodetic Set: Lower Bounds Regarding Vertex Cover

In this section, we follow the same template as in Section 4 and first prove the following theorem.

► **Theorem 15.** *There is an algorithm that, given an instance ψ of 3-PARTITIONED-3-SAT on N variables, runs in $2^{\mathcal{O}(\sqrt{N})}$ time, and constructs an equivalent instance (G, k) of GEODETIC SET such that $\text{vc}(G) + k = \mathcal{O}(\sqrt{N})$ (and $|V(G)| = 2^{\mathcal{O}(\sqrt{N})}$).*

The proofs of the following two corollaries are analogous to those for METRIC DIMENSION.

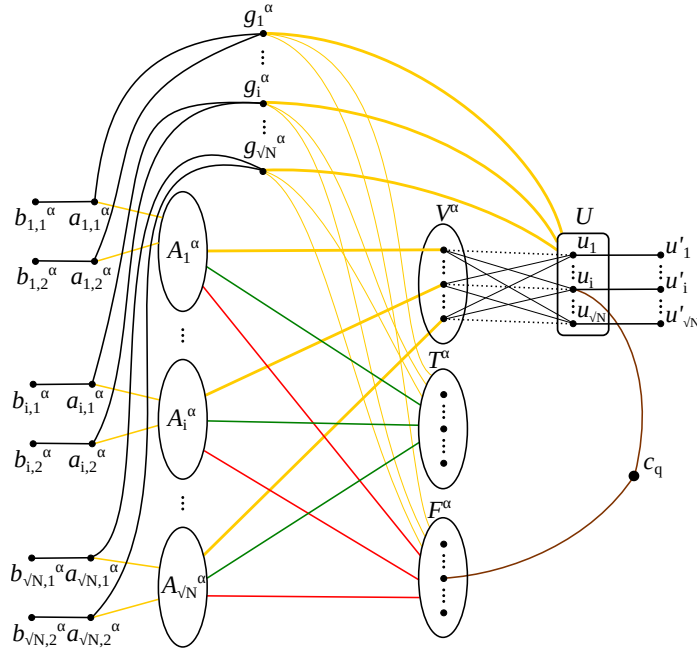
► **Corollary 16.** *Unless the ETH fails, GEODETIC SET does not admit an algorithm running in $2^{\mathcal{O}(\text{vc}^2)} \cdot n^{\mathcal{O}(1)}$ time.*

► **Corollary 17.** *Unless the ETH fails, GEODETIC SET does not admit a kernelization algorithm that does not increase the solution size k and outputs a kernel with $2^{\mathcal{O}(k+\text{vc})}$ vertices.*

6.1 Reduction

Consider an instance ψ of 3-PARTITIONED-3-SAT with $X^\alpha, X^\beta, X^\gamma$ the partition of the variable set, where $|X^\alpha| = |X^\beta| = |X^\gamma| = N$. By adding dummy variables in each of these sets, we can assume that \sqrt{N} is an integer. Further, let $\mathcal{C} = \{C_1, \dots, C_m\}$ be the set of all the clauses of ψ . From ψ , we construct the graph G as follows. We describe the construction for the part of the graph G corresponding to X^α , with the parts corresponding to X^β and X^γ being analogous. We rename the variables in X^α to $x_{i,j}^\alpha$ for $i, j \in [\sqrt{N}]$.

- We partition the variables of X^α into buckets $X_1^\alpha, X_2^\alpha, \dots, X_{\sqrt{N}}^\alpha$ such that each bucket contains \sqrt{N} many variables. Let $X_i^\alpha = \{x_{i,j}^\alpha \mid j \in [\sqrt{N}]\}$ for all $i \in [\sqrt{N}]$.
- For every bucket X_i^α , we add an independent set A_i^α of $2^{\sqrt{N}}$ new vertices, and we add two isolated edges $(a_{i,1}^\alpha, b_{i,1}^\alpha)$ and $(a_{i,2}^\alpha, b_{i,2}^\alpha)$. Let $B^\alpha = \{a_{i,j}^\alpha, b_{i,j}^\alpha \mid i \in [\sqrt{N}], j \in \{1, 2\}\}$. For all $i \in [\sqrt{N}]$ and $u \in A_i^\alpha$, we make both $a_{i,1}^\alpha$ and $a_{i,2}^\alpha$ adjacent to u (see Figure 4). Each vertex in A_i^α corresponds to a certain possible assignment of variables in X_i^α .
- Then, we add three independent sets T^α, F^α , and V^α on \sqrt{N} vertices each: $T^\alpha = \{t_i^\alpha \mid i \in [\sqrt{N}]\}$, $F^\alpha = \{f_i^\alpha \mid i \in [\sqrt{N}]\}$, and $V^\alpha = \{v_i^\alpha \mid i \in [\sqrt{N}]\}$.
- For each $i \in [\sqrt{N}]$, we connect v_i^α with all the vertices in A_i^α .
- For each $i \in [\sqrt{N}]$, we add edges between A_i^α and T^α and between A_i^α and F^α as follows. Consider a vertex $w \in A_i^\alpha$. Recall that this vertex corresponds to an assignment $\pi : X_i^\alpha \mapsto \{\text{True}, \text{False}\}$, where X_i^α is the collection of variables $\{x_{i,j}^\alpha \mid j \in [\sqrt{N}]\}$. If $\pi(x_{i,j}^\alpha) = \text{True}$, then we add the edge (w, t_j^α) , and otherwise, we add the edge (w, f_j^α) .



■ **Figure 4** Overview of the reduction. Sets in ellipses are independent sets, and sets in rectangles are cliques. For each $\delta \in \{\alpha, \beta, \gamma\}$, the sets V^δ and U almost form a complete bipartite graph, except for the matching (marked by dotted edges) that is excluded. Yellow lines from a vertex to a set denote that this vertex is connected to all the vertices in that set. The green and red lines between the A_i^α and $T^\alpha \cup F^\alpha$ transfer, in some sense, for each $w \in A_i^\alpha$, the underlying assignment structure. If an underlying assignment w sets the j^{th} variable to **True**, then we add the green edge (w, t_j^α) , and otherwise, we add the red edge (w, f_j^α) . For all $q \in [m]$ and $\delta \in \{\alpha, \beta, \gamma\}$, let $x_{i,j}^\delta$ be the variable in X^δ that is contained in the clause C_q in ψ . So, for all $q \in [m]$, if assigning **True** (**False**, respectively) to $x_{i,j}^\delta$ satisfies C_q , then we add the edge (c_q, t_j^δ) ((c_q, f_j^δ) , respectively).

- For each $i \in [\sqrt{N}]$, we add a special vertex g_i^α (also referred to as a g -vertex later on) that is adjacent to each vertex in $T^\alpha \cup F^\alpha$. Further, g_i^α is also adjacent to both $a_{i,1}^\alpha$ and $a_{i,2}^\alpha$ (see Figure 4).

This finishes the first part of the construction. The second step is to connect the three previously constructed parts for X^α , X^β , and X^γ .

- We introduce a vertex set $U = \{u_i \mid i \in [\sqrt{N}]\}$ that forms a clique. Then, for each u_i , we add an edge to a new vertex u'_i . Thus, we have a matching $\{(u_i, u'_i) \mid i \in [\sqrt{N}]\}$. Let $U' = \{u'_i \mid i \in [\sqrt{N}]\}$.
- For each $\delta \in \{\alpha, \beta, \gamma\}$, we add edges so that the vertices of $U \cup V^\delta$ almost form a complete bipartite graph, i.e., $E(G)$ contains edges between all pairs $\langle v, w \rangle$ where $v \in U$ and $w \in V^\delta$, except for the matching $\{(v_i^\delta, u_i) \mid i \in [\sqrt{N}]\}$.
- For each $\delta \in \{\alpha, \beta, \gamma\}$ and $i \in [\sqrt{N}]$, we make g_i^δ adjacent to each vertex in U .
- For each $C_q \in \mathcal{C}$, we add a new vertex c_q . Let $C = \{c_q \mid q \in [m]\}$. Since we are considering an instance of 3-PARTITIONED-3-SAT, for each $\delta \in \{\alpha, \beta, \gamma\}$, there is at most one variable in C_q that lies in X^δ . If there is one, then without loss of generality, let it be $x_{i,j}^\delta$ and do the following. Make c_q adjacent to u_i and if $x_{i,j}^\delta = \text{True}$ ($x_{i,j}^\delta = \text{False}$, respectively) satisfies C_q , then $(c_q, t_j^\delta) \in E(G)$ ($(c_q, f_j^\delta) \in E(G)$, respectively).

This concludes the construction of G . The reduction returns (G, k) as an instance of GEODETIC SET where $k = 10\sqrt{N}$.

6.2 Correctness of the Reduction

Suppose, given an instance ψ of 3-PARTITIONED-3-SAT, that the reduction above returns (G, k) as an instance of GEODETIC SET. We first prove the following lemmas which will be helpful in proving the correctness of the reduction, and note that we use distances between vertices to prove that certain vertices are not contained in shortest paths.

► **Lemma 18.** *For all $\delta, \delta' \in \{\alpha, \beta, \gamma\}$, the shortest paths between any two vertices in $B^\delta \cup U \cup U'$ do not cover any vertices in C nor $V^{\delta'}$.*

► **Lemma 19.** *For all $i \in [\sqrt{N}]$ and $\delta \in \{\alpha, \beta, \gamma\}$, v_i^δ can only be covered by a shortest path from a vertex in $A_i^\delta \cup \{v_i^\delta\}$ to another vertex in G .*

► **Lemma 20.** *If G admits a geodetic set of size k , then ψ is a satisfiable 3-PARTITIONED-3-SAT formula.*

► **Lemma 21.** *If ψ is a satisfiable 3-PARTITIONED-3-SAT formula, then G admits a geodetic set of size k .*

Proof of Theorem 15. In Section 6.1, we presented a reduction that takes an instance ψ of 3-PARTITIONED-3-SAT and returns an equivalent instance (G, k) of GEODETIC SET (by Lemmas 20 and 21) in $2^{\mathcal{O}(\sqrt{N})}$ time, where $k = 10\sqrt{N}$. Note that $V(G) = 2^{\mathcal{O}(\sqrt{N})}$. Further, note that taking all the vertices in $B^\delta, V^\delta, T^\delta, F^\delta, U, C$, and g_i^δ for all $i \in [\sqrt{N}]$ and $\delta \in \{\alpha, \beta, \gamma\}$, results in a vertex cover of G . Hence,

$$\text{vc}(G) \leq 3 \cdot (|B^\alpha| + |V^\alpha| + |T^\alpha| + |F^\alpha| + \sqrt{N}) + |U| + |C| = \mathcal{O}(\sqrt{N}).$$

$$\text{Thus, } \text{vc}(G) + k = \mathcal{O}(\sqrt{N}). \quad \blacktriangleleft$$

7 Conclusion

We have seen that both METRIC DIMENSION and GEODETIC SET have a non-trivial $2^{\Theta(\text{vc}^2)}$ running-time dependency (unless the ETH fails) in the vertex cover number parameterization. Both problems are FPT for related parameters, such as vertex integrity, treedepth, distance to (co-)cluster, distance to cograph, etc., as more generally, they are FPT for cliquewidth plus diameter [23, 31]. For both problems, it was proved that the correct dependency in treedepth (and treewidth plus diameter) is in fact double-exponential [20], a fact that is also true for feedback vertex set plus diameter for METRIC DIMENSION [20]. For distance to (co-)cluster, algorithms with double-exponential dependency were given for METRIC DIMENSION in [21]. For the parameter max leaf number ℓ , the algorithm for METRIC DIMENSION from [17] uses ILPs, with a dependency of the form $2^{\mathcal{O}(\ell^6 \log \ell)}$ (a similar algorithm for GEODETIC SET with dependency $2^{\mathcal{O}(f \log f)}$ exists for the feedback edge set number f [31]), which is unknown to be tight. What is the correct dependency for all these parameters? In particular, it seems interesting to determine for which parameter(s) the jump from double-exponential to single-exponential dependency occurs.

For the related problem STRONG METRIC DIMENSION, the correct dependency in the vertex cover number is known to be double-exponential [20]. It would be nice to determine whether similarly intriguing behaviors can be exhibited for related metric-based problems, such as STRONG GEODETIC SET, whose parameterized complexity was recently addressed in [16, 34]. Perhaps our techniques are applicable to such related problems.

References

- 1 A. Agrawal, D. Lokshtanov, S. Saurabh, and M. Zehavi. Split contraction: The untold story. *ACM Trans. Comput. Theory*, 11(3):18:1–18:22, 2019. doi:10.1145/3319909.
- 2 R. Belmonte, F. V. Fomin, P. A. Golovach, and M. S. Ramanujan. Metric dimension of bounded tree-length graphs. *SIAM J. Discrete Math.*, 31(2):1217–1243, 2017. doi:10.1137/16M1057383.
- 3 B. Bergougnoux, O. Defrain, and F. Mc Inerney. Enumerating minimal solution sets for metric graph problems. In *Proc. of the 50th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2024)*, volume 14760 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2024.
- 4 E. Bonnet and N. Purohit. Metric dimension parameterized by treewidth. *Algorithmica*, 83:2606–2633, 2021. doi:10.1007/S00453-021-00808-9.
- 5 N. Bousquet, Q. Deschamps, and A. Parreau. Metric dimension parameterized by treewidth in chordal graphs. In *Proc. of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2023)*, volume 14093 of *Lecture Notes in Computer Science*, pages 130–142. Springer, 2023. doi:10.1007/978-3-031-43380-1_10.
- 6 D. Chakraborty, S. Das, F. Foucaud, H. Gahlawat, D. Lajou, and B. Roy. Algorithms and complexity for geodetic sets on planar and chordal graphs. In *Proc. of the 31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *LIPICs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ISAAC.2020.7.
- 7 D. Chakraborty, F. Foucaud, D. Majumdar, and P. Tale. Tight (double) exponential bounds for identification problems: Locating-dominating set and test cover. In *Proc. of the 35th International Symposium on Algorithms and Computation (ISAAC 2024)*, volume 322 of *LIPICs*, pages 19:1–19:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ISAAC.2024.19.
- 8 J. Chalopin, V. Chepoi, F. Mc Inerney, and S. Ratel. Non-clashing teaching maps for balls in graphs. In *Proc. of the 37th Annual Conference on Learning Theory (COLT 2024)*, volume 247 of *Proceedings of Machine Learning Research*, pages 840–875. PMLR, 2024. URL: <https://proceedings.mlr.press/v247/chalopin24a.html>.
- 9 L. S. Chandran, D. Issac, and A. Karrenbauer. On the parameterized complexity of biclique cover and partition. In *Proc. of the 11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *LIPICs*, pages 11:1–11:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.IPEC.2016.11.
- 10 G. Chartrand, F. Harary, and P. Zhang. On the geodetic number of a graph. *Networks*, 39(1):1–6, 2002. doi:10.1002/NET.10007.
- 11 J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001. doi:10.1006/JAGM.2001.1186.
- 12 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 13 M. Cygan, M. Pilipczuk, and M. Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM J. Comput.*, 45(1):67–83, 2016. doi:10.1137/130947076.
- 14 R. Diestel. *Graph Theory, 6th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2024.
- 15 M. C. Dourado, F. Protti, D. Rautenbach, and J. L. Szwarcfiter. Some remarks on the geodetic number of a graph. *Discrete Mathematics*, 310(4):832–837, 2010. doi:10.1016/J.DISC.2009.09.018.
- 16 M. Dumas, F. Foucaud, A. Perez, and I. Todinca. On graphs coverable by k shortest paths. *SIAM J. Discrete Math.*, 38(2):1840–1862, 2024. doi:10.1137/23M1564511.
- 17 D. Eppstein. Metric dimension parameterized by max leaf number. *Journal of Graph Algorithms and Applications*, 19(1):313–323, 2015. doi:10.7155/JGAA.00360.
- 18 L. Epstein, A. Levin, and G. J. Woeginger. The (weighted) metric dimension of graphs: Hard and easy cases. *Algorithmica*, 72(4):1130–1171, 2015. doi:10.1007/S00453-014-9896-2.

- 19 F. Foucaud, E. Galby, L. Khazaliya, S. Li, F. Mc Inerney, R. Sharma, and P. Tale. Metric dimension and geodetic set parameterized by vertex cover. *CoRR*, abs/2405.01344, 2024. doi:10.48550/arXiv.2405.01344.
- 20 F. Foucaud, E. Galby, L. Khazaliya, S. Li, F. Mc Inerney, R. Sharma, and P. Tale. Problems in NP can admit double-exponential lower bounds when parameterized by treewidth or vertex cover. In *Proc. of the 51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *LIPICs*, pages 66:1–66:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.66.
- 21 E. Galby, L. Khazaliya, F. Mc Inerney, R. Sharma, and P. Tale. Metric dimension parameterized by feedback vertex set and other structural parameters. *SIAM J. Discrete Math.*, 37(4):2241–2264, 2023. doi:10.1137/22M1510911.
- 22 M. R. Garey and D. S. Johnson. *Computers and Intractability - A guide to NP-completeness*. W.H. Freeman and Company, 1979.
- 23 T. Gima, T. Hanaka, M. Kiyomi, Y. Kobayashi, and Y. Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theoretical Comp. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.
- 24 G. Z. Gutin, M. S. Ramanujan, F. Reidl, and M. Wahlström. Alternative parameterizations of metric dimension. *Theoretical Comp. Sci.*, 806:133–143, 2020. doi:10.1016/J.TCS.2019.01.028.
- 25 F. Harary, E. Loukakis, and C. Tsouros. The geodetic number of a graph. *Mathematical and Computer Modelling*, 17(11):89–95, 1993.
- 26 F. Harary and R. A. Melter. On the metric dimension of a graph. *Ars Comb.*, 2:191–195, 1976.
- 27 D. G. Harris and N. S. Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. In *Proc. of the 41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *LIPICs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.STACS.2024.40.
- 28 S. Hartung and A. Nichterlein. On the parameterized and approximation hardness of metric dimension. In *Proc. of the 28th Conference on Computational Complexity, CCC 2013*, pages 266–276. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.36.
- 29 L. Jaffke, O.-J. Kwon, T. J. F. Strømme, and J. A. Telle. Mim-width III. Graph powers and generalized distance domination problems. *Theoretical Comp. Sci.*, 796:216–236, 2019. doi:10.1016/J.TCS.2019.09.012.
- 30 I. Katsikarelis, M. Lampis, and V. Th. Paschos. Structurally parameterized d -scattered set. *Discrete Applied Mathematics*, 308:168–186, 2022. doi:10.1016/J.DAM.2020.03.052.
- 31 L. Kellerhals and T. Koana. Parameterized complexity of geodetic set. *Journal of Graph Algorithms and Applications*, 26(4):401–419, 2022. doi:10.7155/JGAA.00601.
- 32 M. Lampis, N. Melissinos, and M. Vasilakis. Parameterized max min feedback vertex set. In *Proc. of the 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *LIPICs*, pages 62:1–62:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.MFCS.2023.62.
- 33 S. Li and M. Pilipczuk. Hardness of metric dimension in graphs of constant treewidth. *Algorithmica*, 84(11):3110–3155, 2022. doi:10.1007/S00453-022-01005-Y.
- 34 C. V. G. C. Lima, V. F. dos Santos, J. H. G. Sousa, and S. A. Urrutia. On the computational complexity of the strong geodetic recognition problem. *RAIRO Oper. Res.*, 58(5):3755–3770, 2024. doi:10.1051/RO/2024120.
- 35 D. Marx, M. Pilipczuk, and M. Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. In *Proc. of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 474–484, 2018.
- 36 M. Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In *Proc. of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS 2011)*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

33:20 Metric Dimension and Geodetic Set Parameterized by Vertex Cover

- 37 I. Sau and U. dos Santos Souza. Hitting forbidden induced subgraphs on bounded treewidth graphs. *Inf. Comput.*, 281:104812, 2021. doi:10.1016/J.IC.2021.104812.
- 38 P. J. Slater. Leaves of trees. In *Proc. of the 6th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, pages 549–559. Congressus Numerantium, No. XIV. Util. Math., 1975.
- 39 P. Tale. Double exponential lower bound for telephone broadcast. *CoRR*, abs/2403.03501, 2024. doi:10.48550/arXiv.2403.03501.

Agreement Tasks in Fault-Prone Synchronous Networks of Arbitrary Structure

Pierre Fraigniaud   

Institut de Recherche en Informatique Fondamentale (IRIF), CNRS, Université Paris Cité, France

Minh Hang Nguyen   

Institut de Recherche en Informatique Fondamentale (IRIF), CNRS, Université Paris Cité, France

Ami Paz   

Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), CNRS, Université Paris-Saclay, France

Abstract

Consensus is arguably the most studied problem in distributed computing as a whole, and particularly in the distributed message-passing setting. In this latter framework, research on consensus has considered various hypotheses regarding the failure types, the memory constraints, the algorithmic performances (e.g., early stopping and obliviousness), etc. Surprisingly, almost all of this work assumes that messages are passed in a *complete* network, i.e., each process has a direct link to every other process. A noticeable exception is the recent work of Castañeda et al. (Inf. Comput. 2023) who designed a generic oblivious algorithm for consensus running in $\text{radius}(G, t)$ rounds in every graph G , when up to t nodes can crash by irrevocably stopping, where t is smaller than the node-connectivity κ of G . Here, $\text{radius}(G, t)$ denotes a graph parameter called the *radius of G whenever up to t nodes can crash*. For $t = 0$, this parameter coincides with $\text{radius}(G)$, the standard radius of a graph, and, for $G = K_n$, the running time $\text{radius}(K_n, t) = t + 1$ of the algorithm exactly matches the known round-complexity of consensus in the clique K_n .

Our main result is a proof that $\text{radius}(G, t)$ rounds are necessary for oblivious algorithms solving consensus in G when up to t nodes can crash, thus validating a conjecture of Castañeda et al., and demonstrating that their consensus algorithm is optimal for any graph G . We also extend the result of Castañeda et al. to two different settings: First, to the case where the number t of failures is not necessarily smaller than the connectivity κ of the considered graph; Second, to the k -set agreement problem for which agreement is not restricted to be on a single value as in consensus, but on up to k different values.

2012 ACM Subject Classification Theory of computation \rightarrow Distributed algorithms

Keywords and phrases Consensus, set-agreement, fault tolerance, crash failures

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.34

Related Version *Full Version*: <https://arxiv.org/pdf/2410.21538> [16]

Funding *Pierre Fraigniaud*: Additional support from ANR projects DUCAT (ANR-20-CE48-0006), ENEDISC, and QuDATA (ANR-18-CE47-0010).

Minh Hang Nguyen: Additional support from ANR projects DUCAT (ANR-20-CE48-0006), TEMPORAL (ANR-22-CE48-0001), and ENEDISC, and by the European Union's Horizon 2020 program H2020-MSCA -COFUND-2019 Grant agreement n° 945332.

Acknowledgements The authors thank Stephan Felber, Mikaël Rabie, Hugo Rincon Galeana, and Ulrich Schmid for fruitful discussions on this paper.

1 Introduction

For $t \geq 0$, the standard *synchronous t -resilient message-passing* model assumes $n \geq 2$ nodes labeled from 1 to n , and connected as a clique, i.e., as a complete graph K_n . Computation proceeds as a sequence of synchronous rounds, during which every node can send a message



© Pierre Fraigniaud, Minh Hang Nguyen, and Ami Paz;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 34; pp. 34:1–34:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



to each other node, receive the message sent by each other node, and perform some local computation. Up to t nodes may crash during the execution of an algorithm. When a node v crashes at some round $r \geq 1$, it stops functioning after round r and never recovers. Moreover, some (possibly all) of the messages sent by v at round r may be lost, that is, when v crashes, messages sent by v at round r may reach some neighbors, while other neighbors of v may not hear from v at round r . This model has been extensively studied in the literature (see, e.g., [2, 18, 23, 27]). In particular, it is known that consensus can be solved in $t + 1$ rounds in the t -resilient model [13], and this is optimal for every $t < n - 1$ as far as the worst-case complexity is concerned [1, 13].

It is only very recently that the synchronous t -resilient message-passing model has been extended to the setting in which the complete communication graph K_n is replaced by an arbitrary communication graph G (see [4, 8]). Specifically, the graph G is fixed, but arbitrary, and the concern is to design algorithms for G . It was proved in [4] that if the number of failures is smaller than the connectivity of the graph, i.e., if $t < \kappa(G)$, then consensus in G can be solved in $\text{radius}(G, t)$ rounds in the t -resilient model, where $\text{radius}(G, t)$ generalizes the standard notion of graph radius to the scenarios in which up to t nodes may fail by crashing. For $t = 0$, $\text{radius}(G, 0)$ is the standard radius of the graph G . For the complete graph K_n , the $\text{radius}(K_n, t)$ upper bound from [4] coincides with the seminal $t + 1$ upper bound for consensus in K_n .

To get an intuition of $\text{radius}(G, t)$, let us consider the case of the n -node cycle C_n , for $n \geq 3$. We have $\kappa(C_n) = 2$, so we assume $t \leq 1$. The radius of C_n is $\lfloor \frac{n}{2} \rfloor$, i.e., $\text{radius}(C_n, 0) = \lfloor \frac{n}{2} \rfloor$. For $t = 1$, let v be the node that crashes. We have $\text{radius}(C_n, 1) \geq n - 2$, which is the distance between the two neighbors of v in C_n if v crashes “cleanly” at the first round, preventing them to communicate directly through v . However, we actually have $\text{radius}(C_n, 1) = n - 1$. Indeed, v may crash at the first round, yet be capable to send a message to one of its neighbors, and this message needs $n - 2$ additional rounds to reach the other neighbor of v . That is, computing $\text{radius}(G, t)$ requires to take into account not only which nodes crash, but when and how they are crashing – by “how”, it is meant that, for a node v crashing at some round r , to which neighbors they still succeed to communicate at this round, and to which they fail to communicate.

Importantly, the algorithm of [4] is *oblivious*, that is, the output of a node after $\text{radius}(G, t)$ rounds is solely based on the set of pairs (*node-identifier*, *input-value*) collected by that node during $\text{radius}(G, t)$ rounds (and not, e.g., from whom, when, and how many times it received each of these pairs). There are many reasons why to restrict the study to oblivious algorithms. Among them, oblivious algorithms are simple by design, which is desirable for their potential implementation. Moreover, they are known to be efficient, as illustrated by the case of the complete graphs in which optimal solutions can be obtained thanks to oblivious algorithms. As far as this paper is concerned (and maybe also as far as [4] is concerned) obliviousness is highly desirable for the design of *generic* solutions, that is “meta-algorithms” that apply to each and every graph G . In such algorithms, every node forwards pairs (*node-identifier*, *input-value*) during a prescribed number of rounds (e.g., during $\text{radius}(G, t)$ rounds in the generic algorithm from [4]), and then decides on an output value according to a simple function of the set of input values received during these rounds, without having to track of the sequence of rounds at which each pair was received, and from which neighbor(s). Last but not least, intermediate nodes do not need to send complex information about the history of each piece of information transmitted during the execution, hence reducing the bandwidth requirement of the algorithms.

1.1 Objective

The question of the optimality of the consensus algorithm performing in $\text{radius}(G, t)$ rounds in any fixed graph G for every number $t \leq \kappa(G)$ of failures was however left open in [4]. It was conjectured in [4] that, for every graph G , and for every $0 \leq t < \kappa(G)$, no oblivious algorithms can solve consensus in G in less than $\text{radius}(G, t)$ rounds, but this was only proved for the specific case of *symmetric* (a.k.a. *vertex-transitive*) graphs¹. Although the class of symmetric graphs includes, e.g., the complete graphs K_n , the cycles C_n , and the d -dimensional hypercubes Q_d , a lower bound $\text{radius}(G, t)$ for every graph G in this class does not come entirely as a surprise since all nodes of a symmetric graph have the same eccentricity (i.e., maximum distance to any other node, generalized to include crash failures). The fact that all nodes have the same eccentricity implies that they can merely be ordered according to their identifiers for selecting the output value from the received pairs (*node-identifier*, *input-value*). Instead, if the graph is not symmetric, a node that received a pair (*node-identifier*, *input-value*) after $\text{radius}(G, t)$ rounds does not necessarily know whether all the nodes have received this pair, and thus the choice of the output value from the set of received pairs is more subtle. Not only the design of an upper bound is made harder, but it also makes the determination of a strong lower bound more involved. The main question addressed in this paper is therefore the following: For every graph G , and every non-negative integer $t < \kappa(G)$, is there an oblivious algorithm solving consensus in G in less than $\text{radius}(G, t)$ rounds under the t -resilient model (i.e., when up to t nodes may fail by crashing)?

Moreover, the study in [4] let aside the design of a generic (oblivious) algorithm for solving the standard important relaxation of consensus, namely k -set agreement. (Recall that, in k -set agreement, the set of all values outputted by the nodes must be of cardinality at most k .) In fact, several tools developed in [4] do not extend to k -set agreement. Our next step is therefore to question the ability to design a generic algorithm for solving k -set agreement in arbitrary graphs G , for every $k > 1$.

Last but not least, the study in [4] assumed that the number t of failures is smaller than the connectivity $\kappa(G)$ of the graph G at hand. We question what can be said about the case where the number of failures may be larger, that is when $t \geq \kappa(G)$, for both consensus and k -set agreement?

1.2 Our Results

We extend the investigation of the t -resilient model in arbitrary graphs, in various complementary directions.

Lower Bounds for Consensus. We affirmatively prove the conjecture from [4] that their consensus algorithm is indeed optimal (among oblivious algorithms) for *every* graph G , and not only for symmetric graphs. That is, we show that, for every graph G , no oblivious algorithms can solve consensus in G in less than $\text{radius}(G, t)$ rounds under the t -resilient model. This result is achieved by revisiting the notion of *information flow graph* defined in [4] for fixing some inaccuracies in the original definition. We present a more robust (an accurate) definition of information flow graph, and we provide a characterization of the number of rounds required to solve consensus as a function of some structural property of

¹ A graph $G = (V, E)$ is vertex-transitive if, for every two nodes $u \neq v$, there exists an automorphism f of G (i.e., a permutation $f : V \rightarrow V$ preserving the edges and the non-edges of G) such that $f(u) = v$.

that graph. With this characterization at hand, we establish the optimality of the algorithm in [4] by showing that $\text{radius}(G, t)$ rounds are necessary for the information flow graph to satisfy the desired property required for consensus solvability.

Beyond the Connectivity Threshold. Inspired by [8], we extend the study of consensus in the t -resilient model in arbitrary graphs to the case where the number t of crash failures is arbitrary, i.e., not necessarily lower than the connectivity $\kappa(G)$ of the considered graph G . We show that the generic algorithm from [4] can be extended to this framework, at the mere cost of relaxing consensus to impose agreement to hold within each connected component of the graph resulting from removing the faulty nodes from G . Under this somehow unavoidable relaxation, we present extension of the consensus algorithm from [4] to t -resilient models for $t \geq \kappa(G)$, and express the round complexities of these algorithms in term of a non-trivial extension of the radius notion to disconnected graphs.

Extension to Set Agreement. Finally, we extend the study of consensus under the t -resilient model in arbitrary graphs to k -set agreement, for an arbitrary fixed $k \geq 1$. We show that, for every integer $k \geq 1$, and every graph G , there exists an oblivious k -set agreement algorithm performing in $\text{radius}(G, t, k)$ rounds, where $\text{radius}(G, t, k)$ denotes a parameter extending $\text{radius}(G, t)$ to the case where k nodes broadcast instead of just one, with the objective that each (non faulty) node receives a pair (*node-identifier*, *input-value*) from at least one of these k nodes. This extension holds for every t . For $t \geq \kappa(G)$ the k -set agreement tasks must however be relaxed similarly to consensus, so that agreement hold within each connected component of the graph separately. Due to lack of space, our results on k -set agreement are not included in this extended abstract, but they can be found in the full version of the paper (see [16]).

1.3 Related Work

Distributed computing in synchronous networks has a long tradition, including the early studies of the message complexity and round complexity of various tasks such as leader election, spanning tree constructions, BFS and DFS traversals, etc. (see, e.g., [2, 23]). The topic has then flourished in the 2000s under the umbrella of the so-called LOCAL and CONGEST models [19, 25], with the study of numerous graph problems such as coloring, maximal independent set, minimum-weight spanning tree, etc.

Distributed computing in synchronous *fault-prone* networks has also a long history, but it remained for a long time mostly confined to the special case of the message-passing model in the complete networks. That is, n nodes subject to *crash* or *malicious* (a.k.a. Byzantine) failures are connected as a complete graph K_n in which every pair of nodes has a private reliable link allowing them to exchange messages. In this setting, a significant amount of effort has been dedicated to narrowing down the complexity of solving agreement tasks such as consensus and, more generally, k -set agreement for $k \geq 1$. This includes in particular the issue of *early stopping* algorithms whose performances depend on the actual number of failures f experienced during the execution of the algorithm, and not on the upper bound t on the number of failures. We refer to a sequence of surveys on the matter [5, 26, 28].

In the Byzantine case, general communication graphs were studied early on [12], and are still being investigated [20]. In the stop-fault case, on the other hand, it is only recently that this approach has been extended to arbitrary networks, beyond the case of the complete graph K_n [4, 8]. Our paper is carrying on the preliminary investigations in [4], by extending them from consensus to k -set agreement, establishing various lower bounds including one

demonstrating the optimality of the consensus algorithm in [4], and extending the analysis to the case where the number of crashes may exceed the connectivity threshold. The original work in [4] has been extended to solving consensus when *links* are subject to crash failures [8]. Several consensus algorithms were proposed in [8], but their round complexities are expressed as a function of the so-called *stretch*, defined as the number of connected components of the graph after removing the faulty links, plus the sum of the diameters of the connected components. Instead, the round-complexity of the algorithm in [4] is expressed in term of the *radius*, which is a more refined measure. Indeed, we show that the upper bound in [4] is tight (no multiplicative constants, nor even additive constants). The consensus algorithms in [8] however extend to the case where failures may disconnect the graph, and the task is then referred to as “disconnected agreement”. Again, the complexities of the algorithms are expressed in term of the stretch, while we shall express the complexity of our local consensus algorithm as a function of the more refined radius parameter. We actually conjecture that our local consensus algorithm is optimal (with no multiplicative nor additive constants) for all t , no matter whether $t < \kappa(G)$ or $t \geq \kappa(G)$. On the other hand, some consensus algorithms proposed in [8] are early stopping, but the one with round-complexity close to the stretch of the actual failure pattern is not oblivious, and it uses messages with size significantly larger than the size of the messages in oblivious algorithms.

The case of *omission* failures has also attracted a lot of attention. In this context, nodes are reliable but messages may be lost. This is modeled as a sequence $\mathcal{S} = (G_i)_{i \geq 1}$ of directed graphs, where G_i captures the connections that are functioning at round i . The *oblivious message adversary* model allows an adversary to choose each communication graph G_i from a set \mathcal{G} and independently of its choices for the other graphs. The nodes know the set \mathcal{G} a priori, but not the actual graph picked by the adversary at each round). We refer to [9, 24, 29] for recent advances in this domain, including solving consensus. We also refer to the *heard-of* model [6, 7], which bears similarities with the oblivious message adversary model.

The case of *transient* failures is addressed in the context of *self-stabilizing* algorithms [14]. As opposed to most distributed algorithms for networks, which start from a given specific initial configuration, self-stabilizing algorithms must be able to start from any initial configuration (which may result from a corruption of the internal variables of the nodes). Under the synchronous scheduler, a self-stabilizing algorithm performs in a sequence of synchronous rounds, just that it must be able to cope with an arbitrary initial state of the system.

Last but not least, we underline the recent trend related to modeling communication between nodes (under the full-information paradigm) as a topological deformation of the input simplicial complex, and the computation (i.e., the decision of each node regarding its output value) as a simplicial map from the deformed input complex to the output simplicial complex [18]. The KNOW-ALL model [3] has been designed as a first attempt to understand the LOCAL model through the lens of algebraic topology. In particular, it was shown that k -set agreement in a graph G known to all the nodes a priori requires r rounds, where r is the smallest integer such that there exists a k -node dominating set in the r -th transitive closure of G . A follow-up work [17] minimized the involved simplicial complexes, and extended the framework to handle graph problems such as finding a proper coloring.

The study of *anonymous* networks, in which nodes may not be provided with distinct identifiers, and of *asynchronous* communication and computing, is beyond the scope of this paper, and we merely refer the reader to [10, 11, 15, 21, 22] for recent advances in these domains, as far as computing in (non-necessarily complete) networks is concerned.

2 Model and definitions

In this section, we recall the definition of the (synchronous) t -resilient model for networks, and the graph theoretical notions related to this model, all taken from [4], as well as the consensus algorithm presented there.

2.1 The Model

Let $G = (V, E)$ be an n -node undirected graph, which is also connected and simple (i.e., no multiple edges, nor self-loops). Each node $v \in V$ is a computing entity modeled as an infinite state machine. The nodes of G have distinct identifiers, which are positive integers. For the sake of simplifying the notations, we shall not distinguish a node v from its identifier; for instance, by “the smallest node” we mean “the node with the smallest identifier”. Initially, every node knows the graph G , that is, it knows the identifiers of all nodes, and how the nodes are connected. The uncertainty is thus not related to the initial structure of the connections, but is only due to the presence of potential failures, in addition to the fact that, of course, every node is not a priori aware of the inputs of the other nodes.

Computation in G proceeds as a sequence of synchronous rounds. All nodes start simultaneously, at round 1. At each round, each node sends a message to each of its neighbors in G , receives the messages sent by its neighbors, and performs some local computation. Each node may however fail by crashing – when a node crashes, it stops functioning and never recover. However, if a node v crashes at round r , it may still send a message to a non-empty subset of its set $N(v)$ of neighbors during round r . For every positive integer $t \geq 0$, the t -resilient model assumes that at most t nodes may crash. A *failure pattern* is defined as a set

$$\varphi = \{(v, F_v, f_v) \mid v \in F\}$$

where $F \subset V$ is the set of faulty nodes in φ , with $0 \leq |F| \leq t$, and, for each node $v \in F$, we use f_v to specify the round at which v crashes, and $F_v \subseteq N(v)$ to specify the non-empty set of neighbors to which v fails to send messages at round f_v .

A node $v \in F$ such that $F_v = N(v)$ is said to crash *cleanly* in φ (at round f_v). All the nodes in $V \setminus F$ are the correct nodes in φ . The failure pattern in which no nodes fail is denoted by φ_\emptyset . The set of all failure patterns in which at most t nodes fail is denoted by $\Phi_{\text{all}}^{(t)}$. In any execution of an algorithm in graph G under the t -resilient model, the nodes know t and G , but they do not know in advance to which failure pattern they may be exposed. This absence of knowledge is the source of uncertainty in the t -resilient model.

2.2 Eccentricity, connectivity, and radius

The *eccentricity* of a node v in G with respect to a failure pattern φ , denoted by $\text{ecc}(v, \varphi)$, is defined as the minimum number of rounds required for broadcasting a message from v to all *correct* nodes in φ . The broadcast protocol is by flooding, i.e., when a node receives a message at round r , it forwards it to all its neighbors at round $r + 1$. That is $\text{ecc}(v, \varphi)$ is the maximum, taken over all correct nodes v' , of the length of a shortest causal path from v to v' , where a *causal* path with respect to a failure pattern φ from a node v to a node v' is a sequence of nodes u_1, \dots, u_q with $u_1 = v$, $u_q = v'$, and, for every $i \in \{1, \dots, q - 1\}$, $u_{i+1} \in N(u_i)$, u_i has not crashed in φ during rounds $1, \dots, i - 1$, and if u_i crashes in φ at round i , i.e., if $(u_i, F_i, i) \in \varphi$ for some non-empty set $F_i \subseteq N(u_i)$, then $u_{i+1} \notin F_i$.

Note that $\text{ecc}(v, \varphi)$ might be infinite, in case v cannot broadcast to all correct nodes in G under φ . A typical example is when v crashes cleanly at the first round in φ , before sending any message to any of its neighbors. A more elaborate failure pattern φ in which v fails to broadcast is $\varphi = \{(v, N(v) \setminus \{w\}, 1), (w, N(w), 2)\}$ where v crashes at round 1, and sends the message only to its neighbor w , which crashes cleanly at round 2.

The node-connectivity of G , denoted $\kappa(G)$, is the smallest integer q such that removing q nodes disconnects the graph G (or reduces it to a single node whenever G is the complete graph K_n). The following was established in [4].

► **Proposition 1** (Lemma 1 in [4]). *For every graph G , every $t < \kappa(G)$, every node v , and every failure pattern φ in the t -resilient model, $\text{ecc}(v, \varphi) < \infty$ if and only if there exists at least one correct node that becomes aware of the message broadcast from v .*

Note that, in particular, thanks to proposition 1, if v is correct then $\text{ecc}(v, \varphi) < \infty$. Let

$$\Phi_v^* = \{\varphi \in \Phi_{\text{all}}^{(t)} \mid \text{ecc}(v, \varphi) < \infty\}$$

denote the set of failure patterns in the t -resilient model in which v eventually manages to broadcast to all correct nodes. The t -resilient radius is a key parameter defined in [4]:

► **Definition 2.** *The t -resilient radius of G is $\text{radius}(G, t) = \min_{v \in V} \max_{\varphi \in \Phi_v^*} \text{ecc}(v, \varphi)$.*

2.3 Consensus, oblivious algorithms, and the information flow graph

This section defines consensus, and survey the results in [4] regarding the round-complexity of oblivious consensus algorithms, which uses the notion of information flow graph. Note that this latter notion will be revisited, later in our paper.

2.3.1 Oblivious consensus algorithms

In the consensus problem, every node $v \in V$ receives an input value x_v from a set I of cardinality at least 2, and every correct node must decide on an output value $y_v \in I$ such that (1) $y_u = y_v$ for every pair $\{u, v\}$ of correct nodes, and (2) for every correct node $v \in V$, there exists $u \in V$ (not necessarily correct) such that $y_v = x_u$.

Assuming that every node $u \in V$ starts broadcasting the pair (u, x_u) at round 1, we let $\text{view}(v, \varphi, r)$ be the *view* of node v after $r \geq 0$ rounds in failure pattern φ , that is, the set of pairs (u, x_u) received by v after r rounds. An algorithm solving consensus is said to be *oblivious* if the output y_v of every correct node v depends only on the set of values received by v during the execution of the algorithm. That is, in an r -round oblivious algorithm executed under failure pattern φ , every node v outputs a value based solely on the set of pairs $(u, x_u) \in \text{view}(v, \varphi, r)$ (and not, say, on when each value was first received, or from which neighbor it was received). The following result was proved in [4].

► **Proposition 3** (Theorem 2 in [4]). *For every graph G and every $t < \kappa(G)$, consensus in G can be solved by an oblivious algorithm running in $\text{radius}(G, t)$ rounds under the t -resilient model.*

That is, consensus can be solved in the minimal time it takes for a *fixed* node to broadcast in all failure patterns (in which it manages to broadcast). Note that $\text{radius}(G, t)$ might be much larger than $\max_{\varphi \in \Phi_{\text{all}}^{(t)}} \min_{v \in V} \text{ecc}(v, \varphi)$. For instance, the radius of the clique K_n is $t + 1$: consider a path (v_1, \dots, v_{t+1}) in which $v_1 = v$, and, for every $i \in \{1, \dots, t\}$, v_i crashes at round i while sending only to v_{i+1} . On the other hand, $\max_{\varphi \in \Phi_{\text{all}}^{(t)}} \min_{v \in V} \text{ecc}(v, \varphi) = 1$ because,

for every failure pattern φ , there is a (correct) node v that broadcasts to all correct nodes in a single round. Similarly, the cycle C_n has radius $n - 1$, whereas $\max_{\varphi \in \Phi_{\text{all}}^{(t)}} \min_{v \in V} \text{ecc}(v, \varphi)$ is roughly $n/2$.

The consensus algorithm in [4] works as follows. It selects an ordered set of $t + 1$ nodes s_1, \dots, s_{t+1} according to the following rules. Node s_1 is a node with smallest eccentricity, i.e., a node that broadcasts the fastest among all nodes. However, there are failure patterns for which s_1 fails to broadcast (e.g., if s_1 crashes cleanly at round 1). Node s_2 is a node that broadcasts the fastest for all failure patterns in which s_1 fails to broadcast, that is node s_2 is a node that broadcasts the fastest for all failure patterns in $\Phi_{\text{all}}^{(t)} \setminus \Phi_{s_1}^*$. Similarly, node s_3 is a node that broadcasts the fastest for all failure patterns in which s_1 and s_2 fail to broadcast, that is node s_3 is a node that broadcasts the fastest for all failure patterns in $\Phi_{\text{all}}^{(t)} \setminus (\Phi_{s_1}^* \cup \Phi_{s_2}^*)$. And so on, for every $1 < i \leq t + 1$, s_i is a node that broadcasts the fastest for all failure patterns in

$$\Phi_{\text{all}}^{(t)} \setminus \bigcup_{j=1, \dots, i-1} \Phi_{s_j}^*.$$

A key property of the sequence s_1, \dots, s_{t+1} defined as above is that, for all $1 < i \leq t + 1$, the worst-case broadcast time of s_i over all failure patterns in

$$\Phi_{\text{all}}^{(t)} \setminus \bigcup_{j=1, \dots, i-1} \Phi_{s_j}^*$$

is at most the worst-case broadcast time of s_{i-1} over all failure patterns in

$$\Phi_{\text{all}}^{(t)} \setminus \bigcup_{j=1, \dots, i-2} \Phi_{s_j}^*.$$

As a consequence, for every $i \in \{1, \dots, t + 1\}$, the worst-case broadcast time of s_i over all failure patterns in $\Phi_{\text{all}}^{(t)} \setminus \bigcup_{j=1, \dots, i-1} \Phi_{s_j}^*$ is at most $\text{radius}(G, t)$ rounds.

The algorithm in [4] merely consists of letting all nodes s_1, \dots, s_{t+1} broadcast the pairs (s_i, x_{s_i}) by flooding during $\text{radius}(G, t)$ rounds. Every node u then selects as output the input x_{s_i} of the node s_i with smallest index i such that the pair (s_i, x_{s_i}) was received by node u . It was shown that this choice guarantees agreement.

2.4 Information flow graph

The lower bound from [4] on the number of rounds for achieving consensus in vertex-transitive graphs used the core notion of *information flow digraph*. The (directed) graph $\text{IF}(G, r)$ captures the state of mutual knowledge of the nodes at the end of round $r \geq 1$, assuming every node u broadcasts the pair (u, x_u) by flooding throughout the graph G , starting at round 1.

- The vertices of $\text{IF}(G, r)$ are all pairs $(v, \text{view}(v, r, \varphi))$ for $v \in V$ and $\varphi \in \Phi_{\text{all}}^{(t)}$ in which v does not crash in φ during the first r rounds. Note that a same vertex of $\text{IF}(G, r)$ can represent both $(v, \text{view}(v, r, \varphi))$ and $(v, \text{view}(v, r, \psi))$ if v has the same view after r rounds in φ and ψ .
- There is an arc from $(u, \text{view}(u, r, \varphi))$ to $(v, \text{view}(v, r, \varphi))$ whenever $(u, x_u) \in \text{view}(v, r, \varphi)$, where x_u is the input of u .

The *connected components* of $\text{IF}(G, r)$ play an important role, where by connected component we actually refer to the vertices of a connected component of the undirected graph resulting from $\text{IF}(G, r)$ by ignoring the directions of the arcs. A node $v \in V$ of the communication graph $G = (V, E)$ is said to *dominate* a connected component C of $\text{IF}(G, r)$ if,

for every vertex $(u, \text{view}(u, r, \varphi)) \in C$ with $u \neq v$ there is a vertex $(v, \text{view}(v, r, \varphi)) \in C$ with an arc from $(v, \text{view}(v, r, \varphi))$ to $(u, \text{view}(u, r, \varphi))$ in $\text{IF}(G, r)$. The following result characterizes the round-complexity of consensus in G .

► **Proposition 4** (Theorem 3 in [4]). *For every graph $G = (V, E)$ and every $t < \kappa(G)$, consensus in G can be solved by an oblivious algorithm running in r rounds under the t -resilient model if and only if every connected component of $\text{IF}(G, r)$ has a dominating node in V .*

It was proved in [4] that, if G is a symmetric graph then no node in V dominates $\text{IF}(G, \text{radius}(G, t) - 1)$. Property 4 immediately implies that consensus in G cannot be solved by an oblivious algorithm running in less than $\text{radius}(G, t)$ rounds under the t -resilient model. Their proof, however, holds only for symmetric graphs, and does not extend to general graphs.

Remark. The definition of the information flow *digraph* in [4] actually suffers from inconsistencies, and Theorem 3 there is formally incorrect. Roughly, it overlooks the possibility of deciding on an input of a process that already stopped. The “spirit” of the definition and the theorem is nevertheless plausible, and the specific consequences mentioned there are correct. For establishing our lower bound, we had to fix the inaccuracy in the definition of the information flow digraph, and the bugs in the proof of Theorem 3 of [4]. Concretely, we introduce a new information flow *graph* instead of the digraph of [4], and establish a correct version of Theorem 3 using that definition (cf. Theorem 9). See Section 4 for more details.

3 Detailed description of our results

In this section, we survey our results on consensus in detail, and, as already mentioned before, we refer to the full version [16] for our results on k -set agreement.

3.1 Lower bounds for consensus

We show that the consensus algorithm in [4] is optimal for every graph G , and not only for symmetric graphs. Specifically, we establish the following in Section 4.

► **Theorem 5.** *For every graph G and every $t < \kappa(G)$, consensus in G cannot be solved in less than $\text{radius}(G, t)$ rounds by an oblivious algorithm in the t -resilient model.*

This result was conjectured in [4], but only proved to be true for symmetric graphs. The class of symmetric graphs includes cliques, cycles and hypercubes, but remains limited. Moreover, in symmetric graphs, for every two nodes u and v ,

$$\text{ecc}(u, \Phi_{\text{all}}^{(t)}) = \text{ecc}(v, \Phi_{\text{all}}^{(t)}) = \text{radius}(G, t),$$

which implies that a naive algorithm for consensus in which every node outputs the input received from the node with smallest identifier performs in $\text{radius}(G, t)$ rounds. The fact that $\text{radius}(G, t)$ is a tight upper bound for consensus is thus not surprising for the family of symmetric graphs because, essentially, the choice of the $t + 1$ nodes s_1, \dots, s_{t+1} defined in Section 2.3.1 does not matter.

Instead, for an arbitrary graph G , two different nodes may have different eccentricities, which may differ by a multiplicative factor 2 at least. As a consequence, the choice of the source nodes s_1, \dots, s_{t+1} whose input can be adopted as output by the other nodes matters, as well as the ordering of these nodes (in case a node receives the input of two different source nodes).

3.1.1 A naive lower bound

A naive lower bound for the round-complexity of consensus is the maximum, over all failure patterns, of the time it takes *some* node to broadcast in the given pattern, obtained by switching the min and max operator in the definition of $\text{radius}(G, t)$, i.e.,

$$\max_{\varphi \in \Phi_{\text{all}}^{(t)}} \min_{v \in V} \text{ecc}(v, \varphi). \quad (1)$$

Indeed, for every failure pattern φ , even binary consensus under failure pattern φ cannot be solved in less than $R(\varphi) = \min_{v \in V} \text{ecc}(v, \varphi)$ rounds. The proof of this claim is by a standard indistinguishability argument. Specifically, let us assume, for the purpose of contradiction, that there is an algorithm ALG solving consensus in $G = (V, E)$ under failure pattern φ in $R(\varphi) - 1$ rounds. Let us order the nodes of G as v_1, \dots, v_n arbitrarily. Let us consider the input configuration I_0 in which all nodes have input 0. For every $i = 1, \dots, n$, we gradually change the input configuration as follows (see Figure 1).

$$\begin{array}{ccccccccc} 000..0 & \cdots & 100..0 & \cdots & 110..0 & \cdots & 1..100 & \cdots & 1..110 & \cdots & 1..111 \\ I_0 & & I_1 & & I_2 & & I_{n-2} & & I_{n-1} & & I_n \end{array}$$

(Note: In the original image, dots above the transitions indicate nodes $w_1, w_2, \dots, w_{n-1}, w_n$ that do not distinguish the configurations.)

■ **Figure 1** Input configurations I_0, \dots, I_n of a graph $G = (V, E)$, where $V = \{v_1, \dots, v_n\}$.

Since $\text{ecc}(v_i, \varphi) > R(\varphi)$, there exists a node w_i that does not receive the input of v_i in ALG. Let us then switch the input of v_i from 0 to 1, and denote by I_i the resulting input configuration. Note that I_n is the input configuration in which all nodes have input 1. Note also that, for every $i \in \{1, \dots, n\}$, node w_i does not distinguish I_{i-1} from I_i , and therefore ALG must output the same at w_i in both input configurations. Since, for every $i \in \{1, \dots, n\}$, all nodes must output the same value for input configuration I_i , we get that the consensus value returned by ALG for I_0 is the same as for I_n , which contradicts the validity condition.

It was conjectured in [4] that, in the t -resilient model, consensus needs longer time than $\max_{\varphi \in \Phi_{\text{all}}^{(t)}} \min_{v \in V} \text{ecc}(v, \varphi)$, and cannot be solved by an oblivious algorithm in less than $\text{radius}(G, t)$ rounds, i.e., the time it takes a fixed node to broadcast. As said before, this conjecture was however proved only for vertex-transitive graphs.

3.1.2 Sketch of proof of Theorem 5

To show that the consensus algorithm in [4] is optimal, i.e., to establish Theorem 5, we use the characterization of Proposition 4. In fact, we first fix the aforementioned bugs in [4] by defining the information flow graph, and then establish Proposition 4, a correct version of their theorem, using this new definition. With this new definition and new proposition at hand, we show that for every graph $G = (V, E)$ and every $t < \kappa(G)$, there exists a connected component of $\text{IF}(G, \text{radius}(G, t) - 1)$ that has no dominating node in V . To achieve this fact, we show that for every node $v \in V$ there exists a failure pattern φ_v such that

$$\text{ecc}(v, \varphi_v) \geq \text{radius}(G, t),$$

with some additional desirable properties. Then, we define a notion of *successor* of any failure pattern satisfying these desirable properties, which satisfies two key features.

- First, a failure pattern and its successor are in the same connected component of $\text{IF}(G, \text{radius}(G, t) - 1)$. Here we abuse terminology since the vertices of the information flow graph are not failure patterns, but pairs $(\text{node}, \text{view})$. What we formally mean is that the two subgraphs of $\text{IF}(G, \text{radius}(G, t) - 1)$ induced by all the views in the two failures patterns are both in the same connected component of $\text{IF}(G, \text{radius}(G, t) - 1)$.

- Second, for every node $v \in V$, there exists a sequence of failure patterns $\varphi_0, \varphi_1, \dots, \varphi_\ell$ such that $\varphi_0 = \varphi_v$, $\varphi_\ell = \varphi_\emptyset$ (the failure pattern in which no failures occur), and for every $i \in \{0, \dots, \ell - 1\}$, φ_{i+1} is the successor of φ_i .

It follows from these two features that, for every node $v \in V$, φ_v and φ_\emptyset are in the same connected component of $\text{IF}(G, \text{radius}(G, t) - 1)$, namely the connected component of $\text{IF}(G, \text{radius}(G, t) - 1)$ containing φ_\emptyset . Let C be this connected component. For every node $v \in V$, since $\text{ecc}(v, \varphi_v) \geq \text{radius}(G, t)$, we have that v does not dominate C . Therefore, no nodes dominate C , and our new Proposition 4 thus implies that no oblivious algorithm can solve consensus in less than $\text{radius}(G, t)$ rounds.

3.2 Beyond the connectivity threshold

The algorithm from [4] for consensus in the t -resilient model is under the assumption that $t < \kappa(G)$ in graph G , that is, the number of failing nodes is (strictly) smaller than the connectivity of the graph. This assumption is motivated by the mere observation that a set of $\kappa(G)$ nodes that, e.g., fails cleanly at the very first round, might disconnect the graph G , preventing tasks such as consensus to be solved. We show that, by slightly relaxing consensus and set-agreement, one can still consider the case where $t \geq \kappa(G)$, in a meaningful way, in the sense that if the t failing nodes do not disconnect the graph, then the standard consensus and set agreement tasks are solved.

3.2.1 Local consensus

For any given failure pattern φ , let $\text{comp}(G, \varphi)$ be the set of connected components of G resulting by removing from G all nodes that fail in φ . If $t \geq \kappa(G)$, then the nodes in a connected component $C \in \text{comp}(G, \varphi)$ of G may never hear from the nodes in a connected component $C' \neq C$, and vice versa, regardless of the number of rounds. To study consensus and k -set agreement for $t \geq \kappa(G)$, we merely relax the agreement condition: *Agreement must hold component-wise*, i.e., for each connected component separately, in the spirit of [8].

In other words, under φ , for any connected components C and C' of $\text{comp}(G, \varphi)$ all nodes in C must agree (on a single value for consensus, or on at most k values for k -set agreement), and all nodes in C' must agree, but no conditions are imposed the two sets of agreement values corresponding to C and C' . In particular, for consensus, the nodes in C may agree on x , but the nodes in C' may agree on $x' \neq x$.

The validity condition remains unchanged, that is, every output value must be the input value of some node. Note however that a node can return an output value which was the input value of a node from a different connected component.

We refer to these variants of consensus and k -set agreement as *local*, because agreement must hold “locally”, i.e., inside each connected component.

Remark. When $t < \kappa(G)$, consensus and local consensus are the same tasks, and, for every $k \geq 1$, k -set agreement and local k -set agreement are the same tasks. More generally, for every graph G , and for every failure pattern $\varphi \in \Phi_{\text{all}}^{(t)}$, if the nodes failing in φ do not disconnect G , and an algorithm solving local consensus (resp., local k -set agreement) does solve standard consensus (resp., standard k -set agreement).

3.2.2 Consensus beyond the connectivity threshold

We design a local consensus algorithm for an arbitrary graph G in the t -resilient model, for every given t , which does not need to be less than the connectivity $\kappa(G)$ of G . This algorithm satisfied the following property (see proof in the full version [16]).

► **Theorem 6.** *For every connected graph $G = (V, E)$, and every $t \geq 0$, local consensus in G can be solved by an oblivious algorithm running in $\text{radius}(G, t)$ rounds under the t -resilient model.*

In the statement above, $\text{radius}(G, t)$ denotes an extension of the notion of t -resilient radius to the case where $t \geq \kappa(G)$, which coincide to the aforementioned notion of radius whenever $t < \kappa(G)$. For the purpose of extending the notion of radius beyond the connectivity threshold, we revisit the notion of eccentricity entirely. Indeed, given a failure pattern φ , a node v may succeed to broadcast in some connected components but not in all of them. The control of the way information flow through the graph G with respect to the connected components is complex, as the connected components for one failure pattern are typically different from the connected components for another failure pattern.

Despite these difficulties, we are able to design and analyse an oblivious (and hence generic) local consensus algorithm. Given a graph $G = (V, E)$, our algorithm performs in $\text{radius}(G, t) = \min_{v \in V} \text{ecc}(v, \Phi_v^*)$ rounds, where the notion of eccentricity has been redefined and extended for allowing an arbitrary number t of failures. Again, for $t < \kappa(G)$, the extended notion of eccentricity coincides with the notion of eccentricity defined for consensus in [4], which itself coincide with the graph-theoretical notions of eccentricity for $t = 0$. We also note that our extended notion of radius, for all $t \geq 0$, provides a fine grain analysis of our local consensus algorithm, more refined than the notion of stretch defined in [8].

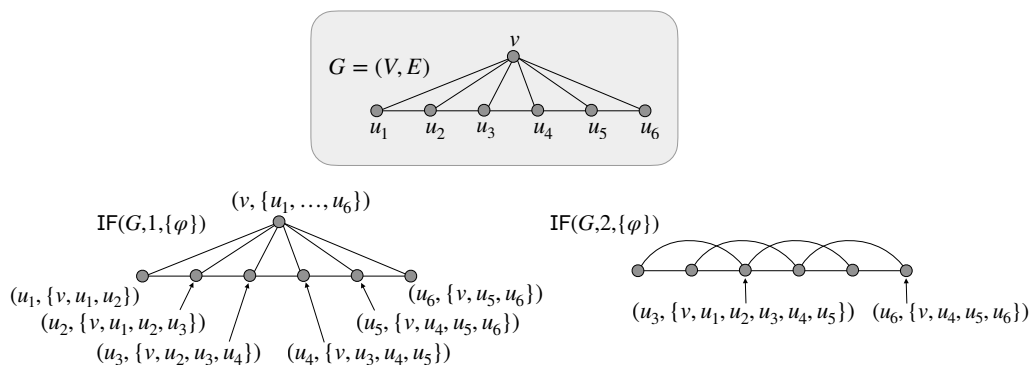
4 Lower Bound for Consensus

This section is entirely devoted to the proof of Theorem 5, that is, we show that, for every graph G and every $t < \kappa(G)$, consensus in G cannot be solved in less than $\text{radius}(G, t)$ rounds by an oblivious algorithm in the t -resilient model. We first establish a consistent notion of information flow graph, which can then be used to characterize consensus solvability, and we fix the bugs in the proof of Theorem 3 in [4] (see Proposition 4) resulting from inconsistencies in the original definition of the information flow digraph. Using our new characterization, we establish our lower bound.

4.1 Information flow graph revisited

The main issue with the notion of information flow *digraph* $\text{IF}(G, r)$ as defined in [4] comes from the fact that this directed graph includes only vertices $(v, \text{view}(v, r, \varphi))$ where v has not crashed in φ during rounds $1, \dots, r$. The main issue is related to the concept of domination, as defined in [4]. A vertex v dominates a connected component C of $\text{IF}(G, r)$ if the set $\{(v, \text{view}(v, r, \varphi)) \mid \varphi \in \Phi_{\text{all}}^{(t)}\}$ dominates C . This is too restrictive, as the correct nodes may agree on the input value of a node v that has already crashed. It follows that, for some failure pattern φ , the vertex $(v, \text{view}(v, r, \varphi))$ may not be present in $\text{IF}(G, r)$ (and therefore cannot dominate any other vertices of $\text{IF}(G, r)$), whereas the nodes that are correct in φ may agree on the input value of v . The characterization of Theorem 3 in [4] is therefore incorrect, even if the “spirit” of the characterization remains conceptually valid, as we shall show in this section.

To provide an illustration of the problems resulting from the original definition of information flow digraph in [4], let us clarify that this definition was aiming for capturing any subset $\Phi \subseteq \Phi_{\text{all}}^{(t)}$ of failure patterns (for instance the subset Φ of failure patterns in which nodes crash cleanly), in which case only the failure patterns $\varphi \in \Phi$ are considered. Let us



■ **Figure 2** The information flow graph $\text{IF}(G, r, \{\varphi\})$ as defined in [4] for $r = 1$ and $r = 2$, where φ is the failure pattern in which v crashes cleanly at the second round. No node dominates $\text{IF}(G, 2, \{\varphi\})$ (right), even though consensus is solvable in G under φ in 2 rounds.

then consider the scenario displayed on Fig. 2. The graph G is a 6-node path plus a universal node v . The set $\Phi = \{\varphi\}$ contains a single failure pattern φ in which v crashes cleanly at the second round.

Fig. 2 displays $\text{IF}(G, 1, \{\varphi\})$ and $\text{IF}(G, 2, \{\varphi\})$ as defined in [4] (the direction of the arcs are omitted, each edge corresponding to two symmetric arcs). A vertex $(v, \text{view}(v, r, \varphi))$ is present in the former but not in the latter, and thus, as opposed to what one might expect since nodes acquire more and more information as time passes, $\text{IF}(G, 2, \{\varphi\})$ is not a denser super graph of $\text{IF}(G, 1, \{\varphi\})$ nor it includes more vertices (with larger views), as some vertices present in $\text{IF}(G, 1, \{\varphi\})$ may disappear in $\text{IF}(G, 2, \{\varphi\})$. In fact, node v dominates $\text{IF}(G, 1, \{\varphi\})$, but it does not dominate $\text{IF}(G, 2, \{\varphi\})$. Therefore, when analyzing G with the set $\{\varphi\}$ of failure patterns using the characterization theorem in [4], consensus should be solvable in 1 round but not in 2 rounds!

We propose below a more robust notion of information flow *graph* (which is not directed anymore). The reader familiar with the algebraic topology interpretation of distributed computing [18] will recognize the mere 1-skeleton of the protocol complex after r rounds. For the purpose of fixing the issues in [4], we introduce $\text{IF}(G, r, \Phi)$ for an arbitrary set of failure patterns $\Phi \subseteq \Phi_{\text{all}}^{(t)}$.

► **Definition 7.** *The information flow graph of a communication graph $G = (V, E)$ after $r \geq 0$ rounds for a set $\Phi \subseteq \Phi_{\text{all}}^{(t)}$, $t \geq 0$, of failure patterns is the graph $\text{IF}(G, r, \Phi)$ defined as follows.*

- *The vertices of $\text{IF}(G, r, \Phi)$ are all pairs $(v, \text{view}(v, r, \varphi))$ for $v \in V$ and $\varphi \in \Phi$, where v is correct in φ .*
- *There is an edge between (v_1, w_1) and (v_2, w_2) in $\text{IF}(G, r, \Phi)$ whenever there exists $\varphi \in \Phi$ such that $w_1 = \text{view}(v_1, r, \varphi)$ and $w_2 = \text{view}(v_2, r, \varphi)$*

Remark. Unlike the definition of [4], this new notion of information-flow graph is not limited to $t \leq \kappa(G)$.

Note that a same vertex (v, ω) of $\text{IF}(G, \Phi, r)$ can represent both $(v, \text{view}(v, r, \varphi))$ and $(v, \text{view}(v, r, \psi))$ if v has the same view after r rounds in $\varphi \in \Phi$ and $\psi \in \Phi$. Note also that, for every $\varphi \in \Phi$, the set

$$\text{config}(G, r, \varphi) = \{(v, \text{view}(v, r, \varphi)) \in \text{IF}(G, r, \Phi) \mid v \in V\}$$

is a clique in $\text{IF}(G, r, \Phi)$. The *connected components* of $\text{IF}(G, r, \Phi)$ play an important role, w.r.t. the following concept of *domination*.

34:14 Agreement Tasks in Fault-Prone Synchronous Networks of Arbitrary Structure

► **Definition 8.** A node $v \in V$ of the communication graph $G = (V, E)$ is said to dominate a connected component C of $\text{IF}(G, r, \Phi)$ if, for every $\varphi \in \Phi$ and every $u \in V$,

$$(u, \text{view}(u, r, \varphi)) \in C \implies (v, x_v) \in \text{view}(u, r, \varphi).$$

Note that only correct nodes need to be dominated, as

$$(u, \text{view}(u, r, \varphi)) \in C \subseteq \text{IF}(G, r, \Phi)$$

implies that u is correct at round r . On the other hand, any node may be dominating. The following result characterizes the round-complexity of consensus in G by fixing the aforementioned inaccuracies in the definition of the information flow graph in [4], with impact on the proof of their characterization theorem (Theorem 3 in [4]).

► **Theorem 9.** For every graph $G = (V, E)$, every $t \geq 0$, and every set of failure patterns $\Phi \subseteq \Phi_{\text{all}}^{(t)}$, consensus in G can be solved by an oblivious algorithm running in r rounds under the t -resilient model with failure patterns in Φ if and only if every connected component of $\text{IF}(G, r, \Phi)$ has a dominating node in V .

Proof. Let us first show that if every connected component of $\text{IF}(G, r, \Phi)$ has a dominating node in V then consensus in G can be solved by an oblivious algorithm running in r rounds. For every connected component C of $\text{IF}(G, r, \Phi)$, let $v_C \in V$ be a node of G that dominates C . The algorithm proceeds as follows. Every node v_C broadcasts by flooding during r rounds. After r rounds, every correct node u considers its view, denoted by $\text{view}(u)$. A crucial point is that $\text{view}(u)$ may not be sufficient for u to determine what is the actual failure pattern $\varphi \in \Phi$ experienced during the execution, merely because one may have

$$\text{view}(u) = \text{view}(u, r, \varphi) = \text{view}(u, r, \psi)$$

for two different failure patterns φ, ψ in Φ . However, $\text{view}(u)$ is sufficient to determine the connected component C of $\text{IF}(G, r, \Phi)$ to which $(u, \text{view}(u))$ belongs. Node u outputs the input x_{v_C} of node v_C .

To establish correctness of this algorithm, observe first that (v_C, x_{v_C}) belongs to the view of node u . To see why, let $\varphi \in \Phi$, and let us consider the execution of the algorithm under φ . Let C be the connected component of $(u, \text{view}(u, r, \varphi))$. Since v_C dominates C , the mere definition of domination implies that $(v_C, x_{v_C}) \in \text{view}(u, r, \varphi)$. As a consequence, the algorithm is well defined. To show agreement, let $u' \neq u$ be another correct node in φ . By definition of the information flow graph, there is an edge between $(u, \text{view}(u, r, \varphi))$ and $(u', \text{view}(u', r, \varphi))$, and thus these two vertices belong to the same connected component C , and both output the same value x_{v_C} .

For the other direction, we show the contrapositive. That is, we let C be a connected component of $\text{IF}(G, r, \Phi)$ that is not dominated, and we aim at showing that there are no oblivious consensus algorithms in G running in r rounds. Let us assume, for the purpose of contradiction, that there exists an oblivious consensus algorithm ALG in G running in r rounds.

▷ **Claim 10.** Let $(u, \text{view}(u, r, \varphi))$ and $(u', \text{view}(u', r, \varphi'))$ be two vertices of C , where u and u' need not be different, nor do φ and φ' . For the same input configuration, node u outputs the same value in ALG under φ as node u' under φ' .

To see why this claim holds, observe that, since $(u, \text{view}(u, r, \varphi))$ and $(u', \text{view}(u', r, \varphi'))$ belong to the same connected component C , there is a sequence

$$(v_0, \text{view}(v_0, r, \psi_0)), \dots, (v_k, \text{view}(v_k, r, \psi_k))$$

of vertices of C such that

$$(v_0, \text{view}(v_0, r, \psi_0)) = (u, \text{view}(u, r, \varphi)), \quad (v_k, \text{view}(v_k, r, \psi_k)) = (u', \text{view}(u', r, \varphi')),$$

and, for every $i \in \{0, \dots, k-1\}$, there is an edge between the two vertices $(v_i, \text{view}(v_i, r, \psi_i))$ and $(v_{i+1}, \text{view}(v_{i+1}, r, \psi_{i+1}))$ in $\text{IF}(G, r, \Phi)$. Note that, for every $i \in \{0, \dots, k\}$, node v_i is correct in ψ_i since $(v_i, \text{view}(v_i, r, \psi_i))$ belongs to the information flow graph. For every $i \in \{0, \dots, k-1\}$, the presence of an edge between $(v_i, \text{view}(v_i, r, \psi_i))$ and $(v_{i+1}, \text{view}(v_{i+1}, r, \psi_{i+1}))$ implies that there exists $\chi \in \Phi$ such that

$$(v_i, \text{view}(v_i, r, \psi_i)) = (v_i, \text{view}(v_i, r, \chi)),$$

and

$$(v_{i+1}, \text{view}(v_{i+1}, r, \psi_{i+1})) = (v_{i+1}, \text{view}(v_{i+1}, r, \chi)).$$

As a consequence, since ALG is a consensus algorithm, ALG outputs the same value at v_{i+1} under ψ_{i+1} as it outputs at v_i under ψ_i , which is the value outputted by ALG under χ . Since this holds for every $i \in \{0, \dots, k-1\}$, we get that, in particular, u outputs the same value in φ as u' in φ' , as claimed.

For establishing a contradiction, let us enumerate the n nodes of G as u_0, \dots, u_{n-1} in arbitrary order. Since C is not dominated, for every node u_i , $i \in \{0, \dots, n-1\}$, there exists a vertex $(v_i, \text{view}(v_i, r, \varphi_i))$ of C such that $(u_i, x_{u_i}) \notin \text{view}(v_i, r, \varphi_i)$, where v_i is correct in φ_i . For $i \in \{0, \dots, n\}$, let us denote by I_i the input configuration in which the $n-i$ nodes $u_0, \dots, u_{n-(i+1)}$ have input 0, and all the other nodes have input 1. Thus, in particular, I_0 is the configuration in which all nodes have input 0, and I_n is the configuration in which all nodes have input 1. Since, for every $i \in \{0, \dots, n-1\}$, $(u_i, x_{u_i}) \notin \text{view}(v_i, r, \varphi_i)$, node u_i does not distinguish I_i from I_{i+1} under φ_i , and thus ALG must output the same at u_i for both configurations.

Since consensus imposes that all (correct) nodes output the same value, this means that, for every $i \in \{0, \dots, n-1\}$, all nodes output the same in ALG for I_i and I_{i+1} under φ_i . By Claim 10, all nodes output the same for I_i under φ_i as they do for I_{i+1} under φ_{i+1} . It follows that all nodes output the same for I_0 under φ_0 as for I_n under φ_n . This is a contradiction as all nodes must output 0 for I_0 , whereas all nodes must output 1 for I_n . \blacktriangleleft

Notation. For a fixed upper bound t on the number of failures, for every graph G , and for every integer $r \geq 0$, we denote by $\text{IF}(G, r)$ the information flow graph for the set of all failure patterns in the t -resilient model, that is, $\text{IF}(G, r) = \text{IF}(G, r, \Phi_{\text{all}}^{(t)})$.

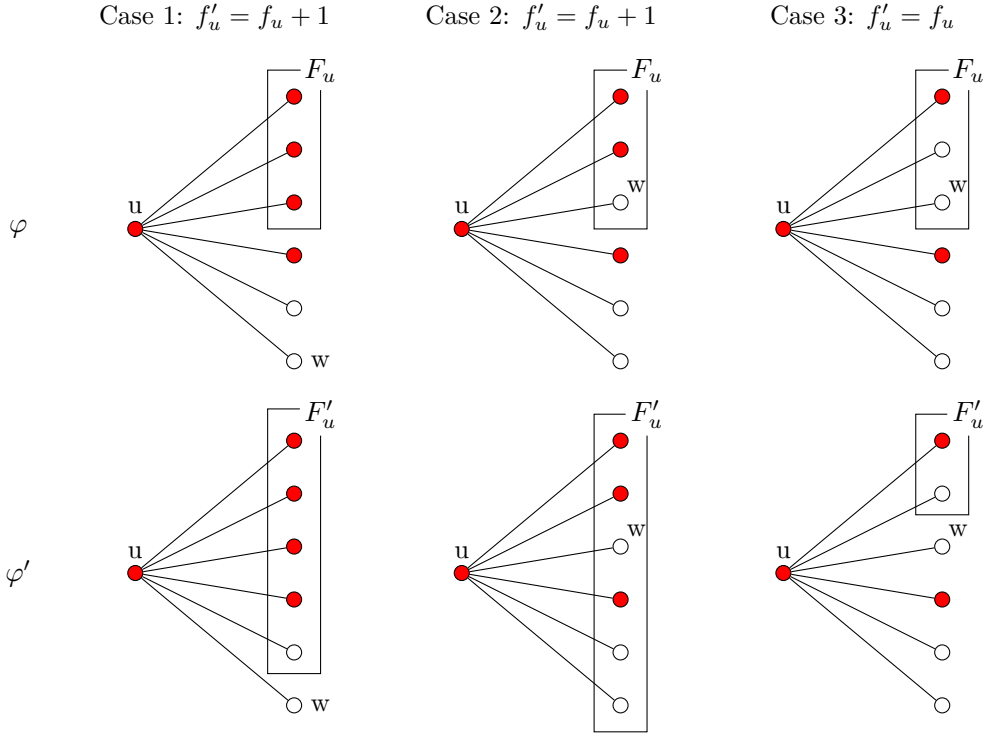
4.2 Proof of Theorem 5

To prove Theorem 5, we define the notion of *successor* of a failure pattern. Given $\varphi \in \Phi_{\text{all}}^{(t)}$, we say that a node u is *crashing last* in φ if there exists a triple $(u, F_u, f_u) \in \varphi$ (i.e., u crashes in φ), and, for every $(v, F_v, f_v) \in \varphi$, $f_u \geq f_v$.

► **Definition 11.** Let $\varphi \in \Phi_{\text{all}}^{(t)}$, let $(u, F_u, f_u) \in \varphi$, and assume that u is crashing last in φ . A successor of φ with respect to u is a failure pattern

$$\text{succ}(\varphi, u) = \left(\varphi \setminus \{(u, F_u, f_u)\} \right) \cup \{(u, F'_u, f'_u)\}$$

where F'_u and f'_u are defined as follows (see Fig. 3):



■ **Figure 3** A successor φ' of a failure pattern φ with respect to node u . Red nodes are faulty in φ and white nodes are correct in it.

1. If F_u contains only faulty nodes in φ , then $f'_u = f_u + 1$, and $F'_u = N(u) \setminus \{w\}$ for some arbitrary correct neighbor w of u .
2. If F_u contains exactly one correct node w in φ , then $f'_u = f_u + 1$, and $F'_u = N(u)$.
3. If F_u contains at least two correct nodes in φ , then $f'_u = f_u$, and $F'_u = F_u \setminus \{w\}$ for some arbitrary correct node $w \in F_u$.

Note that the correct node w in Definition 11 is well defined as the number of failures satisfies $t < \kappa(G) \leq \delta(G) \leq \deg(u)$, where $\delta(G)$ is the minimum degree of the nodes in G . Intuitively, $\text{succ}(\varphi, u)$ is identical to φ , except that u fails at round $f_u + 1$, or it still fails at round f_u but sends its message to one more correct neighbor before crashing.

Note also that a failure pattern may have different successors, which depends on the choice of the node u that crashes last, and on the choice of the correct neighbor w of u in the first and third cases of Definition 11. A correct neighbor w of u in Definition 11 is called a *witness* of the pair (φ, φ') .

Still using the notations of Definition 11, let us set $f''_u = f'_u$ in case 1, and $f''_u = f_u$ in cases 2 and 3. At the end of round f''_u , there is at most one correct node with different views in φ and $\text{succ}(\varphi, u)$. The only correct node may have different views in φ and $\varphi' = \text{succ}(\varphi, u)$ at the end of round f''_u is the *witness* of the pair (φ, φ') . Before applying the notion of successor to derive our lower bound, let us observe the following.

► **Lemma 12.** *For every node v , there exists a failure pattern $\varphi \in \Phi_v^*$ such that no node $u \neq v$ fails at round 1 in φ , and $\text{ecc}(v, \varphi) \geq \text{radius}(G, t)$.*

Proof. By definition of the radius, for every $v \in V$, there exists $\psi \in \Phi_v^*$ such that $\text{ecc}(v, \psi) \geq \text{radius}(G, t)$. The failure pattern φ is identical to ψ , except that, for every node $u \neq v$ that crashes at round 1 in ψ , u crashes cleanly at round 2 in φ . We have $\text{ecc}(v, \varphi) = \text{ecc}(v, \psi)$ because every node that crashes later in φ than in ψ does not send any message to their neighbors after round 1 which may contain information received from v . Thus $\text{ecc}(v, \varphi) \geq \text{radius}(G, t)$. ◀

The premises of the following lemma are justified by Lemma 12.

► **Lemma 13.** *Let $\varphi \in \Phi_{\text{all}}^{(t)}$ such that (1) at most one node crashes at round 1, and (2) if there exists a node v that crashes at round 1 in φ , then $\varphi \in \Phi_v^*$ (i.e., v broadcasts despite the fact that it crashes at round 1). For every successor φ' of φ , the following holds:*

- *at most one node crashes at round 1 in φ' ;*
- *if there is a node v that crashes at round 1 in φ' , then v crashes at round 1 in φ as well;*
- *there exists a correct node with the same view in φ and φ' at the end of round $\text{radius}(G, t) - 1$.*

Proof. Let φ' be a successor of φ , such that the entry (u, F_u, f_u) of φ is replaced by the entry (u, F'_u, f'_u) in φ' . Let w be a witness for the pair (φ, φ') with respect to u . Using the notations from Definition 11, let $f''_u = f'_u$ in Case 1, and $f''_u = f_u$ in Cases 2 and 3.

After f''_u rounds, the only correct node that may have different views in φ and φ' is w . Since u is a node crashing last in φ , we get that, after round f''_u , w needs the same number of rounds in φ and φ' for broadcasting to all correct nodes. Indeed, all nodes that have not crashed in φ nor in φ' up to round f''_u included satisfy: (1) they are correct nodes in both φ and φ' , (2) they have the same view in both φ and φ' , and (3) the subgraph of G induced by the correct nodes in φ is identical to the subgraph of G induced by the correct nodes in φ' .

Let $R = \text{radius}(G, t)$. We consider two cases, depending on whether w broadcasts or not.

Let us first consider the case where, assuming that w starts broadcasting at round $f''_u + 1$, w cannot broadcast to all correct nodes during rounds $f''_u + 1, \dots, R - 1$ under the failure patterns φ' and φ . That is, under φ' , some node s does not receive $\text{view}(w, f''_u, \varphi')$ during rounds $f''_u + 1, \dots, R - 1$. As a consequence, this node s does not detect any difference between $\text{view}(w, f''_u, \varphi)$ and $\text{view}(w, f''_u, \varphi')$. It follows that s has the same view in φ and φ' at the end of $R - 1$ rounds.

Consider now the case where, assuming that w starts broadcasting at round $f''_u + 1$, w does succeed to broadcast to all correct nodes during rounds $f''_u + 1, \dots, R - 1$ under the failure patterns φ' and φ . Since no node fails after round f''_u in both φ and φ' , a causal path from w to a node s in rounds $f''_u + 1, \dots, R - 1$ is also a causal path from s to w in rounds $f''_u + 1, \dots, R - 1$. At the end of round $R - 1$, every correct node can thus send to w its view at the end of round f''_u . Since no node $s \neq v$ fails at round 1, every node $s \neq v$ does send its input to some correct neighbor during round 1. Therefore, $s \in \text{view}(w, R - 1, \varphi)$ and $s \in \text{view}(w, R - 1, \varphi')$. Since $\varphi \in \Phi_v^*$, we get that, at the end of round f''_u , there exists a correct node x that heard from v , i.e., such that $v \in \text{view}(x, f''_u, \varphi)$. At the end of round $R - 1$, this node x will send $\text{view}(x, f''_u, \varphi)$ to w , so $v \in \text{view}(w, R - 1, \varphi)$. Similarly, $v \in \text{view}(w, R - 1, \varphi')$. As a consequence, $\text{view}(w, R - 1, \varphi) = \text{view}(w, R - 1, \varphi')$, and w has a same view in both failure patterns after $R - 1$ rounds, as claimed.

Furthermore, at most one node v crashes at round 1 in φ' , and $\varphi' \in \Phi_v^*$, as desired. ◀

Using the characterization of Theorem 9 of consensus solvability based on the information-flow graph, it is sufficient to prove the following result for establishing our lower bound.

► **Lemma 14.** *The information-flow graph $\text{IF}(G, \text{radius}(G, t) - 1)$ has a connected component that is not dominated by any node of V .*

Proof. Let $R = \text{radius}(G, t)$. For every node $v \in V$, we denote by φ_v a failure pattern in Φ_v^* such that φ_v contains no node $u \neq v$ that fails at round 1, and $\text{ecc}(v, \varphi_v) \geq R$. The existence of φ_v is guaranteed by Lemma 12. Borrowing the notation from [4], for every failure pattern φ , and every $r \geq 1$, let

$$\text{config}(\varphi, r) = \{(v, \text{view}(v, \varphi, r)) \in V(\text{IF}(G, r)) \mid v \in V \text{ is active in } \varphi \text{ at round } r\},$$

where by v is active in φ at round r , we mean that v has not crashed in φ during rounds $1, \dots, r$. It was proved in [4] (see Lemma 4 in there) that, for every failure pattern φ , and every $r \geq 1$, the subgraph of $\text{IF}(G, r)$ induced by the vertices of $\text{config}(\varphi, r)$ is connected.

We now show that, for every $v \in V$, $\text{config}(\varphi_v, R - 1)$ and $\text{config}(\varphi_\emptyset, R - 1)$ are contained in the same connected component of $\text{IF}(G, R - 1)$. Roughly, we shall construct a sequence of intermediate failure patterns from φ_v to φ_\emptyset such that, for every two consecutive failure patterns ψ and ψ' in the sequence, there is a correct node with the same view in ψ and ψ' . Note that the existence of this node implies that the subgraph of $\text{IF}(G, R - 1)$ induced by $\text{config}(\psi, R - 1)$, and the subgraph of $\text{IF}(G, R - 1)$ induced by $\text{config}(\psi', R - 1)$ are included in the same connected component of $\text{IF}(G, R - 1)$.

Let us order the crashing nodes in φ_v in a decreasing order of the rounds at which they crash where ties are broken arbitrarily, and let u_1, \dots, u_{t_v} be the resulting sequence. We have $t_v \leq t$ and, for every $i \in \{1, \dots, t_v - 1\}$, $f_{u_i} \geq f_{u_{i+1}}$. Let us construct a sequence $S = \psi_0, \dots, \psi_\ell$ of failure patterns, where $\psi_0 = \varphi_v$, and $\psi_\ell = \varphi_\emptyset$. This sequence is itself the concatenation of sub-sequences S_i for $i = 1, \dots, t_v$ such that $S_1 = \psi_0, \dots, \psi_{\ell_1}$, and, for every $i \in \{2, \dots, t_v\}$, $S_i = \psi_{\ell_{i-1}+1}, \dots, \psi_{\ell_i}$ with $0 \leq \ell_1 \leq \ell_2 \leq \dots \leq \ell_{t_v} = \ell$. For every sub-sequence S_i , $i \in \{1, \dots, t_v\}$, and for every $j \in \{\ell_{i-1} + 1, \dots, \ell_i - 1\}$, we set

$$\psi_{j+1} = \text{succ}(\psi_j, u_i).$$

Moreover, the first failure pattern $\psi_{\ell_{i-1}+1}$ in the sequence S_i is obtained from φ_v by removing the crashing nodes u_1, \dots, u_{i-1} , i.e., these nodes are correct in $\psi_{\ell_{i-1}+1}$. The last failure pattern ψ_{ℓ_i} of the sequence S_i is when the node u_i that crashes last in ψ_{ℓ_i} fails at round R .

▷ **Claim 15.** For any two consecutive failure patterns ψ_j and ψ_{j+1} in S , there exists a correct node w_j with the same view in both patterns after $R - 1$ rounds, that is,

$$\text{view}(w_j, \psi_j, R - 1) = \text{view}(w_j, \psi_{j+1}, R - 1).$$

To see why the claim holds, let us first assume that ψ_j and ψ_{j+1} belong to a same sub-sequence S_i . In this case, the claim directly follows from Lemma 13. If ψ_j and ψ_{j+1} do not belong to a same sub-sequence S_i , then ψ_j is the last element of a sub-sequence S_i , and ψ_{j+1} is the first element of sub-sequence S_{i+1} , then the claim follows from the fact that the sets of nodes crashing in ψ_j and ψ_{j+1} during round r are the same, for every $r \in \{1, \dots, R - 1\}$. This completes the proof of Claim 15.

From Claim 15, for any two consecutive failure patterns ψ_j and ψ_{j+1} in S , $\text{config}(\psi_j)$ and $\text{config}(\psi_{j+1})$ belong to the same connected component of $\text{IF}(G, R - 1)$. To wrap up, we have shown that, for every $v \in V$, there exists a connected component of $\text{IF}(G, R - 1)$ containing both $\text{config}(\varphi_\emptyset)$ and $\text{config}(\varphi_v)$. Recall that φ_v is a failure pattern in Φ_v^* satisfying that it contains no node different from v that fails at round 1, and $\text{ecc}(v, \varphi_v) \geq R$. At the end of round $R - 1$, no node dominates the component that contains $\text{config}(\varphi_\emptyset)$ because, for every node $v \in V$, v cannot dominates $\text{config}(\varphi_v, R - 1)$. ◀

5 Conclusion

In this paper, we have completed the picture for consensus in the t -resilient model for arbitrary graphs. That is, we have proved that the consensus algorithm in [4] is optimal, i.e., for every graph G and $t < \kappa(G)$, consensus can be solved by an oblivious algorithm performing in $\text{radius}(G, t)$ rounds under the t -resilient model, and no oblivious algorithms can solve consensus in G in less than $\text{radius}(G, t)$ rounds under the t -resilient model. Moreover, we have extended the study of consensus beyond the connectivity threshold. Specifically, we defined the *local* consensus task, a generalization of consensus. We designed and analyzed a generic algorithm for this task, which we believe to be optimal among oblivious algorithms. The technical difficulty of establishing optimality of our algorithm for the local variant of consensus yields from the fact that we miss an analog of our characterization theorem (cf. Theorem 9 in Section 4) for local consensus. Finally, we have generalized the algorithm in [4] for consensus, as well as our algorithm for local consensus, to k -set agreement.

Our results open a vast domain for further investigations. In particular, what could be said for sets of failure patterns Φ distinct from $\Phi_{\text{all}}^{(t)}$? The case Φ_{clean} of clean failures, for which there are no known generic consensus algorithms applying to arbitrary graphs, is particularly intriguing. Another intriguing and potentially challenging area for further research is exploring scenarios where no upper bounds on the number of failing nodes are assumed, by concentrating solely on the set Φ_{connect} of failure patterns that do not result in disconnecting the graph. The main difficulty is that basic results such as Lemma 1 in [4] (cf. Proposition 1) do not hold anymore in this framework. Indeed, some ill behaviors that do not occur when the number of failures is bounded from above by the connectivity of the graph, or when the problems are considered in each connected component separately, pop up when the number of failures is arbitrarily large yet preserving connectivity. Finally, the design of early-stopping algorithms in the t -resilient model for arbitrary graphs is also highly desirable. The early-stopping algorithms in [8] are very promising, but their analysis must be refined to a grain finer than the stretches of the failure patterns, by focusing on, e.g., eccentricities and radii.

References

- 1 Marcos Kawazoe Aguilera and Sam Toueg. A simple bivalency proof that t -resilient consensus requires $t+1$ rounds. *Information Processing Letters*, 71(3-4):155–158, 1999. doi:10.1016/S0020-0190(99)00100-3.
- 2 Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.
- 3 Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. A topological perspective on distributed network algorithms. *Theoretical Computer Science*, 849:121–137, 2021. doi:10.1016/J.TCS.2020.10.012.
- 4 Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. Synchronous t -resilient consensus in arbitrary graphs. *Inf. Comput.*, 292:105035, 2023. doi:10.1016/J.IC.2023.105035.
- 5 Armando Castañeda, Yoram Moses, Michel Raynal, and Matthieu Roy. Early decision and stopping in synchronous consensus: A predicate-based guided tour. In *5th International Conference on Networked Systems – (NETYS)*, volume 10299 of *LNCS*, pages 206–221, 2017. doi:10.1007/978-3-319-59647-1_16.
- 6 Bernadette Charron-Bost and Stephan Merz. Formal verification of a consensus algorithm in the heard-of model. *Int. J. Softw. Informatics*, 3(2-3):273–303, 2009. URL: http://www.ijsi.org/ch/reader/view_abstract.aspx?file_no=273&flag=1.


- 7 Bernadette Charron-Bost and André Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Comput.*, 22(1):49–71, 2009. doi:10.1007/S00446-009-0084-6.
- 8 Bogdan S. Chlebus, Dariusz R. Kowalski, Jan Olkowski, and Jędrzej Olkowski. Disconnected agreement in networks prone to link failures. In *25th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 14310 of *LNCS*, pages 207–222. Springer, 2023. doi:10.1007/978-3-031-44274-2_16.
- 9 Étienne Coulouma and Emmanuel Godard. A characterization of dynamic networks where consensus is solvable. In *International Colloquium on Structural Information and Communication Complexity*, pages 24–35. Springer, 2013. doi:10.1007/978-3-319-03578-9_3.
- 10 Carole Delporte-Gallet, Hugues Fauconnier, Sergio Rajsbaum, and Nayuta Yanagisawa. A characterization of t-resilient colorless task anonymous solvability. In *25th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 11085 of *LNCS*, pages 178–192. Springer, 2018. doi:10.1007/978-3-030-01325-7_18.
- 11 Carole Delporte-Gallet, Hugues Fauconnier, and Andreas Tielmann. Fault-tolerant consensus in unknown and anonymous networks. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 368–375, 2009. doi:10.1109/ICDCS.2009.36.
- 12 Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982. doi:10.1016/0196-6774(82)90004-9.
- 13 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983. doi:10.1137/0212045.
- 14 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000.
- 15 Pierre Fraigniaud, Patrick Lambein-Monette, and Mikaël Rabie. Fault tolerant coloring of the asynchronous cycle. In *36th International Symposium on Distributed Computing (DISC)*, volume 246 of *LIPICs*, pages 23:1–23:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.DISC.2022.23.
- 16 Pierre Fraigniaud, Minh Hang Nguyen, and Ami Paz. Agreement tasks in fault-prone synchronous networks of arbitrary structure. *CoRR arXiv*, abs/2410.21538, 2024. doi:10.48550/arXiv.2410.21538.
- 17 Pierre Fraigniaud and Ami Paz. The topology of local computing in networks. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPICs*, pages 128:1–128:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.128.
- 18 Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.
- 19 Juho Hirvonen and Jukka Suomela. *Distributed Algorithms*. Aalto University, Finland, 2023.
- 20 Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. Exact byzantine consensus on undirected graphs under local broadcast model. In *PODC*, pages 327–336. ACM, 2019. doi:10.1145/3293611.3331619.
- 21 Giuseppe Antonio Di Luna and Giovanni Viglietta. Computing in anonymous dynamic networks is linear. In *63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1122–1133, 2022. doi:10.1109/FOCS54457.2022.00108.
- 22 Giuseppe Antonio Di Luna and Giovanni Viglietta. Optimal computation in leaderless and multi-leader disconnected anonymous dynamic networks. In *37th International Symposium on Distributed Computing (DISC)*, volume 281 of *LIPICs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.DISC.2023.18.
- 23 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- 24 Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *Proceedings of the 2019 ACM symposium on principles of distributed computing*, pages 218–227, 2019. doi:10.1145/3293611.3331624.
- 25 David Peleg. *Distributed Computing: A Locality-sensitive Approach*. SIAM, 2000.

- 26 Michel Raynal. Consensus in synchronous systems: A concise guided tour. In *9th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 221–228. IEEE, 2002. doi:10.1109/PRDC.2002.1185641.
- 27 Michel Raynal. *Fault-tolerant Agreement in Synchronous Message-passing Systems*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2010. doi:10.2200/S00294ED1V01Y201009DCT003.
- 28 Michel Raynal and Corentin Travers. Synchronous set agreement: A concise guided tour. In *12th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 267–274, 2006.
- 29 Kyrill Winkler, Ami Paz, Hugo Rincon Galeana, Stefan Schmid, and Ulrich Schmid. The time complexity of consensus under oblivious message adversaries. *Algorithmica*, pages 1–32, 2024.

Dimension-Free Parameterized Approximation Schemes for Hybrid Clustering

Ameet Gadekar ✉ 

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Tanmay Inamdar ✉ 

Indian Institute of Technology Jodhpur, India

Abstract

HYBRID k -CLUSTERING is a model of clustering that generalizes two of the most widely studied clustering objectives: k -CENTER and k -MEDIAN. In this model, given a set of n points P , the goal is to find k centers such that the sum of the r -distances of each point to its nearest center is minimized. The r -distance between two points p and q is defined as $\max\{\text{dist}(p, q) - r, 0\}$ – this represents the distance of p to the boundary of the r -radius ball around q if p is outside the ball, and 0 otherwise. This problem was recently introduced by Fomin et al. [APPROX 2024], who designed a $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation that runs in time $2^{(kd/\varepsilon)^{O(1)}} \cdot n^{O(1)}$ for inputs in \mathbb{R}^d ; such a bicriteria solution uses balls of radius $(1 + \varepsilon)r$ instead of r , and has a cost at most $1 + \varepsilon$ times the cost of an optimal solution using balls of radius r .

In this paper we significantly improve upon this result by designing an approximation algorithm with the same bicriteria guarantee, but with running time that is FPT only in k and ε – crucially, removing the exponential dependence on the dimension d . This resolves an open question posed in their paper. Our results extend further in several directions. First, our approximation scheme works in a broader class of metric spaces, including doubling spaces, minor-free, and bounded treewidth metrics. Secondly, our techniques yield a similar bicriteria FPT-approximation schemes for other variants of HYBRID k -CLUSTERING, e.g., when the objective features the sum of z -th power of the r -distances. Finally, we also design a coresset for HYBRID k -CLUSTERING in doubling spaces, answering another open question from the work of Fomin et al.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Fixed parameter tractability

Keywords and phrases Clustering, Parameterized algorithms, FPT approximation, k -Median, k -Center

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.35

Related Version Full Version: <https://arxiv.org/abs/2501.03663>

Funding Ameet Gadekar: Partially supported by the Israel Science Foundation (grant No. 1042/22). Tanmay Inamdar: Supported by IITJ Research Initiation Grant (grant number I/RIG/T-NI/20240072).

Acknowledgements Tanmay would like to thank his co-authors from [23] – specifically Fedor V. Fomin – for formulating Hybrid k -Clustering problem and introducing him to it. This work was partially carried out while Ameet was at Bar-Ilan University, Israel.

1 Introduction

k -CENTER, k -MEDIAN, and k -MEANS are among the most popular clustering problems both in theory and practice, with numerous applications in areas such as machine learning [36, 28, 6, 11, 4, 9, 38, 25], facility location problems [32, 41, 30, 29, 35, 16], and computational



© Ameet Gadekar and Tanmay Inamdar;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 35; pp. 35:1–35:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



geometry [7, 34], among others.¹ All of these problems have long been known to be NP-hard, even in the plane [22, 40, 37]. To cope with these intractability results, there has been several decades of research on designing approximation algorithms for these problems, that have lead to polynomial-time constant-factor approximation algorithms for all of them in general metric spaces [27, 20]. Moreover, for more “structured” inputs – such as when the points belong to Euclidean spaces, or planar graph metrics – one can find near-optimal solutions using approximation schemes [24, 17, 19, 5, 34, 7, 2].² Furthermore, given the simplicity and ubiquity of these vanilla clustering objectives, the techniques used for obtaining these results are subsequently used to find clustering with additional constraints, such as capacities [18], fairness [8], and outliers [31, 3].¹ One recurring theme in the literature has been to define a unified clustering objective that also captures classical clustering problems like k -MEDIAN, k -CENTER as special cases [42, 12, 15].

Along this line of research, Fomin et al. [23] recently introduced HYBRID k -CLUSTERING. In this problem, we are given an instance $\mathcal{I} = (M = (P, \mathbb{F}, \text{dist}), k, r)$, where $M = (P, \mathbb{F}, \text{dist})$ is a metric space³ on the set of clients P and a set of facilities \mathbb{F} with distance function dist . Here, k is a positive integer denoting the number of clusters, and r is a non-negative real denoting the radius. The goal is to find a set $X \subseteq \mathbb{F}$ of k centers, such that, $\text{cost}_r(P, X) := \sum_{p \in P} \text{dist}_r(p, X)$ is minimized – here, for any point p , any set $Q \subseteq P \cup \mathbb{F}$, and a real α , $\text{dist}_\alpha(p, Q)$ is defined as $\text{dist}_\alpha(p, Q) := \max\{\text{dist}(p, Q) - \alpha, 0\}$, and $\text{dist}(p, Q) := \min_{q \in Q} \text{dist}(p, q)$. Fomin et al. [23] proposed several motivations for studying this cost function. Indeed, HYBRID k -CLUSTERING can be seen as a *shape fitting* problem, where one wants to find the “best” set of k balls of radius r that fit the given set of points, where the quality of the solution is determined by the sum of distances of each point to the nearest point on the boundary of the ball – this is analogous to the classical regression, where one wants to find the “best” linear function fitting the given set of points. PROJECTIVE CLUSTERING is a well-studied generalization of linear regression, where the aim is to find k affine spaces that minimizes the sum of distances from the points to these spaces (see e.g., [43]), which is also closely related to the model considered in HYBRID k -CLUSTERING. Furthermore, [23] gave another motivation for studying HYBRID k -CLUSTERING – namely, placing k WiFi routers with identical circular coverage, where the clients that lie outside the coverage area need to travel to the boundary of the nearest ball in order to receive coverage. Finally, the name of the problem is motivated from the fact that the objective is a kind of “hybrid” between the k -CENTER and k -MEDIAN costs, and generalizes both of them. Indeed, as observed in [23], HYBRID k -CLUSTERING with $r = 0$ is equivalent to k -MEDIAN, while, when r is set to be the optimal radius for k -CENTER, HYBRID k -CLUSTERING reduces to k -CENTER. These observations immediately rule out *uni-criteria* polynomial-time approximation schemes that violate only the cost, or only the radius by any arbitrary factor $\alpha > 1$ (as formalized in Proposition 1 of [23]). Indeed, such approximation algorithms – even with running times FPT in k – would imply exact FPT algorithms for k -CENTER and k -MEDIAN in low-dimensional continuous Euclidean spaces. To our knowledge, such an algorithm for k -MEDIAN is not known (nor is admittedly a lower bound), whereas [39] shows W[1]-hardness of k -CENTER even in \mathbb{R}^2 . Therefore, given the current state of the art, a *bicriteria* approximation scheme for HYBRID k -CLUSTERING is essentially the best outcome, even if one allows a running time that is FPT in k .

¹ These citations are supposed not to be comprehensive, but representative – an interested reader may use them as a starting point for the relevant literature.

² This includes polynomial-time as well as FPT approximation schemes.

³ Fomin et al. [23] only considered the special case of Euclidean inputs, i.e., when $P \subset \mathbb{F} = \mathbb{R}^d$; however, the underlying problem can be defined for arbitrary metric spaces.

For $\alpha, \beta \geq 1$, an (α, β) -*bicriteria approximation* for HYBRID k -CLUSTERING is an algorithm that returns a solution $X \subseteq \mathbb{F}$ of size at most k satisfying $\text{cost}_{\beta r}(P, X) \leq \alpha \cdot \text{OPT}_r$. Note that the bicriteria solution X is allowed to consider $\text{dist}_{\beta r}(p, X) = \max\{\text{dist}(p, X) - \beta r, 0\}$ instead of $\text{dist}_r(p, X)$ and is also allowed to find a solution of cost αOPT_r , where OPT_r is the cost of an optimal solution w.r.t. radius r , i.e., without any violation of the radius. The main result of Fomin et al. [23] was an $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation for inputs in \mathbb{R}^d in time $2^{(kd/\varepsilon)^{O(1)}} \cdot n^{O(1)}$, where $n = |P|$. An exponential dependence on the dimension d appears inevitable using their approach, although it was not clear whether such a dependence is required. Indeed, for Euclidean k -MEDIAN and k -CENTER, one can obtain *dimension-free* approximation schemes with FPT dependence only on k and ε [7, 33, 1]. This naturally motivates the following question, which was explicitly asked as an open question in [23].

Question 1. “An immediate question is whether improving or removing the FPT dependence on the dimension d is possible[...]”

In this work, we answer this question in the affirmative by designing a (randomized) bicriteria FPT Approximation Scheme (FPT-AS) parameterized by k and ε^4 for the (continuous) Euclidean instances of HYBRID k -CLUSTERING, stated formally in the following theorem.

► **Theorem 1** (Bicriteria FPT-AS for Euclidean Spaces). *There exists a randomized algorithm, that, given an instance of HYBRID k -CLUSTERING in \mathbb{R}^d for any dimension d , runs in time $2^{O(k \log k \cdot (1/\varepsilon^5) \log^2(1/\varepsilon))} \cdot n^{O(1)}$, and returns a $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation with high probability.*

The algorithm of [23] involves a pre-processing step that separately handles “ k -MEDIAN-like”, “ k -CENTER-like” instances, using the techniques specific to these respective problems. Then, the main algorithm handles the remaining instances that cannot be directly reduced to the respective problems. This approach somewhat undermines the goal of defining a unified problem that captures both of these problems as special cases. In contrast, our algorithm adopts a uniform approach by exploiting the intrinsic structure of the hybrid problem, without the need to separately handle some instances. In fact, our algorithm works for a broader class of metric spaces, called metric spaces of *bounded algorithmic scatter dimension* – a notion recently introduced by Abbasi et al. [1], and further studied by Bourneuf and Pilipczuk [10]. These metric spaces capture several interesting and well-studied metric spaces, see below. We give a formal definition of the notion of algorithmic scatter dimension in Section 2; however, a good mental picture to keep is to think of them as essentially doubling spaces, i.e., metric spaces with good “packing-covering” properties.⁵ Our general result is stated below.

► **Theorem 2** (Informal version of Theorem 9). *HYBRID k -CLUSTERING admits a randomized bicriteria FPT-AS in metrics of bounded algorithmic ε -scatter dimension, parameterized by k and ε . In particular, HYBRID k -CLUSTERING admits randomized bicriteria FPT-AS parameterized by k and ε , in continuous Euclidean spaces of any dimension, metrics of bounded doubling dimension, bounded treewidth metrics, and by graphs from any fixed proper minor-closed graph class.*

⁴ Such algorithms run in time $f(k, \varepsilon)n^{O(1)}$ and output a $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria solution. Another term for FPT-AS is EPAS, which stands for Efficient Parameterized Approximation Schemes.

⁵ Although this is a good mental picture, it is ultimately inaccurate since the class of metric spaces of bounded algorithmic scatter dimension is strictly larger than that of doubling spaces. Indeed, continuous Euclidean space of high $(\omega(\log n))$ dimension does not have bounded doubling dimension, yet it does have bounded algorithmic scatter dimension.

We give a technical overview of this result in Section 1.1, and describe the approximation algorithm and its analysis in Section 3.

In their work, [23] also defined HYBRID (k, z) -CLUSTERING problem, which features the z -th power of r -distances, i.e., the objective is to minimize $\text{cost}_r(P, X, z) := \sum_{p \in P} (\text{dist}_r(p, X))^z$, where $z \geq 1$. They designed a bicriteria FPT-AS with similar running time for HYBRID $(k, 2)$ -CLUSTERING, i.e., a hybrid of k -MEANS and k -CENTER; but left the possibility of obtaining a similar result for the general case of $z \geq 1$ conditional on the existence of a certain sampling-based approximation algorithm for the (k, z) -clustering problem for Euclidean inputs. Using our approach, we can obtain a bicriteria FPT-AS for any fixed $z \geq 1$ in a unified way, whose running time is independent of the dimension d . In fact, our approach works for a much more general problem of HYBRID NORM k -CLUSTERING. We discuss these extensions in Section 4.

Next, we turn to another open direction mentioned in the work of Fomin et al. [23].

Question 2. “Another intriguing question is the design of coresets for HYBRID k -CLUSTERING, which could also have some implications for [Question 1].”

In this paper, we also answer this question affirmatively by designing coresets for HYBRID k -CLUSTERING in metric spaces of bounded doubling dimension. Specifically, we prove the following result.

► **Theorem 3 (Coreset for HYBRID k -CLUSTERING).** *There exists an algorithm that takes as input an instance $\mathcal{I} = ((P, \mathbb{F}, \text{dist}), k, r)$ of HYBRID k -CLUSTERING in doubling metric of dimension d and a parameter $\varepsilon \in (0, 1)$, and in time $2^{O(d \log(1/\varepsilon))} |\mathcal{I}|^{O(1)}$ returns a pair (P', w) , where $P' \subseteq P$ has size $2^{O(d \log(1/\varepsilon))} k \log |P|$, and $w : P' \rightarrow \mathbb{N}$, such that the following property is satisfied for any $X \subseteq \mathbb{F}$ of size at most k :*

$$|\text{wcost}_r(P', X) - \text{cost}_r(P, X)| \leq \varepsilon \text{cost}_r(P, X)$$

Here, $\text{wcost}_r(P, X) := \sum_{p \in P'} w(p) \cdot \text{dist}_r(p, X)$.

1.1 Technical Overview

In this section, we highlight our technical and conceptual contributions of our main result (Theorem 2).

A natural direction to design dimension-free parameterized approximation algorithm for HYBRID k -CLUSTERING is to leverage insights from the framework of Abbasi et al. [1], who designed dimension-free parameterized approximations for wide range of clustering problems. However, in HYBRID k -CLUSTERING, the objective is a function of r -distances, instead of true distances, as in [1]. This introduces several technical challenges. For instance, a most obvious roadblock is that the r -distances do not satisfy triangle inequality. Moreover, many of the “nice” properties that are enjoyed by the true distances in “structured” metric spaces (e.g., covering-packing property in doubling spaces, or properties of shortest paths in sparse graphs) do not extend readily to the r -distances. Therefore, we have to overcome several technical and conceptual challenges in order to leverage the ideas presented in the framework of [1]. Let us first recall the key notion from this paper that is crucially used in this framework.⁶

⁶ In fact, the algorithm of our paper and [1] works for a weaker notion called algorithmic scatter dimension. But, for ease of exposition, we work with scatter dimension in this section.

Scatter Dimension. Informally, an ε -scattering sequence in a metric space $M = (P, \mathbb{F}, \text{dist})$ is a sequence of center-point pairs $(x_1, p_1), \dots, (x_\ell, p_\ell)$, where ℓ is some positive integer and for $j \in [\ell], x_j \in \mathbb{F}, p_j \in P$ such that $\text{dist}(p_j, x_i) \leq 1$, for all $1 \leq j < i \leq [\ell]$ and $\text{dist}(p_i, x_i) > (1 + \varepsilon)$, for all $i \in [\ell]$. The ε -scatter dimension of M is the maximum length of ε -scattering sequence contained in M .

Now, consider a natural extension of the framework [1] for HYBRID k -CLUSTERING in a metric space M as follows. Let OPT_r be the optimal cost of the HYBRID k -CLUSTERING instance corresponding to an optimal solution O . The algorithm maintains cluster constraint Q_i for each cluster $i \in [k]$. Each Q_i consists of a collection of requests of the form (p, δ) , where p is a point and δ is a distance, representing the demand that p requires a center within distance δ . The algorithm always maintains a solution X such that $x_i \in X$ satisfy Q_i cluster constraint, for all $i \in [k]$. Now consider the sequence of triples $S_i = (x_i^{(1)}, p_i^{(1)}, \delta_i^{(1)}), \dots, (x_i^{(\ell)}, p_i^{(\ell)}, \delta_i^{(\ell)})$ corresponding to requests in Q_i , where $x_i^{(j)}$ is the i^{th} center maintained by the algorithm just before adding request $(p_i^{(j)}, \delta_i^{(j)})$ to Q_i . If X is a near-optimal solution, then the algorithm terminates successfully and returns X . Otherwise, it identifies a point $p \in P$ whose r -distance to X is much larger than its r -distance to O . Such a point is called a *witness* to X . A simple averaging argument shows that such a witness can be sampled with high probability. The algorithm then guesses the optimal cluster $i \in [k]$ of p and add a new request $(p, \delta_p = \text{dist}(p, X)/(1 + \varepsilon'))$ to Q_i , for some suitable but fixed ε' depending on ε . The center x_i is recomputed to satisfy the updated cluster constraint Q_i , if possible. Otherwise the algorithm reports failure.⁷ The key observation is that the sequence of pairs $(x_i^{(j_1)}, p_i^{(j_1)}), \dots, (x_i^{(j_\ell)}, p_i^{(j_\ell)})$ for a fixed radius $\delta_i^{(j)}$ forms an ε -scattering sequence in M . Thus, if the ε -scatter dimension of M is bounded then the length of these sequences are also bounded. Furthermore, it can be shown that if the aspect ratio of the radii of requests in Q_i for all $i \in [k]$ is bounded, then the number of iterations of the algorithm can be bounded, yielding an FPT-AS.

Working with the inflated radius. A major challenge that arises when the witness is defined in terms of r -distances, but the requests added to the cluster constraints are based on the true distances. Specifically, we need to ensure that a witness whose r -distance to X is significantly larger than its r -distance to O also has a larger true distance to X than to O . This ensures that the request $(p, \text{dist}(p, X)/(1 + \varepsilon'))$ is feasible and the algorithm does not fail. However, this condition may not hold, especially when the true distances are close to r . In fact, the issue is related to a fundamental barrier identified by [23], assuming standard conjectures.⁸ To overcome this fundamental barrier and maintain a sufficient gap between the true distances, we consider the cost of X with respect to $(1 + \varepsilon)r$ radius. In other words, we look for a solution X whose $(1 + \varepsilon)r$ -cost is close to OPT_r . In this case, we redefine a witness as a point in P whose $(1 + \varepsilon)r$ -distance to X is much larger than its r -distance to O . These insights allow us to establish a gap between the true distance of a witness to X and its true distance to O , thereby justifying the requests added by the algorithm.

Bounding the aspect ratio of the radii. Abbasi et al. [1] bound the aspect ratio of the radii by (i) initializing the solution using “good” upper bounds, and (ii) sampling witness from the points that have comparable distance to X with respect to the corresponding upper bounds.

⁷ E.g., if the algorithm failed to sample a witness.

⁸ Such a framework could potentially yield a near-optimal solution that does not violate the radius, contradicting Proposition 1 in [23].

The first step guarantees that the solution X maintained by the algorithm always satisfies the upper bounds within constant factor. This together with the second step allows one to bound the aspect ratio of radii in each Q_i . They show that such witnesses have a good probability mass if the points are sampled proportional to their true distances, and hence the algorithm successfully finds a witness for X with probability that is a function of k and ε . Although, devising feasible and “good” upper bounds for HYBRID k -CLUSTERING can be done with some efforts, the main challenge arises in the second step, which only guarantees to sample a point whose $(1 + \varepsilon)r$ -distance to X is comparable with its upper bound. As mentioned before, these $(1 + \varepsilon)r$ -distances can be much smaller than the corresponding true distances, and hence there is no guarantee that true distance of a witness p to current solution X is comparable to its upper bound. Towards this, we introduce a novel idea of dividing witnesses into two sets – “nearby witness” set, i.e., the set of points within distance $O(r/\varepsilon)$ from X , and “faraway witness” set, which are beyond this distance from X . The observation is that, since the total probability mass on the witness set, now defined using $(1 + \varepsilon)r$ -distances, is still high, it must be that either the nearby or the faraway witnesses have sufficient probability mass. However, since we do not know which of the two sets has enough mass, we perform a randomized branching (i.e., make a guess) – which will be “correct” with probability $1/2$. Then, when a point is sampled proportional to its $(1 + \varepsilon)r$ -distance from either the “nearby” or “faraway” set, it will be a witness with good probability. Now consider each of two cases separately. Since for a nearby witness p , its true distance to X is at least $(1 + \varepsilon)r$ and at most $O(r/\varepsilon)$, the aspect ratio of the radii of requests corresponding to nearby witness set is bounded by $O(1/\varepsilon)$. On the other hand, for requests corresponding to faraway witness set, we show that their radii lie in bounded interval, using ideas similar to [1]. Note that, these two arguments imply that the radii of the requests lie in two (possibly disjoint) intervals that themselves are bounded. However, it is not clear if the length of these requests is bounded, unlike [1], where the radii belonged to a single interval of bounded aspect ratio. Nevertheless, we observe that, using the techniques of [1], the length of requests can be bounded, even when the radii lie in constantly many (here, two) intervals, each with a bounded aspect ratio.

2 Background on Algorithmic Scatter Dimension

In this paper, we consider metric (clustering) space $M = (P, \mathbb{F}, \text{dist})$, where P is a finite set of n points, \mathbb{F} is a (possibly infinite) set of potential cluster centers, and dist is a metric on $(P \cup \mathbb{F})$. A class \mathcal{M} of metric spaces is a (possibly infinite) set of metric spaces.

In this paper, we work with a notion that is weaker (and hence more general) than ε -scatter dimension that was defined in the overview, called *algorithmic ε -scatter dimension*, which we explain next. To this end, we first need the following search problem.

► **Definition 4** (BALL INTERSECTION Problem). *Let \mathcal{M} be a class of metric spaces with possibly infinite set of centers. Given $M = (P, \mathbb{F}, \text{dist}) \in \mathcal{M}$, a finite set $Q \subseteq P \times \mathbb{R}_+$ of distance constraints, and an error parameter $\eta > 0$, the Ball Intersection problem asks to find a center $x \in \mathbb{F}$ that satisfy all distance constraints within η multiplicative error, i.e., $\text{dist}(x, p) \leq (1 + \eta)\delta$, for every $(p, \delta) \in Q$, if such a center exists, and report failure otherwise.*

We say \mathcal{M} admits a BALL INTERSECTION algorithm if it correctly solves the ball intersection problem for every metric space in \mathcal{M} and runs in polynomial time in the size of M and $1/\eta$.

Now, we are ready to define algorithmic scatter dimension.

► **Definition 5** (Algorithmic ε -Scatter Dimension). *Given a class \mathcal{M} of metric spaces with BALL INTERSECTION algorithm $\mathcal{C}_{\mathcal{M}}$, a space $M \in \mathcal{M}$, and $\varepsilon \in (0, 1)$, a $(\mathcal{C}_{\mathcal{M}}, \varepsilon)$ -scattering sequence is a sequence $(x_1, p_1, \delta_1), \dots, (x_\ell, p_\ell, \delta_\ell)$, where ℓ is some positive integer, and for $i \in [\ell]$, $x_i \in \mathbb{F}$, $p_i \in P$ and $\delta_i \in \mathbb{R}_+$ such that*

- (Covering by $\mathcal{C}_{\mathcal{M}}$) $x_i = \mathcal{C}_{\mathcal{M}}(M, \{(p_1, \delta_1), \dots, (p_{i-1}, \delta_{i-1})\}, \varepsilon/2) \quad \forall 2 \leq i \leq \ell$
- (ε -refutation) $\text{dist}(x_i, p_i) > (1 + \varepsilon)\delta_i \quad \forall i \in [\ell]$

The algorithmic $(\varepsilon, \mathcal{C}_{\mathcal{M}})$ -scatter dimension of \mathcal{M} is $\lambda_{\mathcal{M}}(\varepsilon)$ if any $(\mathcal{C}_{\mathcal{M}}, \varepsilon)$ -scattering sequence contains at most $\lambda_{\mathcal{M}}(\varepsilon)$ many triples per radius value. The algorithmic ε -scatter dimension of \mathcal{M} is the minimum algorithmic $(\varepsilon, \mathcal{C}_{\mathcal{M}})$ -scatter dimension over any BALL INTERSECTION algorithm $\mathcal{C}_{\mathcal{M}}$ for \mathcal{M} .

Although, algorithmic scatter dimension restricts the number of triples in the sequence with same radius value, we can use the proof technique from [1] to prove the following stronger guarantee. Results marked with ♠ can be found in the full version of the paper.

► **Lemma 6** (♠). *Let \mathcal{M} be a class of metric spaces of algorithmic ε -scatter dimension $\lambda(\varepsilon)$. Then there exists a BALL INTERSECTION algorithm $\mathcal{C}_{\mathcal{M}}$ with the following property. Given $\varepsilon \in (0, 1)$, a constant $t \geq 1$, and $a_i > 0, \tau_i \geq 2$ for $i \in [t]$, any $(\mathcal{C}_{\mathcal{M}}, \varepsilon)$ -scattering contains $O(\sum_{i \in [t]} \lambda(\varepsilon/2)(\log \tau_i)/\varepsilon)$ many triples whose radii lie in the interval $\cup_{i \in [t]} [a_i, \tau_i a_i]$.*

3 Bicriteria FPT Approximation Scheme

3.1 Algorithm

Our bicriteria FPT-AS for HYBRID k -CLUSTERING is formally stated in Algorithm 1. As an input, we are given an instance $\mathcal{I} = ((P, \mathbb{F}, \text{dist}), k, r)$ of HYBRID k -CLUSTERING, an accuracy parameter ε , access to an algorithm \mathcal{C} for the so-called “BALL INTERSECTION” problem (discussed later), and a guess \mathcal{G} for the optimal cost OPT_r . By a standard exponential search, we will assume that $\text{OPT}_r \leq \mathcal{G} \leq (1 + \varepsilon/3) \cdot \text{OPT}_r$. At a high level, this algorithm can be divided into two steps: *initialization phase* and the *iterative cost-improvement phase*. The initialization phase spans from line 1 to 7.

At a high level, the goal of this phase to compute for each point $p \in P$, an *upper bound* $u(p)$, such that p must have an optimal center within distance $u(p)$. Once we find such upper bounds, we use a subset of them to initialize for each $i \in [k]$, a set of *requests* Q_i , where each request (p, δ_p) demands that the i th center in the solution must be within distance at most δ_p from p in every subsequent solution found by the algorithm.

Now, the algorithm moves to the iterative cost-improvement phase in lines 8 to 20, consisting of a while loop that runs as long as the current solution has not become the bicriteria approximation that we are looking for. Thus, in each iteration of the while loop, we are given a solution X that satisfies all the requests Q_i , and yet satisfies $\text{cost}_{r'}(P, X) > (1 + \varepsilon) \cdot \mathcal{G}$. Then, our algorithm makes a random choice whether there is enough cost-contribution of nearby witnesses, or of faraway witnesses – here a witness is a point p whose distance to X is sufficiently larger than that in the (unknown) optimal solution O . In each of the cases, we sample points from carefully defined sets (cf. N in line 11 and A in 14), proportional to their contribution to the cost to the respective sets (in the analysis, we will argue that with good probability, we will in fact sample witness points). Having sampled such a point p , we guess the index i of the optimal cluster of p in line 17. Finally, assuming p is indeed a witness, we add a request $(p, \frac{\text{dist}(p, X)}{1 + \varepsilon/12})$ to the i th request set Q_i , and in line 19 we recompute x_i using \mathcal{C} . This algorithm either returns a center $x_i \in \mathbb{F}$ that satisfies all the requests (with an error of up to a factor of $\varepsilon/40$), or correctly outputs that there is no point in \mathbb{F} satisfying

■ **Algorithm 1** Approximation Scheme for HYBRID k -CLUSTERING.

Input: Instance $\mathcal{I} = ((P, \mathbb{F}, \text{dist}), k, r)$ of HYBRID k -CLUSTERING, $\varepsilon \in (0, 1)$, and BALL INTERSECTION algorithm \mathcal{C} , and a guess \mathcal{G} for OPT_r

Output: A solution $X \subseteq \mathbb{F}$ such that $\text{cost}_{(1+\varepsilon)r}(P, X) \leq (1 + \varepsilon)\mathcal{G}$, assuming $\text{OPT}_r \leq \mathcal{G} \leq (1 + \varepsilon/3) \cdot \text{OPT}_r$.

- 1: For each $p \in P$, compute $u(p) = 3 \cdot \min\{\alpha > r : |\text{ball}(p, \alpha)| \geq \mathcal{G}/\alpha\}$
- 2: Process P in non-decreasing order of $u(p)$ and mark $p_i \in P$ if $\text{ball}(p_i, u(p_i))$ is disjoint from $\text{ball}(p_j, u(p_j))$ for every marked p_j such that $j < i$
- 3: Let $p^{(1)}, \dots, p^{(k')}$ be the marked points
- 4: **for each** $i \in [k']$, let $Q_i := \{(p^{(i)}, u(p^{(i)}))\}$
- 5: For $k \geq i > k'$, let $Q_i = \emptyset$
- 6: Let $X := (x_1, \dots, x_k)$, where $\forall i \in [k]$, $x_i \in \mathbb{F}$ is any center satisfying requests in Q_i .
- 7: Let $r' := r(1 + \varepsilon/3)$.
- 8: **while** $\text{cost}_{r'}(P, X) > (1 + \varepsilon) \cdot \mathcal{G}$ **do**
- 9: Toss a fair coin to guess whether we are in “nearby witness” or “faraway witness” case
- 10: **if** we guess “nearby witness” case **then**
- 11: $N = \{p \in P : \text{dist}(p, X) \leq \frac{8r}{\varepsilon}\}$
- 12: Sample a point $p \in N$, where $\Pr(p = a) = \frac{\text{dist}_{r'}(a, X_i)}{\sum_{b \in N} \text{dist}_{r'}(b, X_i)}$ for each $a \in P$
- 13: **else** we guess “faraway witness case” **then**
- 14: Let $A := \{p \in P : \text{dist}_{r'}(p, X) > \frac{\varepsilon}{1000k} \cdot u(p)\}$
- 15: Sample a point $p \in A$, where $\Pr(p = a) = \frac{\text{dist}_{r'}(a, X_i)}{\sum_{b \in P} \text{dist}_{r'}(b, X_i)}$ for each $a \in A$
- 16: **end if**
- 17: Sample an integer $i \in [k]$ u.a.r.
- 18: Add (p, δ_p) to Q_i , where $\delta_p = \frac{\text{dist}(p, X)}{1 + \varepsilon/12}$
- 19: $x_i \leftarrow \mathcal{C}(Q_i, \mathbb{F}, \varepsilon/40)$ **if** no x_i was found then **fail**
- 20: **end while**

all the requests. Note that in discrete metric spaces, \mathcal{C} can be simulated by simply scanning each $x \in \mathbb{F}$ and checking whether it satisfies all the requests in Q_i . On the other hand, for continuous Euclidean spaces, such an algorithm was designed in [1]. If the algorithm does not fail at this step, then we update our set X and continue to the next iteration. In the rest of the section, we will prove that the algorithm returns a $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation with good probability.

3.2 Analysis

Throughout the analysis, we assume that we are given a guess \mathcal{G} , such that $\text{OPT}_r \leq \mathcal{G} \leq (1 + \varepsilon/3) \cdot \text{OPT}_r$. We divide the analysis in two parts. In the first part, we bound the running time of the algorithm using the following lemma. The proof of this lemma is present in Section 3.2.1.

► **Lemma 7.** *Algorithm 1 terminates in $O(\frac{k}{\varepsilon} \log(\frac{k}{\varepsilon}) \lambda(\frac{\varepsilon}{40}))$ iterations – with or without failure.*

In the second part, we show the following lemma, which says that the probability that the algorithm terminates without failure is high. The proof the lemma is present in Section 3.2.2.

► **Lemma 8.** *With probability at least $\exp(-O(\frac{k}{\varepsilon} \log(\frac{k}{\varepsilon}) \lambda(\frac{\varepsilon}{40})))$, Algorithm 1 terminates without failure, i.e., returns a solution X satisfying $\text{cost}_{(1+\varepsilon/3)r}(P, X) \leq (1 + \varepsilon)\text{OPT}_r$.*

Using these two lemmas and repeating the algorithm $\exp(O(\frac{k}{\varepsilon} \log(\frac{k}{\varepsilon}) \lambda(\frac{\varepsilon}{40})))$ times, we have our main result.

► **Theorem 9 (Main Theorem).** *Let \mathcal{M} be a class of metric spaces closed under scaling distances by a positive constant. There is a randomized algorithm that takes as input an instance $\mathcal{I} = ((P, \mathbb{F}, \text{dist}), k, r)$ of HYBRID k -CLUSTERING such that $(P, \mathbb{F}, \text{dist}) \in \mathcal{M}$ and $\varepsilon \in (0, 1)$, and outputs a $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria solution for \mathcal{I} when, for all $\varepsilon' > 0$, the algorithmic ε' -scatter dimension of \mathcal{M} is bounded by $\lambda(\varepsilon')$, for some function λ . The running time of the algorithm is $2^{O(\frac{k}{\varepsilon} \cdot \log(k/\varepsilon) \cdot \lambda(\varepsilon/40))} \cdot |\mathcal{I}|^{O(1)}$.*

3.2.1 Bounding runtime using Algorithmic Scatter Dimension

First, we show some properties of the initial upper bounds (line 4), that we need later in the proof of Lemma 7.

► **Lemma 10 (Feasible upper bounds).** *Consider $Q_i = \{(p^{(i)}, u(p^{(i)}))\}$ initialized in Line 4 of Algorithm 1. Then, $\text{dist}(p^{(i)}, O) \leq u(p^{(i)})$.*

Proof. Suppose $\text{dist}(p^{(i)}, O) > u(p^{(i)})$. Letting $\alpha = u(p^{(i)})/3$, we have that $|\text{ball}(p, \alpha)| \geq \mathcal{G}/\alpha$. Since, $\text{dist}(p^{(i)}, O) > 3\alpha$, we have $\text{dist}(p, O) > 2\alpha$ for $p \in \text{ball}(p^{(i)}, \alpha)$. Using $\alpha > r$, this means $\text{dist}_r(p, O) > \alpha$ for $p \in \text{ball}(p, \alpha)$. Therefore,

$$\text{cost}_r(P, O) \geq \sum_{p \in \text{ball}(p^{(i)}, \alpha)} \text{dist}_r(p, O) = \sum_{p \in \text{ball}(p^{(i)}, \alpha)} (\text{dist}(p, O) - r) > \frac{\mathcal{G}}{\alpha} \cdot \alpha = \mathcal{G},$$

contradicting the the cost of O . ◀

Next, we have the following lemma, whose proof is identical to [1], that says that the initialization of X at line 6 is successful and the solution maintained by the algorithm always satisfies the upper bounds within a factor of 3.1.

► **Lemma 11 (Lemma V.5 of [1]).** *The number of marked points in line 3 is at most k , i.e., $k' \leq k$. Hence, the initialization of X at line 6 is successful. Furthermore, at any iteration, the solution X maintained by the algorithm satisfies that $\text{dist}(p, X) \leq 3.1u(p)$, for every $p \in P$.*

Bounding aspect ratio of radii

Towards proving Lemma 7, we bound the aspect ratio of the radii in the requests.

► **Lemma 12.** *Consider a request set $Q_i = \{(p_i^{(1)}, \delta_i^{(1)}), \dots, (p_i^{(\ell)}, \delta_i^{(\ell)})\}$, for $i \in [k]$. Let $X^{(j)}$, $j \in [\ell]$, be the center maintained by the algorithm just before adding the request $(p_i^{(j)}, \delta_i^{(j)})$ to Q_i . Further, let $x_i^{(j)} \in X^{(j)}$ be the center corresponding to cluster $i \in [k]$. Then, the sequence $S_i = (x_i^{(1)}, p_i^{(1)}, \delta_i^{(1)}), \dots, (x_i^{(\ell)}, p_i^{(\ell)}, \delta_i^{(\ell)})$ is an algorithmic $(\mathcal{C}, \varepsilon/20)$ -scattering. Furthermore, the radii of requests in S_i lie in the interval $[r, 8r/\varepsilon] \cup [r_{\min}, \frac{10^5 k}{\varepsilon^2} r_{\min}]$, where r_{\min} is the smallest radii in S_i that is larger than $8r/\varepsilon$.*

Proof. The proof of the first part is similar to [1].

First note that $\delta_i^{(j)} = \frac{\text{dist}(p_i^{(j)}, X^{(j)})}{1 + \varepsilon/12} \leq \frac{\text{dist}(p_i^{(j)}, x_i^{(j)})}{1 + \varepsilon/12}$. Finally, since $x_i^{(j)}$ is computed by \mathcal{C} on $\{(p_i^{(1)}, \delta_i^{(1)}), \dots, (p_i^{(j-1)}, \delta_i^{(j-1)})\}$ and error parameter $\varepsilon/40$, we have that $\text{dist}(p_i^{(j')}, x_i^{(j)}) \leq (1 + \varepsilon/40)\delta_i^{(j')}$, for $j' < j$. Hence, S_i is $(\mathcal{C}, \varepsilon/20)$ -algorithmic scattering.

35:10 Dimension-Free Parameterized Approximation Schemes for Hybrid Clustering

Now we bound that the radii of sequence $S_i, i \in [k]$. Towards this, we partition S_i into S_i^n and S_i^f , based on a partitioning of Q_i , as follows. We say a request $(p_i^{(j)}, \delta_i^{(j)}) \in Q_i$ belongs to Q_i^n if $p_i^{(j)}$ was sampled when the If condition was satisfied (in Line 12), otherwise it belongs to Q_i^f , which corresponds to the Else condition (in Line 15). Correspondingly, $(x_i^{(j)}, p_i^{(j)}, \delta_i^{(j)}) \in S_i^n$ if $(p_i^{(j)}, \delta_i^{(j)}) \in Q_i^n$ and $(x_i^{(j)}, p_i^{(j)}, \delta_i^{(j)}) \in S_i^f$ if $(p_i^{(j)}, \delta_i^{(j)}) \in Q_i^f$. We bound the aspect ratio of each part, S_i^n and S_i^f , separately. For $(x_i^{(j)}, p_i^{(j)}, \delta_i^{(j)}) \in S_i^n$, we have $r \leq \delta_i^{(j)} \leq 8r/\varepsilon$ and hence, the radii of triples in S_i^n lie in interval $[r, 8r/\varepsilon]$. Now consider $(x_i^{(j)}, p_i^{(j)}, \delta_i^{(j)}) \in S_i^f$. Recall that $X^{(j)}$ is the solution maintained by the algorithm when $(p_i^{(j)}, \delta_i^{(j)})$ is added to Q_i . Then, note that $\text{dist}_{r'}(p_i^{(j)}, X^{(j)}) > \varepsilon u(p)/1000k$ (see Line 14), and hence, we have $\text{dist}(p_i^{(j)}, X^{(j)}) > \varepsilon u(p)/1000k$. The following claim, whose proof is similar to [1], shows that the radii of requests in S_i^f are also bounded, finishing the proof of the lemma.

▷ **Claim 13.** Consider requests $(p, \delta_p), (p', \delta_{p'})$ added (in any order) to Q_i in Line 18 of Algorithm 1 such that $(p, \delta_p), (p', \delta_{p'}) \in Q_i^f$. If $\delta_{p'} < \varepsilon^2 \delta_p / 10^5 k$, then the algorithm fails in Line 19 upon making second of the two requests.

Proof. Suppose, for the sake of contradiction, the algorithm does not fail and finds a center x_i such that $\text{dist}(p, x_i) \leq (1 + \varepsilon/40)\delta_p$ and $\text{dist}(p', x_i) \leq (1 + \varepsilon/40)\delta_{p'}$. Thus, $\text{dist}(p, p') \leq \text{dist}(p, x_i) + \text{dist}(p', x_i) \leq (1 + \varepsilon/40)(\delta_p + \delta_{p'})$.

Therefore, we have

$$\text{dist}(p, X) \leq \text{dist}(p, p') + \text{dist}(p', X) \leq (1 + \varepsilon/40)(\delta_p + \delta_{p'}) + 3.1u(p'),$$

using Observation 11. Let X be the center maintained by the algorithm when the request (p, δ_p) is added to Q_i . Then, we have

$$\delta_p = \frac{\text{dist}(p, X)}{(1 + \varepsilon/12)}, \tag{1}$$

due to Line 18. Similarly, let X' be the center maintained by the algorithm when request $(p', \delta_{p'})$ is added to Q_i . Then, since $(p', \delta_{p'}) \in Q_i^f$, we have that $u(p') \leq 900\text{dist}(p', X')/\varepsilon = 1000k\delta_{p'}/\varepsilon$, using $\delta_{p'} = \frac{\text{dist}(p', X')}{(1 + \varepsilon/12)}$. Therefore, using $\delta_{p'} < \varepsilon^2 \delta_p / 10^5 k$,

$$\begin{aligned} \text{dist}(p, X) &\leq (1 + \varepsilon/40)(\delta_p + \delta_{p'}) + 3000k\delta_{p'}/\varepsilon \leq (1 + \varepsilon/40)\delta_p + 3200k\delta_{p'}/\varepsilon \\ &\leq (1 + \varepsilon/40 + \varepsilon/25)\delta_p < (1 + \varepsilon/12)\delta_p \end{aligned}$$

This means $\delta_p > \frac{\text{dist}(p, X)}{1 + \varepsilon/12}$, contradicting (1). ◁

◀

Now we are ready to finish the proof of Lemma 7.

Proof of Lemma 7. We apply Corollary 6 to S_i and note that the radii in S_i lie in $[r, 8r/\varepsilon] \cup [r_{\min}, \frac{10^5 k}{\varepsilon^2} r_{\min}]$, where r_{\min} is the smallest radii in S_i that is larger than $8r/\varepsilon$, due to Lemma 12. This implies that the length of sequence S_i is bounded by $O(\lambda(\varepsilon/40) \frac{(\log(k/\varepsilon))}{\varepsilon})$. Since in each iteration the algorithm adds one request to some Q_i , the total number of iterations is bounded by $O(\frac{k}{\varepsilon} \lambda(\varepsilon/40) (\log(k/\varepsilon)))$. ◀

3.2.2 Bounding success probability

The goal of this subsection is to prove Lemma 8, i.e., give a lower bound on the success probability of the algorithm. To this end, we first introduce the following notion.

► **Definition 14** (Consistency). *Consider a fixed hypothetical optimal solution $O = (o_1, \dots, o_k)$. We say that the current state of execution (specified by (X, Q_1, \dots, Q_k)) of Algorithm 1 is consistent with O if for any request $(p, \delta) \in Q_i, i \in [k]$, we have $\text{dist}(p, o_i) \leq \delta$.*

Note that, Lemma 11 implies that the initial set of requests (line 4) are feasible. Since the balls $\text{ball}(p^{(i)}, u(p^{(i)})), i \in [k']$ are disjoint, we can relabel the optimum centers $O = \{o_1, \dots, o_k\}$ so that $\text{dist}(p^{(i)}, o_i) \leq u(p^{(i)})$. Thus, Lemma 11 implies that the initial state of the algorithm as computed in lines 1 to 6 is consistent with a fixed optimal solution O which is fixed henceforth. To prove Lemma 8, we will inductively argue that the state of the algorithm remains consistent with O – note that the base case is already shown. Most of this subsection is devoted to show the inductive step, i.e., to show that, given a consistent state at the start of an iteration, there is a good probability that the state of the algorithm at the end of the iteration is also consistent (cf. Lemma 22).

To this end, we introduce the following definitions w.r.t. the current solution X .

► **Definition 15** (Contribution and Witness). *For any set $S \subseteq P$, we use $C_S := \sum_{p \in S} \text{dist}_{r'}(p, X)$ to denote the contribution of the set S to the cost of the current solution.*

We say that a point $p \in P$ is an $\varepsilon/3$ -witness (or simply, a witness) w.r.t. the solution X if it satisfies $\text{dist}_{(1+\varepsilon/3)r}(p, X) > (1 + \varepsilon/3) \cdot \text{dist}_r(p, O)$.

The following claim can be proved by a simple averaging argument.

▷ **Claim 16** (♠). $C_W \geq \frac{\varepsilon C_P}{10}$.

Next, we introduce several different classifications of witnesses.

► **Definition 17** (Different subsets of witnesses). *For each $x_j \in X$, let W_j denote the set of witnesses for which x_j is a nearest center in X (breaking ties arbitrarily). Then, $W_{j,\text{near}} := \{p \in W_j : \text{dist}(p, x_j) \leq 8r/\varepsilon\}$, and $W_{j,\text{far}} := W_j \setminus W_{j,\text{near}}$. Further, let $W_{\text{near}} := \bigcup_{j \in [k]} W_{j,\text{near}}$, and $W_{\text{far}} := \bigcup_{j \in [k]} W_{j,\text{far}}$. We will refer to a witness in W_{near} as a nearby witness and a witness in W_{far} as a faraway witness.*

Now, we consider two different cases regarding the behavior of the algorithm.

Case 1: $C_{W_{\text{near}}} \geq \frac{\varepsilon}{100} C_P$. In this case, when we sample a point from $N := \{p \in P : \text{dist}(p, X) \leq 8r/\varepsilon\}$ proportional to their $\text{dist}_{r'}(\cdot, X)$ values, the probability of sampling a nearby witness is at $\frac{\varepsilon}{100}$. This will correspond to the “good event”. We prove this formally in the following lemma.

► **Lemma 18** (Nearby witness lemma). *Suppose the current solution X at the start of an iteration satisfies $\text{cost}_{r'}(P, X) > (1 + \varepsilon) \cdot \mathcal{G}$. Further, suppose $C_{W_{\text{near}}} \geq \frac{\varepsilon}{100} C_P$. Then, with probability at least $\frac{\varepsilon}{200k}$, the point $p \in P$, the index $i \in [k]$, and value δ_p defined in the iteration satisfy the following properties.*

1. $o_i \in O$ is the closest center to $p \in P$, i.e., $\text{dist}(p, o_i) = \text{dist}(p, O)$,
2. $p \in W_{\text{near}}$, i.e., (i) $\text{dist}_{r'}(p, X) > (1 + \varepsilon/3)\text{dist}_r(p, o_i)$, and (ii) $\text{dist}(p, X) \leq \frac{8r}{\varepsilon}$.
3. $\text{dist}(p, o_i) < \frac{\text{dist}(p, X)}{1 + \varepsilon/12} =: \delta_p \leq \frac{8r}{\varepsilon}$.

Proof. First, in line 10 with probability $1/2$, we correctly move to the “nearby witness” (if) case of line 10. We condition on this event. Next, note that $W_{\text{near}} \subseteq N$, and $C_{W_{\text{near}}} \geq \frac{\varepsilon}{100}C_P \geq \frac{\varepsilon}{100}C_N$, where the first inequality is due to the case assumption. Therefore, in line 12, the point p sampled from N , will belong to W_{near} with probability at least $\frac{\varepsilon}{100}$. Let $i \in [k]$ denote the index such that $o_i \in O$ is the closest center in the optimal solution O , and we correctly guess the index i in line 17. Note that the probability of the algorithm making the “correct” random choices is at least $\frac{1}{2} \cdot \frac{\varepsilon}{100} \cdot \frac{1}{k} = \frac{\varepsilon}{200k}$.

We condition on the said events. Since p is a witness, we know that $\text{dist}_{r'}(p, X) > (1 + \varepsilon/3) \cdot \text{dist}_r(p, o_i)$. We first observe that $\text{dist}_{r'}(p, X)$ must be positive, which implies that $\text{dist}(p, X) > r' = (1 + \varepsilon/3)r$. Now, we consider two cases based on the value of $\text{dist}(p, O)$.

If $\text{dist}(p, o_i) < r$, then $\text{dist}_r(p, o_i) = 0$, in which case,

$$\text{dist}(p, o_i) < r \leq \frac{r'}{1 + \varepsilon/12} \leq \frac{\text{dist}(p, X)}{1 + \varepsilon/12} \quad (2)$$

Otherwise, $\text{dist}(p, o_i) \geq r$, in which case $\text{dist}_r(p, o_i) = \text{dist}(p, o_i) - r$. Then, consider

$$\begin{aligned} \text{dist}(p, X) - (1 + \frac{\varepsilon}{3})r &= \text{dist}_{r'}(p, X) > (1 + \frac{\varepsilon}{3}) \cdot \text{dist}_r(p, o_i) = (1 + \frac{\varepsilon}{3})(\text{dist}(p, o_i) - r) \\ \implies \text{dist}(p, X) - (1 + \frac{\varepsilon}{3})r &> (1 + \frac{\varepsilon}{3}) \cdot \text{dist}(p, o_i) - (1 + \frac{\varepsilon}{3})r \\ \implies \text{dist}(p, o_i) < \frac{\text{dist}(p, X)}{1 + \varepsilon/3} &\leq \frac{\text{dist}(p, X)}{1 + \varepsilon/12} \end{aligned} \quad (3)$$

Thus, regardless of the value of $\text{dist}(p, o_i)$, we have established the third item, hence completing the proof of the lemma. \blacktriangleleft

Case 2: $C_{W_{\text{near}}} < \frac{\varepsilon}{100}C_P$. In this case, most of contribution of witnesses is coming from “faraway witnesses”. In this case, the “correct choice” corresponds to the case when we sample points from the set $A = \{p \in P : \text{dist}_{r'}(p, X) > \frac{\varepsilon}{1000k} \cdot u(p)\}$ as defined in line 14. In this case, we will show that with good probability, the sampled point is a “faraway witness”. Specifically, we show the following lemma.

► **Lemma 19 (Faraway witness lemma).** *Suppose the current solution X at the start of an iteration satisfies $\text{cost}_{r'}(P, X) > (1 + \varepsilon) \cdot \mathcal{G}$. Further, suppose $C_{W_{\text{near}}} < \frac{\varepsilon}{100}C_P$. Then, with probability at least $\frac{\varepsilon}{16k}$, the point $p \in P$, the index $i \in [k]$, and value δ_p defined in the iteration satisfy the following properties.*

1. $o_i \in O$ is the closest center to p , i.e., $\text{dist}(p, o_i) = \text{dist}(p, O)$,
2. $p \in W_{\text{far}}$, i.e., (i) $\text{dist}_{r'}(p, X) > (1 + \varepsilon/3)\text{dist}_r(p, o_i)$, and (ii) $\text{dist}(p, X) > \frac{8r}{\varepsilon}$,
3. $\text{dist}_{r'}(p, X) > \frac{\varepsilon}{1000k}u(p)$, and
4. $\text{dist}(p, o_i) < \frac{\text{dist}(p, X)}{1 + \varepsilon/12} =: \delta_p \leq \frac{8r}{\varepsilon}$.

Proof. First, in line 10 with probability $1/2$, we correctly move to the “faraway witness” (else) case of line 13. We condition on this event. Now, by combining Claim 16 and the case assumption we obtain,

$$C_{W_{\text{far}}} = C_W - C_{W_{\text{near}}} \geq C_P \cdot \left(\frac{\varepsilon}{10} - \frac{\varepsilon}{100}\right) = \frac{\varepsilon}{9}C_P. \quad (4)$$

Next, let $H := \{j \in [k] : C_{W_{j, \text{far}}} \geq \frac{\varepsilon C_P}{100k}\}$. It follows that,

$$\begin{aligned} \sum_{j \in [k] \setminus H} C_{W_{j, \text{far}}} &\leq k \cdot \frac{\varepsilon C_P}{100k} \\ \implies \sum_{j \in H} C_{W_{j, \text{far}}} &\geq C_{W_{\text{far}}} - \sum_{j \in [k] \setminus H} C_{W_{j, \text{far}}} \geq \frac{\varepsilon C_P}{9} - \frac{\varepsilon C_P}{100} \geq \frac{\varepsilon C_P}{8} \end{aligned} \quad (5)$$

Fix a $j \in H$, and let us index the points in $W_{j,\text{far}} = \{z_1, z_2, \dots, z_\ell\}$ in the non-decreasing order of their distances $\text{dist}(z, x_j)$ (and equivalently, $\text{dist}_r(z, x_j)$). First, we state the following simple consequences of various definitions for future reference.

► **Observation 20** (♠). *For each $z \in W_{j,\text{far}}$, the following bounds hold.*

1. $\text{dist}(z, x_j) \geq \frac{8r}{\varepsilon}$,
2. $\text{dist}(z, O) > (1 + \frac{\varepsilon}{3}) \cdot \text{dist}(z, x_j)$
3. $\text{dist}_{r'}(z, x_j) \geq \frac{6r}{\varepsilon}$, and
4. $\text{dist}_r(z, x_j) \leq \text{dist}(z, x_j) \leq (1 + \frac{\varepsilon}{5}) \cdot \text{dist}_{r'}(z, x_j) \leq (1 + \frac{\varepsilon}{5}) \cdot \text{dist}_r(z, x_j)$.

Let us return to the points in $W_{j,\text{far}} = \{z_1, z_2, \dots, z_\ell\}$. Let $q \in [\ell]$ denote the minimum index such that, the contribution of the set $W_{j,\text{far}}^- = \{z_1, z_2, \dots, z_q\}$, is at least $\frac{C_{W_{j,\text{far}}}}{2}$. Note that by a simple argument, this implies that the contribution of the set $W_{j,\text{far}}^+ = W_{j,\text{far}} \setminus W_{j,\text{far}}^-$ is also at least $\frac{C_{W_{j,\text{far}}}}{3}$ (see, e.g., Lemma 7.20 in [21]). Hence, we have the following lower bounds on the contribution of both the sets, by recalling that $j \in H$.

$$C_{W_{j,\text{far}}^-} \geq \frac{\varepsilon C_P}{200k}, \quad C_{W_{j,\text{far}}^+} \geq \frac{\varepsilon C_P}{300k} \quad (6)$$

Next, we first prove the following technical claim.

▷ **Claim 21.** Let $j \in H$. For all $p \in W_{j,\text{far}}^+$, it holds that $u(p) \leq \frac{900k \text{dist}(p, x_j)}{\varepsilon}$. Therefore, $W_{j,\text{far}}^+ \subseteq A$.

Proof. For this proof, we use the following shorthand: $W^+ := W_{j,\text{far}}^+$, and $W^- := W_{j,\text{far}}^-$. Now, fix an arbitrary point $p \in W^+$. For any $q \in W^-$, $\text{dist}(p, x_j) \geq \text{dist}(q, x_j)$, which implies that

$$\text{dist}(p, q) \leq \text{dist}(p, x_j) + \text{dist}(q, x_j) \leq 2 \cdot \text{dist}(p, x_j) \leq 2(1 + \frac{\varepsilon}{5}) \cdot \text{dist}_{r'}(p, x_j) \leq 4 \cdot \text{dist}_{r'}(p, x_j) \quad (7)$$

Here, we use the property 4 from Observation 20 in the third inequality in the above. On the other hand, note that,

$$\frac{\varepsilon \text{OPT}_r}{200k} \leq \frac{\varepsilon C_P}{200k} \leq C_{W^-} = \sum_{q \in W^-} \text{dist}_{r'}(q, x_j) \leq \text{dist}_{r'}(p, x_j) \cdot |W^-| \quad (8)$$

Then, if we set $\alpha = 12 \text{dist}_{r'}(p, x_j)$, from (7), we obtain that $W^- \subseteq \text{ball}(p, \alpha/3)$. Now, combining this with (8), we obtain that,

$$|\text{ball}(p, \alpha/3)| \geq |W^-| \geq \frac{\varepsilon \text{OPT}_r}{300k \cdot \text{dist}_{r'}(p, x_j)} \geq \frac{\varepsilon \text{OPT}_r}{25k\alpha} = \frac{\varepsilon \text{OPT}_r}{75k(\alpha/3)} \quad (9)$$

Hence, we have that $|\text{ball}(p, \frac{75k\alpha}{3\varepsilon})| \geq \frac{\varepsilon \text{OPT}_r}{75k(\alpha/3)}$. Using $\text{dist}(p, x_j) \geq \frac{8r}{\varepsilon}$ from Observation 20 and the fact $\text{dist}_r(p, x_j) \geq 5 \text{dist}(p, x_j)/6$, we have $\frac{75k\alpha}{3\varepsilon} = \frac{300k \text{dist}_{r'}(p, x_j)}{\varepsilon} > r$. Therefore, $u(p) \leq \frac{900k \text{dist}_{r'}(p, x_j)}{\varepsilon} \leq \frac{900k \text{dist}(p, x_j)}{\varepsilon}$, as desired. ◁

Recall from (5) that $C_{W_{\text{far}} \cap A} \geq \sum_{j \in H} C_{W_{j,\text{far}}^+} \geq \frac{\varepsilon C_P}{8} \geq \frac{\varepsilon C_A}{8}$, therefore, when we sample a point from A , the probability that the sampled point p belongs to $\bigcup_{j \in H} W_{j,\text{far}}^+$ is at least $\frac{\varepsilon}{8}$. Now, we condition on this event. Let $o_i \in O$ be the nearest center to p , and in line 17, the probability that we sample the correct index is $\frac{1}{k}$. Thus, the total probability of the algorithm making the ‘‘correct choices’’ in this case is at least $\frac{1}{2} \cdot \frac{\varepsilon}{8} \cdot \frac{1}{k} = \frac{\varepsilon}{16k}$. We condition on these choices. Note that, item 1 is thus satisfied due to the correct sampling of i , item 2 is satisfied due $p \in \bigcup_{j \in H} W_{j,\text{far}}^+ \subseteq W_{\text{far}} \subseteq W$, and item 3 is satisfied since $p \in A$ by construction. Thus, we are only left with showing item 4, to which end, we consider different cases for the distance between p and its closest optimal center, o_i .

If $\text{dist}(p, o_i) < \frac{6r}{\varepsilon}$, then since $\text{dist}(p, X) \geq \frac{8r}{\varepsilon}$, it easily follows that,

$$\text{dist}(p, o_i) < \frac{3}{4} \cdot \text{dist}(p, X) \leq \frac{\text{dist}(p, X)}{1 + \varepsilon/12} \quad (10)$$

Otherwise, if $\text{dist}(p, o_i) \geq \frac{6r}{\varepsilon}$, then similar item 4 of Observation 20, we can show that,

$$\text{dist}_r(p, o_i) < \text{dist}(p, o_i) \leq (1 + \frac{\varepsilon}{3}) \cdot \text{dist}_r(p, o_i) \quad (11)$$

However, since p is an $\varepsilon/3$ -witness, it follows that, $\text{dist}_{r'}(p, X) > (1 + \frac{\varepsilon}{3}) \cdot \text{dist}_r(p, O)$. Therefore, we obtain that,

$$\text{dist}(p, x_j) \geq \text{dist}_{r'}(p, x_j) > (1 + \frac{\varepsilon}{3}) \cdot \text{dist}_r(p, o_i) \geq \frac{1 + \varepsilon/3}{1 + \varepsilon/5} \cdot \text{dist}(p, o_i) \geq (1 + \frac{\varepsilon}{12}) \cdot \text{dist}(p, o_i) \quad (12)$$

Thus, in each of the sub-cases, in (10) and (12), we have shown that $\text{dist}(p, o_i) < \frac{\text{dist}(p, X)}{1 + \varepsilon/12}$, thus completing the proof of the lemma. \blacktriangleleft

Lemma 18 and Lemma 19 can be combined in a straightforward way to obtain the following lemma completing the inductive step.

► Lemma 22. *Consider an iteration of the algorithm such that the state of execution (X, Q_1, \dots, Q_k) at the start of the iteration is consistent with a fixed optimal solution O , and further $\text{cost}_{r'}(P, X) > (1 + \varepsilon) \cdot \mathcal{G}$. Then, with probability at least $\frac{\varepsilon}{200k}$, the state of the algorithm at the end of the iteration $(X', Q'_1, Q'_2, \dots, Q'_k)$ is also consistent with O .*

Proof. Consider the scenario described in the premise of the lemma. Then, the solution X at the start of the iteration either satisfies that $C_{W_{\text{near}}} \geq \frac{\varepsilon}{100} C_P$ (nearby witness case), or $C_{W_{\text{near}}} \geq \frac{\varepsilon}{100} C_P$ (faraway witness case). Then, Lemmas 18 and 19; along with the correctness of \mathcal{C} , together imply that, conditioned on the respective case assumption, the probability that the state of the algorithm is consistent with O is at least $\frac{\varepsilon}{200k}$.⁹ \blacktriangleleft

Armed with Lemma 22, it is easy to conclude the proof of Lemma 8.

Proof of Lemma 8. As argued initially, the state of the algorithm at line 6 is consistent with the optimal solution O . Then, Lemma 22 implies that, for any $\ell \geq 1$, if the algorithm runs for ℓ iterations, then with probability at least $(\frac{\varepsilon}{200k})^\ell$, the state of the algorithm remains consistent. Further, note that as long as the state of the algorithm remains consistent with O , then the algorithm cannot fail. Finally, Lemma 7 implies that the algorithm terminates within $O(\frac{k}{\varepsilon} \lambda(\frac{\varepsilon}{40}) \log(\frac{k}{40}))$ iterations. Therefore, the probability that the algorithm terminates without failure – i.e., by successfully breaking the **while** loop by finding a solution X satisfying $\text{cost}_{r'}(P, X) \leq (1 + \varepsilon) \text{OPT}_r$, is at least $(\frac{\varepsilon}{k})^{O(\frac{k}{\varepsilon} \lambda(\frac{\varepsilon}{40}) \log(\frac{k}{40}))} = \exp(-O(\frac{k}{\varepsilon} \log(\frac{k}{\varepsilon}) \lambda(\frac{\varepsilon}{40})))$. \blacktriangleleft

4 Extensions of the Bicriteria FPT-AS

Fomin et al. [23] defined a generalization of HYBRID k -CLUSTERING, which they call HYBRID (k, z) -CLUSTERING, wherein the objective is $\sum_{p \in P} (\text{dist}_r(p, X))^z$, for fixed $z \geq 1$, which simultaneously generalizes (k, z) -CLUSTERING and k -CENTER. They generalized their algorithm for $z = 2$, i.e., HYBRID k -MEANS, but left the possibility of extending to an arbitrary value of z conditional on the existence of a certain kind of sampling algorithm. In this paper, we consider a much more general HYBRID NORM k -CLUSTERING problem that captures all

⁹ Note that this probability also accounts for the result of the coin toss in line 10.

(k, z) -CLUSTERING problems, and much more. Here, the objective is to find a set $X \subseteq \mathbb{F}$ that minimizes $f(\mathbf{d}_r(P, X))$, where $\mathbf{d}_r(P, X) = (\text{dist}_r(p_1, X), \text{dist}_r(p_2, X), \dots, \text{dist}_r(p_n, X))$ is the vector of r -distances to the solution X , and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a *monotone norm*¹⁰ – we refer the reader to [1] for a detailed background on norms and related notions.

Algorithm 1 and its analysis can be extended to the general norm objective following the approach of [1]. Here, we only sketch the modifications required to the algorithm and its proof, since this is not the main focus of this work.

The initial upper bounds $u(p)$ can be found in a similar manner. The while loop runs as long as the solution X in hand satisfies that $f(\mathbf{d}_{r'}(P, X)) > (1 + \varepsilon/3) \cdot \text{OPT}_r$, where $\mathbf{d}_{r'}$ denotes the vector of distances $(\text{dist}_{r'}(p, X))_{p \in X}$, and $r' = (1 + \varepsilon/3)r$. If this is the case, then the first step is to compute¹¹ a *subgradient* of f at $\mathbf{d}_{r'}$, which is, loosely speaking, a weight function $w : P \rightarrow \mathbb{R}_{\geq 0}$, such that it suffices to focus the attention in the current iteration on $\text{wcost}_{r'}(P, X) := \sum_{p \in P} w(p) \text{dist}_r(p, X)$ that satisfies $\text{wcost}_{r'}(P, X) = f(\mathbf{d}_{r'}(P, X)) > (1 + \varepsilon/3) \cdot \text{OPT}_r$. Our algorithm proceeds in a similar fashion, except that whenever points are sampled from the set N (in line 12) or set A (in line 15), the probability of sampling a point p is proportional to $w(p) \cdot \text{dist}_r(p, X)$. The rest of the algorithm remains unchanged.

There are a few minor modifications required in the analysis – mainly in Section 3.2.2. First, while the definition of an $(\varepsilon/3)$ -witness (or simply a *witness*) remains unchanged, we redefine the *contribution* of a subset $S \subseteq P$ to be $C_S := \sum_{p \in S} w(p) \cdot \text{dist}_r(p, X)$. The nearby and faraway witness cases are then considered exactly as in Section 3.2.2. The proof of the analogue of Lemma 18 goes through without any modifications; whereas we need to be slightly more careful in the proof of the analogue of Lemma 19. Here, in inequalities (8 and 9) we need to take the weighted distances into account and relate C_P to $f(\mathbf{d}_{r'}(P, X))$, which requires a slight worsening of constants. With these minor changes in place, we can conclude that the state of the algorithm after ℓ iterations is consistent with a fixed optimal solution O with probability at least $\Omega((\frac{\varepsilon}{k}))^\ell$. The analysis for bounding the number of iterations remains unchanged, and hence we obtain an FPT-AS with a similar running time, modulo the constants hidden in the big-Oh notation. We omit the details.

5 Coreset

In this section, we design a coreset for HYBRID k -CLUSTERING in doubling metric of bounded dimension. More specifically, we prove the following (restated for convenience).

► **Theorem 3** (Coreset for HYBRID k -CLUSTERING). *There exists an algorithm that takes as input an instance $\mathcal{I} = ((P, \mathbb{F}, \text{dist}), k, r)$ of HYBRID k -CLUSTERING in doubling metric of dimension d and a parameter $\varepsilon \in (0, 1)$, and in time $2^{O(d \log(1/\varepsilon))} |\mathcal{I}|^{O(1)}$ returns a pair (P', w) , where $P' \subseteq P$ has size $2^{O(d \log(1/\varepsilon))} k \log |P|$, and $w : P' \rightarrow \mathbb{N}$, such that the following property is satisfied for any $X \subseteq \mathbb{F}$ of size at most k :*

$$|\text{wcost}_r(P', X) - \text{cost}_r(P, X)| \leq \varepsilon \text{cost}_r(P, X)$$

Here, $\text{wcost}_r(P, X) := \sum_{p \in P'} w(p) \cdot \text{dist}_r(p, X)$.

Proof. A formal description of our algorithm can be found in the full version of the paper. Here, we provide an overview of the overview, which is based on grid construction approach of [26, 2]. Let $T \subseteq P \cup \mathbb{F}$ be an approximate solution that satisfies the following properties:

¹⁰ $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a *norm* if it satisfies following properties for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and any $\lambda \in \mathbf{R}$, (i) $f(\mathbf{x}) = 0$ iff $\mathbf{x} = \mathbf{0}$, (ii) $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$, (iii) $f(\lambda \mathbf{x}) = |\lambda| f(\mathbf{x})$; furthermore, f is *monotone* if $f(\mathbf{x}) \leq f(\mathbf{y})$ whenever $\mathbf{x} \leq \mathbf{y}$, where \leq denotes point-wise inequality.

¹¹ Actually, it is sufficient to compute an approximate subgradient, as done in [1].

(i) $|T| \leq 2^{O(d)}k$, and (ii) $\text{cost}_r(P, T) \leq 36 \cdot \text{OPT}_r$ ¹². For each $t_i \in T$, consider the balls $B_i^j = \text{ball}(t_i, 2^j R)$, for $j \in \{0, 1, \dots, 2 \log \lceil \alpha n \rceil\}$. Note that, since $\text{dist}(p, T) \leq \text{cost}(P, T) = \alpha n \cdot R$, we have that p lies in some ball B_i^j , for $i \in [\gamma k]$ and $j \in \{0, 1, \dots, 2 \log \lceil \alpha n \rceil\}$.

The idea is to decompose each B_i^j into smaller balls each of radius $\frac{\varepsilon 2^j R}{4\alpha}$, and associate each point $p \in P$ to a unique smallest ball containing p , breaking ties arbitrary. We say a small ball b is non-empty if there is a point in P associated with b . Next, for each non-empty ball b , pick an arbitrary point p associated with b and add p to P' with weight $w(p)$ equal to the total number of points associated with b ; we call such a point p as the representative of points associated with b .

To bound the size of P' , note that, in doubling metric of dimension d , a unit ball can be covered by $2^{O(d \log(1/\varepsilon))}$ balls, each of radius ε . Hence, we have $|P'| = O(2^{O(d \log(1/\varepsilon))} \gamma k \log(\alpha n))$.

Recall that, for $X \subseteq \mathbb{F}$, $|X| \leq k$, we define the weighted cost of X with respect to $(P', \{w(p')\}_{p' \in P'})$ as $\text{wcost}_r(P', X) = \sum_{p' \in P'} w(p') d_r(p', X)$. Now we show that the cost of X with respect to $(P', \{w(p')\}_{p' \in P'})$ approximately preserves the cost of X with respect to P . Consider a point $p \in P$ and let $p' \in P'$ be its representative in P' . Now, the contribution of p towards $\text{cost}_r(P, X)$ is $\text{dist}_r(p, X)$. On the other hand, its contribution towards $\text{wcost}_r(P', X)$ is $\text{dist}_r(p', X)$. Hence,

$$|\text{wcost}_r(P', X) - \text{cost}_r(P, X)| \leq \sum_{p \in P} |\text{dist}_r(p, X) - \text{dist}_r(p', X)|.$$

Now if $p \in \bigcup_{i \in [\gamma k]} B_i^0$, then $\text{dist}(p, X) - \frac{\varepsilon R}{4\alpha} \leq \text{dist}(p', X) \leq \text{dist}(p, X) + \frac{\varepsilon R}{4\alpha}$. Hence,

$$\max\{\text{dist}(p, X) - \varepsilon R/4\alpha - r, 0\} \leq \text{dist}_r(p', X) \leq \max\{\text{dist}(p, X) + \varepsilon R/4\alpha - r, 0\}$$

Therefore, $\text{dist}_r(p, X) - \varepsilon R/4\alpha \leq \text{dist}_r(p', X) \leq \text{dist}_r(p, X) + \varepsilon R/4\alpha$. Hence, we have

$$\sum_{p \in P_0} |\text{dist}_r(p, X) - \text{dist}_r(p', X)| \leq \frac{\varepsilon R n}{4\alpha} \leq \frac{\varepsilon \text{OPT}_r}{2} \leq \frac{\varepsilon \text{cost}_r(P, X)}{2}.$$

Now, suppose $p \in P \setminus \bigcup_{i \in [\gamma k]} B_i^0$, then $\text{dist}(p, T) > R$. Let $j \geq 1$ be such that $2^{j-1} R \leq \text{dist}(p, T) \leq 2^j R$. Hence, we have that $2^j R \leq 2 \text{dist}(p, T)$. Therefore, using $\text{dist}(p, X) - \frac{\varepsilon d(p, T)}{2\alpha} \leq \text{dist}(p', X) \leq \text{dist}(p, X) + \frac{\varepsilon d(p, T)}{2\alpha}$, we have

$$\max\{\text{dist}(p, X) - \varepsilon d(p, T)/2\alpha - r, 0\} \leq \text{dist}_r(p', X) \leq \max\{\text{dist}(p, X) + \varepsilon d(p, T)/2\alpha - r, 0\}$$

This implies, $\text{dist}_r(p, X) - \varepsilon d(p, T)/2\alpha \leq \text{dist}_r(p', X) \leq \text{dist}_r(p, X) + \varepsilon d(p, T)/2\alpha$. Thus, we have

$$\sum_{p \in P \setminus P_0} |\text{dist}_r(p, X) - \text{dist}_r(p', X)| \leq \sum_{p \in P \setminus P_0} \frac{\varepsilon}{2\alpha} \text{dist}(p, T) \leq \frac{\text{OPT}_r}{2} \leq \frac{\varepsilon \text{cost}_r(P, X)}{2}.$$

Finally, we have $|\text{wcost}_r(P', X) - \text{cost}_r(P, X)| \leq \varepsilon \text{cost}_r(P, X)$, as desired.

To finish the proof, we invoke the coreset construction algorithm using the approximate set T obtained from the following lemma. At a high level, to obtain this lemma, we start from an (18, 6)-bicriteria approximation for HYBRID k -CLUSTERING from [23, 13], and use the fact that, a ball of radius $O(r)$ can be decomposed into $2^{O(d)}$ balls of radius r , converting the guarantee from $\text{dist}_{6r}(\cdot, \cdot)$ to $\text{dist}_r(\cdot, \cdot)$.

¹²Note the set T is a bicriteria solution, but in a different sense, since it approximates the size and the cost while using r -distances.

► **Lemma 23** (Approximate solution T for coresets construction). *There is a polynomial-time algorithm that, given an instance $\mathcal{I} = (P, \mathbb{F}, \text{dist}, r)$ of HYBRID k -CLUSTERING in doubling metric of dimension d , computes $T \subseteq P \cup \mathbb{F}$, $|T| \leq 2^{O(d)}k$ such that $\text{cost}_r(P, T) \leq 36 \cdot \text{OPT}_r$, where OPT_r is the optimal cost for \mathcal{I} .* ◀

Proof. Let A be $(18, 6)$ -bicriteria solution for \mathcal{I} obtained from [14], as mentioned in [23]. That is $\text{cost}_{6r}(P, A) \leq 18\text{OPT}_r$. Consider $B_a := \text{ball}(a, 12r)$ for $a \in A$, and decompose B_a into $2^{O(d)}$ smaller balls, each of radius $r/2$ – note that this follows from the fact that the metric space has doubling dimension d . For each smaller ball b such that $b \cap (P \cup \mathbb{F}) \neq \emptyset$, add an arbitrary $t \in b \cap (P \cup \mathbb{F})$ to T . Finally, add A to T . Hence, $|T| = |A| + 2^{O(d)}|A| = k2^{O(d)}$.

It is easy to see that, for every $p \in \bigcup_{c \in A} \text{ball}(c, 12r)$, there is some $q \in T$, such that $\text{dist}(p, q) \leq r$. Now, consider some $p' \notin \bigcup_{c \in A} \text{ball}(c, 12r)$. Note that $\text{dist}_{6r}(p, A) \geq 6r$, which implies that $\text{dist}(p, A) \leq 2 \cdot \text{dist}_{6r}(p, A)$. Therefore, $\text{dist}_r(p, T) \leq \text{dist}_r(p, A) \leq \text{dist}(p, A) \leq 2\text{dist}_{6r}(p, A)$. Therefore, it follows that for all $p \in P$, $\text{dist}_r(p, T) \leq 2 \cdot \text{dist}_{6r}(p, A)$. Which implies that,

$$\text{cost}_r(p, T) \leq 2 \cdot \text{cost}_r(p, A) \leq 36 \cdot \text{OPT}_r.$$

This concludes the proof of the lemma. ◀

6 Conclusion

In this paper, we revisit HYBRID k -CLUSTERING, which was introduced and studied recently by Fomin et al. [23]. We resolve two open questions explicitly asked in their work, namely, for continuous Euclidean instances from \mathbb{R}^d , (1) we obtain an FPT-AS for HYBRID k -CLUSTERING that does not have an FPT dependence on the dimension d (and in fact, the dependence on k is $2^{O(k \log k)}$ instead $2^{\text{poly}(k)}$ as in [23]), and (2) we design coresets for the same. Indeed, our technique also generalizes to the (k, z) -clustering variant of the problem, which was implicitly considered in [23], but was not completely resolved. To obtain our algorithmic result, we build upon insights from the recent framework of Abbasi et al. [1] for clustering in metric spaces of bounded *algorithmic scatter dimension* that encapsulates a broad class of metric spaces. Thus, our result shows that the potential of the framework introduced in [1] extends to clustering problems beyond that captured by the monotone norm setting, thus paving the way for obtaining FPT-ASes for other clustering problems using the similar technique. However, several interesting questions remain.

Firstly, the framework of [1] is inherently randomized due to key sampling steps, and its derandomization remains an intriguing open question. In particular, derandomizing the step that samples a witness in our algorithm is an interesting challenge. In another direction, now that the approximation landscape of the vanilla version of HYBRID k -CLUSTERING is beginning to be well-understood, it is natural to explore *constrained* variants of the problem such as those involving fairness or capacity constraints.

References

- 1 Fateme Abbasi, Sandip Banerjee, Jaroslav Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1377–1399. IEEE, 2023. doi:10.1109/FOCS57990.2023.00085.

- 2 Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation for robust clustering in discrete geometric spaces. 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 6:1–6:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.6.
- 3 Akanksha Agrawal, Tanmay Inamdar, Saket Saurabh, and Jie Xue. Clustering what matters: Optimal approximation for clustering with outliers. *J. Artif. Intell. Res.*, 78:143–166, 2023. doi:10.1613/JAIR.1.14883.
- 4 Barbara Anthony, Vineet Goyal, Anupam Gupta, and Viswanath Nagarajan. A plant location guide for the unsure: Approximation algorithms for min-max location problems. *Mathematics of Operations Research*, 35(1):pages 79–101, 2010.
- 5 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k -medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98*, pages 106–113, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276718.
- 6 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k -median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.
- 7 Mihai Badoiu, Sarel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, (STOC)*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.
- 8 Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On coresets for fair clustering in metric and euclidean spaces and their applications. *J. Comput. Syst. Sci.*, 142:103506, 2024. doi:10.1016/J.JCSS.2024.103506.
- 9 Sayan Bhattacharya, Parinya Chalermsook, Kurt Mehlhorn, and Adrian Neumann. New approximability results for the robust k -median problem. In *Scandinavian Workshop on Algorithm Theory (SWAT'14)*, pages 50–61. Springer, 2014. doi:10.1007/978-3-319-08404-6_5.
- 10 Romain Bourneuf and Marcin Pilipczuk. Bounding ε -scatter dimension via metric sparsity. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (to appear)*, 2025.
- 11 Vladimir Braverman, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for ordered weighted clustering. In *International Conference on Machine Learning (ICML'19)*, pages 744–753. PMLR, 2019. URL: <http://proceedings.mlr.press/v97/braverman19a.html>.
- 12 Jaroslaw Byrka, Krzysztof Sornat, and Joachim Spoerhase. Constant-factor approximation for ordered k -median. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 620–631. ACM, 2018. doi:10.1145/3188745.3188930.
- 13 Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k -center problem. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 67:1–67:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.67.
- 14 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between k -median and k -center: Approximation algorithms for ordered k -median. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPICs*, pages 29:1–29:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.29.
- 15 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 126–137. ACM, 2019. doi:10.1145/3313276.3316322.

- 16 Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, and Chris Schwiegelshohn. Breaching the 2 LMP approximation barrier for facility location with applications to k -median. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 940–986. SIAM, 2023. doi:10.1137/1.9781611977554.CH37.
- 17 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k -means and k -median in euclidean and minor-free metrics. *SIAM J. Comput.*, 48(2):644–667, 2019. doi:10.1137/17M112717X.
- 18 Vincent Cohen-Addad and Jason Li. On the fixed-parameter tractability of capacitated clustering. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPICs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.41.
- 19 Vincent Cohen-Addad, Michał Pilipczuk, and Marcin Pilipczuk. A polynomial-time approximation scheme for facility location on planar graphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 560–581. IEEE, 2019.
- 20 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coresets framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 169–182. ACM, 2021. doi:10.1145/3406325.3451022.
- 21 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 22 Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 434–444. ACM, 1988. doi:10.1145/62212.62255.
- 23 Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Saket Saurabh, and Meirav Zehavi. Hybrid k -clustering: Blending k -median and k -center. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPICs*, pages 4:1–4:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.APPROX/RANDOM.2024.4.
- 24 Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k -means in doubling metrics. *SIAM J. Comput.*, 48(2):452–480, 2019. doi:10.1137/17M1127181.
- 25 Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. Socially fair k -means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 438–448, 2021. doi:10.1145/3442188.3445906.
- 26 Sarel Har-Peled and Soham Mazumdar. On coresets for k -means and k -median clustering. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300. ACM, 2004. doi:10.1145/1007352.1007400.
- 27 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.
- 28 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k -center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. URL: <http://www.jstor.org/stable/3689371>, doi:10.1287/MOOR.10.2.180.
- 29 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 731–740. ACM, 2002. doi:10.1145/509907.510012.

- 30 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.
- 31 Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k -median and k -means with outliers via iterative rounding. 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 646–659. ACM, 2018. doi:10.1145/3188745.3188882.
- 32 Alfred A Kuehn and Michael J Hamburger. A heuristic program for locating warehouses. *Management science*, 9(4):643–666, 1963.
- 33 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In *32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1374–1385. Springer, 2005. doi:10.1007/11523468_111.
- 34 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010. doi:10.1145/1667053.1667054.
- 35 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013. doi:10.1016/J.IC.2012.01.007.
- 36 S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. doi:10.1109/TIT.1982.1056489.
- 37 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.
- 38 Yury Makarychev and Ali Vakilian. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory (COLT'21)*, pages 3246–3264. PMLR, 2021. URL: <http://proceedings.mlr.press/v134/makarychev21a.html>.
- 39 Dániel Marx. Parameterized complexity of independence and domination on geometric graphs. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 154–165. Springer, 2006. doi:10.1007/11847250_14.
- 40 Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984. doi:10.1137/0213014.
- 41 John F. Stollsteimer. A working model for plant numbers and locations. *Journal of Farm Economics*, 45(3):631–645, 1963. URL: <http://www.jstor.org/stable/1235442>.
- 42 Arie Tamir. The k -centrum multi-facility location problem. *Discret. Appl. Math.*, 109(3):293–307, 2001. doi:10.1016/S0166-218X(00)00253-5.
- 43 Murad Tukan, Xuan Wu, Samson Zhou, Vladimir Braverman, and Dan Feldman. New coresets for projective clustering and applications. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pages 5391–5415. PMLR, 2022. URL: <https://proceedings.mlr.press/v151/tukan22a.html>.

MaxMin Separation Problems: FPT Algorithms for st -Separator and Odd Cycle Transversal

Ajinkya Gaikwad ✉ 

Indian Institute of Science Education and Research, Pune, India

Hitendra Kumar ✉

Indian Institute of Science Education and Research, Pune, India

Soumen Maity ✉

Indian Institute of Science Education and Research, Pune, India

Saket Saurabh ✉ 

The Institute of Mathematical Sciences, Chennai, India

University of Bergen, Norway

Roohani Sharma ✉ 

University of Bergen, Norway

Abstract

In this paper, we study the parameterized complexity of the MAXMIN versions of two fundamental separation problems: MAXIMUM MINIMAL st -SEPARATOR and MAXIMUM MINIMAL ODD CYCLE TRANSVERSAL (OCT), both parameterized by the solution size. In the MAXIMUM MINIMAL st -SEPARATOR problem, given a graph G , two distinct vertices s and t and a positive integer k , the goal is to determine whether there exists a minimal st -separator in G of size at least k . Similarly, the MAXIMUM MINIMAL OCT problem seeks to determine if there exists a minimal set of vertices whose deletion results in a bipartite graph, and whose size is at least k . We demonstrate that both problems are fixed-parameter tractable parameterized by k . Our FPT algorithm for MAXIMUM MINIMAL st -SEPARATOR answers the open question by Hanaka, Bodlaender, van der Zanden & Ono [TCS 2019].

One unique insight from this work is the following. We use the meta-result of Lokshtanov, Ramanujan, Saurabh & Zehavi [ICALP 2018] that enables us to reduce our problems to highly unbreakable graphs. This is interesting, as an explicit use of the recursive understanding and randomized contractions framework of Chitnis, Cygan, Hajiaghayi, Pilipczuk & Pilipczuk [SICOMP 2016] to reduce to the highly unbreakable graphs setting (which is the result that Lokshtanov et al. tries to abstract out in their meta-theorem) does not seem obvious because certain “extension” variants of our problems are $W[1]$ -hard.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Parameterized Complexity, FPT, MaxMin problems, Maximum Minimal st -separator, Maximum Minimal Odd Cycle Transversal, Unbreakable Graphs, CMSO, Long Induced Odd Cycles, Sunflower Lemma

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.36

Funding *Ajinkya Gaikwad*: The author is supported by the Ministry of Human Resource Development, Government of India, under Prime Minister’s Research Fellowship Scheme (No. MRF-192002-211).

Saket Saurabh: The author is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416); and he also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.



© Ajinkya Gaikwad, Hitendra Kumar, Soumen Maity, Saket Saurabh, and Roohani Sharma;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 36; pp. 36:1–36:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

In this work we study fixed-parameter tractability of two fundamental MaxMin separation problems: MAXIMUM MINIMUM st -SEPARATOR (MAXMIN st -SEP) and MAXIMUM MINIMUM ODD CYCLE TRANSVERSAL (MAXMIN OCT). In the MAXMIN st -SEP problem, the input is an undirected graph G , two distinct vertices s, t of G and a positive integer k . The goal is to determine if there exists a subset of vertices Z , of size *at least* k , such that Z is a *minimal* st -separator in G . That is, the deletion of Z disconnects s and t and the deletion of any proper subset of Z results in a graph where s and t are connected. Similarly, in the MAXMIN OCT problem, given G, k , the goal is to determine if there exists a set of vertices Z , of size *at least* k , such that Z intersects all odd length cycles in G and Z is minimal, that is no proper subset of Z intersects all odd length cycles in G .

In contrast to the classical polynomial-time solvable st -SEPARATOR problem, where the goal is to find an st -separator of size *at most* k , the MAXMIN st -SEP problem is NP-hard [20]. The MAXMIN OCT problem can also be shown to be NP-hard by giving a reduction from the MAXMIN st -SEP problem (see Section 2.1).

In this work we show that both MAXMIN st -SEP and MAXMIN OCT are fixed-parameter tractable with respect to the parameter k . The first result, in fact, resolves an open problem which was explicitly posed by Hanaka, Bodlaender, van der Zanden & Ono [20].

MAXMIN versions of several classical vertex/edge deletion minimization problems have been studied in the literature. The original motivation behind studying such versions is that the size of the solution of the MAXMIN versions reflects on the worst-case guarantees of a greedy heuristic. In addition to this, the MAXMIN problems have received a lot of attention also because of their deep combinatorial structure which makes them stubborn even towards basic algorithmic ideas. As we will highlight later, neither greedy, nor an exhaustive-search strategy like branching, works for the MAXMIN versions of even the “simplest” problems without significant effort. The MAXMIN versions are even harder to approximate. For example, the classic VERTEX COVER and FEEDBACK VERTEX SET problems admit 2-approximation algorithms [4], which are tight under the Unique Games Conjecture [21]. But the MAXMIN VERTEX COVER admits a $n^{1/2}$ -approximation, which is tight unless $P = NP$ [8, 31], and the MAXMIN FEEDBACK VERTEX SET admits a tight $n^{2/3}$ -approximation [13].

We note here that the study of MaxMin versions of classical deletion problems is not the only proposed way of understanding worst-case heuristics guarantees (though in this work we only focus on such versions). Several variations of different problems have been defined whose core is similar. This includes problems like b -COLORING, GRUNDY COLORING etc, which are analogs for the classic CHROMATIC NUMBER problem.

Parameterized Complexity of MaxMin problems in literature. Given the extensive study of the VERTEX COVER problem in parameterized complexity, the parameterized complexity MAXMIN VERTEX COVER has naturally garnered significant attention [30, 8, 31, 2]. More recently, MAXMIN FEEDBACK VERTEX SET problem has also been explored in [13, 24], where several faster FPT algorithms are proposed. The MAXMIN UPPER DOMINATING SET problem has been studied in [1, 3, 5, 14], and its edge variant, the MAXMIN UPPER EDGE DOMINATING SET, is addressed in [28, 17]. Additionally, MAXMIN and MINMAX formulations have been investigated for a range of other problems, including cut and separation problems [23, 12], knapsack problems [18, 16], matching problems [9], and coloring problems [6]. We elaborate more on the literature around MAXMIN versions of cut and separation problems in the later paragraph.

We remark that for most problems mentioned in the paragraph above, showing that they are FPT parameterized by the solution size is not very difficult (though designing faster FPT algorithms could be much challenging and require deep problem-specific combinatorial insights). The reason is that it is easy to bound the treewidth of the input graph: find a greedy packing of obstructions (edges for MAXMIN VERTEX COVER and cycles for MAXMIN FEEDBACK VERTEX SET); if the packing size is at least k , then the instance is a yes-instance, otherwise there exists a vertex cover (resp. feedback vertex set) of the input graph, of size at most $2k$ (resp. $\mathcal{O}(k \log k)$ from the Erdős-Pósa theorem). In both case, the treewidth of the graph is bounded by a function of k . Since these problems are expressible in Monadic Second Order (MSO) Logic, from Courcelle’s theorem a linear-time FPT in k algorithm follows.

Parameterized Complexity of MAXMIN cut and separation problems in literature. The key challenge in the study of cut and separation problems, in contrast to the vertex/edge-deletion problems mentioned in the paragraph above, is that it is not always easy to bound the treewidth of the instances. Having said that the existing work on MAXMIN versions of cut and separation problems are still based on treewidth win-win approaches as explained below.

Hanaka et al. [20] studied the parameterized complexity of the MAXMIN SEPARATOR problem. Here the input is only a connected graph G and a positive integer k , and the goal is to find a minimal vertex set of size at least k whose deletion disconnects the graph. This problem has an easy FPT algorithm parameterized by k based on a win-win approach: if the input graph has large treewidth, then it has a large grid-minor which implies the existence of a large solution; otherwise the treewidth is bounded and since the problem is expressible in MSO, one can solve the problem in linear time on bounded treewidth graphs. Note that this approach completely fails when we are looking for an st -separator (which is the problem we attack in the present work) for fixed vertices s and t . In particular, a large grid minor does not necessarily imply a large st -separator.

In the same work Hanaka et al. designed an explicit FPT algorithm parameterized by treewidth for the MAXMIN st -SEP problem and left open the question of determining the parameterized complexity of MAXMIN st -SEP parameterized by the solution size k .

Common techniques like flow augmentation [22] and treewidth reduction [27] are not suitable for solving the MAXIMUM MINIMAL st -SEPARATOR problem. This is because the size of the solution in this problem can be arbitrarily large, rather than being bounded by a function of k . When the solution size is unbounded, these methods fail to simplify the problem or give meaningful reductions. We also considered the idea to reduce the problem to a scenario where the solution size is bounded by some function of k using some win-win approach, and then use techniques like treewidth-reduction, unbreakable tree decompositions or even flow-augmentation. But showing the first part seemed tricky and even if one shows that the solution size is bounded, it is still not straightforward to use these classical cut-based techniques as maintaining certificates of minimality of a partial solution is hard in general.

Later Duarte et al. [12] studied the edge-deletion variant of the MAXMIN st -SEP problem. They termed it the LARGEST st -BOND problem and showed, amongst other results, that it is FPT parameterized by the solution size k . At the core of this algorithm is another treewidth-based win-win approach, which seems hard to generalize to the vertex-deletion case.

Problem 1: MaxMin st -Sep. In this work, for the first time, we use the power of highly unbreakable instances to show fixed-parameter tractability of some MAXMIN separation problems. For two positive integers q, k , a graph G is called (q, k) -unbreakable if *no* vertex

set of size *at most* k can disconnect two (large) sets of size at least q each. For the purposes of an informal discussion, we say a graph is unbreakable if it is (q, k) -unbreakable for some values of q and k .

Chitnis et al. [10] developed a win-win approach based on this unbreakable structure of the graph. The approach has two parts: recursive understanding and randomized contractions. In the first part, a large enough part of the input graph is detected that is unbreakable and has small boundary to the rest of the graph. In the second part, a family of “partial solutions” are computed for this unbreakable part of the graph depending on how the solution for the whole graph interacts with its boundary. If the unbreakable part is large enough, then there exists an irrelevant edge/vertex that does not participate in the computed partial solutions, which helps in reducing the size of the graph.

Unfortunately, at the first glance it looks impossible to use this approach for solving any of the two problems MAXMIN *st*-SEP and MAXMIN OCT. The problem is that in the above approach one needs to find partial solutions on unbreakable graphs for a more general “extension-kind” of problem. In particular, the family of partial solutions should be such that if there exists a solution for the whole graph, then one should be able to replace the part of this solution that intersects with the unbreakable part, with one of the computed partial solutions. To do so, for example, it seems necessary to, in some implicit way at least, guess how an optimum solution of the whole graph intersects with the boundary of this unbreakable part and the partial solution should at least try to be “compatible” with this guess. The bottleneck here is that given a vertex v , finding whether the graph G has *any* minimal *st*-separator *containing* v is NP-hard (see Lemmas 1 and 2). Thus, the problem of determining, given a subset of vertices X , whether G has a minimal *st*-separator of size at least k , that contains X , is W[1]-hard (it is in fact para-NP-hard).

► **Lemma 1.** *Let G be a graph containing vertices s and t . A vertex $v \in V(G)$ is in some minimal *st*-separator Z if and only if there is an induced path between s and t containing v .*

Proof. For the first part, assume there exists a minimal *st*-separator Z that contains v . By the minimality of Z , there must be a path P between s and t in the graph $G - (Z \setminus \{v\})$, which passes through v . In other words, path P is such that no vertex in Z other than v appears on it. However, since there is no induced path between s and t through v , two vertices, say a and b , on P must be adjacent, with a lying between s and v , and b lying between v and t . This creates a new path between s and t that does not include any vertex from Z , contradicting the assumption that Z is an *st*-separator.

For the second part, if there exists an induced path between s and t passing through v , we can construct a minimal *st*-separator Z that includes v . According to Definition 13, let S be the set of all vertices on the induced path from s to a , and T the set of all vertices on the induced path from b to t , where a is a predecessor of v and b a successor of v . These sets, S and T , serve as a *certificate* for the *st*-separator minimality for v . By Lemma 14, we can then construct a minimal *st*-separator that contains v . ◀

► **Lemma 2** ([19]). *Given a graph G and any three arbitrary vertices s, t and v , determining if there exist an induced path between s and t through v is NP-hard.*

Therefore, a first glance suggests that computing “partial solutions” may be W[1]-hard in general. One can ask if the hardness holds even when the graph is unbreakable (which is our scenario). It turns out yes, because a result by Lokshtanov et al. [26], shows that the FPT algorithm for unbreakable graphs can be lifted to an FPT algorithm on general graphs for problems definable in Counting Monadic Second Order (CMSO) Logic. Since the extension version is also CMSO definable, there shouldn’t be an FPT algorithm for the extension version even on unbreakable graphs.

Despite this issue, we show that the core lies in solving the problem on unbreakable graphs, and in fact the problem on unbreakable graphs is FPT using non-trivial insights. In fact the result of Lokshtanov et al. [26] comes to rescue. In [26] Lokshtanov et al. show that, in order to show fixed-parameter tractability of CMSO definable problems parameterized by the solution size (say k), it is enough to design FPT algorithms for such problems when the input graph is (q, k) -unbreakable, for some large enough q that depends only on k and the problem. The highlight of this result, compared to explicitly doing the above-mentioned approach of recursive understanding and randomized contractions is that, this result says that it is enough to solve the *same* problem on unbreakable graphs. This allows us to skip taking the provably hard route of dealing with extension-kind problems.

After overcoming the above issues, we can safely say that the core lies in designing an FPT algorithm for the unbreakable case, which itself is far from obvious. We give a technical overview of this phase in Section 1.1.

Problem 2: MaxMin OCT. The challenges continue for the MAXMIN OCT problem. In parameterized complexity, the first FPT algorithm for the classical ODD CYCLE TRANSVERSAL (OCT) problem parameterized by the solution size, introduced the technique of Iterative Compression [29]. Using this, it was shown that OCT reduces to solving at most $3^k \cdot n$ instances of the polynomial-time *st*-SEPARATOR problem. Most algorithms for different variants of the OCT problem, like INDEPENDENT OCT, use the same framework and reduce to essentially solving the variant of *st*-SEPARATOR, like INDEPENDENT *st*-SEPARATOR [25].

For the MAXMIN variant, unfortunately this is not the case. In fact our FPT algorithm for MAXMIN OCT is independent of our FPT algorithm for MAXMIN *st*-SEP. The reason is again that finding a minimal odd cycle transversal set (oct) that extends a given vertex is NP-hard (see Observation 3 and Lemma 4).

► **Observation 3.** *A vertex $v \in V(G)$ does not participate in any induced odd cycle of G if and only if v is not in any minimal OCT.*

Proof. For the forward direction, let Z be a minimal oct of G of size at least k . Then $G - Z$ is bipartite. Additionally, for any $z \in Z$, the graph $G - (Z \setminus \{z\})$ contains an odd cycle through z , implying the existence of an induced odd cycle C_z containing z . As the cycle is an induced odd cycle, we know that the vertex v does not lie on this cycle. We will show that Z is also a minimal oct in $G - v$. Clearly, $G - (Z \cup \{v\})$ is a bipartite graph. Therefore, Z is an oct of $G - v$. For any vertex $z \in Z$, there is an induced odd cycle C_z in $G - (Z \cup \{v\})$. Therefore, Z is also a minimal odd cycle transversal of $G - v$.

For the other direction, let Z be a minimal oct of $G - v$ with $|Z| \geq k$. Then, $G - (Z \cup \{v\})$ is bipartite. For each $z \in Z$, the graph $G - ((Z \setminus \{z\}) \cup \{v\})$ contains an induced odd cycle C_z . Since v is not part of any induced odd cycle, $G - Z$ is bipartite, and Z remains a minimal odd cycle transversal in G . ◀

► **Lemma 4.** *Given a graph G and a vertex $v \in V(G)$, determining whether there is an induced odd cycle containing a given vertex is NP-complete.*

Proof. We know that given two vertices a and b , determining if a graph contains an induced path of odd length between a and b is NP-complete [7]. Construct a graph G' by adding a new vertex x to G and making it adjacent to vertices a and b . It is clear that there exists an induced odd cycle through x in G' if and only if there is an induced path of odd length between a and b in G . This finishes the proof. ◀

At the core of the Iterative Compression based approach for OCT, a subset of vertices X is guessed to be in the solution, and after deleting X , the problem reduces to finding an st -separator, for some s, t . In particular, the final solution is this set X union a minimum st -separator. For the MAXMIN case, this amounts to finding a minimal st -separator that together with X forms a minimal oct. Since the extension version of both the MAXMIN st -SEPARATOR and MAXMIN OCT are para-NP-hard, this leaves little hope to use the same approach for MAXMIN versions.

Having eliminated this approach, we again go back to the approach via unbreakable graphs. This time again the core lies in designing an FPT algorithm when the graph is highly unbreakable and this algorithm require lot more insights than that of the MAXMIN st -SEP. We give a technical overview of this stage in Section 1.1.

1.1 Our results and technical overview

Below we state two main theorems of this work.

► **Theorem 5.** MAXIMUM MINIMAL st -SEPARATOR parameterized by k is FPT.

► **Theorem 6.** MAXIMUM MINIMAL OCT parameterized by k is FPT.

As discussed earlier, to prove both our theorems we first show that both MAXMIN st -SEP and MAXMIN OCT are CMSO definable. We then use Proposition 7, to reduce to solving the problem on unbreakable graphs.

► **Proposition 7** (Theorem 1, [26]). *Let ψ be a CMSO formula. For all $c \in \mathbb{N}$, there exists $s \in \mathbb{N}$ such that if there exists an algorithm that solves CMSO $[\psi]$ on (s, c) -unbreakable structures in time $\mathcal{O}(n^d)$ for some $d > 4$, then there exists an algorithm that solves CMSO $[\psi]$ on general structures in time $\mathcal{O}(n^d)$.*

In particular, to prove Theorems 5 and 6, it is enough to prove Theorems 8 and 9.

► **Theorem 8.** For positive integers $q, k \geq 1$, MAXIMUM MINIMAL st -SEPARATOR on (q, k) -unbreakable graphs on n vertices can be solved in time $(k-1)^{2q} \cdot n^{\mathcal{O}(1)}$.

► **Theorem 9.** For any positive integers $q, k \geq 1$, MAXIMUM MINIMAL OCT on (q, k) -unbreakable graphs on n vertices can be solved in time $2^{(qk)^{\mathcal{O}(q)}} \cdot n^{\mathcal{O}(1)}$.

MaxMin st -Sep on (q, k) -unbreakable graphs. To prove Theorem 8, we develop a branching algorithm which exploits the unbreakability of the input graph. The key observation behind the algorithm is that for any two vertex sets S and T such that $s \in S$, $t \in T$ and $G[S]$ and $G[T]$ are connected, every minimal ST -separator is also a minimal st -separator. As we are working on (q, k) -unbreakable graphs, if we manage to find such sets where $|S| > q$ and $|T| > q$, then every minimal ST -separator has size at least k . In this case, we can construct a minimal ST -separator greedily in polynomial time, which can then be returned as a solution. Therefore, the goal of our branching algorithm is to construct such sets S and T .

The algorithm begins with $S = \{s\}$ and $T = \{t\}$. We use a reduction rule which ensures that at any point $N(S)$ and $N(T)$ (the neighborhoods of S and T) are minimal ST -separators in G . Clearly, we can assume that $|N(S)| < k$ and $|N(T)| < k$; otherwise, we have already found a solution. A crucial observation is that if there exists a minimal ST -separator Z of size at least k , then there exists a vertex $u \in N(S) \setminus N(T)$ (resp. $u \in N(T) \setminus N(S)$) such that $u \notin Z$. This implies that such a vertex u remains reachable from S even after removing

the solution Z . This observation allows us to grow the set S to $S \cup \{u\}$ (resp. T to $T \cup \{u\}$). Since $|N(S)| < k$ (resp. $|N(T)| < k$), one can branch on all possible vertices $u \in N(S)$ (resp. $u \in N(T)$) and in each branch grow S (resp. T). Based on whether $\min(|S|, |T|)$ is $|S|$ or $|T|$, we branch on the vertices in $N(S) \setminus N(T)$ or $N(T) \setminus N(S)$, respectively, to increase the size of these sets. Once both S and T have sizes of at least q , we can greedily construct and output a minimal ST -separator (which is also a minimal st -separator) of size at least k .

MaxMin OCT on (q, k) -unbreakable graphs. The algorithm for this case uses two key lemmas, both of which provide sufficient conditions for a yes instance. Here the input is a $(q, 2k)$ -unbreakable graph.

Our first key lemma (Lemma 26) says that if there exists an induced odd cycle of length at least $2q + 2$ in G , then there always exist a minimal oct in G of size at least k . The proof of this result is based on a branching algorithm, which works similarly to the branching algorithm for the MAXMIN st -SEPARATOR problem on (q, k) -unbreakable graphs, by carefully selecting the sets S and T at the beginning of the algorithm. Note that we cannot use this lemma, on its own as a stopping criterion, because one does not know how to find a long induced odd length cycle efficiently. Nevertheless, as we see later it becomes useful together with our second sufficient condition.

Our second key lemma (Lemma 28), which states a second sufficient condition, says that if there exists a large enough family of distinct (and not necessarily disjoint) induced odd cycles in G of lengths at most $2q + 1$, then there always exists a minimal oct in G of size at least k . The proof of this result relies on the Sunflower Lemma [15, 11], along with the observation that the subgraph induced by the core of the sunflower must be bipartite. Such a large sunflower then serves as a certificate that any oct which is disjoint from the core of the sunflower is large. Also since the core is bipartite, there exists an oct that is disjoint from it.

Another simple, yet important observation is that, if a vertex is not part of any induced odd cycle, then deleting this vertex from the graph does not affect the solution. Again, the problem here is that determining whether a vertex passes through an induced odd cycle is NP-hard (see Lemma 4), therefore we cannot use it as a reduction rule.

Finally, using these two sufficient conditions and the observation above, we provide an FPT algorithm that, given a vertex x , either outputs an induced odd cycle containing x if it exists, or concludes correctly that one of the two scenarios mentioned above occurs. In the later situation we are done. Also if the induced odd cycle containing x , which is returned, is long, then also we are done because of the first sufficient condition. If the algorithm outputs, that there is no induced odd cycle through x , then we can safely delete x from the graph and reduce its size. Because of deleting such vertices, the new graph may no longer be unbreakable, in which case we start solving the problem on the reduced graph completely from scratch.

By running this algorithm for each $x \in V(G)$, we reduce the graph to one where each vertex is contained in some small induced odd cycle. If such a graph contains a sufficiently large number of vertices, it guarantees the existence of a large family of distinct small induced odd cycles in G , each of length at most $2q + 2$, in which case we can return a yes instance because of the second sufficient condition. This result finally allows us to bound the number of vertices in the graph by $(qk)^{\mathcal{O}(q)}$. We can then solve the problem using a brute-force algorithm on this graph.

2 Preliminaries

Throughout this article, $G = (V, E)$ denotes a finite, simple and undirected graph of order $|V| = n$. The *open neighbourhood* $N_G(v)$ of a vertex $v \in V(G)$ is the set $\{u \mid (u, v) \in E(G)\}$. The *closed neighbourhood* $N_G[v]$ of a vertex $v \in V(G)$ is the set $\{v\} \cup N_G(v)$. The *degree* of $v \in V(G)$ is $|N_G(v)|$ and is denoted by $d_G(v)$. The subgraph induced by $D \subseteq V(G)$ is denoted by $G[D]$. For $X, Y \subseteq V(G)$ such that $X \cap Y = \emptyset$, $E_G(X, Y)$ denote the edges of G with one endpoint in X and the other in Y . We will drop the subscripts in the above notation, whenever it is clear from the context. For $s, t \in V(G)$, by an st -path in G we mean a path from s to t in G . For $S, T \subseteq V(G)$, by an ST -path we mean a path between a vertex of S to a vertex of T . For $i \in \mathbb{N}$, $[i]$ denotes the set $\{1, \dots, i\}$.

Let G be a graph. A pair (X, Y) , where $X \cup Y = V(G)$, is called a *separation* if $E(X \setminus Y, Y \setminus X) = \emptyset$. The order of (X, Y) is $|X \cap Y|$. If there exists a separation (X, Y) of order at most k such that $|X \setminus Y| \geq q$ and $|Y \setminus X| \geq q$, then G is (q, k) -breakable and the separation (X, Y) is called a *witnessing separation* for the (q, k) -breakability of G . Otherwise, G is (q, k) -unbreakable.

We refer to [11] for the formal definition of Counting Monadic Second Order (CMSO) logic. We will crucially use the following result of Lokshantov et al. [26] that allows one to show that a CMSO-expressible graph problem is FPT by designing an FPT algorithm for the problem on (q, k) -unbreakable graphs, for any k and a sufficiently large q that depends only on k .

► **Proposition 7** (Theorem 1, [26]). *Let ψ be a CMSO formula. For all $c \in \mathbb{N}$, there exists $s \in \mathbb{N}$ such that if there exists an algorithm that solves CMSO $[\psi]$ on (s, c) -unbreakable structures in time $\mathcal{O}(n^d)$ for some $d > 4$, then there exists an algorithm that solves CMSO $[\psi]$ on general structures in time $\mathcal{O}(n^d)$.*

Next, we state the Sunflower Lemma which is used in this paper. We first define the terminology used in the statement of the next lemma. Given a set system (U, \mathcal{F}) , where U is a set and \mathcal{F} is a family containing distinct subsets of U , a *sunflower* with k petals and a core Y is a collection of sets $S_1, S_2, \dots, S_k \in \mathcal{F}$ such that $S_i \cap S_j = Y$ for all $i \neq j$. The sets $S_i \setminus Y$ are called the *petals* of this sunflower. If the sets in \mathcal{F} are distinct and $k \geq 2$, then none of the petals of the sunflower are empty. Note that a family of pairwise disjoint sets is a sunflower (with an empty core).

► **Theorem 10** (Sunflower Lemma, [15, 11]). *Let \mathcal{F} be a family of distinct sets over a universe U , such that each set in \mathcal{F} has cardinality exactly d . If $|\mathcal{F}| > d!(k-1)^d$, then \mathcal{F} contains a sunflower with k petals and such a sunflower can be computed in time polynomial in $|\mathcal{F}|$, $|U|$, and k .*

2.1 NP-hardness of Maximum Minimal OCT

► **Lemma 11.** MAXIMUM MINIMAL OCT is NP-hard.

Proof. It was shown in [20] that MAXIMUM WEIGHT MINIMAL st -SEPARATOR is NP-hard on bipartite graphs, even when all vertex weights are identical. This implies that MAXIMUM MINIMAL st -SEPARATOR is NP-hard on bipartite graphs. We provide a polynomial-time reduction from the MAXIMUM MINIMAL st -SEPARATOR to the MAXIMUM MINIMAL OCT. Given an instance $I = (G, s, t, k)$ of the MAXIMUM MINIMAL st -SEPARATOR, we construct an instance $I' = (G', k' = k)$ of the MAXIMUM MINIMAL OCT as follows. We assume that $k > 1$. We consider two cases based on the bipartition of G :

Case 1. If s and t are on the same side of the bipartition, we add an edge between s and t , and subdivide it by adding two vertices u and v , creating a new graph G' .

Case 2. If s and t are on opposite sides of the bipartition, we add a subdivided edge between s and t with one new vertex u' , resulting in G' .

In both cases, the newly added path between s and t in G' is denoted by P' .

We now prove that the instances I and I' are equivalent.

In the forward direction, let Z be a minimal st -separator in G of size at least k . We claim that Z is a minimal oct in G' . Since G is bipartite, any odd cycle in G' must involve s and t . Removing Z from G separates s and t , meaning $G' - Z$ contains only the newly added path P' , ensuring no odd cycles in G' . Thus, Z is an oct in G' .

Next, we show that Z is minimal. For every $z \in Z$, the graph $G - (Z \setminus \{z\})$ contains a path between s and t . Let's call it P . In Case 1, this path is even-length, and in Case 2, it is odd-length. In both cases, the graph $G' - (Z \setminus \{z\})$ contains two vertex-disjoint paths P and P' between s and t of different parities, forming an odd cycle. Therefore, Z is a minimal oct in G' .

In the backward direction, let Z' be a minimal oct in G' of size at least k . Since $k > 1$, Z' does not include the newly added vertices u, v (in Case 1) or u' (in Case 2). We claim that Z' is a minimal st -separator in G .

If Z' were not an st -separator in G , then there would exist a path between s and t in $G - Z'$, implying the presence of an odd cycle in $G' - Z'$ involving the path P' . Since Z' is a minimal oct in G' , for every $z \in Z'$, the graph $G' - (Z' \setminus \{z\})$ contains an odd cycle. Therefore, $G' - (Z' \setminus \{z\})$ must contain a path P which is a vertex-disjoint st -path from P' . This implies that $G - (Z' \setminus \{z\})$ contains an st -path, proving that Z' is a minimal st -separator in G . Thus, I and I' are equivalent, completing the proof. ◀

3 Maximum Minimal st -Separator parameterized by the solution size

The goal of this section is to prove Theorem 5.

► **Theorem 5.** MAXIMUM MINIMAL st -SEPARATOR parameterized by k is FPT.

Let (G, k) be an instance of MAXMIN st -SEPARATOR. The goal is to reduce the task to designing an algorithm for (q, k) -unbreakable graphs. For this we first show that the problem can be expressed in CMSO (in fact in MSO). Since MAXMIN st -SEPARATOR is a maximization problem, the size of the solution can potentially be as large as $\mathcal{O}(n)$. To formulate a CMSO sentence that is bounded by a function of k , we focus on a k -sized subset of the solution and encode the minimality of each of its vertices in a way that allows for its extension to a “full-blown” minimal solution.

► **Lemma 12.** MAXIMUM MINIMAL st -SEPARATOR is CMSO-definable with a formula of length $\mathcal{O}(k)$.

Proof. The instance (G, k) is a yes instance of MAXMIN st -SEP if there exists $Z \subseteq V(G)$ of size at least k such that $G - Z$ has no st -path (equivalently s and t are in different connected components of $G - Z$), and for each $v \in Z$, $G - (Z \setminus \{v\})$ has an st -path (that is s and t are in the same connected component of $G - (Z \setminus \{v\})$).

Alternately, suppose $Z \subseteq V(G)$ (of arbitrarily large size) such that $G - Z$ has no st -path, and Z contains k distinct vertices v_1, \dots, v_k such that for each $i \in [k]$, $G - (Z \setminus \{v_i\})$ contains an st -path. Then Z may not be a minimal st -separator but it always contains a minimal st -separator of size at least k . In fact, (G, k) is a yes instance if and only if such a set Z

exists. These properties of Z can be incorporated as a CMSO formula ψ as follows, where $\mathbf{conn}(U)$ is a CMSO sentence that checks whether a vertex set U induces a connected graph. The CMSO description of CMSO can be, for example, found in [11].

$$\begin{aligned} \psi = & \exists Z \subseteq V(G) \left(\exists v_1, v_2, \dots, v_k \in Z \left(\bigwedge_{1 \leq i < j \leq k} v_i \neq v_j \right) \right. \\ & \wedge \exists U \subseteq V(G) \setminus Z \left((s \in U) \wedge (t \in U) \wedge \mathbf{conn}(U) \right) \\ & \left. \bigwedge_{i=1}^k \neg \exists U \in V(G) \setminus (Z \setminus \{v_i\}) \left((s \in U) \wedge (t \in U) \wedge \mathbf{conn}(U) \right) \right) \end{aligned}$$

It is clear that the size of the above formula ψ depends linearly on k . ◀

From Lemma 12 and Proposition 7, to prove Theorem 5, it is enough to prove Theorem 8.

► **Theorem 8.** *For positive integers $q, k \geq 1$, MAXIMUM MINIMAL st -SEPARATOR on (q, k) -unbreakable graphs on n vertices can be solved in time $(k-1)^{2q} \cdot n^{\mathcal{O}(1)}$.*

We prove Theorem 8 in Section 3.1. As mentioned earlier, given a vertex set V' , it may not always be possible to extend it to a minimal st -separator. Below, we give a definition for a *certificate* for the st -separator minimality of a set $V' \subseteq V(G)$, which guarantees the existence of a minimal st -separator that contains (extends) the set V' .

► **Definition 13.** *Let G be a graph, $s, t \in V(G)$ and $V' \subseteq V(G)$. We say that two sets of vertices S and T serve as a certificate for the st -separator minimality for V' if the following conditions hold: $s \in S$ and $t \in T$, $S \cap T = \emptyset$, $G[S]$ and $G[T]$ are connected subgraphs, $E_G(S, T) = \emptyset$, and for every $v \in V'$, the subgraph $G[S \cup T \cup \{v\}]$ is connected. Note that $V' \cap (S \cup T) = \emptyset$.*

► **Lemma 14.** *Let G be a graph, and let $s, t \in V(G)$. If there exists a certificate for the st -separator minimality of $V' \subseteq V(G)$, then there exists a minimal st -separator in G that includes all the vertices of V' .*

Proof. Let S and T serve as a certificate for the st -separator minimality of V' . We will construct a set $V' \subseteq Z \subseteq V(G) \setminus (S \cup T)$ which is a minimal st -separator in G . The set Z is constructed iteratively. Initialize $Z := \emptyset$ and $G' := G[S \cup T]$. Fix an arbitrary ordering of the vertices in $V(G) \setminus (S \cup T)$.

For each vertex v in the prescribed order: (i) if $G[S \cup T \cup \{v\}]$ is connected, then update $Z := Z \cup \{v\}$, otherwise (ii) if $G[S \cup T \cup \{v\}]$ is not connected, then update $G' := G[V(G') \cup \{v\}]$, update S to be the vertices reachable from old S in G' , and update T to be the vertices reachable from old T in G' .

The process continues until all vertices in $V(G) \setminus V(G')$ have been processed. The final set Z is then returned as a minimal st -separator of G . Note that if the initial sets S and T served as a *certificate* for the minimality of the st -separator V' , then $V' \subseteq Z$, as the subgraph $G[S \cup T \cup \{v\}]$ will always be connected for each $v \in V'$ during every stage of the above process. ◀

3.1 Maximum Minimal st -Separator on (q, k) -unbreakable graphs

In this section, we prove Theorem 8. To prove Theorem 8 we design a branching algorithm that maintains a tuple (G, S, T, k, q) where G is a (q, k) -unbreakable graph, $S, T \subseteq V(G)$ and q, k are positive integers. Additionally the sets S, T satisfy the following properties.

1. $s \in S$ and $t \in T$,
2. $S \cap T = \emptyset$,
3. both $G[S]$ and $G[T]$ are connected and
4. $E(S, T) = \emptyset$.

An instance (G, S, T, k, q) satisfying the above properties is called a *valid* instance. Given a valid instance, we design a branching algorithm that outputs a minimal ST -separator of G , which is disjoint from $S \cup T$, and has size at least k , if it exists. The algorithm initializes $S := \{s\}$ and $T := \{t\}$. Note that the above-mentioned properties of the sets S, T ensure that at each stage of the algorithm, every minimal ST -separator is also a minimal st -separator.

The algorithm has one reduction rule (Reduction Rule 3), four stopping criteria (Reduction Rules 1, 2, 4 and 5) and one branching rule (Branching Rule 1). The branching rule is applied when neither the reduction rule nor the three stopping criterion can be applied. The overall idea is the following. Observe that $N(S) \cap N(T)$ is a part of any minimal ST -separator. Therefore, if $|N(S) \cap N(T)| \geq k$, then we can correctly report that G has a minimal ST -separator of size at least k . Reduction Rule 3 ensures that $N(S)$ (resp. $N(T)$) is a *minimal* ST -separator. Therefore when Reduction Rule 3 is no longer applicable, if $|N(S)| \geq k$ (resp. $|N(T)| \geq k$), then we can correctly report that G has a minimal ST -separator of size at least k . In fact, $|N(S)|$ (resp. $|N(T)|$) is a minimal ST -separator of size at least k . Otherwise, we have that both $|N(S)| < k$ and $|N(T)| < k$. In this case, we use the branching rule. Say, without loss of generality that $|S| \leq |T|$. Since $N(S)$ is a minimal ST -separator, but its size is strictly less than k and every minimal ST -separator contains $N(S) \cap N(T)$, there exists a vertex in $N(S) \setminus N(T)$ that does not belong to the solution (if there exists a solution of size at least k). In this case we branch on the vertices of $N(S) \setminus N(T)$. If we guess that a vertex $v \in N(S)$ does not belong to the solution, since $v \in N(S)$, v remains reachable from S after removing the solution. In this case, we update $S := S \cup \{v\}$. Therefore, each application of the branching rule increases the size of the smaller of the two sets S or T . When both $|S| \geq q$ and $|T| \geq q$, from the (q, k) -unbreakability of G , we know that *every* ST -separator of G has size at least k . And hence there is a minimal st -separator of size at least k .

Below we formalize the above arguments. Given $I = (G, S, T, k, q)$, we define its measure $\mu(I) = q - \min(|S|, |T|)$. We state the reduction rules and a branching rule below. We apply the reduction rules in order exhaustively before applying the branching rule.

► **Lemma 15.** *Let $I = (G, S, T, k, q)$ where G is (q, k) -unbreakable. If $\mu(I) \leq 0$, then every minimal ST -separator, which is disjoint from $S \cup T$, is of size at least k .*

Proof. If $\mu(I) \leq 0$, then $q \leq \min(|S|, |T|)$. For the sake of contradiction, let us assume that there exists a minimal ST -separator Z in G , which is disjoint from $S \cup T$, and has size strictly less than k . Note that $G \setminus Z$ contains two connected components of size at least q each: one containing S , say C_S , and the other containing T . Consider the separation $(C_S \cup Z, V(G) \setminus C_S)$ of G . This is a witnessing separation that G is (q, k) -breakable, which is a contradiction. ◀

The safeness of Reduction Rule 1 is immediate from Lemma 15.

► **Reduction Rule 1.** *If $\mu(I) \leq 0$, then report a yes instance.*

► **Reduction Rule 2.** *If $|N(S) \cap N(T)| \geq k$, then report a yes instance.*

► **Lemma 16.** *Reduction Rule 2 is safe.*

36:12 MaxMin Separation Problems

Proof. The above reduction rule is safe because the sets S and T serve as a *certificate* for the st -separator minimality of the vertex set $N(S) \cap N(T)$. From Lemma 14, there exists a minimal st -separator in G that contains $N(S) \cap N(T)$. Since $|N(S) \cap N(T)| \geq k$, we conclude that G contains a minimal st -separator of size at least k . ◀

► **Lemma 17.** *If there exists $v \in N(S)$ (resp. $v \in N(T)$), such that every path from v to any vertex of T (resp. S) intersects $N(S) \setminus \{v\}$ (resp. $N(T) \setminus \{v\}$), or there is no path from v to any vertex of T (resp. S), then there is no minimal ST -separator which is disjoint from $S \cup T$ and that contains v .*

Proof. When v has no path to any vertex of T , then such a vertex cannot lie on any ST -path and hence, is not a part of any minimal ST -separator.

Suppose now that $v \in N(S)$ and every path from v to any vertex of T intersects $N(S) \setminus \{v\}$. The other case when $v \in N(T)$ is symmetric. For the sake of contradiction, say there exists a minimal ST -separator Z such that $v \in Z$ and $Z \cap (S \cup T) = \emptyset$. This implies that in the graph $G - Z$, there is no path from any vertex in S to any vertex in T , but in the graph $G - (Z \setminus \{v\})$, such a path, say P , exists. Let P be a path from s' to t' in $G - (Z \setminus \{v\})$, where $s' \in S$ and $t' \in T$.

Since v cannot reach any vertex of T (in particular t') in G , without traversing another vertex, say u , in $N(S)$, consider the u to t' subpath of P (which does not contain v). Since $u \in N(S)$, let $s'' \in N(u) \cap S$. Since $Z \cap (S \cup T) = \emptyset$, there exists a path P' from s'' to t' in $G - (Z \setminus \{v\})$ that does not contain v (take the edge (s'', u) , followed by the u to t' subpath of P). This contradicts that Z is an ST -separator. ◀

The following reduction rule ensures that both $N(S)$ and $N(T)$ are minimal ST -separators.

► **Reduction Rule 3.** *If there exists $v \in N(S)$ (resp. $v \in N(T)$), such that every path from v to any vertex of T (resp. S) intersects $N(S)$ (resp. $N(T)$), or there is no path from v to any vertex of T (resp. S), then update $S := S \cup \{v\}$ (resp. $T := T \cup \{v\}$).*

► **Lemma 18.** *Reduction Rule 3 is safe.*

Proof. From Lemma 17, no minimal ST -separator, that is disjoint from $S \cup T$, contains v . Since $v \in N(S)$ (resp. $v \in N(T)$), for any minimal ST -separator Z , v is reachable from S in $G - Z$, since $Z \cap S = \emptyset$ (resp. $Z \cap T = \emptyset$). Thus, Z is also a minimal separator between $S \cup \{v\}$ and T . ◀

► **Lemma 19.** *When Reduction Rule 3 is no longer applicable, $N(S)$ (resp. $N(T)$) is a minimal ST -separator.*

Proof. First note that $N(S)$ (resp. $N(T)$) is an ST -separator in G which is disjoint from $S \cup T$, since $N(S) \cap T = \emptyset$ because $E(S, T) = \emptyset$.

For the sake of contradiction, say $N(S)$ is not a minimal ST -separator in G . In particular, there exists $v \in N(S)$ such that $N(S) \setminus \{v\}$ is also an ST -separator. Since Reduction Rule 3 is no longer applicable, there exists a path from v to a vertex of T , say t' , which has no other vertex of $N(S)$. Such a path together with an edge from v to a vertex of S , gives an ST -path, which intersects $N(S)$ only at v . ◀

From Lemma 19, the safeness of Reduction Rule 4 is immediate.

► **Reduction Rule 4.** *If $|N(S)| \geq k$ or $|N(T)| \geq k$ then report a yes instance.*

► **Reduction Rule 5.** *If $N(S) \setminus N(T) = \emptyset$ or $N(T) \setminus N(S) = \emptyset$, then report a no instance.*

► **Lemma 20.** *Reduction Rule 5 is safe.*

Proof. Suppose $N(S) \setminus N(T) = \emptyset$. The other case is symmetric. Then any ST -path uses only the vertices of $S \cup T \cup (N(S) \cap N(T))$. In this case there is a unique ST -separator in G which is disjoint from $S \cup T$. This separator is $N(S) \cap N(T)$. Since Reduction Rule 2 is no longer applicable, $|N(S) \cap N(T)| < k$. Thus G has no ST -separator of size at least k . ◀

► **Branching Rule 1.** *If $|S| \leq |T|$ (resp. $|T| < |S|$) and $N(S) \setminus N(T) \neq \emptyset$ (resp. $N(T) \setminus N(S) \neq \emptyset$), then for each vertex $x \in N(S) \setminus N(T)$ (resp. $x \in N(T) \setminus N(S)$), we recursively solve the instance $(G, S \cup \{x\}, T, k, q)$ (resp. $(G, S, T \cup \{x\}, k, q)$).*

First observe that the new instances created in this branching rule are all valid, that is, the sets $S \cup \{x\}$, T (respectively S , $T \cup \{x\}$) satisfy the desired properties: the two sets $S \cup \{x\}$ and T (resp. S and $T \cup \{x\}$) are disjoint, $G[S \cup \{x\}]$ (resp. $G[T \cup \{x\}]$) is connected and $E_G(S \cup \{x\}, T) = \emptyset$ (resp. $E_G(S, T \cup \{x\}) = \emptyset$) because $x \in N(S) \setminus N(T)$.

► **Lemma 21.** *Branching Rule 1 is exhaustive, that is, G has a minimal ST -separator of size at least k which is disjoint from $S \cup T$ if and only if there exists $x \in N(S) \setminus N(T)$ (resp. $x \in N(T) \setminus N(S)$) such that G has a minimal separator of size at least k between $S \cup \{x\}$ and T (resp. S and $T \cup \{x\}$), which is disjoint from $S \cup T \cup \{x\}$.*

Proof. Assume that $|S| \leq |T|$. The other case is symmetric. Since Reduction Rule 4 is no longer applicable, $|N(S)| < k$. Since Reduction Rule 3 is no longer applicable and because of Lemma 19, $N(S)$ is a minimal ST -separator. Also because $N(S) \cap N(T)$ is contained in every minimal ST -separator in G (from the proof of Lemma 16), for any minimal ST -separator in G of size at least k , say Z , which is disjoint from $S \cup T$, there exists a vertex $x \in N(S) \setminus N(T)$ such that $x \notin Z$. Since $x \in N(S)$, x remains reachable from S in $G - Z$. In particular, Z is also a minimal separator between $S \cup \{x\}$ and T .

In the other direction say the instance $(G, S \cup \{x\}, T, k, q)$ reports a minimal separator, say Z , between $S \cup \{x\}$ and T of size at least k . Since $S \cup \{x\}$ and T satisfy the desired properties of a valid instance, Z is also a minimal ST -separator in G . ◀

Proof of Theorem 8. Initialize an instance $I = (G, S, T, k, q)$ where $S = \{s\}$ and $T = \{t\}$. Note that $\mu(I) = q - 1$. Apply Reduction Rules 1-5 exhaustively in-order. Without loss of generality, say $|S| \leq |T|$, the other case is analogous. Since none of the reduction rules are applicable, $1 \leq |N(S) \setminus N(T)| < k$. Now apply Branching Rule 1. After every application of the Branching Rule 1, $\mu(I')$, where I' is the new instance, strictly decreases if $|S| \neq |T|$. If $|S| = |T|$, then after every two applications of the Branching Rule 1, the measure μ decreases by 1. From Reduction Rule 1, if μ of an instance is at most 0, then we stop and report a yes instance. The correctness of this algorithm follows from the safeness of the Reduction Rules 1-5 and Branching Rule 1. We now argue about the running time.

Note that all reduction rules can be applied in polynomial time. Also all reduction rules, except Reduction Rule 3, is applied only once throughout the algorithm. Reduction Rule 3 is applied at most $2q$ times (or until both S and T grow to a size of q each). Thus only polynomial time is spent on all applications of all reduction rules. The branching rule branches in at most $k - 1$ instances and has depth bounded by $2q$. Therefore the overall running time is $(k - 1)^{2q} \cdot n^{\mathcal{O}(1)}$. ◀

4 MAXIMUM MINIMAL OCT parameterized by solution size

In this section, we prove Theorem 6.

► **Theorem 6.** *MAXIMUM MINIMAL OCT parameterized by k is FPT.*

Throughout this section, we call a set of vertices of G whose deletion results in a bipartite graph, an *oct* of G . To prove Theorem 6, we first show that the problem is CMSO definable (Lemma 22). Using Proposition 7, one can reduce to solving this problem on $(q, 2k)$ -unbreakable graphs. On $(q, 2k)$ -unbreakable graphs, we then list and prove two sufficient conditions (Lemmas 28 and 26) which always imply a yes instance (in fact the first one implies a yes instance even when the input graph is not $(q, 2k)$ -unbreakable). We also make an observation about irrelevant vertices that can be deleted without changing the solution of the instance (Observation 3). Even though checking whether any one of these sufficient conditions hold or finding these irrelevant vertices, may not be efficient, nonetheless we design an FPT algorithm that correctly concludes that at least one of the sufficient conditions is met, or outputs an irrelevant vertex, whenever the number of vertices in the graph is strictly more than a number with is a function of q and k (Theorem 25). In the case when an irrelevant vertex is outputted, deleting them reduces the size of the graph but the resulting graph may not be $(q, 2k)$ -unbreakable. In this case, we start from the beginning and solve the problem from scratch (on general graphs). If none of the above hold, then the number of vertices in the graph is bounded, and we can solve the problem using brute-force.

► **Lemma 22.** *MAXIMUM MINIMAL OCT is CMSO-definable by a formula of length $\mathcal{O}(k)$.*

Proof. Let (G, k) be an instance of MAXIMUM MINIMAL OCT. Then (G, k) is a yes instance if there exists $Z \subseteq V(G)$ of size at least k such that $G - Z$ is bipartite and for each $v \in Z$, $G - (Z \setminus \{v\})$ is not bipartite.

Alternately, let $Z \subseteq V(G)$ such that Z contains k distinct vertices v_1, \dots, v_k , such that $G - Z$ is bipartite and for each $i \in [k]$, $G - (Z \setminus \{v_i\})$ is not bipartite. Observe that if such a set Z exists, it may not be a minimal oct of G , but it definitely contains a minimal oct of size at least k . In fact, (G, k) is a yes instance if and only if such a set Z exists. We phrase this description of Z as the CMSO formula ψ as defined below.

$$\begin{aligned} \varphi \equiv \exists Z \subseteq V(G) & \left(\exists v_1, v_2, \dots, v_k \in Z \left(\bigwedge_{1 \leq i < j \leq k} v_i \neq v_j \right) \right. \\ & \wedge \mathbf{bipartite}(V(G) \setminus Z) \\ & \left. \wedge \left(\bigwedge_{i=1}^k \neg \mathbf{bipartite}(V(G) \setminus (Z \setminus \{v_i\})) \right) \right) \end{aligned}$$

where $\mathbf{bipartite}(W)$ is a CMSO sentence given below, which checks whether the graph induced by the vertices in W is bipartite.

$$\begin{aligned} \mathbf{bipartite}(W) \equiv \exists X \subseteq W, \exists Y \subseteq W \\ & \left((X \cap Y = \emptyset) \wedge (X \cup Y = W) \right. \\ & \left. \wedge \forall u, v \in W (E(u, v) \implies (u \in X \iff v \in Y)) \right). \end{aligned}$$

It is clear that the size of the above formula φ depends linearly on k . ◀

Proof of Observation 3. For the forward direction, let Z be a minimal oct of G of size at least k . Then $G - Z$ is bipartite. Additionally, for any $z \in Z$, the graph $G - (Z \setminus \{z\})$ contains an odd cycle through z , implying the existence of an induced odd cycle C_z containing z . As

the cycle is an induced odd cycle, we know that the vertex v does not lie on this cycle. We will show that Z is also a minimal oct in $G - v$. Clearly, $G - (Z \cup \{v\})$ is a bipartite graph. Therefore, Z is an oct of $G - v$. For any vertex $z \in Z$, there is an induced odd cycle C_z in $G - (Z \cup \{v\})$. Therefore, Z is also a minimal odd cycle transversal of $G - v$.

For the other direction, let Z be a minimal oct of $G - v$ with $|Z| \geq k$. Then, $G - (Z \cup \{v\})$ is bipartite. For each $z \in Z$, the graph $G - ((Z \setminus \{z\}) \cup \{v\})$ contains an induced odd cycle C_z . Since v is not part of any induced odd cycle, $G - Z$ is bipartite, and Z remains a minimal odd cycle transversal in G . ◀

Because of Lemma 22 we can invoke Proposition 7. We invoke Proposition 7 with $c = 2k$. Let q be the s from this proposition that corresponds to this choice of c . We conclude that to prove Theorem 6 it is enough to prove Theorem 9.

► **Theorem 9.** *For any positive integers $q, k \geq 1$, MAXIMUM MINIMAL OCT on (q, k) -unbreakable graphs on n vertices can be solved in time $2^{(qk)^{\mathcal{O}(q)}} \cdot n^{\mathcal{O}(1)}$.*

We prove Theorem 9 in Section 4.1. Next we define a certificate for minimality of oct for a vertex set V' . The existence of such certificates guarantees the existence of a minimal oct which contains V' .

► **Definition 23.** *Given a graph G and a set of vertices $V' \subseteq V(G)$, we say that an induced subgraph G' of G is a certificate for the oct-minimality of V' , if G' is bipartite and for every $v \in V'$, $G[V(G') \cup \{v\}]$ contains an odd cycle.*

► **Lemma 24.** *Given a graph G and a set of vertices V' , if there is a certificate for the oct-minimality of V' then there exists a minimal oct of G that contains all the vertices in V' .*

Proof. Let G' be a certificate for minimality of V' . First observe that $V' \cap V(G') = \emptyset$ because G' is bipartite, but $G[V(G') \cup \{v\}]$, for any $v \in V'$, contains an odd cycle. We now construct a minimal oct of G , say Z , iteratively as follows. Initialize $Z = \emptyset$. Fix an arbitrary ordering of the vertices in $V(G) \setminus V(G')$ (note that $V' \subseteq V(G) \setminus V(G')$). Traverse the vertices of $V(G) \setminus V(G')$ in this order. For any vertex $v \in V(G) \setminus V(G')$ in the chosen order:

- if $G[V(G') \cup \{v\}]$ is bipartite, update $G' := G[V(G') \cup \{v\}]$, that is add v to G' , otherwise
- $G[V(G') \cup \{v\}]$ contains an odd cycle, in which case add v to the set Z .

When all the vertices in $V(G) \setminus V(G')$ have been processed as stated above, then the set Z is a minimal oct of G . Moreover, one can observe that if we start with G' which a certificate for minimality of V' then $V' \subseteq Z$. ◀

4.1 Maximum Minimal OCT on $(q, 2k)$ -unbreakable graphs

The goal of this section is to prove Theorem 9. Let (G, k) be an instance of MAXMIN OCT. A vertex $v \in V(G)$ is called *irrelevant* if (G, k) is equivalent to $(G - v, k)$. To prove Theorem 9 it is enough to prove Theorem 25.

► **Theorem 25.** *For positive integers $q, k \geq 1$, given as input a graph G which is $(q, 2k)$ -unbreakable on at least $(2q + 2)^2(2q + 2)!(k - 1)^{2q+2} + 1$ vertices, there exists an algorithm that runs in time $(qk)^{\mathcal{O}(q)} \cdot n^{\mathcal{O}(1)}$, and either returns a minimal oct of G of size at least k or, outputs an irrelevant vertex v .*

To see the proof of Theorem 9 assuming Theorem 25, observe that if the number of vertices is at most $(2q + 2)^2(2q + 2)!(k - 1)^{2q+2}$, then the problem can be solved using brute-force. Otherwise, the algorithm of Theorem 25 either reports a yes instance, or finds

36:16 MaxMin Separation Problems

an irrelevant vertex v , in which case, delete v from the graph and solve the problem on $G - v$ (which is not necessarily $(q, 2k)$ -unbreakable). The rest of the section is devoted to the proof of Theorem 25.

Irrelevant vertices. Recall Observation 3 from Section 1.

► **Observation 3.** *A vertex $v \in V(G)$ does not participate in any induced odd cycle of G if and only if v is not in any minimal OCT.*

Note that we cannot explicitly design a (polynomial-time) reduction rule based on the above observation, because determining whether there is an induced odd cycle containing a given vertex is NP-complete (see Lemma 4).

Sufficient condition 1 [Long induced odd cycle in G].

► **Lemma 26.** *For any positive integers q, k , if G is $(q, 2k)$ -unbreakable and there exists an induced odd cycle in G of length at least $2q + 2$, then G has a minimal oct of size at least k .*

Proof. Let C be an induced odd cycle of length at least $2q + 2$ in G . Let $x, y \in V(C)$ be arbitrarily chosen vertices such that $C \setminus \{x, y\}$ contains exactly two paths S and T each of length at least q each. Moreover since C is an odd cycle one of these two paths is odd and the other is even. Without loss of generality, we can assume that S is a path of even length and T is a path of odd length. Let $Z_x := \{x\}$ and $Z_y := \{y\}$.

Below we define a procedure that iteratively grows the sets in (S, T, Z_x, Z_y) while maintaining the following invariants.

- The sets S, T and $Z_x \cup Z_y$ are pairwise disjoint.
- $G[S]$ is connected, bipartite and $|S| \geq q$.
- $G[T]$ is connected, bipartite and $|T| \geq q$.
- $G[S \cup T \cup \{y\}]$ is a certificate for the oct-minimality for Z_x (and in particular, $G[S \cup T \cup \{y\}]$ is bipartite).
- $G[S \cup T \cup \{x\}]$ is a certificate for the oct-minimality for Z_y (and in particular, $G[S \cup T \cup \{x\}]$ is bipartite).
- $N(x) \cap S \neq \emptyset, N(x) \cap T \neq \emptyset, N(y) \cap S \neq \emptyset$ and $N(y) \cap T \neq \emptyset$.

Observe that the starting sets (S, T, Z_x, Z_y) defined earlier satisfy these invariant. Note that in invariants 4 and 5, the sets $G[S \cup T \cup \{y\}]$ and $G[S \cup T \cup \{x\}]$, which serve as a certificate of minimality for Z_x and Z_y respectively, rely on the fact that C is an induced odd cycle. Since the iterative procedure grows these sets, the last property always hold. Also $G[S \cup T \cup \{x\} \cup \{y\}]$ contains the odd cycle C . The idea is to grow these sets until Z_x or Z_y has size at least k . If this happens then we can use Lemma 24, to conclude that G has a minimal oct of size at least k .

Since G is $(q, 2k)$ -unbreakable and $|S|, |T| \geq q$, we have that every ST -separator in G has size at least $2k + 1$. In particular, $|N(S) \cup N(T)| \geq 2k + 1$. Thus, if we guarantee that $(N(S) \cup N(T)) \subseteq (Z_x \cup Z_y)$, then $|Z_x \cup Z_y| \geq 2k + 1$. Hence either $|Z_x| \geq k$ or $|Z_y| \geq k$.

Towards this we grow the sets in (S, T, Z_x, Z_y) as follows. Let us call the vertices in $S \cup T \cup Z_x \cup Z_y$ as marked.

► **Claim 27.** Let $v \in N(S) \cap N(T)$ be an unmarked vertex. Either $G[S \cup T \cup \{y, v\}]$ or $G[S \cup T \cup \{x, v\}]$ contains an odd cycle.

Proof. Since both $G[S]$ and $G[T]$ are connected bipartite graphs, there exists a unique bipartition, say $S = A_S \cup B_S$ and $T = A_T \cup B_T$ of S and T respectively. Recall that x has neighbours in both the sets S and T . Since the graph $G[S \cup \{x\}]$ is bipartite, without loss of generality, assume that $(N(x) \cap S) \subseteq B_S$. Since $G[T \cup \{x\}]$ is also bipartite, without loss of generality assume that $N(x) \cap T \subseteq B_T$. Since the graph $G[S \cup T \cup \{x\}]$ is connected and bipartite, we know that there is a unique bipartition of $G[S \cup T \cup \{x\}]$ which is $G[S \cup T \cup \{x\}] = ((A_S \cup A_T \cup \{x\}) \cup (B_S \cup B_T))$.

Now, let us focus on the connected graph $G[S \cup T \cup \{y\}]$. Because $G[S \cup \{y\}]$ is bipartite, without loss of generality, we can assume that $(N(y) \cap S) \subseteq B_S$. We now show that $(N(y) \cap T) \subseteq A_T$. For the sake of contradiction, assume that this is not the case. This implies that there are two possibilities. If y has neighbours in both the sets A_T and B_T then it leads to a contradiction as $G[T \cup \{y\}]$ is bipartite. If the neighbours of y are contained in the set B_T then it leads to a contradiction as it would imply that $G[S \cup T \cup \{x, y\}]$ is bipartite.

Therefore, we get a unique bipartition $G[S \cup T \cup \{y\}] = ((A_S \cup B_T \cup \{y\}) \cup (B_S \cup A_T))$ of $G[S \cup T \cup \{y\}]$. As the vertex v has neighbours in both the sets S and T , there are four possibilities: (i) if v has a neighbour in A_S and A_T then $G[S \cup T \cup \{y, v\}]$ contains an odd cycle, (ii) if v has a neighbour in A_S and B_T then $G[S \cup T \cup \{x, v\}]$ contains an odd cycle, (iii) if v has a neighbour in A_T and B_S then $G[S \cup T \cup \{x, v\}]$ contains an odd cycle, or (iv) if v has a neighbour in A_T and B_T then $G[S \cup T \cup \{y, v\}]$ contains an odd cycle. This finishes the proof of the claim. \triangleleft

Let v be an arbitrarily chosen unmarked vertex.

Case 1: $v \in N(S) \cap N(T)$. From Lemma 27, either $G[S \cup T \cup \{x, v\}]$ or $G[S \cup T \cup \{y, v\}]$ or both contain an odd cycle. If $G[S \cup T \cup \{x, v\}]$ contain an odd cycle, then update $Z_y := Z_y \cup \{v\}$. If $G[S \cup T \cup \{y, v\}]$ contain an odd cycle, then update $Z_x := Z_x \cup \{v\}$. If both are true then update $Z_x := Z_x \cup \{v\}$ and $Z_y := Z_y \cup \{v\}$. The sets S, T remain the same. Observe that in either case, the updated sets satisfy all the invariants.

Case 2: $v \in N(S) \setminus N(T)$. In this case, if $G[S \cup \{v\}]$ is bipartite, then update $S := S \cup \{v\}$. The other sets T, Z_x, Z_y remain the same. Note that the updated sets (in particular S) maintains the invariant. Otherwise $G[S \cup \{v\}]$ contain an odd cycle. In this case update $Z_x := Z_x \cup \{v\}$ and $Z_y := Z_y \cup \{v\}$.

Case 3: $v \in N(T) \setminus N(S)$. This case is symmetric to Case 2.

We repeat the above process until we have sets (S, T, Z_x, Z_y) such that all vertices in $N(S) \cup N(T)$ are marked. In particular, in this case $(N(S) \cup N(T)) \subseteq (Z_x \cup Z_y)$. As argued earlier, this implies either $|Z_x| \geq k$ or $|Z_y| \geq k$. In both cases, we report that G has a minimal oct of size at least k from Lemma 24. \blacktriangleleft

Sufficient condition 2 [Large family of short induced odd cycles].

► **Lemma 28.** *Let (G, k) be an instance of MAXIMUM MINIMAL OCT. Let d be any positive integer. If \mathcal{F} is a family containing distinct induced odd cycles of G of length at most d and $|\mathcal{F}| > d(d!)(k-1)^d$ then G has a minimal oct of size at least k .*

Proof. From the Sunflower Lemma (Theorem 10), we can conclude that there exist at least k induced odd cycles $\{F_1, F_2, \dots, F_k\} \subseteq \mathcal{F}$ such that $V(F_i) \cap V(F_j) = Y$ for all $i, j \in [k]$. As the cycles in \mathcal{F} are induced odd cycles, the graph induced by the set of vertices in Y must be bipartite. Note that Y could possibly be empty. We claim that G has a minimal odd cycle transversal of size at least k . Such a minimal odd cycle transversal can be obtained by the greedy algorithm described in the proof of Lemma 24. We start with the induced

subgraph $G[Y]$. Clearly, any minimal odd cycle transversal obtained by this algorithm will contain at least one vertex from $V(F_i) \setminus Y$ for each $i \in [k]$. Since the sets $V(F_i) \setminus Y$ for each $i \in [k]$ are disjoint, a minimal odd cycle transversal must have a size of at least k . ◀

The combination lemma.

► **Lemma 29.** *Given a graph G , a vertex $x \in V(G)$ and positive integers d, k , there is an algorithm that runs in $(kd)^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$ time, and correctly outputs one of the following:*

1. *an induced odd cycle containing x ,*
2. *an induced odd cycle of length at least d ,*
3. *a family \mathcal{F} of distinct induced odd cycles, each of length at most $d - 1$, such that $|\mathcal{F}| \geq d \cdot d! \cdot (k - 1)^d$,*
4. *a determination that there is no induced odd cycle containing x in G .*

Proof. Suppose G contains an induced odd cycle containing x . Let C_x be one such cycle. We design an iterative algorithm that maintains a pair (G', \mathcal{F}) , where G' is an induced subgraph of G and \mathcal{F} is a family of induced odd cycles of length at most $d - 1$ in G , with the following additional properties. The graph G' is guaranteed to contain the cycle C_x , if C_x existed in the first place in G , and every induced odd cycle in G' is distinct from any cycle in the family \mathcal{F} .

Initialize $G' := G$ and $\mathcal{F} = \emptyset$. In each iteration, the algorithm finds an arbitrary induced odd cycle of G , say F , in polynomial time, if it exists. The following cases can now arise.

1. The algorithm fails to find the cycle F . That is G' has no induced odd cycle. In this case, report that G has no induced odd cycle passing through x .
This is correct because if G had such an induced odd cycle passing through x , then C_x exists in G and by the invariants of the algorithm C_x also exists in G' , which is a candidate for the cycle F .
2. If F contains x , then return F as the induced odd cycle containing x .
3. If the length of F is at least d , then return F as an induced odd cycle of length at least d in G .
4. Otherwise, F exists but does not contain x and has a length of at most $d - 1$.

By the invariants of the pair (G', \mathcal{F}) , F is distinct from all the cycles in \mathcal{F} . Update \mathcal{F} by adding F to it. Then the updated \mathcal{F} satisfies the required invariants.

To update G' proceed as follows. Guess the intersection of $V(F)$ with $V(C_x)$ (in G'). Let this be F' . The number of guesses is bounded by $2^{|F|} \leq 2^{d-1}$. Since F does not contain x , F is not equal to C_x . Hence, $F \setminus F'$ is non-empty. Update $G' := G' - (F \setminus F')$. Observe that the updated G' contains C_x if the old G_x contained it. Also every induced odd cycle in the updated G' is distinct from F (that has been newly added to \mathcal{F}) because a non-empty subset of $V(F)$ (that is $V(F) \setminus V(F')$) has been deleted in the updated G' .

If the first, second or third condition mentioned above does not apply in each of the first $d \cdot d! \cdot (k - 1)^d$ iterations, then at the end of the i -th iteration where $i = d \cdot d! \cdot (k - 1)^d$, $|\mathcal{F}| = d \cdot d! \cdot (k - 1)^d$. In this case, we return \mathcal{F} as a large family of induced odd cycles of length at most $d - 1$ of G .

This finishes the description and correctness of the algorithm. For the running time observe that in each iteration, the algorithm runs in polynomial time to find F and check the first three conditions. The fourth condition in each iteration makes 2^{d-1} guesses and the number of iterations is at most $d \cdot d! \cdot (k - 1)^d$. Therefore the overall running time is $(kd)^{\mathcal{O}(d)} \cdot n^{\mathcal{O}(1)}$. ◀

Proof of Theorem 25. The algorithm for Theorem 25 proceeds as follows. Recall that G is a (q, k) -unbreakable graph. For each $x \in V(G)$, run the algorithm of Lemma 29 on input $(G, x, 2q + 2, k)$.

If for some $x \in V(G)$, the algorithm returns an induced odd cycle of length at least $2q + 2$ or a family \mathcal{F} of distinct induced odd cycles of length at most $2q + 1$ such that $|\mathcal{F}| \geq (2q + 2)(2q + 2)!(k - 1)^{2q+2}$ then we return a yes instance due to Lemma 26 or 28, respectively. If the algorithm returns that there is no induced odd cycle containing x in G then we can delete x from G and solve the problem on the reduced graph. This is safe because of Observation 3. Finally, if the algorithm returns an induced odd cycle containing x of length at least $2q + 2$, then again report that it is a yes instance because of Lemma 26.

If none of the above conditions hold, then for each $x \in V(G)$, the algorithm of Lemma 29 returns an induced odd cycle containing x , say C_x , of length at most $2q + 1$. Note that the cycles returned for each $x \in V(G)$ in this case may not be distinct.

The following claim shows that if the number of vertices in G is large (and there is an induced odd cycle of small length for each vertex of the graph), then there exists a large family containing *distinct* induced odd cycles of small length in G , in which case we can report a yes instance by Lemma 28.

▷ **Claim 30.** If the number of vertices in G is at least $(2q + 2)^2(2q + 2)!(k - 1)^{2q+2} + 1$, then G contains a minimal oct of size at least k .

Proof. We will construct a \mathcal{F} of distinct induced odd cycles in G of length at most $2q + 1$. For every vertex $x \in V(G)$, we take a cycle C_x . Unless the cycle C_x is already in \mathcal{F} , we will update $\mathcal{F} = \mathcal{F} \cup \{C_x\}$. As the length of cycles in C_x returned by the above algorithm is bounded by $2q + 1$, if the number of vertices in G is more than $(2q + 1)(2q + 2)(2q + 2)!(k - 1)^{2q+2}$ then $|\mathcal{F}| \geq (2q + 2)(2q + 2)!(k - 1)^{2q+2}$. Due to Lemma 28, we return a yes instance. ◁

This finishes the proof of Theorem 25. ◀

5 Conclusion

In this paper, we established that both MAXIMUM MINIMAL st -SEPARATOR and MAXIMUM MINIMAL ODD CYCLE TRANSVERSAL (OCT) are fixed-parameter tractable parameterized by the solution size. Instead of using treewidth-based win-win approaches, we design FPT algorithms for highly unbreakable graphs for both these problems. While we have demonstrated the FPT nature of these problems, the challenge of developing efficient FPT algorithms remains open. Additionally, the edge-deletion version of the MAXIMUM MINIMAL OCT can be shown to be FPT using similar techniques, but with much simpler ideas. But designing an faster/explicit FPT algorithm even for this version remains an interesting direction for future research. Finally, the parameterized complexity of the weighted version of MAXIMUM MINIMAL st -SEPARATOR and MAXIMUM MINIMAL WEIGHT OCT remain open as very large weights cause problems while formulating the problem in CMSO, in order to reduce it to the unbreakable case.



References

- 1 Hassan AbouEisha, Shahid Hussain, Vadim V. Lozin, Jérôme Monnot, Bernard Ries, and Viktor Zamaraev. Upper domination: Towards a dichotomy through boundary properties. *Algorithmica*, 80(10):2799–2817, 2018. doi:10.1007/S00453-017-0346-9.

- 2 Júlio Araújo, Marin Bougeret, Victor Campos, and Ignasi Sau. Introducing lop-kernels: A framework for kernelization lower bounds. *Algorithmica*, 84(11):3365–3406, November 2022. doi:10.1007/s00453-022-00979-z.
- 3 Júlio Araújo, Marin Bougeret, Victor A. Campos, and Ignasi Sau. Parameterized complexity of computing maximum minimal blocking and hitting sets. *Algorithmica*, 85(2):444–491, September 2022. doi:10.1007/s00453-022-01036-5.
- 4 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. doi:10.1137/S0895480196305124.
- 5 Cristina Bazgan, Ljiljana Brankovic, Katrin Casel, Henning Fernau, Klaus Jansen, Kim-Manuel Klein, Michael Lampis, Mathieu Liedloff, Jérôme Monnot, and Vangelis Th. Paschos. The many facets of upper domination. *Theor. Comput. Sci.*, 717:2–25, 2018. doi:10.1016/J.TCS.2017.05.042.
- 6 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. *SIAM J. Discret. Math.*, 36(3):1761–1787, 2022. doi:10.1137/20M1385779.
- 7 Dan Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991. doi:10.1016/0012-365X(91)90098-M.
- 8 Nicolas Boria, Federico Della Croce, and Vangelis Th. Paschos. On the max min vertex cover problem. *Discrete Applied Mathematics*, 196:62–71, 2015. Advances in Combinatorial Optimization. doi:10.1016/j.dam.2014.06.001.
- 9 Juhi Chaudhary, Sounaka Mishra, and B. S. Panda. Minimum maximal acyclic matching in proper interval graphs. In Amitabha Bagchi and Rahul Muthu, editors, *Algorithms and Discrete Applied Mathematics - 9th International Conference, CALDAM 2023, Gandhinagar, India, February 9-11, 2023, Proceedings*, volume 13947 of *Lecture Notes in Computer Science*, pages 377–388. Springer, 2023. doi:10.1007/978-3-031-25211-2_29.
- 10 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016. doi:10.1137/15M1032077.
- 11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 12 Gabriel L. Duarte, Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Daniel Lokshantov, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, and Uéverton S. Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83(5):1421–1458, 2021. doi:10.1007/S00453-020-00789-1.
- 13 Louis Dublois, Tesshu Hanaka, Mehdi Khosravian Ghadikolaei, Michael Lampis, and Nikolaos Melissinos. (in)approximability of maximum minimal fvs. *Journal of Computer and System Sciences*, 124:26–40, 2022. doi:10.1016/j.jcss.2021.09.001.
- 14 Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. Upper dominating set: Tight algorithms for pathwidth and sub-exponential approximation. *Theor. Comput. Sci.*, 923:271–291, 2022. doi:10.1016/J.TCS.2022.05.013.
- 15 Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.
- 16 Fabio Furini, Ivana Ljubić, and Markus Sinnl. An effective dynamic programming algorithm for the minimum-cost maximal knapsack packing problem. *European Journal of Operational Research*, 262(2):438–448, 2017. doi:10.1016/j.ejor.2017.03.061.
- 17 Ajinkya Gaikwad and Soumen Maity. Parameterized complexity of upper edge domination. *CoRR*, abs/2208.02522, 2022. doi:10.48550/arXiv.2208.02522.
- 18 Laurent Gourvès, Jérôme Monnot, and Aris T. Pagourtzis. The lazy bureaucrat problem with common arrivals and deadlines: Approximation and mechanism design. In Leszek Gąsieniec and Frank Wolter, editors, *Fundamentals of Computation Theory*, pages 171–182, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40164-0_18.

- 19 Robert Haas and Michael Hoffmann. Chordless paths through three vertices. *Theoretical Computer Science*, 351(3):360–371, 2006. Parameterized and Exact Computation. doi:10.1016/j.tcs.2005.10.021.
- 20 Tesshu Hanaka, Hans L. Bodlaender, Tom C. van der Zanden, and Hirotaka Ono. On the maximum weight minimal separator. *Theoretical Computer Science*, 796:294–308, 2019. doi:10.1016/j.tcs.2019.09.025.
- 21 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.
- 22 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation ii: Undirected graphs. *ACM Trans. Algorithms*, 20(2), March 2024. doi:10.1145/3641105.
- 23 Michael Lampis. Minimum stable cut and treewidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 92:1–92:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.92.
- 24 Michael Lampis, Nikolaos Melissinos, and Manolis Vasilakis. Parameterized Max Min Feedback Vertex Set. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 62:1–62:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.MFCS.2023.62.
- 25 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. *ACM Trans. Algorithms*, 16(3):32:1–32:31, 2020. doi:10.1145/3379698.
- 26 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing CMSO model checking to highly connected graphs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPICs*, pages 135:1–135:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.135.
- 27 Dániel Marx, Barry O’sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4), October 2013. doi:10.1145/2500119.
- 28 Jérôme Monnot, Henning Fernau, and David F. Manlove. Algorithmic aspects of upper edge domination. *Theor. Comput. Sci.*, 877:46–57, 2021. doi:10.1016/J.TCS.2021.03.038.
- 29 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/J.ORL.2003.10.009.
- 30 Meirav Zehavi. Maximum minimal vertex cover parameterized by vertex cover. *SIAM Journal on Discrete Mathematics*, 31(4):2440–2456, 2017. doi:10.1137/16M109017X.
- 31 Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. Time-approximation trade-offs for inapproximable problems. *Journal of Computer and System Sciences*, 92:171–180, 2018. doi:10.1016/j.jcss.2017.09.009.

On the Existential Theory of the Reals Enriched with Integer Powers of a Computable Number

Jorge Gallego-Hernández  

IMDEA Software Institute, Madrid, Spain
Universidad Politécnica de Madrid, Spain

Alessio Mansutti  

IMDEA Software Institute, Madrid, Spain

Abstract

This paper investigates $\exists\mathbb{R}(\xi^{\mathbb{Z}})$, that is the extension of the existential theory of the reals by an additional unary predicate $\xi^{\mathbb{Z}}$ for the integer powers of a fixed computable real number $\xi > 0$. If all we have access to is a Turing machine computing ξ , it is not possible to decide whether an input formula from this theory is satisfiable. However, we show an algorithm to decide this problem when

- ξ is known to be transcendental, or
- ξ is a root of some given integer polynomial (that is, ξ is algebraic).

In other words, knowing the algebraicity of ξ suffices to circumvent undecidability. Furthermore, we establish complexity results under the proviso that ξ enjoys what we call a *polynomial root barrier*. Using this notion, we show that the satisfiability problem of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is

- in EXPSPACE if ξ is an algebraic number, and
- in 3EXP if ξ is a logarithm of an algebraic number, Euler’s e , or the number π , among others.

To establish our results, we first observe that the satisfiability problem of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ reduces in exponential time to the problem of solving quantifier-free instances of the theory of the reals where variables range over $\xi^{\mathbb{Z}}$. We then prove that these instances have a *small witness property*: only finitely many integer powers of ξ must be considered to find whether a formula is satisfiable. Our complexity results are shown by relying on well-established machinery from Diophantine approximation and transcendental number theory, such as bounds for the transcendence measure of numbers.

As a by-product of our results, we are able to remove the appeal to Schanuel’s conjecture from the proof of decidability of the entropic risk threshold problem for stochastic games with rational probabilities, rewards and threshold [Baier et al., *MFCS*, 2023]: when the base of the entropic risk is e and the aversion factor is a fixed algebraic number, the problem is (unconditionally) in EXP.

2012 ACM Subject Classification Computing methodologies → Symbolic and algebraic algorithms

Keywords and phrases Theory of the reals with exponentiation, decision procedures, computability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.37

Related Version *Extended Version*: <https://arxiv.org/abs/2502.02220>

Funding This work is part of a project that is partially funded by the Madrid Regional Government (César Nombela grant 2023-T1/COM-29001), and by MCIN/AEI/10.13039/501100011033/FEDER, EU (grant PID2022-138072OB-I00).

Acknowledgements We would like to thank Michael Benedikt and Dmitry Chistikov for the insightful discussions on the paper [Avigad and Yin, *Theor. Comput. Sci.*, 2007], and Andrew Scoones and James Worrell for providing guidance through the number theory literature. We are also grateful to the anonymous referees for comments and corrections.

1 Introduction

Tarski’s exponential function problem asks to determine the decidability of the validity problem from the first-order (FO) theory of the structure $(\mathbb{R}; 0, 1, +, \cdot, e^x, <, =)$. This theory, hereinafter denoted $\mathbb{R}(e^x)$, extends the FO theory of the reals (a.k.a. Tarski arithmetic) with



the exponential function $x \mapsto e^x$. A celebrated result by Macintyre and Wilkie establishes an affirmative answer to Tarski's problem conditionally to the truth of Schanuel's conjecture, a profound conjecture from transcendental number theory [24]. Recent years have seen this result being used as a black-box to establish conditional decidability results for numerous problems stemming from dynamical systems [14, 2] automata theory [15, 13], neural networks verification [19, 21], the theory of stochastic games [5], and differential privacy [7].

As it is often the case when appealing to a result as a black-box, some of the computational tasks resolved by relying on the work in [24] do not require the full power of $\mathbb{R}(e^x)$. Consequently, it is natural to ask whether some of these tasks can be tackled without relying on unproven conjectures, perhaps by reduction to tame fragments or variants of $\mathbb{R}(e^x)$. A few results align with this question:

- In the papers [3, 1, 28], Achatz, Anai, McCallum and Weispfenning introduce a procedure to decide sentences of the form $\exists x \exists y : y = \text{trans}(x) \wedge \varphi(x, y)$, where φ is a formula from Tarski arithmetic, and $x \mapsto \text{trans}(x)$ is any analytic and strongly transcendental function (see [28, Section 2] for the precise definition). Since $x \mapsto e^x$ enjoys such properties, this result shows a non-trivial fragment of $\mathbb{R}(e^x)$ that is unconditionally decidable. The procedure is implemented in the tool Redlog [16]. No complexity bound is known.
- In [17], van den Dries proves decidability of the extension of Tarski arithmetic with the unary predicate $2^{\mathbb{Z}}$ interpreted as the set $\{2^i : i \in \mathbb{Z}\}$, i.e., the set of all integer powers of 2. While this result is achieved by model-theoretic arguments, an effective quantifier elimination procedure was later given by Avigad and Yin [4]. Their procedure runs in TOWER, and in fact it requires non-elementary time already for the elimination of a single quantified variable. The choice of the base 2 for the integer powers is somewhat arbitrary: in [18], the decidability is extended to any fixed algebraic number (i.e., a number that is root of some polynomial equation; see Section 3 for background knowledge on computable, algebraic and transcendental numbers), and in fact Avigad and Yin's procedure is also effective for any such number. Considering any two $\alpha, \beta \in \mathbb{R}$ satisfying $\alpha^{\mathbb{Z}} \cap \beta^{\mathbb{Z}} = \{1\}$ yields undecidability, as shown by Hieronymi in [20].

When comparing the two lines of work discussed above, it becomes apparent that there is a balance to be struck between reasoning about transcendental numbers, the path followed by the first set of works, and developing algorithms that are well-behaved from a complexity standpoint, the path taken in particular in [4]. Our aim with this paper is to somewhat bridge this gap: we add to the second line of work by studying predicates for integer powers of bases that may be transcendental, all the while maintaining complexity upper bounds.

From now on, we write $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ to denote the existential fragment of the FO theory of the structure $(\mathbb{R}; 0, 1, \xi, +, \cdot, \xi^{\mathbb{Z}}, <, =)$, where $\xi > 0$ is a fixed real number. In this paper, we examine the complexity of deciding the satisfiability problem of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ for different choices of the number ξ . The following theorem summarises our results.

- **Theorem 1.** *Fix a real number $\xi > 0$. The satisfiability problem for $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is*
1. *in EXPSPACE whenever ξ is an algebraic number;*
 2. *in 3EXP if $\xi \in \{\pi, e^{\pi}, e^{\eta}, \alpha^{\eta}, \ln(\alpha), \frac{\ln(\alpha)}{\ln(\beta)} : \alpha, \beta, \eta \text{ algebraic with } \alpha > 0 \text{ and } 1 \neq \beta > 0\}$;*
 3. *decidable whenever ξ is a computable transcendental number.*

Theorem 1 has a catch, however. To be effective, the algorithm for deciding $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ requires:

- For Theorem 1.1, to have access to a canonical representation (see Section 3) of ξ .
- In the cases covered by Theorem 1.2, to have access to representations of α , β , and η .

- In the case of ξ computable transcendental number (Theorem 1.3), to have access to a Turing machine T that computes ξ (that is, given an input $n \in \mathbb{N}$ written in unary, T returns a rational number T_n such that $|\xi - T_n| \leq 2^{-n}$).

In summary, Theorem 1 shows that $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is decidable for every fixed computable number $\xi > 0$, as long as it is known whether ξ is algebraic or transcendental, and in the former case having access to a canonical representation of ξ .

The results in Theorem 1 are obtained by (i) reducing the satisfiability problem for $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ to the problem of solving instances of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ where all variables range over $\xi^{\mathbb{Z}}$, and (ii) showing that a solution over $\xi^{\mathbb{Z}}$ can be found by only looking at a “small” set of integer powers of ξ (a *small witness property*). In proving Step (ii), we also obtain a quantifier elimination procedure for *sentences* of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$, that is formulae where no variable occurs free. This procedure provides a partial answer to the question raised in [4] regarding the complexity of removing a single existential variable in Tarski arithmetic extended with $2^{\mathbb{Z}}$: within sentences of the existential fragment, such an elimination step can be performed in elementary time.

Coming back to our initial question on identifying computational tasks that might not need the full power of $\mathbb{R}(e^x)$, as a by-product of our results we show that the entropic risk threshold problem for stochastic games studied by Baier, Chatterjee, Meggendorfer and Piribauer [5] is unconditionally decidable in EXP even when the base of the entropic risk is e (or algebraic) and the aversion factor is any (fixed) algebraic number.

2 Approaching complexity bounds with root barriers

Theorems 1.1 and 1.2 are instances of a more general result concerning classes of computable real numbers. To properly introduce this result, it is beneficial to go back to Macintyre and Wilkie’s work on $\mathbb{R}(e^x)$. The exact statement made in [24] is that $\mathbb{R}(e^x)$ is decidable as soon as the following computational problem, implied by Schanuel’s conjecture, is established:

► **Conjecture 2.** *There is a procedure that for input $f_1, \dots, f_n, g \in \mathbb{Z}[x_1, \dots, x_n, e^{x_1}, \dots, e^{x_n}]$, with $n \geq 1$, returns a positive integer t with the following property: for every non-singular¹ solution $\alpha \in \mathbb{R}^n$ of the system of equalities $\bigwedge_{i=1}^n f_i(\mathbf{x}) = 0$, either $g(\alpha) = 0$ or $|g(\alpha)| > t^{-1}$.*

Above, $\mathbb{Z}[x_1, \dots, x_n, e^{x_1}, \dots, e^{x_n}]$ is the set of all n -variate exponential-polynomials with integer coefficients. As remarked in [24], t is guaranteed to exist by Khovanskii’s theorem [22], hence the crux of the problem concerns how to effectively compute such a number starting from f_1, \dots, f_n and g . The purpose of the dichotomy “either $g(\alpha) = 0$ or $|g(\alpha)| > t^{-1}$ ” is in part to resolve what is a fundamental problem when working with computable real numbers. Let α to be a vector of computable numbers. Consider the problem of establishing, given in input a polynomial p with integer coefficients, whether $p(\alpha)$ is positive, negative, or zero. This *polynomial sign evaluation* task is a well-known undecidable problem. Intuitively, the undecidability arises from the possibility that any approximation α^* of α might yield $p(\alpha^*) \neq 0$, even though $p(\alpha) = 0$. However, when working under the hypothesis that either $p(\alpha) = 0$ or $|p(\alpha)| > t^{-1}$, the problem becomes decidable: it suffices to compute an approximation α^* enjoying $|p(\alpha) - p(\alpha^*)| < (2t)^{-1}$, and then check whether $|p(\alpha^*)| \leq (2t)^{-1}$. If the answer is positive, then $p(\alpha) = 0$, otherwise $p(\alpha)$ and $p(\alpha^*)$ have the same sign.

¹ A solution α of $\bigwedge_{i=1}^n f_i(\mathbf{x}) = 0$ is said to be non-singular whenever the determinant of the $n \times n$ Jacobian matrix $\frac{\partial(f_1, \dots, f_n)}{\partial(x_1, \dots, x_n)}$ is, once evaluated at α , non-zero. We give this definition only for completeness of the discussion on Conjecture 2. It is not used in this paper.

The same issue occurs in $\exists\mathbb{R}(\xi^{\mathbb{Z}})$: under the sole hypothesis that ξ is computable, we cannot even check if $\xi = 2$ holds. However, what we can do is to draw some inspiration from Conjecture 2, and introduce as a further assumption the existence of what we call a *root barrier* of ξ . Below, $\mathbb{N}_{\geq 1} = \{1, 2, 3, \dots\}$, and given a polynomial p we write $\deg(p)$ for its *degree* and $h(p)$ for its *height* (i.e., the maximum absolute value of a coefficient of p).

► **Definition 3.** *A function $\sigma: (\mathbb{N}_{\geq 1})^2 \rightarrow \mathbb{N}$ is a root barrier of $\xi \in \mathbb{R}$ if for every non-constant polynomial $p(x)$ with integer coefficients, either $p(\xi) = 0$ or $\ln |p(\xi)| \geq -\sigma(\deg(p), h(p))$.*

To avoid non-elementary bounds on the runtime of our algorithms, we focus on computable numbers having root barriers $\sigma(d, h)$ that are polynomial expressions of the form $c \cdot (d + \lceil \ln h \rceil)^k$, where $c, k \in \mathbb{N}$ are some positive constants and $\lceil \cdot \rceil$ is the ceiling function. We call such functions *polynomial root barriers*, highlighting the fact that then $\sigma(\deg(p), h(p))$ in Definition 3 is bounded by a polynomial in the bit size of p . The aforesaid Theorem 1.2 is obtained by instantiating the following Theorem 4.2 to natural choices of ξ .

► **Theorem 4.** *Let $\xi > 0$ be a real number computable by a polynomial-time Turing machine, and let $\sigma(d, h) := c \cdot (d + \lceil \ln h \rceil)^k$ be a root barrier of ξ , for some $c, k \in \mathbb{N}_{\geq 1}$.*

1. *If $k = 1$, then the satisfiability problem for $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is in 2EXP.*
2. *If $k > 1$, then the satisfiability problem for $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is in 3EXP.*

As we will see in Section 6, whenever algebraic, the base ξ has a root barrier with exponent $k = 1$, and the related satisfiability problem for $\exists(\xi^{\mathbb{Z}})$ thus lie in 2EXP. However, a small trick will allow us to further improve this result to EXPSPACE, establishing Theorem 1.1.

3 Preliminaries

In this section, we fix our notation, introduce background knowledge on computable, algebraic and transcendental numbers, and define the existential theory $\exists\mathbb{R}(\xi^{\mathbb{Z}})$.

Sets, vectors, and basic functions. Given a finite set S , we write $|S|$ for its cardinality. Given $a, b \in \mathbb{R}$, we write $[a, b]$ for the closed interval $\{c \in \mathbb{R} : a \leq c \leq b\}$. We use parenthesis (and) for open intervals, hence writing, e.g., (a, b) for the set $\{c \in \mathbb{R} : a < c < b\}$. We write $[a..b]$ for the set of integers $[a, b] \cap \mathbb{Z}$. Given $A \subseteq \mathbb{R}$, $c \in \mathbb{R}$, and a binary relation \sim (e.g., \geq), we define $A_{\sim c} := \{a \in A : a \sim c\}$. The *endpoints* of A are its supremum and infimum, if they exist. For instance, the endpoints of the interval $[a, b]$ are the numbers a and b , while the endpoints of $[a..b]$ are the numbers $\lfloor a \rfloor$ and $\lfloor b \rfloor$, where $\lfloor \cdot \rfloor$ stands for the floor function.

Given a positive real number b with $b \neq 1$, we write $\log_b(\cdot)$ for the logarithm function of base b . We abbreviate $\log_2(\cdot)$ and $\log_e(\cdot)$ as $\log(\cdot)$ and $\ln(\cdot)$, respectively.

Unless stated explicitly, all integers encountered by our algorithms are encoded in binary; note that $n \in \mathbb{Z}$ can be represented using $1 + \lceil \log(n + 1) \rceil$ bits. Similarly, each rational is encoded as a ratio $\frac{n}{d}$ of two coprime integers n and d encoded in binary, with $d \geq 1$.

Integer polynomials. An *integer polynomial* in variables $\mathbf{x} = (x_1, \dots, x_n)$ is an expression $p(\mathbf{x}) := \sum_{j=1}^m (a_j \cdot \prod_{i=1}^n x_i^{d_{j,i}})$, where $a_j \in \mathbb{Z}$ and $d_{j,i} \in \mathbb{N}$ for every $j \in [1..m]$ and $i \in [1..n]$. In the context of algorithms, we assume the coefficients a_j to be given in binary encoding, and the exponents $d_{i,j}$ to be given in unary encoding. We rely on the following notions:

- The *height* of p , denoted $h(p)$, is defined as $\max\{|a_j| : j \in [1..m]\}$.
- The *degree* of p , denoted $\deg(p)$, is defined as $\max\{\sum_{i=1}^n d_{j,i} : j \in [1..m]\}$.
- Given $i \in [1..n]$, the *partial degree of p in x_i* , denoted $\deg(x_i, p)$, is $\max\{d_{j,i} : j \in [1..m]\}$.
- The *bit size* of p , denoted $\text{size}(p)$, is defined as $m \cdot (\lceil \log(h(p) + 1) \rceil + n \cdot \deg(p))$.

Computable numbers, and algebraic and transcendental numbers. A real number $\xi \in \mathbb{R}$ is said to be *computable* whenever there is a (deterministic) Turing machine $T: \mathbb{N} \rightarrow \mathbb{Q}$ that given in input $n \in \mathbb{N}$ written in unary (e.g., over the alphabet $\{1\}^*$) returns a rational number T_n (represented as described above) such that $|\xi - T_n| \leq 2^{-n}$. We thus have $\xi = \lim_{n \rightarrow \infty} T_n$, and for this reason ξ is said to be *computed by T* (or T *computes* ξ). The computable numbers form a field [31]; we will later need the following two statements regarding their closure under product and reciprocal.

► **Lemma 5.** *Given Turing machines T and T' computing reals a and b , one can construct a Turing machine T'' computing $a \cdot b$. If T and T' run in polynomial time, then so does T'' .*

► **Lemma 6.** *Given a Turing machine T computing a non-zero real number r , one can construct a Turing machine T' computing $\frac{1}{r}$. If T runs in polynomial time, then so does T' .*

A real number ξ is *algebraic* if it is a root of some univariate non-zero integer polynomial. Otherwise, ξ is *transcendental*. We often denote algebraic numbers by $\alpha, \beta, \eta, \dots$. Throughout the paper, we consider the following canonical representation: an algebraic number α is represented by a triple (q, ℓ, u) where q is a non-zero integer polynomial and ℓ, u are (representations of) rational numbers such that α is the only root of q belonging to $[\ell, u]$.

The existential theory $\exists\mathbb{R}(\xi^{\mathbb{Z}})$. Let $\xi > 0$ be a computable real number. We consider the structure $(\mathbb{R}; 0, 1, \xi, +, \cdot, \xi^{\mathbb{Z}}, <, =)$ extending the signature of the FO theory of the reals with the constant ξ and the unary *integer power* predicate $\xi^{\mathbb{Z}}$ interpreted as $\{\xi^i : i \in \mathbb{Z}\}$. Formulae from the existential theory of this structure, denoted $\exists\mathbb{R}(\xi^{\mathbb{Z}})$, are built from the grammar

$$\varphi, \psi ::= p(\xi, \mathbf{x}) \sim 0 \mid \xi^{\mathbb{Z}}(x) \mid \top \mid \perp \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x \varphi,$$

where \sim belongs to $\{<, =\}$, the argument x of the predicate $\xi^{\mathbb{Z}}(x)$ is a variable, and p is an integer polynomial involving ξ and variables \mathbf{x} . For convenience of notation, ξ is in this context seen as a variable of the polynomial p , so that we can rely on the previously defined notions of height, degree and bit size. We remark that, then, $h(p)$ is independent of ξ whereas $\deg(p)$ depends on the integers occurring as powers of ξ . The bit size of a formula φ , denoted as $\text{size}(\varphi)$, is the number of bits required to write down φ (where ξ is stored symbolically, using a constant number of symbols). Similarly, we write $\deg(\varphi)$ and $h(\varphi)$ for the maximum degree and height of polynomials occurring in φ , respectively.

The semantics of formulae from $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is standard; it is the one of the FO theory of the reals, plus a rule stating that $\xi^{\mathbb{Z}}(x)$ is true whenever x evaluates to a number in $\xi^{\mathbb{Z}}$. The grammar above features disjunctions (\vee), conjunctions (\wedge), true (\top) and false (\perp), but it does not feature negation (\neg) on top of atomic formulae. This restriction is w.l.o.g.: $\neg \xi^{\mathbb{Z}}(x)$ is equivalent to the formula $x \leq 0 \vee \exists y : \xi^{\mathbb{Z}}(y) \wedge y < x \wedge x < \xi \cdot y$ stating that x is either non-positive or strictly between two successive integer powers of ξ , whereas $\neg(p(\xi, \mathbf{x}) < 0)$ and $\neg(p(\xi, \mathbf{x}) = 0)$ are equivalent to $p(\xi, \mathbf{x}) = 0 \vee \neg p(\xi, \mathbf{x}) < 0$, and $p(\xi, \mathbf{x}) < 0 \vee \neg p(\xi, \mathbf{x}) < 0$, respectively. We still sometimes write negations in formulae, but these occurrences should be seen as shortcuts. The grammar also avoids polynomials in the scope of $\xi^{\mathbb{Z}}(\cdot)$, since $\xi^{\mathbb{Z}}(p(\xi, \mathbf{x}))$ is equivalent to $\exists y : y = p(\xi, \mathbf{x}) \wedge \xi^{\mathbb{Z}}(y)$. We write $\varphi \models \psi$ whenever φ *entails* ψ .

4 An algorithm for deciding $\exists\mathbb{R}(\xi^{\mathbb{Z}})$

Fix a computable number $\xi > 0$ that is either transcendental or has a polynomial root barrier. In this section, we discuss our procedure for deciding the satisfiability of formulae in $\exists\mathbb{R}(\xi^{\mathbb{Z}})$. For simplicity, we assume for now $\xi > 1$. The general case of $\xi > 0$ is handled in Section 4.5.

■ **Algorithm 1** A procedure deciding the satisfiability problem for $\exists\mathbb{R}(\xi^{\mathbb{Z}})$.

Fixed: $\xi > 1$ computable number that is transcendental or has a polynomial root barrier.

Input: $\varphi(x_1, \dots, x_n)$: quantifier-free formula from $\exists\mathbb{R}(\xi^{\mathbb{Z}})$.

Output: True (\top) if φ is satisfiable, and otherwise False (\perp).

```

1: for  $i \in [1..n]$  do
2:   let  $u_i$  and  $v_i$  be two fresh variables
3:   update  $\varphi$ : replace every occurrence of  $\xi^{\mathbb{Z}}(x_i)$  with  $v_i = 1$ 
4:   update  $\varphi$ : replace every occurrence of  $x_i$  with  $u_i \cdot v_i$ 
5:    $\varphi \leftarrow \varphi \wedge (v_i = 0 \vee 1 \leq |v_i| < \xi)$ 
6:  $\psi(u_1, \dots, u_n) \leftarrow \text{REALQE}(\exists v_1 \dots \exists v_n : \varphi)$   $\triangleright$  eliminate  $v_1, \dots, v_n$  (see Theorem 7)
7: for  $i \in [1..n]$  do  $\triangleright g_i$  below is encoded in unary
8:   guess  $g_i \leftarrow$  an element of  $P_\psi$   $\triangleright P_\psi \subseteq \mathbb{Z}$  is the set from in Proposition 8
9: return evaluate whether the assignment  $(u_1 = \xi^{g_1}, \dots, u_n = \xi^{g_n})$  is a solution to  $\psi$ 

```

■ **Algorithm 2** Algorithm for solving SIGN_ξ when ξ has a root barrier.

Fixed: A number $\xi \in \mathbb{R}$ computed by a Turing machine T and having a root barrier σ .

Input: A univariate integer polynomial $p(x)$ of degree d and height h .

Output: The symbol \sim from $\{<, >, =\}$ such that $p(\xi) \sim 0$.

```

1:  $n \leftarrow 1 + 2\sigma(d, h) + 3d \lceil \log(h + 4) \rceil$ 
2: if  $|p(T_n)| \leq 2^{-2\sigma(d, h)-1}$  and  $|T_n| < h + 2$  then return the symbol =
3: else return the sign of  $p(T_n)$ 

```

The pseudocode of the procedure is given in Algorithm 1. To keep it as simple as possible, we use nondeterminism in line 8 instead of implementing, e.g., a routine backtracking algorithm. The procedure assumes the input formula $\varphi(x_1, \dots, x_n)$ to be quantifier-free (this is without loss of generality, since $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is an existential theory), and it is split into three steps, which we discuss in the forthcoming three subsections.

4.1 Step I (lines 1–6): reducing the variables to integer powers of ξ

The first step reduces the problem of finding a solution over \mathbb{R} to the problem of finding a solution over $\xi^{\mathbb{Z}}$. Below, we denote by $\exists\xi^{\mathbb{Z}}$ the existential theory of the structure $(\xi^{\mathbb{Z}}; 0, 1, \xi, +, \cdot, <, =)$. Formulae from this theory are built from the grammar of $\exists\mathbb{R}(\xi^{\mathbb{Z}})$, except they do not feature predicates $\xi^{\mathbb{Z}}(x)$, as they are now trivially true.

For reducing $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ to $\exists\xi^{\mathbb{Z}}$, we observe that every $x \in \mathbb{R}$ can be factored as $u \cdot v$ where u belongs to $\xi^{\mathbb{Z}}$ and v is either 0 (if $x = 0$) or it belongs, in absolute value, to the interval $[1, \xi)$. In the case of $x \neq 0$, this factorisation is unique, and u corresponds to the largest element of $\xi^{\mathbb{Z}}$ that is less or equal to the absolute value of x , i.e., $u \leq |x| < \xi \cdot u$. The procedure uses this fact to replace every occurrence of a variable x_i in the input formula $\varphi(x_1, \dots, x_n)$ with two fresh variables u_i and v_i (see the **for** loop of line 1), where v_i is set to satisfy either $v_i = 0$ or $1 \leq |v_i| < \xi$ (the latter is short for the formula $(1 \leq v_i < \xi) \vee (-\xi < v_i \leq -1)$), and u_i is (implicitly) assumed to belong to $\xi^{\mathbb{Z}}$. This allows to replace all occurrences of the predicate $\xi^{\mathbb{Z}}(x_i)$ with $v_i = 1$ (line 3). We obtain in this way an equivalent formula from the existential theory of the reals, but where the variables u_1, \dots, u_n are assumed to range over $\xi^{\mathbb{Z}}$.

After the updates performed by the **for** loop, the procedure eliminates the variables v_1, \dots, v_n by appealing to a quantifier elimination procedure for the FO theory of the reals, named REALQE in the pseudocode. We remind the reader that a quantifier elimination procedure is an algorithm that, from an input (quantified) formula, produces an *equivalent* quantifier-free formula. Since such a procedure preserves formula equivalence, we can use it to eliminate v_1, \dots, v_n even if u_1, \dots, u_n are assumed to range over $\xi^{\mathbb{Z}}$. The constant ξ appearing in the formula is treated as an additional free variable by REALQE. The output formula $\psi(u_1, \dots, u_n)$ belongs to $\exists \xi^{\mathbb{Z}}$, as required. This concludes the first step of the algorithm.

To perform the quantifier elimination step, we rely on the quantifier elimination procedure for the (full) FO theory of the reals developed by Basu, Pollack and Roy [8]. This procedure achieves the theoretically best-known bounds for the output formula, not only for arbitrary quantifier alternation but also for the existential fragment (i.e., when taking $\omega = 1$ below).

► **Theorem 7** ([8, Theorem 1.3.1]). *There is an algorithm with the following specification:*

Input: A formula $\varphi(\mathbf{y})$ from the first-order theory of $(\mathbb{R}; 0, 1, +, \cdot, <, =)$.

Output: A quantifier-free formula $\gamma(\mathbf{y}) = \bigvee_{i=1}^I \bigwedge_{j=1}^J p_{i,j}(\mathbf{y}) \sim_{i,j} 0$ equivalent to φ , where every $\sim_{i,j}$ is from $\{<, =\}$.

Suppose the input formula φ to be of the form $Q_1 \mathbf{x}_1 \in \mathbb{R}^{n_1} \dots Q_\omega \mathbf{x}_\omega \in \mathbb{R}^{n_\omega} : \psi(\mathbf{y}, \mathbf{x}_1, \dots, \mathbf{x}_\omega)$, where $\mathbf{y} = (y_1, \dots, y_k)$, every Q_i is \exists or \forall , and ψ is a quantifier-free formula with m atomic formulae $g_i \sim 0$ satisfying $\deg(g_i) \leq d$ and $h(g_i) \leq h$. Then, the output formula γ satisfies

$$\begin{aligned} I &\leq (m \cdot d + 1)^{(k+1)\Pi_{i=1}^\omega O(n_i)}, & \deg(p_{i,j}) &\leq d^{\Pi_{i=1}^\omega O(n_i)}, \\ J &\leq (m \cdot d + 1)^{\Pi_{i=1}^\omega O(n_i)}, & h(p_{i,j}) &\leq (h + 1)^{d^{(k+1)\Pi_{i=1}^\omega O(n_i)}}, \end{aligned}$$

and the algorithm runs in time $\text{size}(\varphi)^{O(1)}(m \cdot d + 1)^{(k+1)\Pi_{i=1}^\omega O(n_i)}$.

4.2 Step II (lines 7 and 8): solving $\exists \xi^{\mathbb{Z}}$

The second step of the procedure searches for a solution to the quantifier-free formula ψ in line 6. For every variable u_i in ψ , the algorithm guesses an integer g_i , encoded in unary, from a finite set P_ψ . Implicitly, this guess is setting $u_i = \xi^{g_i}$. The next proposition shows that P_ψ can be computed from ψ and the base ξ , i.e., $\exists \xi^{\mathbb{Z}}$ has a *small witness property*.

► **Proposition 8.** *Fix $\xi > 1$. There is an algorithm with the following specification:*

Input: A quantifier-free formula $\psi(u_1, \dots, u_n)$ from $\exists \xi^{\mathbb{Z}}$.

Output: A finite set $P_\psi \subseteq \mathbb{Z}$ such that ψ is satisfiable if and only if ψ has a solution in the set $\{(\xi^{j_1}, \dots, \xi^{j_n}) : j_1, \dots, j_n \in P_\psi\}$.

To be effective, the algorithm requires knowing either that ξ is a computable transcendental number, or two integers $c, k \in \mathbb{N}_{\geq 1}$ for which $\sigma(d, h) := c \cdot (d + \lceil \ln(h) \rceil)^k$ is a root barrier of ξ . In the latter case, the elements in P_ψ are bounded in absolute value by $(2^c \lceil \ln(H) \rceil)^{D^{2^5 n^2 k^{D^{8n}}}}$, where $H := \max(8, h(\psi))$ and $D := \deg(\psi) + 2$.

We defer a sketch of the proof of Proposition 8 (perhaps the main technical contribution of the paper) to Section 5. Note that the bound on P_ψ given in the final statement of Proposition 8 is in general triply exponential in $\text{size}(\psi)$, but it becomes doubly exponential if the root barrier σ is such that $k = 1$. The two statements in Theorem 4 stem from this distinction.

4.3 Step III (line 9): polynomial sign evaluation

The last step of the procedure checks if the assignment $u_1 = \xi^{g_1}, \dots, u_n = \xi^{g_n}$ is a solution to $\psi(u_1, \dots, u_n)$. Observe that $\psi(\xi^{g_1}, \dots, \xi^{g_n})$ is a Boolean combination of polynomial (in)equalities $p(\xi) \sim 0$, where ξ may occur with negative powers (as some g_i may be negative). This is unproblematic, as one can make all powers non-negative by rewriting each (in)equality $p(\xi) \sim 0$ as $\xi^{-d} \cdot p \sim 0$, where d is the smallest negative integer occurring as a power of ξ in p (or 0 if such an integer does not exist). After this small update, line 9 boils down to determining the sign that each polynomial in the formula has when evaluated at ξ . This enables us to simplify all inequalities to either \top or \perp , to then return \top or \perp depending on the Boolean structure of ψ . Let us thus focus on the required sign evaluation problem, which we denote by SIGN_ξ . Its specification is the following:

Input: A univariate integer polynomial $p(x)$.

Output: The symbol \sim from $\{<, >, =\}$ such that $p(\xi) \sim 0$.

Solving SIGN_ξ when $\xi \in \mathbb{R}$ is transcendental. It is a standard fact that SIGN_ξ becomes solvable whenever ξ is any computable transcendental number. Indeed, in this case $p(\xi)$ must be different from 0, and one can rely on the fast-convergence sequence of rational numbers T_0, T_1, \dots to find $n \in \mathbb{N}$ such that $|p(\xi) - p(T_n)|$ is guaranteed to be less than $|p(T_n)|$. The sign of $p(\xi)$ then agrees with the sign of $p(T_n)$, and the latter can be easily computed. In general, the asymptotic running time of this algorithm cannot be bounded.

Solving SIGN_ξ when $\xi \in \mathbb{R}$ has a (polynomial) root barrier. A similar algorithm as the one given for transcendental numbers can be defined for numbers with a polynomial root barrier; and in this case its running time can be properly analysed. The pseudocode of such a procedure is given in Algorithm 2, and it should be self-explanatory. We stress that running this algorithm requires access to the root barrier σ and the Turing machine T .

► **Lemma 9.** *Algorithm 2 respects its specification.*

Proof sketch. If $|T_n| \geq h + 2$, then $p(\xi)$ and $p(T_n)$ have the same sign, because $h + 1$ is an upper bound to the absolute value of every root of $p(x)$ [30, Chapter 8]. If $|T_n| < h + 2$ instead, by studying the derivative of p in the interval $[-(h + 3), h + 3]$ containing ξ , one finds $|p(\xi) - p(T_n)| \leq 2^{-2\sigma(d,h)-1}$, with n defined as in line 1. Then, either $|p(T_n)| \leq 2^{-2\cdot\sigma(d,h)-1}$ and $p(\xi) = 0$, or $|p(T_n)| > 2^{-2\cdot\sigma(d,h)-1}$ and $p(\xi)$ and $p(T_n)$ have the same sign. ◀

When σ is a polynomial root barrier, the integer n from line 1 can be written in unary using polynomially many digits with respect to $\text{size}(p)$. This yields the following lemma.

► **Lemma 10.** *Let $\xi \in \mathbb{R}$ be a number computed by a Turing machine T and having a polynomial root barrier σ . If T runs in polynomial time, then so does Algorithm 2.*

4.4 Correctness and running time of Algorithm 1

Since lines 1–5 preserve the satisfiability the input formula, by chaining Theorem 7, Proposition 8, and Lemma 9, we conclude that Algorithm 1 is correct.

► **Lemma 11.** *Algorithm 1 respects its specification.*

This establishes Theorem 1.3 restricted to bases $\xi > 1$. Analogously, when ξ is a number with a polynomial root barrier $\sigma(d, h) := c \cdot (d + \lceil \log_e h \rceil)^k$, by pairing Lemma 11 with a complexity analysis of Algorithm 1, one shows Theorem 4 restricted to bases $\xi > 1$. In performing this analysis, we observe that the bottleneck of the procedure is given by the guesses of the integers g_i performed lines 7 and 8. The absolute value of these integers is either doubly or triply exponential in the size of the input formula φ , depending on whether $k = 1$. A deterministic implementation of the procedure can iterate through all their values in doubly or triply exponential time.

4.5 Handling small bases

We now extend our algorithm so that it works assuming $\xi > 0$ instead of just $\xi > 1$, hence completing the proofs of Theorem 1.3 and Theorem 4. Let ξ be computable and either transcendental or with a polynomial root barrier. First, observe that we can call the procedure for SIGN_ξ on input $x - 1$ in order to check if $\xi \in (0, 1)$, $\xi = 1$ or $\xi > 1$.

If $\xi = 1$, we replace in the input formula φ every occurrence of $\xi^{\mathbb{Z}}(x)$ with $x = 1$, obtaining a formula from the existential theory of the reals, which we can solve by Theorem 7. If $\xi > 1$, we call Algorithm 1. Suppose then $\xi \in (0, 1)$. In this case, we replace every occurrence of $\xi^{\mathbb{Z}}(x)$ with $(\frac{1}{\xi})^{\mathbb{Z}}(x)$, and opportunely multiply by integer powers of $\frac{1}{\xi}$ both sides of polynomial inequalities in order to eliminate the constant ξ . In this way, we obtain from φ an equivalent formula in $\exists \mathbb{R}((\frac{1}{\xi})^{\mathbb{Z}})$. Since $\frac{1}{\xi} > 1$, we can now call Algorithm 1; provided we first establish the properties of $\frac{1}{\xi}$ required to run this algorithm. These properties indeed hold:

1. If ξ is transcendental, then so is $\frac{1}{\xi}$. This is because the algebraic numbers form a field.
2. If ξ has a polynomial root barrier σ , then σ is also a root barrier of $\frac{1}{\xi}$. Indeed, consider an integer polynomial $p(x) = \sum_{i=0}^d a_i \cdot x^i$ with height h , and assume $p(\frac{1}{\xi}) \neq 0$. Since σ is a root barrier of ξ , we have $\xi^d \cdot |p(\frac{1}{\xi})| = |\sum_{i=0}^d a_i \cdot \xi^{d-i}| \geq e^{-\sigma(h, d)}$, which in turns implies that $|p(\frac{1}{\xi})| \geq e^{-\sigma(h, d)} \cdot \xi^{-d} \geq e^{-\sigma(h, d)}$, where the last inequality uses $\frac{1}{\xi} \geq 1$.
3. From a Turing machine T computing ξ , we can construct a Turing machine T' computing $\frac{1}{\xi}$. Lemma 6 gives this construction, and shows that T' runs in polynomial time if so does T .

5 Finding solutions over integer powers of ξ

In this section we give a sketch of the proof of Proposition 8, i.e., we show that $\exists \xi^{\mathbb{Z}}$ has a *small witness property*. The proof is split into two parts:

1. We first give a quantifier-elimination-like procedure for $\exists \xi^{\mathbb{Z}}$. Instead of targeting formula equivalence, we only focus on equisatisfiability: given a formula $\exists y \varphi(y, \mathbf{x})$, with φ quantifier-free, the procedure derives an *equisatisfiable* quantifier-free formula $\psi(\mathbf{x})$. Preserving equisatisfiability, instead of equivalence, is advantageous complexity-wise. (Our procedure preserves equivalence for sentences, as these are equivalent to \top or \perp .)
2. By analysing our quantifier elimination procedure, we derive the bounds on the set P_ψ from Proposition 8 that are required to complete the proof. This step is similar to the *quantifier relativisation* technique for Presburger arithmetic (see, e.g., [34, Theorem 2.2]).

Some of the core mechanisms of our quantifier-elimination-like procedure follow observations done by Avigad and Yin for their (equivalence-preserving) quantifier elimination procedure [4]. Apart from targeting equisatisfiability, a key property of our procedure is that it does not require appealing to a quantifier elimination procedure for the theory of the reals. The procedure in [4] calls such a procedure once for each eliminated variable instead.

5.1 Quantifier elimination

Fix a real number $\xi > 1$. In this section, we rely on some auxiliary notation and definitions:

- We often see an integer polynomial $p(\xi, \mathbf{x})$ as a polynomial in variables $\mathbf{x} = (x_1, \dots, x_m)$ having as coefficients univariate integer polynomials on ξ , i.e., $p(\xi, \mathbf{x}) = \sum_{i=1}^n q_i(\xi) \cdot \mathbf{x}^{\mathbf{d}_i}$, where the notation $\mathbf{x}^{\mathbf{d}_i}$ is short for the monomial $\prod_{j=1}^m x_j^{d_{i,j}}$, with $\mathbf{d}_i = (d_{i,1}, \dots, d_{i,m})$.
- We sometimes write polynomial (in)equalities using Laurent polynomials, i.e., polynomials with negative powers. For instance, Lemma 12 below features equalities with monomials $\xi^g \cdot \mathbf{x}^{\mathbf{d}_i}$ where g may be a negative integer. Laurent polynomials are just a shortcut for us, as one can opportunely manipulate the (in)equalities to make all powers non-negative (as we did in Section 4.3): a polynomial (in)equality $p(\xi, x_1, \dots, x_m) \sim 0$ is rewritten as $p(\xi, x_1, \dots, x_m) \cdot \xi^{-d} \cdot x_1^{-d_1} \cdot \dots \cdot x_m^{-d_m} \sim 0$, where d_i (resp. d) is the smallest negative integer occurring as a power of x_i (resp. ξ) in p (or 0 if such a negative integer does not exist). Observe that this transformation does not change the number of monomials nor the height of the polynomial p , but it may double the degree of each variable and of ξ .
- Given a formula φ , a variable x and a Laurent polynomial $q(\mathbf{y})$, we write $\varphi[q(\mathbf{y})/x]$ for the formula obtained from φ by replacing every occurrence of x by $q(\mathbf{y})$, and then updating all polynomial (in)equalities with negative degrees in the way described above.
- We write $\lambda: \mathbb{R}_{>0} \rightarrow \xi^{\mathbb{Z}}$ for the function mapping $a \in \mathbb{R}_{>0}$ to the largest integer power of ξ that is less or equal than a , i.e., $\lambda(a)$ is the only element of $\xi^{\mathbb{Z}}$ satisfying $\lambda(a) \leq a < \xi \cdot \lambda(a)$.

The relation $\lambda(p(\xi, \mathbf{x})) = y$, where p is an integer polynomial, is definable in $\exists \xi^{\mathbb{Z}}$ as $p(\xi, \mathbf{x}) > 0 \wedge y \leq p(\xi, \mathbf{x}) < \xi \cdot y$. To obtain a quantifier elimination procedure, we must first understand what values can y take given $p(\xi, \mathbf{x})$. The next lemma answers this question.

► **Lemma 12.** *Let $p(\xi, \mathbf{x}) := \sum_{i=1}^n (q_i(\xi) \cdot \mathbf{x}^{\mathbf{d}_i})$, where each q_i is a univariate integer polynomial. In the theory $\exists \xi^{\mathbb{Z}}$, the formula $p(\xi, \mathbf{x}) > 0$ entails the formula $\bigvee_{i=1}^n \bigvee_{g \in G} \lambda(p(\xi, \mathbf{x})) = \xi^g \cdot \mathbf{x}^{\mathbf{d}_i}$, for some finite set $G \subseteq \mathbb{Z}$. Moreover:*

- I. *If ξ is a computable transcendental number, there is an algorithm computing G from p .*
- II. *If ξ has a root barrier $\sigma(d, h) := c \cdot (d + \lceil \ln(h) \rceil)^k$, for some $c, k \in \mathbb{N}_{\geq 1}$, then*

$$G := [-L..L], \quad \text{where } L := (2^{3c} D \lceil \ln(H) \rceil)^{6nk^{3n}},$$

with $H := \max\{8, h(q_i) : i \in [1, n]\}$, and $D := \max\{\deg(q_i) + 2 : i \in [1, n]\}$.

Proof sketch. A suitable set G can be found as follows. Let \mathcal{Q} be the set of all univariate integer polynomials $Q(z)$ for which there are $j \leq \ell \in [1..n]$, numbers $g_j, \dots, g_{\ell-1} \in \mathbb{N}$, and integer polynomials $Q_j(z), \dots, Q_{\ell}(z)$ such that $Q_{\ell} = Q$ and

1. the polynomials Q_j, \dots, Q_{ℓ} are recursively defined as

$$\begin{aligned} Q_j(z) &:= q_j(z), \\ Q_r(z) &:= Q_{r-1}(z) \cdot z^{g_{r-1}} + q_r(z), \end{aligned} \quad \text{for every } r \in [j+1, \ell],$$

2. the real numbers $Q_j(\xi), \dots, Q_{\ell-1}(\xi)$ are all non-zero, and $Q_{\ell}(\xi)$ is (strictly) positive,
 3. for every $r \in [j.. \ell-1]$, the number ξ^{g_r} belongs to the interval $[1, \frac{|q_{r+1}(\xi)| + \dots + |q_n(\xi)|}{|Q_r(\xi)|}]$.
- Items 1–3 ensure the set \mathcal{Q} to be finite. We define the (finite) set

$$B := \left\{ \beta \in \mathbb{Z} : \text{there is } Q \in \mathcal{Q} \text{ such that } \xi^{\beta} \in \left\{ \lambda(Q(\xi)), \frac{\lambda(Q(\xi) \cdot (\xi-1))}{\xi}, \frac{\lambda(Q(\xi) \cdot (\xi+1))}{\xi} \right\} \right\}.$$

By induction on n , one can prove that any finite set G that includes $[\min B.. \max B]$ respects the property in the first statement of the lemma. To prove the remaining statements of the

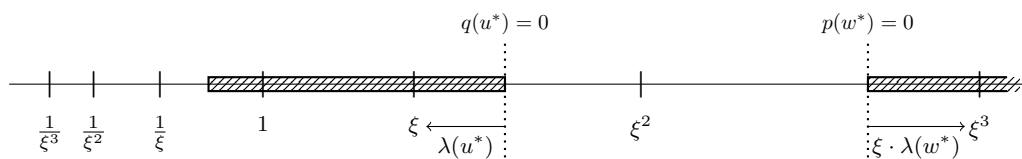


Figure 1 High-level idea of the quantifier elimination procedure. Dashed rectangles are intervals corresponding to the set of solutions over \mathbb{R} of a (univariate) formula φ . To search for a solution over $\xi^{\mathbb{Z}}$, it suffices to look for elements of $\xi^{\mathbb{Z}}$ that are close to the endpoints of these intervals. At each endpoint, a polynomial in φ must evaluate to zero (since around endpoints the truth of φ changes), so it suffices to look for integer powers of ξ that are close to roots or polynomials in φ .

lemma (Items (I) and (II)) one shows how to effectively compute an overapproximation of the set B . In the case of ξ having a polynomial root barrier, this overapproximation is obtained by bounding the values of $\lambda(Q(\xi))$, $\frac{\lambda(Q(\xi) \cdot (\xi - 1))}{\xi}$, and $\frac{\lambda(Q(\xi) \cdot (\xi + 1))}{\xi}$, for every $Q \in \mathcal{Q}$. ◀

We now give the high-level idea of the quantifier elimination procedure, which is also depicted in Figure 1. Let $\psi(u, \mathbf{y})$ be a quantifier-free formula of $\exists \xi^{\mathbb{Z}}$, and u be the variable we want to eliminate. Suppose to evaluate the variables \mathbf{y} with elements in $\xi^{\mathbb{Z}}$, hence obtaining a univariate formula $\varphi(u)$. The set of all solutions *over the reals* of $\varphi(u)$ can be decomposed into a finite set of disjoint intervals. (This follows from the o-minimality of the FO theory of the reals [26, Chapter 3.3].) Figure 1 shows these intervals as dashed rectangles. Around the endpoints of these intervals the truth of φ changes, and therefore for each such endpoint u^* there must be a non-constant polynomial in φ such that $q(u^*) = 0$. If an interval with endpoint $u^* \in \mathbb{R}_{>0}$ contains an element of $\xi^{\mathbb{Z}}$, then it contains one that is “close” to u^* :

- If $u^* \in \mathbb{R}_{>0}$ is the *right endpoint* of an interval, at least one among $\lambda(u^*)$ and $\xi^{-1} \cdot \lambda(u^*)$ belongs to the interval. The first case is depicted in Figure 1. The latter case occurs when u^* belongs to $\xi^{\mathbb{Z}}$ but not to the interval.
- If u^* is the *left endpoint* of an interval, then $\xi \cdot \lambda(u^*)$ or $\lambda(u^*)$ belongs to the interval. The latter case occurs when u^* belongs to $\xi^{\mathbb{Z}}$ and also to the interval.

Note that we have restricted the endpoint u^* to be positive, so that $\lambda(u^*)$ is well-defined. The only case were we may not find such an endpoint is when $\varphi(u)$ is true for every $u > 0$. But finding an element of $\xi^{\mathbb{Z}}$ is in this case simple: we can just pick $1 \in \xi^{\mathbb{Z}}$. Since u^* is positive, we can split it into $x^* \cdot v^*$ with $x^* \in \xi^{\mathbb{Z}}$ and $1 \leq v^* < \xi$ (so, $\lambda(u^*) = x^*$). To obtain quantifier elimination, our goal is then to characterise, symbolically as a finite set of polynomials $\tau(\mathbf{y})$, the set of all possible values for x^* . In this way, we will be able to eliminate the variable u by considering the polynomials $\xi^{-1} \cdot \tau(\mathbf{y})$, $\tau(\mathbf{y})$ and $\xi \cdot \tau(\mathbf{y})$ representing the integer powers of ξ that are “close” to endpoints. The following lemma provides the required characterisation.

► **Lemma 13.** Let $r(x, v, \mathbf{y}) := \sum_{i=0}^n p_i(\xi, \mathbf{y}) \cdot (x \cdot v)^i$, where each p_i is an integer polynomial, M be the set of monomials \mathbf{y}^ℓ occurring in some p_i , and $N := \{\mathbf{y}^{\ell_1 - \ell_2} : \mathbf{y}^{\ell_1}, \mathbf{y}^{\ell_2} \in M\}$. Then,

$$\xi^{\mathbb{Z}}(x) \wedge 1 \leq v < \xi \wedge r(x, v, \mathbf{y}) = 0 \wedge \left(\bigvee_{i=0}^n p_i(\xi, \mathbf{y}) \neq 0 \right) \wedge \bigwedge_{\mathbf{y} \text{ from } \mathbf{y}} \xi^{\mathbb{Z}}(\mathbf{y}) \models \bigvee_{(j, g, \mathbf{y}^\ell) \in F} x^j = \xi^g \cdot \mathbf{y}^\ell$$

holds (in the theory $\exists \mathbb{R}(\xi^{\mathbb{Z}})$) for some finite set $F \subseteq [1..n] \times \mathbb{Z} \times N$. Moreover:

1. If ξ is a computable transcendental number, there is an algorithm computing F from r .
- II. If ξ has a root barrier $\sigma(d, h) := c \cdot (d + \lceil \ln(h) \rceil)^k$, for some $c, k \in \mathbb{N}_{\geq 1}$, then,

$$F := [1..n] \times [-L..L] \times N, \quad \text{where } L := n \left(2^{4cD} \lceil \ln(H) \rceil \right)^{6|M| \cdot k^{31|M|}},$$

with $H := \max\{8, h(p_i) : i \in [1, n]\}$, and $D := \max\{\deg(\xi, p_i) + 2 : i \in [0, n]\}$.

37:12 On the Existential Theory of the Reals with Integer Powers of a Computable Number

Proof sketch. By following the arguments in [4, Lemma 3.9], one shows that the premise of the entailment in the statement entails a disjunction over formulae of the form

$$x^{k-j} = \frac{\xi^s \cdot \lambda(\pm p_j(\xi, \mathbf{y}))}{\lambda(\mp p_k(\xi, \mathbf{y}))} \wedge \pm p_j(\xi, \mathbf{y}) > 0 \wedge \mp p_k(\xi, \mathbf{y}) > 0,$$

where $0 \leq j < k \leq n$, $s \in [-g..g]$ with $g := 1 + \lceil \log_\xi(n) \rceil$, and $m \leq n^2 \cdot (2 \cdot \lceil \log_\xi(n) \rceil + 3)$. Afterwards, we rely on Lemma 12 to remove the occurrences of λ from the above formulae, establishing in this way the first statement of the lemma. Items (I) and (II) follow from the analogous items in Lemma 12. To achieve the bounds in Item (II) we also rely on the fact that $\lceil \log_\xi(n) \rceil \leq 2^{2c} \lceil \ln(n) \rceil$. This follows from a simple computation, noticing that since ξ is not a root of the polynomial $x - 1$, by the definition of root barrier we have $\xi > 1 + \frac{1}{e^c}$. ◀

By relying on the characterisation, given in Lemma 13, of the values that $\lambda(u^*)$ can take, where $u^* > 0$ is the root of some polynomial, and by applying our previous observation that satisfiability can be witnessed by picking elements of $\xi^{\mathbb{Z}}$ that are “close” to u^* (i.e., the numbers $\xi^{-1} \cdot \lambda(u^*)$, $\lambda(u^*)$ or $\xi \cdot \lambda(u^*)$), we obtain the following key lemma.

► **Lemma 14.** *Let $\varphi(u, \mathbf{y})$ be a quantifier-free formula from $\exists \xi^{\mathbb{Z}}$. Then, $\exists u \varphi$ is equivalent to*

$$\bigvee_{\ell \in [-1..1]} \bigvee_{q \in Q} \bigvee_{(j, g, \mathbf{y}^\ell) \in F_q} \exists u : u^j = \xi^{j \cdot \ell + g} \cdot \mathbf{y}^\ell \wedge \varphi \quad (\dagger)$$

where Q is the set of all polynomials in φ featuring u , plus the polynomial $u - 1$, and each F_q is the set obtained by applying Lemma 13 to $r(x, v, \mathbf{y}) := q[x \cdot v / u]$, with x and v fresh variables.

To eliminate the variable u , we now consider each disjunct $\exists u (u^j = \xi^k \cdot \mathbf{y}^\ell \wedge \varphi)$ from Formula (†) and, roughly speaking, substitute u with $\sqrt[j]{\xi^k \cdot \mathbf{y}^\ell}$. We do not need however to introduce j th roots, as shown in the following lemma.

► **Lemma 15.** *Let $\varphi(u, \mathbf{y})$ be a quantifier-free formula from $\exists \xi^{\mathbb{Z}}$, with $\mathbf{y} = (y_1, \dots, y_n)$. Let $j \in \mathbb{N}_{\geq 1}$, $k \in \mathbb{Z}$ and $\boldsymbol{\ell} := (\ell_1, \dots, \ell_n) \in \mathbb{Z}$. Then, $\exists \mathbf{y} \exists u : u^j = \xi^k \cdot \mathbf{y}^\ell \wedge \varphi$ is equivalent to*

$$\bigvee_{\mathbf{r} := (r_1, \dots, r_n) \in R} \exists \mathbf{z} : \varphi[z_i^j \cdot \xi^{r_i} / y_i : i \in [1..n]] [\xi^{\frac{k + \boldsymbol{\ell} \cdot \mathbf{r}}{j}} \cdot \mathbf{z}^\ell / u],$$

where $R := \{(r_1, \dots, r_n) \in [0..j-1]^n : j \text{ divides } k + \sum_{i=1}^n r_i \cdot \ell_i\}$, $\boldsymbol{\ell} \cdot \mathbf{r} := \sum_{i=1}^n r_i \cdot \ell_i$, and $\mathbf{z} := (z_1, \dots, z_n)$ is a vector of fresh variables.

Proof sketch. Consider a solution to the equality $u^j = \xi^k \cdot \mathbf{y}^\ell$. Each y_i evaluates to a number of the form $\xi^{q_i \cdot j + r_i}$, with $q_i \in \mathbb{Z}$ and $r_i \in [0..j-1]$. Since u^j is of the form $\xi^{j \cdot q}$ for some $q \in \mathbb{Z}$, we must have that $k + \sum_{i=1}^n r_i \cdot \ell_i$ is divisible by j . Observe that the set R in the statement of the lemma contains all possible vectors $\mathbf{r} = (r_1, \dots, r_n)$ satisfying this divisibility condition.

At the formula level, consider a vector $\mathbf{r} = (r_1, \dots, r_n) \in R$, and replace in $u^j = \xi^k \cdot \mathbf{y}^\ell \wedge \varphi$ every variable y_i with the term $z_i^j \cdot \xi^{r_i}$. After this replacement, the equality $u^j = \xi^k \cdot \mathbf{y}^\ell$ can be rewritten as $u = \xi^{\frac{k + \boldsymbol{\ell} \cdot \mathbf{r}}{j}} \cdot \mathbf{z}^\ell$, where the division is without remainder. We can therefore substitute u with $\xi^{\frac{k + \boldsymbol{\ell} \cdot \mathbf{r}}{j}} \cdot \mathbf{z}^\ell$ in φ , eliminating it. ◀

By chaining Lemmas 14 and 15, one can eliminate all variables from a quantifier-free formula $\varphi(\mathbf{x})$, obtaining an equisatisfiable formula with no variables.

5.2 Quantifier relativisation

Looking closely at how a quantifier-free formula $\varphi(u_1, \dots, u_n)$ of $\exists\xi^{\mathbb{Z}}$ evolves as we chain Lemmas 14 and 15 to eliminate all variables, we see that the resulting variable-free formula is a finite disjunction $\bigvee_i \psi_i$ of formulae ψ_i that are obtained from φ via a sequence of substitutions stemming from Lemma 15. As an example, for a formula in three variables $\varphi(u_1, u_2, u_3)$, each ψ_i is obtained by applying a sequence of substitutions of the form:

$$\begin{array}{ccc} \left\{ \begin{array}{l} u_1 = \xi^{k_1} \cdot z_1^{\ell_1} \cdot z_2^{\ell_2} \\ u_2 = z_1^{j_1} \cdot \xi^{r_1} \\ u_3 = z_2^{j_2} \cdot \xi^{r_2} \end{array} \right. & \longrightarrow & \left\{ \begin{array}{l} z_1 = \xi^{k_2} \cdot z_3^{\ell_3} \\ z_2 = z_3^{j_2} \cdot \xi^{r_3} \end{array} \right. & \longrightarrow & \left\{ z_3 = \xi^{k_3} \right. \\ \text{elimination of } u_1 & & \text{elimination of } z_1 & & \text{elimination of } z_3 \end{array}$$

We can “backpropagate” these substitutions to the initial variables u_1, \dots, u_n , associating to each one of them an integer power of ξ . In the above example, we obtain the system

$$\left\{ \begin{array}{l} u_1 = \xi^{k_1} \cdot (\xi^{k_2} \cdot (\xi^{k_3})^{\ell_3})^{\ell_1} \cdot ((\xi^{k_3})^{j_2} \cdot \xi^{r_3})^{\ell_2} \\ u_2 = (\xi^{k_2} \cdot (\xi^{k_3})^{\ell_3})^{j_1} \cdot \xi^{r_1} \\ u_3 = ((\xi^{k_3})^{j_2} \cdot \xi^{r_3})^{j_1} \cdot \xi^{r_2} \end{array} \right.$$

By Lemmas 13–15, we can restrict the integers occurring as powers of ξ in the resulting system of substitutions to a finite set. Since the disjunction $\bigvee_i \psi_i$ is finite, this implies that, under the hypothesis that ξ is a computable number that is either transcendental or has a polynomial root barrier, it is possible to compute a finite set $P_\varphi \subseteq \mathbb{Z}$ witnessing the satisfiability of φ . That is, the sentence $\exists u_1 \dots \exists u_n \varphi$ is equivalent to

$$\exists u_1 \dots \exists u_n \bigvee_{(g_1, \dots, g_n) \in (P_\varphi)^n} (\varphi \wedge \bigwedge_{i=1}^n u_i = \xi^{g_i}).$$

Proposition 8 follows (in particular, the bound on P_φ for the case of ξ with a polynomial root barrier is derived by iteratively applying the bounds in Lemmas 13–15).

6 Proof of Theorem 1: classical numbers with polynomial root barriers

In this section, we complete the proof of Theorem 1 by establishing Theorem 1.1 and Theorem 1.2. Following Theorem 4, we discuss natural choices for the base $\xi > 0$ that (i) can be computed with polynomial-time Turing machines and (ii) have polynomial root barriers.

The case of ξ algebraic. Let ξ be a fixed algebraic number represented by (q, ℓ, u) . The following two results (the first one based on performing a dichotomy search to refine the interval $[\ell, u]$) show that one can construct a polynomial-time Turing machine for ξ , and that ξ has a polynomial root barrier where the integer k from Theorem 4 equals 1.

► **Lemma 16.** *Given an algebraic number α represented by (q, ℓ, u) , one can construct a polynomial-time Turing machine computing α .*

► **Theorem 17** ([10, Theorem A.1]). *Let $\alpha \in \mathbb{R}$ be a zero of a non-zero integer polynomial $q(x)$, and consider a non-constant integer polynomial $p(x)$. Then, either $p(\alpha) = 0$ or $\ln |p(\alpha)| \geq -\deg(q) \cdot (\ln(\deg(p) + 1) + \ln h(p)) - \deg(p) \cdot (\ln(\deg(q) + 1) + \ln h(q))$.*

■ **Table 1** Transcendence measures for some classical real numbers. For convenience only, the table assumes $h \geq 16$ (so that $\ln \ln h \geq 1$; replace h by $h + 15$ to avoid this assumption). The numbers $\alpha > 0$, $\beta > 0$ and η are fixed algebraic numbers, with $\beta \neq 1$. The integers c_η , $c_{\alpha,\eta}$, c_α and $c_{\alpha,\beta}$ are constants that depend on, and can be computed from, polynomials representing α , β and η . In the case of α^η , η is assumed to be irrational. In the last line of the table, $\frac{\ln \alpha}{\ln \beta}$ is assumed to be irrational.

Number	Transcendence measure from [33]	Simplified bound (α, β, η fixed)
π	$2^{40} d (\ln h + d \ln d) (1 + \ln d)$	$O(d^2 (\ln d)^2 \ln h)$
e^π	$2^{60} d^2 (\ln h + \ln d) (\ln \ln h + \ln d) (1 + \ln d)$	$O(d^2 (\ln d)^3 (\ln h) (\ln \ln h))$
e^η	$c_\eta \cdot d^2 (\ln h + \ln d) \left(\frac{\ln \ln h + \ln d}{\ln \ln h + \ln \max(1, \ln d)} \right)^2$	$O(d^2 (\ln d)^3 (\ln h) (\ln \ln h)^2)$
α^η	$c_{\alpha,\eta} \cdot d^3 (\ln h + \ln d) \frac{\ln \ln h + \ln d}{(1 + \ln d)^2}$	$O(d^3 (\ln d)^2 (\ln h) (\ln \ln h))$
$\ln \alpha$	$c_\alpha \cdot d^2 \frac{\ln h + d \ln d}{1 + \ln d}$	$O(d^3 (\ln d) \ln h)$
$\frac{\ln \alpha}{\ln \beta}$	$c_{\alpha,\beta} \cdot d^3 \frac{\ln h + d \ln d}{(1 + \ln d)^2}$	$O(d^4 (\ln d) \ln h)$

By applying Theorem 4.1, Lemma 16 and Theorem 17, we deduce that the satisfiability problem for $\exists \mathbb{R}(\xi^{\mathbb{Z}})$ is in 2EXP. However, for algebraic numbers it is possible to obtain a better complexity result (EXPSpace) by slightly modifying Steps II and III of Algorithm 1.

Proof of Theorem 1.1. Let φ be a formula in input of Algorithm 1, and $\psi(u_1, \dots, u_n)$ to be the formula obtained from φ after executing lines 1–6. In lines 7 and 8, guess the integers g_1, \dots, g_n in binary, instead of unary. These numbers have at most m bits where, by Theorem 7 and Proposition 8, m is exponential in $\text{size}(\varphi)$. Let $g_i = \pm_i \sum_{j=0}^{m-1} d_{i,j} 2^j$, with $d_{i,j} \in \{0, 1\}$ and $\pm_i \in \{+1, -1\}$, so that $\xi^{g_i} = \prod_{j=0}^{m-1} \xi^{\pm_i d_{i,j} 2^j}$. Note that the formula

$$\gamma(x_0, \dots, x_{m-1}) := q(x_0) = 0 \wedge \ell \leq x_0 \leq u \wedge \bigwedge_{i=1}^{m-1} x_i = (x_{i-1})^2$$

has a unique solution: for every $j \in [0..m-1]$, x_j must be equal to ξ^{2^j} . The formula ψ is therefore equisatisfiable with the formula $\psi' := \psi[x_0 / \xi] \wedge \gamma \wedge \bigwedge_{i=1}^n u_i = \prod_{j=0}^{m-1} x_j^{\pm_i d_{i,j}}$, which (after rewriting $u_i = \prod_{j=0}^{m-1} x_j^{\pm_i d_{i,j}}$ into $u_i \prod_{j=0}^{m-1} x_j^{d_{i,j}} = 1$ when $\pm_i = -1$) is a formula from the existential theory of the reals of size exponential in $\text{size}(\varphi)$. Since the satisfiability problem for the existential theory of the reals is in PSPACE [12], we conclude that checking whether ψ' is satisfiable can be done in EXPSpace. Accounting for Steps I and II, we thus obtain a procedure running in non-deterministic exponential space (because of the guesses in lines 7 and 8), which can be determined by Savitch's theorem [32]. ◀

The case of ξ among some classical transcendental numbers (proof sketch of Theorem 1.2).

In the context of transcendental numbers, root barriers are usually called *transcendence measures*. Several fundamental results in number theory concern deriving a transcendence measure for “illustrious” numbers, such as Euler's e , π , or logarithms of algebraic numbers [29, 25, 33]. A few of these results are summarised in Table 1, which is taken almost verbatim from [33, Fig. 1 and Corollary 4.2]. All transcendence measures in the table are *polynomial* root barriers. Note that in the cases of α^η and $\frac{\ln \alpha}{\ln \beta}$, the transcendence measures hold under further assumptions, which are given in the caption of the table.

Following Theorem 4.2, to prove Theorem 1.2 it suffices to show how to construct a polynomial-time Turing machine for every number in Table 1, and derive polynomial root barriers for the cases $\xi = \alpha^\eta$ and $\xi = \frac{\ln \alpha}{\ln \beta}$ without relying on the additional assumptions in the table. The following two results solve the first of these two issues.

► **Theorem 18** ([6]). *One can construct a polynomial-time Turing machine computing π .*

► **Lemma 19.** *Given a polynomial-time Turing machine computing $r \in \mathbb{R}$,*

1. *one can construct a polynomial-time Turing machine computing e^r ;*
2. *if $r > 0$, one can construct a polynomial-time Turing machine computing $\ln(r)$.*

Proof idea. The two Turing machines use the power series in the identities $e^x = \sum_{j=0}^{\infty} \frac{x^j}{j!}$ and $\ln(x) = 2 \sum_{j=0}^{\infty} \left(\frac{1}{2j+1} \left(\frac{x-1}{x+1} \right)^{2j+1} \right)$, truncated to obtain the required accuracy. ◀

As an example, to construct the Turing machine for $\frac{\ln(\alpha)}{\ln(\beta)}$ we construct machines for the following sequence of numbers: α and β (applying Lemma 16), $\ln(\alpha)$ and $\ln(\beta)$ (Lemma 19.2), $\frac{1}{\ln(\beta)}$ (Lemma 6) and $\frac{1}{\ln(\beta)} \cdot \ln(\alpha)$ (Lemma 5). For α^η , we follow the operations in $e^{\eta \cdot \ln(\alpha)}$.

Let us now discuss how to derive polynomial root barriers when $\xi = \alpha^\eta$ or $\xi = \frac{\ln(\alpha)}{\ln(\beta)}$. In the case $\xi = \alpha^\eta$, Table 1 assumes η to be irrational. To check whether an algebraic number represented by (q, ℓ, u) is rational, it suffices to factor $q(x)$ into a product of irreducible polynomials with rational coefficients, and test for any degree 1 factor $n \cdot x - m$ whether the rational number $\frac{m}{n}$ belongs to $[\ell, u]$. The factorisation of q can be computed (in fact, in polynomial time) using LLL [23]. If such a rational number does not exist, then η is irrational and the polynomial root barrier for α^η is given in Table 1. Otherwise, $\eta = \frac{m}{n}$ and the number $\alpha^{\frac{m}{n}}$ is algebraic. In this case, rely on the following lemma to construct a representation of $\alpha^{\frac{m}{n}}$, and then derive a polynomial root barrier by applying Theorem 17.

► **Lemma 20.** *There is an algorithm that given a rational r and an algebraic number $\alpha > 0$ represented by (q, ℓ, u) , computes a representation (q', ℓ', u') of the algebraic number α^r .*

We move to the case $\xi = \frac{\ln(\alpha)}{\ln(\beta)}$, which Table 1 assumes to be irrational. Since ξ is positive, $\alpha, \beta \notin \{0, 1\}$. We observe that for every $\frac{m}{n} \in \mathbb{Q}$, we have $\xi = \frac{m}{n}$ if and only if $\alpha^n \beta^{-m} = 1$. (In other words, $\frac{\ln(\alpha)}{\ln(\beta)} \in \mathbb{Q}$ if and only if α and β are multiplicatively dependent.) From a celebrated result of Masser [27], the set $\{(m, n) \in \mathbb{Z}^2 : \alpha^n \beta^{-m} = 1\}$ is a finitely-generated integer lattice for which we can explicitly construct a basis K (see [11] for a polynomial-time procedure). If $K = \{(0, 0)\}$, then ξ is irrational and its polynomial root barrier is given in Table 1. Otherwise, since $\alpha, \beta \notin \{0, 1\}$, there is $(m, n) \in K$ with $n \neq 0$, and $\xi = \frac{m}{n}$. We can then derive a polynomial root barrier by applying Theorem 17.

7 An application: the entropic risk threshold problem

We now apply some of the machinery developed for $\exists \mathbb{R}(\xi^{\mathbb{Z}})$ to remove the appeal to Schanuel’s conjecture from the decidability proof of the entropic risk threshold problem for stochastic games from [5]. Briefly, a *(turn-based) stochastic game* is a tuple $\mathsf{G} = (S_{\max}, S_{\min}, A, \Delta)$ where S_{\max} and S_{\min} are disjoint finite set of *states* controlled by two players, A is a function from states to a finite set of *actions*, and Δ is a function taking as input a state s and an action from $A(s)$, and returning a *probability distribution* on the set of states. Below, we write $\Delta(s, a, s')$ for the probability associated to s' in $\Delta(s, a)$, and set $S := S_{\max} \cup S_{\min}$.

Starting from an initial state \hat{s} , a play of the game produces an infinite sequence of states $\rho = s_1 s_2 s_3 \dots$ (a path), to which we associate the *total reward* $\sum_{i=1}^{\infty} r(s_i)$, where $r: S \rightarrow \mathbb{R}_{\geq 0}$ is a given *reward function*. A classical problem is to determine the strategy for one of the players that optimises (minimises or maximises) its expected total reward. Instead of expectation, the *entropic risk* yields the normalised logarithm of the average of the function $b^{-\eta X}$, where the *base* $b > 1$ and the *risk aversion factor* $\eta > 0$ are real numbers, and X is a random variable ranging over total rewards. We refer the reader to [5] for motivations behind this notion, as well as all formal definitions.

Fix a base $b > 1$ and a risk aversion factor $\eta \in \mathbb{R}$. The *entropic risk threshold problem* $\text{ERISK}[b^{-\eta}]$ asks to determine if the entropic risk is above a threshold t . The inputs of this problem are a stochastic game G having rational probabilities $\Delta(s, a, s')$, an initial state \hat{s} , a reward function $r: S \rightarrow \mathbb{Q}_{>0}$ and a threshold $t \in \mathbb{Q}$. In [5], this problem is proven to be in PSPACE for b and η rationals, and decidable subject to Schanuel's conjecture if $b = e$ and $\eta \in \mathbb{Q}$ (both results also hold when b and η are not fixed). We improve upon the latter result, by establishing the following theorem (that assumes having representations of α and η):

► **Theorem 21.** *The problems $\text{ERISK}[e^{-\eta}]$ and $\text{ERISK}[\alpha^{-\eta}]$ are in EXP for every fixed algebraic numbers α, η . When α, η are not fixed but part of the input, these problems are decidable.*

Proof sketch. Ultimately, in [5] the authors show that the problem $\text{ERISK}[b^{-\eta}]$ is reducible in polynomial time to the problem of checking the satisfiability of a system of constraints of the following form (see [5, Equation 7] for an equivalent formula):

$$v(\hat{s}) \leq (b^{-\eta})^t \wedge \bigwedge_{s \in T} v(s) = d_s \wedge \bigwedge_{s \in S} v(s) = \bigoplus_{a \in A(s)} \left((b^{-\eta})^{r(s)} \sum_{s' \in S} \Delta(s, a, s') \cdot v(s') \right), \quad (1)$$

where T is some subset of the states S of the game, $d_s \in \{0, 1\}$, and in the notation $\bigoplus_{a \in A(s)}$ the symbol \bigoplus stands for the functions min or max, depending on which of the two players controls s . The formula has one variable $v(s)$ for every $s \in S$, ranging over \mathbb{R} .

Since $z = \max(x, y)$ is equivalent to $z \geq x \wedge z \geq y \wedge (z = x \vee z = y)$, and $z = \min(x, y)$ is equivalent to $z \leq x \wedge z \leq y \wedge (z = x \vee z = y)$, except for the rationality of the exponents t and $r(s)$ (which we handle below), Formula 1 belongs to $\exists\mathbb{R}((b^{-\eta})^{\mathbb{Z}})$.

Fix $b > 1$ to be either e or algebraic, and $\eta > 0$ to be algebraic. Assume to have access to representations for these algebraic numbers, so that if η is represented by $(q(x), \ell, u)$, then $-\eta$ is represented by $(q(-x), -u, -\ell)$. Consider the problem of checking whether a formula φ of the form given by Formula 1 is satisfiable. Since φ does not feature predicates $(b^{-\eta})^{\mathbb{Z}}$, but only the constant $b^{-\eta}$, instead of Algorithm 1 we can run the following simplified procedure:

- I. *Update all exponents t and $r(s)$ of φ to be over \mathbb{N} and written in unary. (1) Compute the l.c.m. $d \geq 1$ of the denominators of these exponents. (2) Rewrite every term $(b^{-\eta})^{\frac{p}{q}}$, where $\frac{p}{q}$ is one such exponent, into $(b^{-\frac{\eta}{d}})^{\frac{p \cdot d}{q}}$. Note that $\frac{p \cdot d}{q} \in \mathbb{Z}$. (3) Rewrite φ into $\varphi[x / b^{-\frac{\eta}{d}}] \wedge x^d = b^{-\eta} \wedge x \geq 0$, with x fresh variable. (4) Opportunely multiply both sides of inequalities by integer powers of x to make all exponents range over \mathbb{N} . (5) Change to a unary encoding for the exponents by adding further variables, as done in the proof of Theorem 1.1 (Section 6). Overall, this step takes polynomial time in $\text{size}(\varphi)$.*
- II. *Eliminate x and all variables $v(s)$ with $s \in S$.* This is done by appealing to Theorem 7, treating $b^{-\eta}$ as a free variable. The result is a Boolean combination ψ of polynomial inequalities over $b^{-\eta}$. This step runs in time exponential in $\text{size}(\varphi)$.
- III. *Evaluate ψ .* Call Algorithm 2 on each inequality, to then return \top or \perp according to the Boolean structure of ψ . Since we can construct a polynomial-time Turing machine for $b^{-\eta}$ (Section 6), by Lemma 10 this step takes polynomial time in $\text{size}(\psi)$. ◀

8 Conclusion and future directions

We have studied the complexity of the theory $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ for different choices of $\xi > 0$. Particularly valuable turned out to be the introduction of root barriers (Definition 3): by relying on this notion, we have established that $\exists\mathbb{R}(\xi^{\mathbb{Z}})$ is in EXPSPACE if ξ is algebraic, and in 3EXP for natural choices of ξ among the transcendental numbers, such as e and π .

A first natural question is how far are we from the exact complexity of these existential theories, considering that the only known lower bound is inherited from the existential theory of the reals, which lies in PSPACE [12]. While we have no answer to this question, we remark that strengthening the hypotheses on ξ may lead to better complexity bounds. For example, we claim that our EXPSPACE result for algebraic numbers improves to EXP when ξ is an integer (we aim at including this result in an extended version of this paper).

We have presented natural examples of bases ξ having polynomial root barriers. More exotic instances are known: setting $\xi = q(\pi, \Gamma(\frac{1}{4}))$, where q is an integer polynomial and Γ is Euler's Gamma function, results in one such base. This follows from a theorem by Bruiliet [9, Theorem B'] on the algebraic independence of π and $\Gamma(\frac{1}{4})$. This leads to a second natural question: are there real numbers a, b satisfying $a^{\mathbb{Z}} \cap b^{\mathbb{Z}} = \{1\}$ for which the existential theory of the reals enriched with both the predicates $a^{\mathbb{Z}}$ and $b^{\mathbb{Z}}$ is decidable?

References



- 1 Melanie Achatz, Scott McCallum, and Volker Weispfenning. Deciding polynomial-exponential problems. In *ISSAC*, pages 215–222, 2008. doi:10.1145/1390768.1390799.
- 2 Shaull Almagor, Dmitry Chistikov, Joël Ouaknine, and James Worrell. O-minimal invariants for discrete-time dynamical systems. *ACM Trans. Comput. Log.*, 23(2), 2022. doi:10.1145/3501299.
- 3 Hirokazu Anai and Volker Weispfenning. Deciding linear-trigonometric problems. In *ISSAC*, pages 14–22, 2000. doi:10.1145/345542.345567.
- 4 Jeremy Avigad and Yimu Yin. Quantifier elimination for the reals with a predicate for the powers of two. *Theor. Comput. Sci.*, 370(1-3):48–59, 2007. doi:10.1016/J.TCS.2006.10.005.
- 5 Christel Baier, Krishnendu Chatterjee, Tobias Meggendorfer, and Jakob Piribauer. Entropic risk for turn-based stochastic games. In *MFCSS*, volume 272, pages 15:1–15:16, 2023. doi:10.4230/LIPICSS.MFCSS.2023.15.
- 6 David H. Bailey, Peter B. Borwein, and Simon Plouffe. On the rapid computation of various polylogarithmic constants. *Math. Comput.*, 66:903–913, 1997. doi:10.1090/S0025-5718-97-00856-9.
- 7 Gilles Barthe, Rohit Chadha, Paul Krogmeier, A. Prasad Sistla, and Mahesh Viswanathan. Deciding accuracy of differential privacy schemes. *Proc. ACM Program. Lang.*, 5(POPL):1–30, 2021. doi:10.1145/3434289.
- 8 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996. doi:10.1145/235809.235813.
- 9 Sylvain Bruiliet. D'une mesure d'approximation simultanée à une mesure d'irrationalité: le cas de $\Gamma(1/4)$ et $\Gamma(1/3)$. *Acta Arith.*, 104(3):243–281, 2002. doi:10.4064/aa104-3-3.
- 10 Yann Bugeaud. *Approximation by Algebraic Numbers*. Cambridge Tracts in Mathematics. Cambridge University Press, 2004. doi:10.1017/CB09780511542886.
- 11 Jin-yi Cai, Richard J. Lipton, and Yechezkel Zalcstein. The complexity of the A B C problem. *SIAM J. Comput.*, 29(6):1878–1888, 2000. doi:10.1137/S0097539794276853.
- 12 John Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467, 1988. doi:10.1145/62212.62257.
- 13 Dmitry Chistikov, Stefan Kiefer, Andrzej S. Murawski, and David Purser. The big-o problem. *Log. Methods Comput. Sci.*, 18(1), 2022. doi:10.46298/LMCS-18(1:40)2022.
- 14 Mohan Dantam and Amaury Pouly. On the decidability of reachability in continuous time linear time-invariant systems. In *HSCC*, 2021. doi:10.1145/3447928.3456705.
- 15 Laure Daviaud, Marcin Jurdziński, Ranko Lazić, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When are emptiness and containment decidable for probabilistic automata? *JCSS*, 119:78–96, 2021. doi:10.1016/j.jcss.2021.01.006.

- 16 Andreas Dolzmann and Thomas Sturm. REDLOG: computer algebra meets computer logic. *SIGSAM Bull.*, 31(2):2–9, 1997. doi:10.1145/261320.261324.
- 17 Lou van den Dries. The field of reals with a predicate for the powers of two. *Manuscripta Math.*, 54:187–196, 1986. doi:10.1007/BF01171706.
- 18 Lou van den Dries and Ayhan Günaydin. The fields of real and complex numbers with a small multiplicative group. *Proc. Lond. Math. Soc.*, 93(1):43–81, 2006. doi:10.1017/S0024611506015747.
- 19 Teemu Hankala, Miika Hannula, Juha Kontinen, and Jonni Virtema. Complexity of neural network training and ETR: extensions with effectively continuous functions. In *AAAI*, pages 12278–12285, 2024. doi:10.1609/AAAI.V38I11.29118.
- 20 Philipp Hieronymi. Defining the set of integers in expansions of the real field by a closed discrete set. *Proc. Am. Math. Soc.*, 138(6):2163–2168, 2010. doi:10.1090/S0002-9939-10-10268-8.
- 21 Omri Isac, Yoni Zohar, Clark W. Barrett, and Guy Katz. DNN verification, reachability, and the exponential function problem. In *CONCUR*, pages 26:1–26:18, 2023. doi:10.4230/LIPICS.CONCUR.2023.26.
- 22 A. G. Khovanskii. Fewnomials. *Transl. Math. Monogr.*, 88, 1991. Translated by Smilka Zdravkovska. doi:10.1090/mmono/088.
- 23 Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982. doi:10.1007/bf01457454.
- 24 Angus Macintyre and Alex J. Wilkie. On the decidability of the real exponential field. In Piergiorgio Odifreddi, editor, *Kreiseliana. About and Around Georg Kreisel*, pages 441–467. A K Peters, 1996.
- 25 Kurt Mahler. Zur Approximation der Exponentialfunktion und des Logarithmus. Teil I. *Journal für die reine und angewandte Mathematik*, 166:118–150, 1932.
- 26 David Marker. *Model Theory: An Introduction*. Graduate Texts in Mathematics. Springer, 2002. doi:10.1007/b98860.
- 27 D. W. Masser. *Linear relations on algebraic groups*, pages 248–262. Cambridge University Press, 1988.
- 28 Scott McCallum and Volker Weispfenning. Deciding polynomial-transcendental problems. *J. Symb. Comput.*, 47(1):16–31, 2012. doi:10.1016/J.JSC.2011.08.004.
- 29 J. Popken. Zur Transzendenz von e. *Mathematische Zeitschrift*, 29:525–541, 1929.
- 30 Q. I. Rahman and G. Schmeisser. *Analytic Theory of Polynomials*. Oxford University Press, September 2002. doi:10.1093/oso/9780198534938.001.0001.
- 31 H. G. Rice. Recursive real numbers. *Proc. Am. Math. Soc.*, 5(5):784–791, 1954. doi:10.2307/2031867.
- 32 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *JCSS*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.
- 33 Michel Waldschmidt. Transcendence measures for exponentials and logarithms. *J. Aust. Math. Soc.*, 25(4):445–465, 1978. doi:10.1017/S1446788700021431.
- 34 Volker Weispfenning. The complexity of almost linear diophantine problems. *J. Symb. Comput.*, 10(5):395–404, 1990. doi:10.1016/S0747-7171(08)80051-X.

Two-Dimensional Longest Common Extension Queries in Compact Space

Arnab Ganguly  



University of Wisconsin, Whitewater, WI, USA

Daniel Gibney  

University of Texas at Dallas, TX, USA

Rahul Shah  

Louisiana State University, Baton Rouge, LA, USA

Sharma V. Thankachan  

North Carolina State University, Raleigh, NC, USA

Abstract

For a length n text over an alphabet of size σ , we can encode the suffix tree data structure in $\mathcal{O}(n \log \sigma)$ bits of space. It supports suffix array (SA), inverse suffix array (ISA), and longest common extension (LCE) queries in $\mathcal{O}(\log^\epsilon n)$ time, which enables efficient pattern matching; here $\epsilon > 0$ is an arbitrarily small constant. Further improvements are possible for LCE queries, where $\mathcal{O}(1)$ time queries can be achieved using an index of space $\mathcal{O}(n \log \sigma)$ bits. However, compactly indexing a two-dimensional text (i.e., an $n \times n$ matrix) has been a major open problem. We show progress in this direction by first presenting an $\mathcal{O}(n^2 \log \sigma)$ -bit structure supporting LCE queries in near $\mathcal{O}((\log_\sigma n)^{2/3})$ time. We then present an $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ -bit structure supporting ISA queries in near $\mathcal{O}(\log n \cdot (\log_\sigma n)^{2/3})$ time. Within a similar space, achieving SA queries in poly-logarithmic (even strongly sub-linear) time is a significant challenge. However, our $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ -bit structure can support SA queries in $\mathcal{O}(n^2 / (\sigma \log n)^c)$ time, where c is an arbitrarily large constant, which enables pattern matching in time faster than what is possible without preprocessing.

We then design a repetition-aware data structure. The δ_{2D} compressibility measure for two-dimensional texts was recently introduced by Carfagna and Manzini [SPIRE 2023]. The measure ranges from 1 to n^2 , with smaller δ_{2D} indicating a highly compressible two-dimensional text. The current data structure utilizing δ_{2D} allows only element access. We obtain the first structure based on δ_{2D} for LCE queries. It takes $\tilde{\mathcal{O}}(n^{5/3} + n^{8/5} \delta_{2D}^{1/5})$ space and answers queries in $\mathcal{O}(\log n)$ time.

2012 ACM Subject Classification Theory of computation \rightarrow Pattern matching

Keywords and phrases String matching, text indexing, two-dimensional text

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.38

Funding Supported by the US National Science Foundation (NSF) under Grant Numbers 2315822 (S Thankachan) and 2137057 (R Shah).

1 Introduction

A two-dimensional text $T[0..n][0..n]$ can be viewed as an $n \times n$ matrix, where each entry is a character from an alphabet set Σ of size σ . Data structures for two-dimensional texts have been studied for decades. In particular, there has been extensive work on generalizing suffix trees [16, 17, 23] and suffix arrays [16, 22] to 2D text. These data structures, although capable of answering most queries in optimal (or near optimal) time, require $\mathcal{O}(n^2)$ words, or $\mathcal{O}(n^2 \log n)$ bits, of space.

On the other hand, in the case of 1D texts of length n , there exist data structures with the same functionality as suffix trees/arrays but requiring only $\mathcal{O}(n \log \sigma)$ bits of space [18, 32], or even smaller in the case where the text is compressible [11, 21]. This is true even for



© Arnab Ganguly, Daniel Gibney, Rahul Shah, and Sharma V. Thankachan; licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 38; pp. 38:1–38:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



some variants of suffix trees, such as parameterized [14, 13] and order-isomorphic [12] suffix trees [33]. The query times of these space-efficient versions are often polylogarithmic, with the exception of LCE queries, for which Kempa and Kociumaka demonstrated that the query time can be made constant [19]. For 2D texts, the only results known in this direction include a data structure by Patel and Shah that uses $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ bits and supports inverse suffix array (ISA) queries in $\mathcal{O}(\log^4 n / (\log \log n)^3)$ time [28]. In this work, we make further progress in this direction. In particular, we focus on space-efficient data structures for longest common extension (LCE) queries in the 2D setting. The problem is formally defined as follows:

► **Problem 1** (2D LCE). *Preprocess a 2D text $T[0..n][0..n]$ over an alphabet Σ of size σ into a data structure that can answer 2D LCE queries efficiently. A 2D LCE query consists of points (i_1, j_1) , (i_2, j_2) and asks to return the largest L such that $T[i_1..i_1+L][j_1..j_1+L]$ and $T[i_2..i_2+L][j_2..j_2+L]$ are matching square submatrices of T .*

A 2D suffix tree of size $\mathcal{O}(n^2 \log n)$ bits can answer LCE queries in constant time. Our first result is an LCE data structure that occupies $\mathcal{O}(n^2 \log \sigma)$ bits of space.

► **Theorem 1.** *By maintaining an $\mathcal{O}(n^2 \log \sigma)$ -bit data structure, we can answer 2D LCE queries in $\mathcal{O}((\log_\sigma n)^{2/3} \cdot (\log \log_\sigma n)^{5/3})$ time.*

Turning now to highly compressible 2D texts, we consider repetition-aware compression measures. The δ measure is an important and well-studied compressibility measure for 1D text [26]. Only recently has it been extended to 2D text by Carfagna and Manzini with a δ_{2D} -measure [5]. They demonstrate that the data structure of Brisaboa et al. [3] occupies $\mathcal{O}((\delta_{2D} + \sqrt{n\delta_{2D}}) \log \frac{n \log \sigma}{\sqrt{\delta_{2D} \log n}})$ space. However, this data structure only supports access to the elements of T . We provide the first repetition-aware data structure supporting the more advanced LCE queries. Note that the measure δ_{2D} ranges from 1 to n^2 , with a smaller δ_{2D} value implying higher compressibility.

► **Theorem 2.** *By maintaining an $\mathcal{O}((n^{5/3} + n^{8/5} \delta_{2D}^{1/5}) \log \beta)$ word data structure, we can answer 2D LCE queries in $\mathcal{O}(1 + \log \beta)$ time, where β is always $\mathcal{O}(n)$ and goes to $\mathcal{O}(1)$ as δ_{2D} approaches n^2 . In particular,*

$$\beta = \begin{cases} n & \text{if } \delta_{2D} < n^{9/5} \\ n^{9/5} / \delta_{2D}^{9/10} & \text{if } \delta_{2D} \geq n^{9/5}. \end{cases}$$

When $\delta_{2D} = \Theta(n^2)$, our data structure takes $\mathcal{O}(n^2)$ words of space and answers LCE queries in $\mathcal{O}(1)$ time. When $\delta_{2D} = o(n^2)$, the space becomes $o(n^2)$ and LCE queries are answered in logarithmic time. Our approach builds off many of the same techniques as our compact index but also introduces a matrix representation of the leaves of a truncated suffix tree. We call this a *macro-matrix*. We prove that if the original 2D text is compressible, then this macro-matrix remains compressible for appropriately chosen parameters. This is then combined with the data structure of Brisaboa et al. [3] to achieve Theorem 2.

As the first steps towards obtaining the other functionalities of the suffix tree, we apply our 2D LCE query structure from Theorem 1 to get the following results. Definitions of suffix array (SA) and inverse suffix array (ISA) are deferred to Section 1.1.

The following theorem significantly improves on the results by Patel and Shah [28].

► **Theorem 3** (2D ISA queries). *By maintaining an $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ -bit data structure, we can answer inverse suffix array queries in $\mathcal{O}(\log n \cdot (\log_\sigma n)^{2/3} \cdot (\log \log_\sigma n)^{5/3})$ time.*

We also provide the first known results regarding a nearly compact index for 2D suffix array queries.

► **Theorem 4** (2D SA queries). *By maintaining an $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ -bit data structure, we can answer suffix array (SA) queries in $\mathcal{O}(n^2/(\sigma \log n)^c)$ time, where c is an arbitrarily large constant fixed at the time of construction.*

A fundamental problem is to find all submatrices of T that match with a given square pattern $P[0..m][0..m]$. After building the 2D suffix tree, given P as a query, the number of occurrences of P (denoted by occ) can be obtained in $\mathcal{O}(m^2)$ time, and all occurrences can be reported in $\mathcal{O}(m^2 + occ)$ time. Our result, which uses a smaller index, is the following.

► **Theorem 5** (PM queries). *By maintaining an $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ -bit data structure, we can count the occurrences of an $m \times m$ query pattern in time $\mathcal{O}(m^2 + n^2/(\sigma \log n)^c)$ and report all occurrences in time $\mathcal{O}(m^2 + occ + n^2/(\sigma \log n)^c)$, where c is an arbitrarily large constant fixed at the time of construction.*

Although the time complexities in Theorems 4 and 5 are far from satisfactory, these are the first results demonstrating subquadratic query times in compact space are possible for 2D SA and PM queries.

1.1 Preliminaries

Notation and Strings. We denote the interval $i, i + 1, \dots, j$ with $[i..j]$ and the interval $i, i + 1, \dots, j - 1$, with $[i..j)$. For a string S of length n we use $S[i]$ to refer to i^{th} character, $i \in [0..n)$. We use $S_1 \cdot S_2$ to denote the concatenation of two strings S_1 and S_2 . For notation, $S[i..j] = S[i] \cdot S[i+1] \cdot \dots \cdot S[j]$, $S[i..j) = S[i] \cdot S[i+1] \cdot \dots \cdot S[j-1]$, and $S[i..] = S[i..n)$. Arrays and strings are zero-indexed throughout this work.

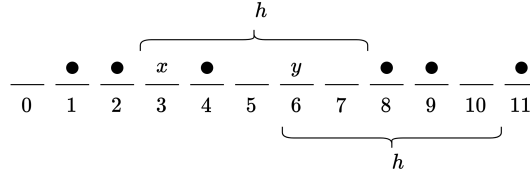
For a single string $S[0..n)$ and $i, j \in [0..n)$, $\text{LCE}(i, j)$ is defined as the length of the longest common prefix of $S[i..)$ and $S[j..)$. In the case of two strings, $S_1[0..n_1)$ and $S_2[0..n_2)$, we overload the notation so that for $i \in [0..n_1)$, $j \in [0..n_2)$, $\text{LCE}(i, j)$ is the length of the longest common prefix of $S_1[i..)$ and $S_2[j..)$. For a given string S , the suffix tree [34] is a compact trie of all suffixes of S with leaves ordered according to the lexicographic rank of the corresponding suffixes. The classical suffix tree takes $\mathcal{O}(n)$ words of space and can be constructed in $\mathcal{O}(n)$ time for polynomially sized integer alphabets [9]. The suffix array $\text{SA}[0..n)$ of a string $S[0..n)$ is the unique array such that $S[\text{SA}[i)..)$ is the i^{th} smallest suffix lexicographically. The inverse suffix array $\text{ISA}[0..n)$ is the unique array such that $\text{ISA}[\text{SA}[i)] = i$, or equivalently, $\text{ISA}[i]$ gives the lexicographic rank of $S[i..)$. The suffix tree can answer LCE queries in $\mathcal{O}(1)$ time. We call a compact trie with lexicographically ordered leaves for a subset of suffixes a *sparse suffix tree*. Observe that the number of nodes in a sparse suffix tree remains proportional to the number of suffixes it is built from.

We will utilize the following result by Kempa and Kociumaka, which provides an LCE data structure smaller than a classical suffix tree.

► **Lemma 6** ([19]). *1D LCE queries on a text $S[0..n)$ over an alphabet set $\Sigma = [0..\sigma)$ can be answered in $\mathcal{O}(1)$ time by maintaining a data structure of size $\mathcal{O}(n \log \sigma)$ bits.*

The next result by Bille et al. allows for a trade-off between space and query time. We will utilize it in Section 2.2.

► **Lemma 7** ([1]). *Suppose we have the text $S[0..n)$ as read-only, such that we can determine the lexicographic order of any of its two characters in constant time. Then we can answer 1D LCE queries on S in time $\mathcal{O}(\tau)$ by maintaining an $\mathcal{O}(n/\tau)$ words of space auxiliary structure, where $1 \leq \tau \leq n$ is any parameter fixed at the time of construction.*



■ **Figure 1** An example d -cover for $n = 12$ and $d = 7$. Here the difference cover used is $\mathcal{D} = \{1, 2, 4\}$, resulting in a d -cover $\mathcal{C} = \{1, 2, 4, 8, 9, 11\}$ (elements indicated with ‘●’) and a lookup table $A = [1, 1, 2, 1, 4, 4, 2]$. For the positions $x = 3$ and $y = 6$, we have $h = A[(6 - 3) \bmod 7] - 3 \bmod 7 \equiv 5$. Observe that $3 + 5, 6 + 5 \in \mathcal{C}$.

d-Covers. A d -cover of an interval $[0..n)$ is a subset of positions, denoted by \mathcal{C} , such that for any $x \in [0..n - d)$ and $y \in [0..n - d)$ there exists $h \in [0..d)$ where $x + h, y + h \in \mathcal{C}$. It was shown by Burkhardt and Kärkkäinen that there exists a d -cover of size $\mathcal{O}(n/\sqrt{d})$ that can be computed in $\mathcal{O}(n/\sqrt{d})$ time [4]. d -Covers have been used previously for LCE queries in the 1D case by Gawrychowski et al. [15] and Bille et al. [2]. Since we need a small data structure that lets us find an h value as described above in constant time, we briefly outline the construction given in [4].

A *difference cover* modulo d is a subset $\mathcal{D} \subseteq \{0, 1, \dots, d-1\}$ where for all $w \in \{0, 1, \dots, d-1\}$ there exist $u, v \in \mathcal{D}$ such that $w \equiv u - v \pmod{d}$. Colbourn and Ling showed there exists \mathcal{D} such that $|\mathcal{D}| = \Theta(\sqrt{d})$ [8]. A d -cover \mathcal{C} is constructed from a difference cover \mathcal{D} as follows: For $j \in [0..n)$, if $(j \bmod d) \in \mathcal{D}$, then j is added to \mathcal{C} . We also build a look-up table A of size d such that for all $i \in \{0, 1, \dots, d-1\}$ both $A[i]$ and $(A[i] + i) \bmod d$ are in \mathcal{D} . This is always possible, thanks to the definition of the difference cover. See Figure 1.

► **Lemma 8** ([4]). *For a d -cover \mathcal{C} of an interval $[0..n)$, there exists a data structure of size $\mathcal{O}(d)$ that given $x, y \in [0..n - d)$, outputs an $h \in [0..d)$ such that $x + h, y + h \in \mathcal{C}$ in $\mathcal{O}(1)$ time.*

Proof. We maintain the $\mathcal{O}(d)$ space look-up table A as described above. We assume without loss of generality, $y \geq x$. Let $h := (A[(y - x) \bmod d] - x) \bmod d$. Observe that

$$x + h \equiv A[(y - x) \bmod d] \pmod{d}$$

Hence, $(x + h \bmod d) \in \mathcal{D}$ and $x + h \in \mathcal{C}$. Also,

$$y + h \equiv A[(y - x) \bmod d] + (y - x) \pmod{d}$$

Hence, $(y + h \bmod d) \in \mathcal{D}$ and $y + h \in \mathcal{C}$. ◀

2D Suffix Trees and 2D Suffix Arrays. We utilize the generalization of suffix trees to 2D texts presented by Giancarlo [16]. This suffix tree is created from the *Lstrings* of the 2D text T . LStrings are over an alphabet $\cup_{i=1}^n \Sigma^{2i-1}$. For a position $(i, j) \in [0..n)^2$ the suffix $T[i..][j..]$ is $a_0 \cdot a_1 \cdot \dots \cdot a_l$ where $l = n - \max(i, j)$ and $a_0 = T[i][j]$ and $a_k = T[i + k][j..j + k] \cdot T[i..i + k][j + k]$ for $k > 0$. See Figure 2. The characters are maintained implicitly as references to T , resulting in the 2D suffix tree over all suffixes $T[i..][j..]$, $(i, j) \in [0, n)^2$ occupying $\mathcal{O}(n^2)$ words of space. Once constructed, the 2D suffix tree allows us to find the LCE of two positions in $\mathcal{O}(1)$ time through a lowest common ancestor (LCA) query. The 2D suffix tree also enables pattern matching in optimal $\mathcal{O}(m^2 + occ)$ time.

T	0	1	2	3	4	
0	a	a	b	b	$\$$	
1	a	b	b	c	$\$$	suffix starting at $(0, 0)$: $a \cdot aab \cdot bbbba \cdot bcabcab \cdot \$\$\$\$\$\$\$\$\$$
2	b	b	a	a	$\$$	suffix starting at $(0, 1)$: $a \cdot bbb \cdot babca \cdot cab\$\$\$\$$
3	b	c	a	b	$\$$	suffix starting at $(1, 0)$: $a \cdot bbb \cdot bcbaa \cdot \$\$cab\$$
4	$\$$	$\$$	$\$$	$\$$	$\$$	

■ **Figure 2** An example 2D text and the suffixes starting at positions $(0, 0)$, $(0, 1)$, $(1, 0)$. The “.” denotes concatenation, and consecutive symbols without “.” between them are treated as a single character.

The order between characters a and a' of Lstrings is defined as the lexicographic order induced by the base alphabet Σ . The lexicographic order of two Lstrings (and corresponding submatrices) is induced by the order of their characters. We additionally assume that the bottom row and rightmost column of T consist of only a $\$$ symbol, which is the smallest in the alphabet order and occurs nowhere else in T .

The suffix array $SA[0..n^2]$ of a 2D text $T[0..n][0..n]$ is an array containing 2D points such that if $(i, j) = SA[h]$, then $T[i..][j..]$ is the h^{th} smallest suffix lexicographically. The inverse suffix array maps each $(i, j) \in [0..n]^2$ to its position in SA, i.e. $ISA[SA[h]] = h$.

The δ_{2D} Measure and 2D Block Trees. The δ measure is a well-studied compressibility measure for 1D texts [7, 20, 24, 25, 30]. It is defined as $\delta(T) = \max_{1 \leq t \leq n} d_t(T)/t$ where $d_t(T)$ denotes the number of distinct length t substrings of $T[0, n)$.

Carfagna and Manzini recently generalized the δ measure to 2D texts [5, 6]. Letting $d_t(T)$ denote the number of distinct $t \times t$ submatrices of $T[0..n][0..n)$, $\delta_{2D}(T) = \max_{1 \leq t \leq n} d_t(T)/t^2$. Observe that $\delta_{2D}(T)$ can range between 1, e.g., in case where all elements of T are the same character, and n^2 , i.e., the case where all elements of T are distinct. Carfagna and Manzini showed that the 2D *block tree* data structure of Brisaboa, et al. [3] occupies $\mathcal{O}(\delta_{2D}(T) + \sqrt{n\delta_{2D}(T)}) \log \frac{n \log \sigma}{\sqrt{\delta_{2D}(T)} \log n}$ words of space and provides access to any entry of T in $\mathcal{O}(1 + \log \frac{n \log \sigma}{\sqrt{\delta_{2D}(T)} \log n})$ time. A further generalization of the δ measure to 2D allowing for non-square matrices was introduced by Romana et al. and related to other potential 2D compressibility measures [31]. In this work, we will only consider the δ_{2D} measure based on square submatrices. We hereafter refer to δ_{2D} as δ and omit the text T when it is clear from context.

2 Compact Data Structures for 2D LCE Queries

We start with some definitions. Let R_i denote the i^{th} row and C_j denote the j^{th} column of our 2D text T , where $0 \leq i, j < n$. Specifically, $R_i[0..n)$ (resp., $C_j[0..n)$) is a text of length n over the alphabet Σ , such that its k^{th} character is $T[i][k]$ (resp., $T[k][j]$), where $k \in [0..n)$.

We define a set of sampled positions on the diagonals of T , that is $T[n-1][0]$, $T[n-2][0] \cdot T[n-1][1]$, \dots , $T[0][n-2] \cdot T[1][n-1]$, $T[0][n-1]$, using d -cover with $d = \Theta(\log_\sigma^2 n)$. This is obtained by taking a d -cover \mathcal{C} for $[0..n)$ and using it to define sample positions starting from the top left of each diagonal. Formally, the sample positions are

$$\mathcal{C}_D = \{(i, j) \mid i, j \in [0..n), \min(i, j) \in \mathcal{C}\}.$$

See Figure 3.

	0	1	2	3	4	5	6
0							
1		•	•	•	•	•	•
2		•	•	•	•	•	•
3		•	•				
4		•	•		•	•	•
5		•	•		•		
6		•	•		•		

■ **Figure 3** An example 7-cover $\mathcal{C} = \{1, 2, 4\}$ used for the diagonal sample positions of a 7×7 text. Note that this $d = 7$ value is for illustrative purposes. Sample positions are indicated with a “•”.

We maintain a sparse suffix tree over the suffixes starting from these sampled positions. As this is a compact trie with $|\mathcal{C}_D| = \mathcal{O}(n^2/\sqrt{d})$ leaves, the space required for this sparse suffix tree is $\mathcal{O}(n^2/\sqrt{d})$ words. By our above choice of d , this is $\mathcal{O}(n^2 \log \sigma)$ bits. Using this sparse suffix tree, we can obtain LCE for any two sampled positions in $\mathcal{O}(1)$ time.

Additionally, we maintain the data structure from Lemma 6 for the concatenation of columns C_0, \dots, C_{n-1} and rows R_0, \dots, R_{n-1} , which adds another $\mathcal{O}(n^2 \log \sigma)$ bits. This allows us to find the LCE between $R_i[x..]$ and $R_j[y..]$ (or $C_i[x..]$ and $C_j[y..]$) in $\mathcal{O}(1)$ time. We can take a minimum between the LCE value and $\min(n-x, n-y)$ to avoid common prefixes crossing row or column boundaries.

In what follows, we first present a simple preliminary solution. We then develop these ideas further with two refinements that lead us to Theorem 1. The components defined above (sparse suffix tree from diagonal samples and LCE data structures for concatenated rows and columns) are used in all three solutions.

2.1 Achieving $\mathcal{O}(\log_\sigma^2 n)$ Query Time

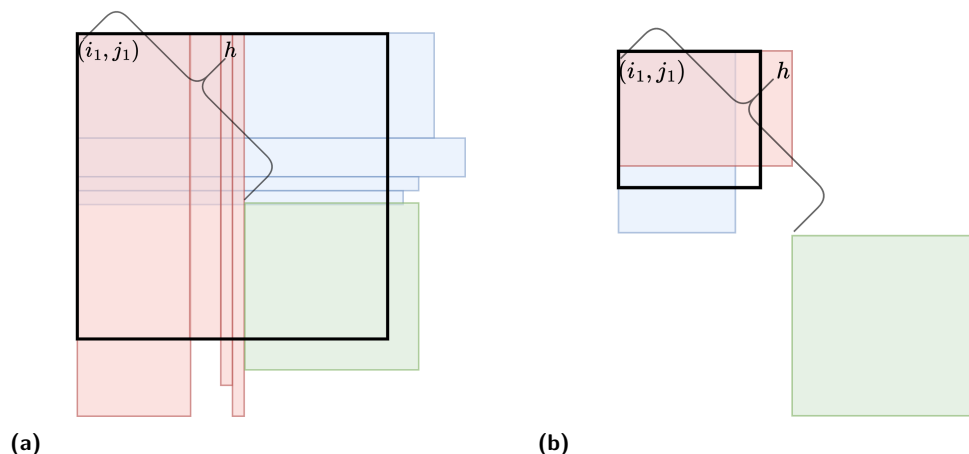
To answer an LCE query $(i_1, j_1), (i_2, j_2)$, we use the look-up structure discussed in Lemma 8 to obtain an $h \in [0..d)$ such that (i_1+h, j_1+h) and (i_2+h, j_2+h) are sampled diagonal positions. For convenience, in the case where no such h in the look-up structure exists, because either (i_1, j_1) or (i_2, j_2) is near the boundary of T , we consider h as being one less than the minimum diagonal offset to a boundary of T . We first obtain $\text{LCE}((i_1+h, j_1+h), (i_2+h, j_2+h))$ in $\mathcal{O}(1)$ time. Next, for $k \in [0..h)$, we compute the LCEs between $R_{i_1+k}[j_1..]$ and $R_{i_2+k}[j_2..]$, and between $C_{j_1+k}[i_1..]$ and $C_{j_2+k}[i_2..]$. While iterating from $k = 1$ to $k = h-1$, if for some k either the LCE between $R_{i_1+k}[j_1..]$ and $R_{i_2+k}[j_2..]$ or between $C_{j_1+k}[i_1..]$ and $C_{j_2+k}[i_2..]$ becomes less than k , we output $k-1$. Otherwise, we output the minimum over $h + \text{LCE}((i_1+h, j_1+h), (i_2+h, j_2+h))$ and all of the LCE values computed for the rows and columns specified above.

Only one constant time query for a diagonal sampled position is required, and the number of 1D LCE queries needed is at most $2d$. Since $d = \Theta(\log_\sigma^2 n)$ and each 1D LCE query takes $\mathcal{O}(1)$ time, the total time is $\mathcal{O}(\log_\sigma^2 n)$.

2.2 Achieving $\mathcal{O}(\log_\sigma n \cdot (\log \log_\sigma n)^2)$ Query Time

First, we define $R_{i,t}$ and $C_{j,t}$. These are texts of length n over an alphabet Σ^t , such that $0 \leq i, j$ and $i+t-1, j+t-1 < n$. The k^{th} character of $R_{i,t}$ and $C_{j,t}$ are length t strings over Σ defined as follows:

$$R_{i,t}[k] = R_i[k] \cdot R_{i+1}[k] \cdots R_{i+t-1}[k]$$



■ **Figure 4** The two cases considered when querying. The actual LCE is shown as the black square. $\text{LCE}((i_1 + h, j_1 + h), (i_2 + h, j_2 + h))$ is shown with a green square. The LCE of slabs are shown in red and blue. Further binary search is necessary in Case (b).

$$C_{j,t}[k] = C_j[k] \cdot C_{j+1}[k] \cdots C_{j+t-1}[k].$$

We call these *meta characters*. We also call $R_{i,t}$ and $C_{j,t}$ *slabs* of length t . Applying the structure from Lemma 6 over the concatenation of rows and the concatenation of columns, we can compare two meta characters in $\mathcal{O}(1)$ time.

Data Structure. In addition to the previous components, we maintain the structure from Lemma 7 over the text obtained by concatenating $R_{i,t}$ for $i \in [0..n]$ and $t = 1, 2, 4, 8, \dots, \min(n - i, 2^{\lceil \log d \rceil})$. We also maintain the structure from Lemma 7 over the text obtained by concatenating $C_{j,t}$ for $j \in [0..n]$ and $t = 1, 2, 4, 8, \dots, \min(n - j, 2^{\lceil \log d \rceil})$. We leave the parameter τ appearing in Lemma 7 to be optimized later.

Querying. Given an LCE query $(i_1, j_1), (i_2, j_2)$, we first find an $h \in [0..d]$ such that $(i_1 + h, j_1 + h)$ and $(i_2 + h, j_2 + h)$ are sampled positions. We then decompose the interval $[i_1..i_1 + h]$ and $[j_1..j_1 + h]$ into $\mathcal{O}(\log d)$ slabs that have lengths that are powers of two. We perform an LCE query for each corresponding slab for both rows and columns. A minimum is taken over all these LCE values and $h + \text{LCE}((i_1 + h, j_1 + h), (i_2 + h, j_2 + h))$. Denote this minimum with m . There are two possible cases.

- $m > h$. See Figure 4a. In this case, m is reported as the result.
- $m \leq h$. See Figure 4b. In this case, we still need to find the largest value y such that the minimum LCE of the slabs covering $C_{j_1}[i_1..], \dots, C_{j_1+y}[i_1..]$ (with slabs covering $C_{j_2}[i_2..], \dots, C_{j_2+y}[i_2..]$, respectively) and $R_{i_1}[j_1..], \dots, R_{i_1+y}[j_1..]$ (with slabs covering $R_{i_2}[j_2..], \dots, R_{i_2+y}[j_2..]$, respectively) is at least y . To accomplish this, we perform a modified binary search while keeping track of the minimum LCE values for both the column and row slabs. The only difference compared to standard binary search is that rather than always dividing the current range under consideration in half, we consider the power of two closest to half the size of the current range. This is done to ensure that we always use slabs for which we have prepared LCE data structures.

Analysis. Letting $T(l)$ be the number of LCE queries on slabs for the binary search on a range of length l , the resulting recurrence is

$$T(l) \leq T(2^{\lceil \log l/2 \rceil}) + 1 = \mathcal{O}(\log l).$$

Hence, $T(h) = \mathcal{O}(\log d)$. We now fix $\tau = \log_\sigma n \cdot \log \log_\sigma n$. Since each LCE query on a slab takes $\mathcal{O}(\tau)$ time, the overall query time is $\tau \cdot \log d = \mathcal{O}(\log_\sigma n \cdot (\log \log_\sigma n)^2)$, where we used that $d = \Theta(\log_\sigma^2 n)$. The total added space relative to the previous solution is $\mathcal{O}(\log d \cdot n^2/\tau)$ words. Using our definitions of d and τ , the space remains $\mathcal{O}(n^2 \log \sigma)$ bits.

2.3 Achieving $\mathcal{O}(\log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{5/3})$ Query Time

Data Structure. Let x be a parameter to be defined later. In addition to the previously defined diagonal sample positions, we now define sample positions for the rows and columns using an x -cover, denoted by \mathcal{X} . We maintain the structure in Lemma 7 (with parameter τ left open for optimizing later) over the text obtained by concatenating slabs $R_{i,t}$ for $t = 1, 2, 4, 8, \dots, \min(n - i, 2^{\lceil \log d \rceil})$, whenever $i \in \mathcal{X}$. We do the same for slabs $R_{i,t}$ for $t = 1, 2, 4, 8, \dots, \min(2^{\lceil \log d \rceil})$ whenever $i + t - 1 \in \mathcal{X}$ and $i \geq 0$. Similarly, we maintain the structure from Lemma 7 for the concatenation of $C_{j,t}$ for $t = 1, 2, 4, 8, \dots, \min(2^{\lceil \log d \rceil})$ for $j \in \mathcal{X}$. We do the same for $C_{j,t}$ for $t = 1, 2, 4, 8, \dots, \min(2^{\lceil \log d \rceil})$ whenever $j + t - 1 \in \mathcal{X}$ and $j \geq 0$. Note that these slabs do not need to be explicitly constructed and can be simulated directly using the input text.

Querying. Given a query $(i_1, j_1), (i_2, j_2)$, we first find $h \in [0..d]$ such that $(i_1 + h, j_1 + h)$ and $(i_2 + h, j_2 + h)$ are diagonal sample positions. Let find $y \in [0..x]$ such that $i_1 + y$ and $i_2 + y$ are in \mathcal{X} . We find the LCEs of columns $C_{i_1}[j_1..], \dots, C_{i_1+y-1}[j_1..]$ with $C_{i_2}[j_2..], \dots, C_{i_2+y-1}[j_2..]$, respectively. We next find $y' \in [0..x]$ such that $i_1 + h - 1 - y'$ and $i_2 + h - 1 - y'$ are in \mathcal{X} . We then find the LCEs of columns $C_{i_1+h-y'}[j_1..], \dots, C_{i_1+h-1}[j_1..]$, with $C_{i_2+h-y'}[j_2..], \dots, C_{i_2+h-1}[j_2..]$, respectively. We then take the largest power of two, say 2^a , such that $i_1 + y + 2^a \leq i_1 + h - 1 - y'$, and obtain the LCE of the slab $C_{i_1+y,2^a}[j_1..]$ with $C_{i_2+y,2^a}[j_2..]$. We also obtain the LCE of the slabs $C_{i_1+h-y'-2^a,2^a}[j_1..]$ and $C_{i_2+h-y'-2^a,2^a}[j_2..]$. We perform a symmetric procedure on the rows. A minimum is taken among all of these LCE values as well as $h + \text{LCE}((i_1 + h, j_1 + h), (i_2 + h, j_2 + h))$. Let m denote this minimum. We consider two cases like in Section 2.2.

- $m > h$. In this case, m is reported as the result.
- $m \leq h$. As in Section 2.2, we want to output the largest value y such that the minimum LCE of the slabs covering $C_{j_1}[i_1..], \dots, C_{j_1+y}[i_1..]$ (with LCE relative to slabs covering $C_{j_2}[i_2..], \dots, C_{j_2+y}[i_2..]$) and $R_{i_1}[j_1..], \dots, R_{i_1+y}[j_1..]$ (with LCE relative to slabs covering $R_{i_2}[j_2..], \dots, R_{i_2+y}[j_2..]$) is at least y . The modification to the binary search algorithm from Section 2.2 is that we intermix at most x single row/column evaluations to reach the next position in \mathcal{X} . After this position in \mathcal{X} is reached, the power of two that most evenly splits the remaining range can be used.

Analysis. We claim that answering a query requires $\mathcal{O}(x \cdot \log d)$ number of LCE queries for single rows/columns and $\mathcal{O}(\log d)$ number of LCE queries on slabs. To see this, let $S(l)$ be the number of single row/column LCE queries on a range of length l , and $T(l)$ be the number of slab LCE queries. Then we have

$$S(l) \leq S(2^{\lceil \log l/2 \rceil}) + \mathcal{O}(x) = \mathcal{O}(x \log l)$$

$$T(l) \leq T(2^{\lceil \log l/2 \rceil}) + 1 = \mathcal{O}(\log l).$$

Hence, $S(h) = \mathcal{O}(x \log d)$ and $T(h) = \mathcal{O}(\log d)$. Each single row/column LCE query takes $\mathcal{O}(1)$ time and each LCE query on a slab takes $\mathcal{O}(\tau)$ time. As a result, the total query time is $\mathcal{O}(x \cdot \log d + \log d \cdot \tau)$. To optimize, we keep $d = \Theta(\log_\sigma^2 n)$ and now fix $x = \tau = (\log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{2/3})$ and obtain the query time of $\mathcal{O}(\log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{5/3})$.

The (extra) space is $\mathcal{O}(\log d \cdot n^2 / (\sqrt{x} \cdot \tau))$ words. This is because we take $\mathcal{O}(\log d)$ larger slabs for each column/row sample position, creating an overall string of length $\mathcal{O}(\log d \cdot n^2 / \sqrt{x})$. The LCE structure from Lemma 7, then occupies $\mathcal{O}(\log d \cdot n^2 / (\sqrt{x} \cdot \tau))$ words. With the above choice of x and τ , the total space is $\mathcal{O}(n^2 \log \sigma)$ bits. This completes the proof of Theorem 1.

3 Repetition-Aware LCE Data Structure

Overview. We use a parameter τ that we will optimize over later. We aim to use a truncated suffix tree in conjunction with a sparse suffix tree on sampled positions from a τ -cover to efficiently perform LCE queries. If we truncate the 2D suffix tree at a string depth of τ , then the δ measure provides an upper bound of $\tau^2 \delta$ on the number of leaves at depth τ . As we argue, one can also upper bound the number of additional leaves in the truncated suffix tree in terms of τ and n .

The first challenge in using the above ideas is that, for these LCE queries from sampled positions to provide information on the overall LCE result, the matching submatrices starting at sampled positions should overlap. This is accomplished by using a string depth of 2τ for the truncated suffix tree while still using a τ -cover. The second challenge is that given our LCE query, we need to know which leaves to consider in the truncated suffix tree. Moreover, we should accomplish this in $o(n^2)$ space when δ is small. To this end, we introduce the notion of a *macro-matrix* M , which stores the leaf in the truncated suffix tree to examine for a specified position in T . We then relate the δ measure of this macro-matrix to the δ measure of the matrix T . This relationship enables us to use the 2D block tree data structure of Brisboa et al. [3] on M , which occupies sublinear space for compressible matrices and supports efficient access to the elements of M .

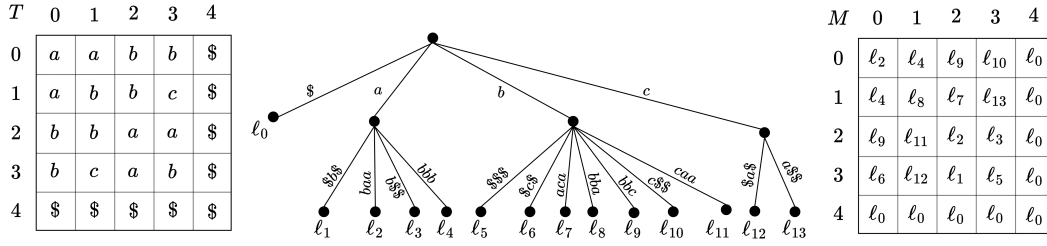
3.1 Data Structures

Truncated Suffix Tree. We first construct a 2D suffix tree of T truncated at a string depth of 2τ . Call this $\mathcal{T}_{\leq 2\tau}$. We use ℓ_1, ℓ_2, \dots to denote the leaves of $\mathcal{T}_{\leq 2\tau}$.

Compressed Representation of Macro-Matrix. We next define the *macro-matrix*. The elements of a macro-matrix are essentially meta symbols, where two meta-symbols are the same if and only if the corresponding $2\tau \times 2\tau$ square substrings are identical. Formally, the macro-matrix M is the matrix obtained as follows: for $i, j \in [0..n)$,

- if there exists a $2\tau \times 2\tau$ matrix with upper left corner (i, j) , i.e., $i, j \leq n - 2\tau$, then we make $M[i][j] = \ell$ where ℓ is a pointer to the leaf of $\mathcal{T}_{\leq 2\tau}$ corresponding to $T[i..i + 2\tau - 1][j..j + 2\tau - 1]$;
- if $i > n - 2\tau$ or $j > n - 2\tau$, then let $M[i][j] = \ell$ where ℓ is a pointer to the leaf in $\mathcal{T}_{\leq 2\tau}$ corresponding to the $(n - \max(i, j)) \times (n - \max(i, j))$ matrix with upper left corner (i, j) .

See Figure 5. We then construct the 2D block tree of M , denoted as $\text{BT}(M)$.



■ **Figure 5** An example 2D text T , a truncated suffix tree with $\tau = 1$, i.e., truncated at a string depth of $2\tau = 2$, and the resulting macro-matrix M .

Sparse Suffix Tree. We define sample positions based on a τ -cover \mathcal{C} of $[0..n)$. These consist of sample positions for the rows,

$$\mathcal{C}_R = \{(i, j) \mid i \in \mathcal{C}, j \in [0..n)\}$$

for the columns,

$$\mathcal{C}_C = \{(i, j) \mid i \in [0..n), j \in \mathcal{C}\}$$

and for the diagonals,

$$\mathcal{C}_D = \{(i, j) \mid i, j \in [0..n), \min(i, j) \in \mathcal{C}\}.$$

Let $\mathcal{C}' = \mathcal{C}_R \cup \mathcal{C}_C \cup \mathcal{C}_D$. Observe that $|\mathcal{C}'| = \Theta(n^2/\sqrt{\tau})$. We build a sparse suffix tree over the suffixes starting at sampled positions in \mathcal{C}' , denoted as \mathcal{T}_s . We also maintain the lookup data structure from Lemma 8. As before, this allows us to find in $\mathcal{O}(1)$ time equally far sampled positions at most τ away from the queried positions in each row, column, and diagonal.

3.2 Querying

Given LCE query $(i_1, j_1), (i_2, j_2)$, we first use $\text{BT}(M)$ to get the corresponding values in M . Say these correspond to the leaves ℓ_1 and ℓ_2 in $\mathcal{T}_{\leq 2\tau}$ respectively. If $\ell_1 \neq \ell_2$, then the string depth of the LCA of ℓ_1 and ℓ_2 gives us the LCE of $(i_1, j_1), (i_2, j_2)$.

If $\ell_1 = \ell_2$ then we use the lookup data structure from Lemma 8 to find:

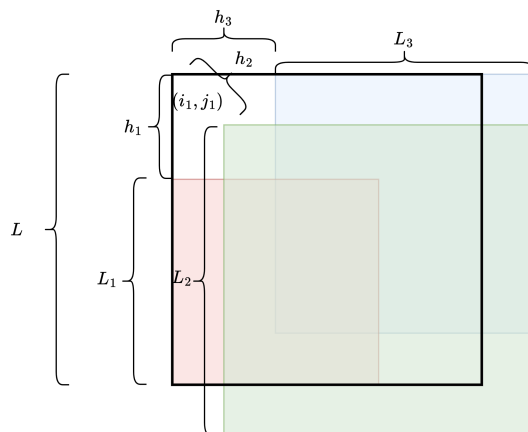
- $h_1 \in [0.. \tau)$ such that $(i_1 + h_1, j_1)$ and $(i_2 + h_1, j_2)$ are sampled positions. We then use an $\mathcal{O}(1)$ time query on \mathcal{T}_s to get the LCE of $(i_1 + h_1, j_1)$ and $(i_2 + h_1, j_2)$. Denote this LCE value as L_1 .
- $h_2 \in [0.. \tau)$ such that $(i_1 + h_2, j_1 + h_2)$ and $(i_2 + h_2, j_2 + h_2)$ are sampled positions. We use an $\mathcal{O}(1)$ time query on \mathcal{T}_s to get the LCE of $(i_1 + h_2, j_1 + h_2)$ and $(i_2 + h_2, j_2 + h_2)$. Denote this LCE value as L_2 .
- $h_3 \in [0.. \tau)$ such that $(i_1, j_1 + h_3)$ and $(i_2, j_2 + h_3)$ are sampled positions. We use an $\mathcal{O}(1)$ time query on \mathcal{T}_s to get the LCE of $(i_1, j_1 + h_3)$ and $(i_2, j_2 + h_3)$. Denote this LCE value as L_3 .

We report $\min(h_1 + L_1, h_2 + L_2, h_3 + L_3)$ as the solution.

3.3 Correctness

The first lemma is immediate.

► **Lemma 9.** *When $\ell_1 \neq \ell_2$, $\text{LCE}((i_1, j_1), (i_2, j_2))$ is the string depth of $\text{LCA}(\ell_1, \ell_2)$.*



■ **Figure 6** The solution LCE L is shown as the black square. Submatrix T_1 matrix in red, submatrix T_2 in green, and submatrix T_3 matrix in blue.

► **Lemma 10.** *When $\ell_1 = \ell_2$, $\text{LCE}((i_1, j_1), (i_2, j_2)) = \min(h_1 + L_1, h_2 + L_2, h_3 + L_3)$.*

Proof. Define $L := \text{LCE}((i_1, j_1), (i_2, j_2))$. First, we show that $L \leq \min(h_1 + L_1, h_2 + L_2, h_3 + L_3)$. Starting from $(i_1, j_1 + h_1)$ there exists a matching submatrix (with respect to position $(i_2, j_2 + h_1)$) of size at least $L - h_1$, thus we have that $L_1 \geq L - h_1$. Hence, $L_1 + h_1 \geq L$. A similar argument holds for h_2 and h_3 .

Next, we show $L \not\leq \min(h_1 + L_1, h_2 + L_2, h_3 + L_3)$.

- We denote the submatrix $T[i_1 + h_1 \dots i_1 + h_1 + L_1][j_1 \dots j_1 + L_1]$ as T_1 .
- We denote the submatrix $T[i_1 + h_2 \dots i_1 + h_2 + L_2][j_1 + h_2 \dots j_1 + h_2 + L_2]$ as T_2 .
- We denote the submatrix $T[i_1 \dots i_1 + L_3][j_1 + h_3 \dots j_1 + h_3 + L_3]$ as T_3

See Figure 6.

Observe that $h_1, h_2, h_3 \leq \tau - 1$ and since $L \geq 2\tau$, we have $L_1, L_2, L_3 \geq \tau$. Submatrix T_2 has lower left corner in column $j_1 + h_2 \leq j_1 + L_1 - 1$ and in row $i_1 + h_2 + L_2 - 1 \geq i_1 + h_1$ making it overlap with T_1 . Also, T_2 has upper right corner in column $j_1 + h_2 + L_2 - 1 \geq j_1 + h_3$ and row $i_1 + h_2 \leq i_1 + h_3 + L_3 - 1$. Hence, T_2 overlaps with T_3 as well.

Now, suppose for the sake of contradiction that $h_1 + L_1, h_2 + L_2, h_3 + L_3 > L$. For any positions in row $x = i_1 + L$ and column y where $j_1 \leq y \leq j_1 + L$ we have

$$i_1 \leq x = i_1 + L \leq i_1 + h_1 + L_1 - 1, \quad i_1 + h_2 + L_2 - 1$$

and

$$j_1 \leq y \leq j_1 + L \leq j_1 + h_1 + L_1 - 1, \quad j_1 + h_2 + L_2 - 1.$$

Similarly, for any position in column $y = j_1 + L$ and row x where $i_1 \leq x \leq i_1 + L$ we have

$$j_1 \leq y = j_1 + L \leq j_1 + h_2 + L_2 - 1, \quad j_1 + h_3 + L_3 - 1$$

and

$$i_1 \leq x \leq i_1 + L \leq i_1 + h_2 + L_2 - 1, \quad i_1 + h_3 + L_3 - 1.$$

Based on the above inequalities and the fact that submatrices T_1 , T_2 , and T_3 overlap, this implies that the matching submatrices with upper left corners (i_1, j_1) and (i_2, j_2) can be extended further by at least one row and column. This contradicts the definition of L . ◀

3.4 Analysis and Optimization

3.4.1 Space Analysis

Space for τ -Cover lookup structure and Sparse Suffix Tree. According to Lemma 8, the lookup structure requires $\mathcal{O}(\tau)$ space. Since $|\mathcal{C}'| = \mathcal{O}(n^2/\sqrt{\tau})$, we have that the sparse suffix tree \mathcal{T}_s uses $\mathcal{O}(n^2/\sqrt{\tau})$ space.

Space for $\mathcal{T}_{\leq 2\tau}$. The space for the truncated suffix tree $\mathcal{T}_{\leq 2\tau}$ is bounded by the number of distinct $2\tau \times 2\tau$ submatrices of T , denoted $d_{2\tau}(T)$, plus the number of distinct matrices of size less than 2τ that can not be further extended down and to the right (due to a boundary of T). There are at most $\mathcal{O}(\tau n)$ of the latter since, for every length from 1 to 2τ , at most $2n$ submatrices cannot be further extended. By the definition of δ , $d_{2\tau}(T) \leq 4\tau^2\delta(T)$, making the space for $\mathcal{T}_{\leq 2\tau}$ bound by $\mathcal{O}(\tau^2\delta(T) + \tau n)$.

Space for Macro-Matrix. The space for $\text{BT}(M)$ depends on $\delta(M)$. We prove the following lemma relating $\delta(T)$ and $\delta(M)$.

► **Lemma 11.** $\delta(M) = \Omega(\max(1, \delta(T)/\tau^2 - n/\tau))$ and $\delta(M) = \mathcal{O}(\tau^2\delta(T) + \tau n)$.

Proof. First, the lower bound. Observe that for an arbitrary $t \in [2\tau..n]$, two matching $t \times t$ submatrices in T cause two matching $(t - 2\tau + 1) \times (t - 2\tau + 1)$ submatrices in M (with the same upper left corners as the corresponding submatrices in T). In this way, every distinct $t \times t$ submatrix in T maps to one distinct $(t - 2\tau + 1) \times (t - 2\tau + 1)$ submatrix in M , and we have $d_t(T) \leq d_{(t-2\tau+1)}(M)$. Then for $t \geq 2\tau$, we have

$$\frac{d_t(T)}{t^2} \leq \frac{d_{(t-2\tau+1)}(M)}{t^2} \leq \frac{(t-2\tau+1)^2\delta(M)}{t^2} \leq \delta(M) \quad (1)$$

implying $d_t(T) \leq t^2\delta(M)$ for $t \geq 2\tau$.

Next, consider $t \in [1..2\tau)$. Note that the number of distinct $t \times t$ submatrices in T is almost upper bounded by the number of distinct $(t + 2\tau) \times (t + 2\tau)$ submatrices in T , except that some of the distinct matrices with sizes between $t \times t$ and $(t + 2\tau) \times (t + 2\tau)$ may be prevented from being extended due to the right and bottom boundaries of T . The number of such submatrices is bounded by $2n(t + 2\tau - t) = \mathcal{O}(\tau n)$. Hence, for $t < 2\tau$,

$$d_t(T) \leq d_{(t+2\tau)}(T) + \mathcal{O}(\tau n)$$

Applying Inequality (1), we can then write

$$\frac{d_t(T)}{t^2} \leq \frac{d_{(t+2\tau)}(T)}{t^2} + \frac{\mathcal{O}(\tau n)}{t^2} \leq \frac{(t+2\tau)^2}{t^2}\delta(M) + \mathcal{O}(\tau n) = \mathcal{O}(\tau^2\delta(M) + \tau n).$$

Taking the maximum over both cases, yields that $\delta(T) = \mathcal{O}(\tau^2\delta(M) + \tau n)$.

For the upper bound, we claim that, for an arbitrary $t \in [1..n]$,

$$d_t(M) \leq d_{(t+2\tau-1)}(T) + \mathcal{O}(\tau n),$$

where we take $d_{(t+2\tau-1)}(T) = 0$ if $t + 2\tau - 1 > n$. The above inequality follows from the fact that every distinct $(t + 2\tau - 1) \times (t + 2\tau - 1)$ submatrix in T maps to one distinct $t \times t$ submatrix in M . What remains to be counted for $d_t(M)$ are distinct $t \times t$ submatrices in M that are not resulting from some $(t + 2\tau - 1) \times (t + 2\tau - 1)$ submatrix in T . That is, submatrices on the bottom and/or right boundary. Again, the number of such $t \times t$ submatrices is bounded by $2n((t + 2\tau - 1) - t) = \mathcal{O}(\tau n)$.

To complete the proof, we have the bound

$$\begin{aligned} \delta(M) &= \max_t \frac{d_t(M)}{t^2} \leq \max_t \frac{d_{(t+2\tau-1)}(T) + \mathcal{O}(\tau n)}{t^2} \\ &\leq \max_t \frac{(t+2\tau-1)^2 \delta(T) + \mathcal{O}(\tau n)}{t^2} = \mathcal{O}(\tau^2 \delta(T) + \tau n). \quad \blacktriangleleft \end{aligned}$$

Let σ' be the alphabet size of the macro-matrix M . The space for the block tree $\text{BT}(M)$ is

$$\mathcal{O} \left((\delta(M) + \sqrt{n\delta(M)}) \log \left(\frac{n \log \sigma'}{\sqrt{\delta(M)} \log n} \right) \right).$$

Applying that $\sigma' \leq n^2$ and Lemma 11, this space is bound by

$$\mathcal{O} \left((\tau^2 \delta(T) + \tau \sqrt{n\delta(T)} + \tau n) \log \left(\frac{n}{\sqrt{\max(1, \frac{\delta(T)}{\tau^2} - \frac{n}{\tau})}} \right) \right).$$

Total Space. Summing the total data structure sizes, the combined space is

$$\mathcal{O} \left((\tau^2 \delta(T) + \tau \sqrt{n\delta(T)} + \tau n) \log \left(\frac{n}{\sqrt{\max(1, \frac{\delta(T)}{\tau^2} - \frac{n}{\tau})}} \right) + \frac{n^2}{\sqrt{\tau}} + \tau \right).$$

3.4.2 Optimizing

We consider two cases based on $\delta(T)$, which we now denote as just δ . If $\delta > n^{1/3}$, we set $\tau = \lceil n^{4/5} / (2\delta^{2/5}) \rceil$ and let $\beta = n / \sqrt{\max(1, 4\delta^{9/5} / n^{8/5} - 2n^{1/5} \delta^{2/5})}$. The space is (up to constant factors)

$$\left(n^{8/5} \delta^{1/5} + n^{13/10} \delta^{1/10} + \frac{n^{9/5}}{\delta^{2/5}} \right) \log \beta + n^{8/5} \delta^{1/5} + \frac{n^{4/5}}{\delta^{2/5}} = \mathcal{O} \left(n^{8/5} \delta^{1/5} \cdot \log \beta \right).$$

Observe that as δ approaches n^2 , β approaches $\mathcal{O}(1)$.

If $\delta \leq n^{1/3}$, we make $\tau = n^{2/3}$. The resulting space complexity is

$$(n^{4/3} \delta + n^{7/6} \sqrt{\delta} + n^{5/3}) \log \beta + n^{5/3} + n^{2/3} = \mathcal{O} \left(n^{5/3} \log \beta \right).$$

For this case, the argument of the logarithm β is $\mathcal{O}(n)$. One can also readily check that β as defined above is bound by the expression for β appearing in Theorem 2.

3.4.3 Query Time

The query time is dominated by the access to $\text{BT}(M)$, which takes $1 + \log \frac{n}{\sqrt{\delta(M)}} = \mathcal{O}(1 + \log \beta)$ time, where β is defined as above. The remaining queries take $\mathcal{O}(1)$ time. This completes the proof of Theorem 2.

4 Applications

We next demonstrate some applications of Theorem 1 by proving Theorems 3, 4, 5.

4.1 ISA Queries

We maintain a sampled suffix array. Specifically, we sample the suffix array values for every $(\log_\sigma n)$ leaf of the suffix tree. The space required for this is $\mathcal{O}(n^2 \log \sigma)$ bits. Additionally, for each text position, we maintain how far away its predecessor sampled leaf is relative to its leaf in the suffix tree. This requires $\mathcal{O}(\log \log_\sigma n)$ bits per entry. The resulting total space is $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ bits.

To find the ISA value of a text position (i, j) , we perform a binary search on the sampled leaves to find the lexicographic predecessor of (i, j) within the sampled set. Once the predecessor is found, we add the offset associated with (i, j) . This gives us the suffix array position associated with (i, j) , i.e., its ISA value. The binary search requires $\mathcal{O}(\log n)$ number of LCE queries. Each LCE query takes $\mathcal{O}(\log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{5/3})$ time, resulting in an overall time complexity of $\mathcal{O}(\log n \cdot \log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{5/3})$.

4.2 SA queries

Let τ be a parameter. We divide the leaves of the suffix tree into contiguous blocks of size $\lceil n^2/\tau \rceil$ (except for perhaps the last block, which can be smaller). There are $\Theta(\tau)$ blocks. We associate each position in T with the block in which its leaf lies in the suffix tree. This information is stored as follows: consider a binary array B_b associated with each block b . Each binary array is of length n^2 and represents a linearization of T . For a block B_b , we consider a 1 in a position if the corresponding suffix tree leaf is in block b and 0 otherwise. Note that there are at most $m := \lceil n^2/\tau \rceil$ 1's in B_b . We build a data structure representing B_b using $m \log \frac{n^2}{m} + \mathcal{O}(m)$ bits of space, or equivalently, $n^2/\tau \cdot \log \tau + \mathcal{O}(n^2/\tau)$ bits of space, such that select queries can be answered in constant time [29]. The total space for select data structures over all $\Theta(\tau)$ bit vectors, is $n^2 \log \tau + \mathcal{O}(n^2) = \mathcal{O}(n^2 \log \tau)$ bits. We also maintain the ISA data structure described previously.

Given an SA query for index i , we first identify which block i is in. Say this is block b . We use select queries to iterate through the text positions contained in block b . For each text position iterated over, we perform an ISA query and check whether its ISA value equals i .

The space required for the ISA data structure is $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ bits. The space for the select data structures is $\mathcal{O}(n^2 \log \tau)$ bits. The query time is $\mathcal{O}(n^2/\tau \cdot \log n \cdot \log_\sigma^{2/3} n \cdot (\log \log_\sigma n)^{5/3})$. We obtain Theorem 4 by making $\tau = (\sigma \log n)^c$, where c is an arbitrarily large constant that can absorb the additional logarithmic factors in the query time.

4.3 Pattern Matching

Counting. In addition to the previous structures, we maintain the LCE data structure from Lemma 6 over the rows and columns. First, a binary search is done to find the leaf for the lexicographically smallest suffix with P as a prefix (if one exists). We start by using an SA query to obtain $\text{SA}[\lceil n^2/2 \rceil]$. Using that we have read access to the original text, we match characters in P in Lstring order to the submatrix starting at $\text{SA}[\lceil n^2/2 \rceil]$ until we reach our first mismatch. At this point, we know our lexicographical order relative to our current leaf. When we transition to a new leaf in the binary search, we perform an SA query followed by LCE queries with the position from the preceding leaf. This avoids repeatedly iterating over characters in P . Assuming the LCE query is at least the length already matched, we continue matching from the last matched position. A similar binary search finds the lexicographically largest suffix with P as a prefix. We return the suffix range length.

The total number of LCE and SA queries performed is $\mathcal{O}(\log n)$. The time is dominated by the SA queries, which require $\mathcal{O}(n^2/(\sigma \log n)^c)$ time.

Reporting. We start with the suffix range obtained previously, say $[x..y]$. We use the same blocking scheme for the suffix leaves described for SA queries, also using constant time select data structures. We first identify the block that x lies in, say B_b . We use the select data structure to iterate through all of the text positions corresponding to suffixes in block b . For each position, we perform an ISA query and check whether its position in the suffix array is at least x . If it is, we output it. We perform a similar procedure for the block containing y , now checking if the position in the suffix array is at most y . For the remaining blocks, those completely contained in $[x..y]$, we use their select data structures to output all occurrences with suffixes in that block.

The space complexity is the same as the SA data structure. For the query time, each block has size $\mathcal{O}(n^2/\tau)$, and with $\tau = (\sigma \log n)^c$, the time spent on the blocks containing x and y is absorbed by $n^2/(\sigma \log n)^c$ already appearing due to SA queries.

5 Open Problems

We leave open many directions for potential improvement, for example:

- Can we design a data structure with faster SA query time that uses $\mathcal{O}(n^2 \log \sigma + n^2 \log \log n)$ bits of space (or better)? This seems significantly harder than ISA queries. Suffix array sampling, like in the FM-index [10], is not immediately adaptable.
- Can we design a data structure in repetition-aware compressed space that supports ISA, SA, or pattern-matching queries? Also, can the space for a data structure for LCE queries be improved? Grammar-based compression has proven useful for repetition-aware compressed data structures supporting LCE queries in the 1D case, particularly run-length straight-line programs (RL-SLP). For 1D text, it is possible to construct RL-SLPs with size close to the δ measure [25], which can be used for LCE [27] and pattern matching queries [24]. Although Romana et al. [31] introduce a version of RL-SLP for 2D text, it is open how such a RL-SLP could be utilized for LCE queries and other types of queries, e.g., SA and pattern matching queries.

References

- 1 Philip Bille, Inge Li Gørtz, Mathias Bæk Tejs Knudsen, Moshe Lewenstein, and Hjalte Wedel Vildhøj. Longest common extensions in sublinear space. In Ferdinando Cicalese, Ely Porat, and Ugo Vaccaro, editors, *Combinatorial Pattern Matching - 26th Annual Symposium, CPM 2015, Ischia Island, Italy, June 29 - July 1, 2015, Proceedings*, volume 9133 of *Lecture Notes in Computer Science*, pages 65–76. Springer, 2015. doi:10.1007/978-3-319-19929-0_6.
- 2 Philip Bille, Inge Li Gørtz, Benjamin Sach, and Hjalte Wedel Vildhøj. Time-space trade-offs for longest common extensions. *J. Discrete Algorithms*, 25:42–50, 2014. doi:10.1016/J.JDA.2013.06.003.
- 3 Nieves R. Brisaboa, Travis Gagie, Adrián Gómez-Brandón, and Gonzalo Navarro. Two-dimensional block trees. *Comput. J.*, 67(1):391–406, 2024. doi:10.1093/COMJNL/BXAC182.
- 4 Stefan Burkhardt and Juha Kärkkäinen. Fast lightweight suffix array construction and checking. In Ricardo A. Baeza-Yates, Edgar Chávez, and Maxime Crochemore, editors, *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocán, Mexico, June 25-27, 2003, Proceedings*, volume 2676 of *Lecture Notes in Computer Science*, pages 55–69. Springer, 2003. doi:10.1007/3-540-44888-8_5.
- 5 Lorenzo Carfagna and Giovanni Manzini. Compressibility measures for two-dimensional data. In Franco Maria Nardini, Nadia Pisanti, and Rossano Venturini, editors, *String Processing and Information Retrieval - 30th International Symposium, SPIRE 2023, Pisa, Italy, September 26-28, 2023, Proceedings*, volume 14240 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2023. doi:10.1007/978-3-031-43980-3_9.

- 6 Lorenzo Carfagna and Giovanni Manzini. The landscape of compressibility measures for two-dimensional data. *IEEE Access*, 12:87268–87283, 2024. doi:10.1109/ACCESS.2024.3417621.
- 7 Anders Roy Christiansen, Mikko Berggren Ettienne, Tomasz Kociumaka, Gonzalo Navarro, and Nicola Prezza. Optimal-time dictionary-compressed indexes. *ACM Trans. Algorithms*, 17(1):8:1–8:39, 2021. doi:10.1145/3426473.
- 8 Charles J. Colbourn and Alan C. H. Ling. Quorums from difference covers. *Inf. Process. Lett.*, 75(1-2):9–12, 2000. doi:10.1016/S0020-0190(00)00080-6.
- 9 Martin Farach. Optimal suffix tree construction with large alphabets. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 137–143. IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646102.
- 10 Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 390–398. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892127.
- 11 Travis Gagie, Gonzalo Navarro, and Nicola Prezza. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *J. ACM*, 67(1):2:1–2:54, 2020. doi:10.1145/3375890.
- 12 Arnab Ganguly, Dhruvil Patel, Rahul Shah, and Sharma V. Thankachan. LF successor: Compact space indexing for order-isomorphic pattern matching. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 71:1–71:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.71.
- 13 Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. pbwt: Achieving succinct data structures for parameterized pattern matching and related problems. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 397–407. SIAM, 2017. doi:10.1137/1.9781611974782.25.
- 14 Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. Fully functional parameterized suffix trees in compact space. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 65:1–65:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.65.
- 15 Pawel Gawrychowski, Tomasz Kociumaka, Wojciech Rytter, and Tomasz Walen. Faster longest common extension queries in strings over general alphabets. In Roberto Grossi and Moshe Lewenstein, editors, *27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, June 27-29, 2016, Tel Aviv, Israel*, volume 54 of *LIPICs*, pages 5:1–5:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.CPM.2016.5.
- 16 Raffaele Giancarlo. A generalization of the suffix tree to square matrices, with applications. *SIAM J. Comput.*, 24(3):520–562, 1995. doi:10.1137/S0097539792231982.
- 17 Gaston H Gonnet. *Efficient searching of text and pictures*. UW Centre for the New Oxford English Dictionary, 1990.
- 18 Roberto Grossi and Jeffrey Scott Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM J. Comput.*, 35(2):378–407, 2005. doi:10.1137/S0097539702402354.
- 19 Dominik Kempa and Tomasz Kociumaka. String synchronizing sets: sublinear-time BWT construction and optimal LCE data structure. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 756–767. ACM, 2019. doi:10.1145/3313276.3316368.
- 20 Dominik Kempa and Tomasz Kociumaka. Resolution of the Burrows-Wheeler transform conjecture. *Commun. ACM*, 65(6):91–98, 2022. doi:10.1145/3531445.

- 21 Dominik Kempa and Tomasz Kociumaka. Collapsing the hierarchy of compressed data structures: Suffix arrays in optimal compressed space. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1877–1886. IEEE, 2023. doi:10.1109/FOCS57990.2023.00114.
- 22 Dong Kyue Kim, Yoo Ah Kim, and Kunsoo Park. Generalizations of suffix arrays to multi-dimensional matrices. *Theor. Comput. Sci.*, 302(1-3):223–238, 2003. doi:10.1016/S0304-3975(02)00828-9.
- 23 Dong Kyue Kim, Joong Chae Na, Jeong Seop Sim, and Kunsoo Park. Linear-time construction of two-dimensional suffix trees. *Algorithmica*, 59(2):269–297, 2011. doi:10.1007/S00453-009-9350-Z.
- 24 Tomasz Kociumaka, Gonzalo Navarro, and Francisco Olivares. Near-optimal search time in δ -optimal space, and vice versa. *Algorithmica*, 86(4):1031–1056, 2024. doi:10.1007/S00453-023-01186-0.
- 25 Tomasz Kociumaka, Gonzalo Navarro, and Nicola Prezza. Toward a definitive compressibility measure for repetitive sequences. *IEEE Trans. Inf. Theory*, 69(4):2074–2092, 2023. doi:10.1109/TIT.2022.3224382.
- 26 Gonzalo Navarro. Indexing highly repetitive string collections, part I: repetitiveness measures. *ACM Comput. Surv.*, 54(2):29:1–29:31, 2022. doi:10.1145/3434399.
- 27 Takaaki Nishimoto, Tomohiro I, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Fully dynamic data structure for LCE queries in compressed space. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, volume 58 of *LIPICs*, pages 72:1–72:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.MFCS.2016.72.
- 28 Dhrumil Patel and Rahul Shah. Inverse suffix array queries for 2-dimensional pattern matching in near-compact space. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPICs*, pages 60:1–60:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ISAAC.2021.60.
- 29 Rajeev Raman, Venkatesh Raman, and Srinivasa Rao Satti. Succinct indexable dictionaries with applications to encoding k -ary trees, prefix sums and multisets. *ACM Trans. Algorithms*, 3(4):43, 2007. doi:10.1145/1290672.1290680.
- 30 Sofya Raskhodnikova, Dana Ron, Ronitt Rubinfeld, and Adam D. Smith. Sublinear algorithms for approximating string compressibility. *Algorithmica*, 65(3):685–709, 2013. doi:10.1007/S00453-012-9618-6.
- 31 Giuseppe Romana, Marinella Sciortino, and Cristian Urbina. Exploring repetitiveness measures for two-dimensional strings, 2024. doi:10.48550/arXiv.2404.07030.
- 32 Kunihiko Sadakane. Succinct representations of LCP information and improvements in the compressed suffix arrays. In David Eppstein, editor, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 225–232. ACM/SIAM, 2002. URL: <http://dl.acm.org/citation.cfm?id=545381.545410>.
- 33 Sharma V. Thankachan. Compact text indexing for advanced pattern matching problems: Parameterized, order-isomorphic, 2d, etc. (invited talk). In Hideo Bannai and Jan Holub, editors, *33rd Annual Symposium on Combinatorial Pattern Matching, CPM 2022, June 27-29, 2022, Prague, Czech Republic*, volume 223 of *LIPICs*, pages 3:1–3:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CPM.2022.3.
- 34 Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973*, pages 1–11. IEEE Computer Society, 1973. doi:10.1109/SWAT.1973.13.

A Quasi-Polynomial Time Algorithm for Multi-Arrival on Tree-Like Multigraphs

Ebrahim Ghorbani ✉ 

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Jonah Leander Hoff ✉

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Matthias Mnich ✉ 

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Abstract

Propp machines, or rotor-router models, are a classic tool to simulate random systems in forms of Markov chains by deterministic systems. To this end, the nodes of the Markov chain are replaced by switching nodes, which maintain a queue over their outgoing arcs, and a particle sent through the system traverses the top arc of the queue which is then moved to the end of the queue and the particle arrives at the next node. A key question to answer about such systems is whether a single particle can reach a particular target node, given as input an initial configuration of the queues at all switching nodes. This question was introduced by Dohrau et al. (2017) under the name of ARRIVAL. A major open question is whether ARRIVAL can be solved in polynomial time, as it is known to lie in $\text{NP} \cap \text{co-NP}$; yet the fastest known algorithm for general instances takes subexponential time (Gärtner et al., ICALP 2021).

We consider a generalized version of ARRIVAL introduced by Auger et al. (RP 2023), which requires routing multiple (potentially exponentially many) particles through a rotor graph. The MULTI-ARRIVAL problem is to determine the particle configuration that results from moving all particles from a given initial configuration to sinks. Auger et al. showed that for path-like rotor graphs with a certain uniform rotor order, the problem can be solved in polynomial time.

Our main result is a quasi-polynomial-time algorithm for MULTI-ARRIVAL on tree-like rotor graphs for arbitrary rotor orders. Tree-like rotor graphs are directed multigraphs which can be obtained from undirected trees by replacing each edge by an arbitrary number of arcs in either or both directions. For trees of bounded contracted height, such as paths, the algorithm runs in polynomial time and thereby generalizes the result by Auger et al.. Moreover, we give a polynomial-time algorithm for MULTI-ARRIVAL on tree-like rotor graphs without parallel arcs.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Arrival, Rotor-routing, Tree-like Multigraph, Path-Like Multigraph, Fixed-Parameter Tractability

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.39

Funding Funded by the project “Hamburg Quantum Computing”, co-financed by ERDF and Fonds of the Hamburg Ministry of Science, Research, Equalities and Districts (BWFGB).

1 Introduction

In 2017, Dohrau et al. [3] introduced the ARRIVAL problem, which they intuitively defined as follows:

“Suppose that a train is running along a railway network, starting from a designated origin, with the goal of reaching a designated destination. The network, however, is of a special nature: every time the train traverses a switch, the switch will change its position immediately afterwards. Hence, the next time the train traverses the



© Ebrahim Ghorbani, Jonah Leander Hoff, and Matthias Mnich;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 39; pp. 39:1–39:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



same switch, the other direction will be taken, so that directions alternate with each traversal of the switch. Given a network with origin and destination, what is the complexity of deciding whether the train, starting at the origin, will eventually reach the destination?”

One may solve ARRIVAL by running the train iteratively according to the outlined rules until reaching a destination. This may require a number of steps which is exponential in the number of switches. Gärtner et al. [6] devised an algorithm for ARRIVAL which runs in subexponential time. It is further known that ARRIVAL lies in $\text{NP} \cap \text{co-NP}$ [3], which motivated the conjecture that the problem is polynomial-time decidable. As of now, this conjecture is still open, despite some effort [4, 5, 6].

The routing behaviour in ARRIVAL is also known as *switching networks*, *rotor routing*, *Propp machines*, and *Eulerian walkers* [4, 7, 10]. We adopt the nomenclature of rotor routing and refer to the trains as *particles*. In rotor routing, particles are *routed* from each node along its outgoing arcs in a fixed cyclic order called the *rotor order*. A directed multigraph equipped with a rotor order at each of its nodes is referred to as a *rotor graph*. ARRIVAL can be seen as determining the unique *particle configuration* resulting from routing a single particle until it reaches a sink. As the complexity of ARRIVAL in general is unknown, recent efforts focus on identifying families of instances on which ARRIVAL can be solved in polynomial-time. One such family are *tree-like* rotor graphs, which are rotor graphs whose underlying undirected simple graphs are trees. Note that the class of tree-like rotor graphs is richer than its name may suggest: they can contain arcs in opposite direction, as well as parallel arcs, between the same two nodes. Tree-like rotor graphs are directed multi-graphs that can be obtained from undirected trees by replacing edge $\{u, v\}$ by any number $a_{uv} \geq 0$ of arcs (u, v) and any number $a_{vu} \geq 0$ of arcs (v, u) , where a_{uv} and a_{vu} can be different (not both of them should be zero). On tree-like rotor graphs, ARRIVAL was shown to be solvable in polynomial time [1]. More recently, the case of routing multiple (potentially exponentially many) particles on *path-like* rotor graphs has been studied. MULTI-ARRIVAL refers to the problem of determining the particle configuration resulting from a routing that moves all particles from a given initial configuration to the sinks. This resulting configuration has been proved to be unique [2]. Auger et al. [2] showed that for path-like rotor graphs with a certain uniform rotor order, MULTI-ARRIVAL can be solved in polynomial-time. Results by Gärtner et al. [6] show that polynomial-time algorithms for MULTI-ARRIVAL for a certain family \mathcal{F} of rotor graphs yield polynomial-time algorithms for ARRIVAL for rotor graphs that belong to \mathcal{F} after deleting constantly many nodes.

Our Contributions. Our main contributions are algorithms for MULTI-ARRIVAL on tree-like rotor graphs. To state them, we introduce the notion of *contracted height* $\text{ch}(R)$ of a simple rooted tree R as the minimum height over all trees obtained from R by contracting, for each node v of R , one of the arcs to its children. We now state our main results informally, where $\kappa(T)$ is the maximum possible $\text{ch}(\langle T \rangle)$ over all choices of a root, where $\langle T \rangle$ is the simple undirected underlying tree of T .

► **Theorem 1 (Informal).** MULTI-ARRIVAL can be solved on tree-like rotor graph T with $|A(T)|$ arcs in time $\mathcal{O}^*(\log^{\kappa(T)+1} |A(T)|)$, where \mathcal{O}^* hides factors polynomial in the input. In particular, MULTI-ARRIVAL can be solved in polynomial time if $\kappa(T) = \mathcal{O}\left(\frac{\log |A(T)|}{\log(\log |A(T)|)}\right)$.

We further show that if $\kappa(T) = \mathcal{O}(1)$ (which includes path-like rotor graphs), MULTI-ARRIVAL can be solved in time polynomial in $|\hat{A}(T)|$, where $\hat{A}(T)$ represents the *succinct* encoding of the arcs, only recording the number of consecutive parallel arcs. Therefore, our results widely generalize the previous algorithms by Auger et al. [2] for path-like rotor graphs with uniform rotor order.

As an important corollary, we deduce that MULTI-ARRIVAL on tree-like rotor graphs is fixed-parameter tractable when parameterized by $\kappa(T)$. That is, MULTI-ARRIVAL can be solved in time $f(\kappa(T)) \cdot m^{\mathcal{O}(1)}$ on tree-like rotor graphs of size m . Such fixed-parameter algorithms are considered to be advantageous over so-called XP-algorithms, where the degree of the polynomial depends on the parameter. Indeed, for the related parameter “feedback vertex set size” $\text{fvs}(G)$, which measures the node deletion distance of the rotor graph G to an acyclic digraph, Gärtner et al. [6] provided an XP-algorithm with run time $m^{\mathcal{O}(\text{fvs}(G))}$.

Finally, we show:

► **Theorem 2** (Informal). MULTI-ARRIVAL is solvable in polynomial time for tree-like rotor graphs without parallel arcs.

2 Preliminaries

2.1 Directed Multigraphs

Throughout, we consider *directed multigraphs* G with finite node set $V(G)$ and arc set $A(G)$. An arc with tail v and head w is said to be *from* v *to* w . A directed multigraph may have *parallel* arcs which share the same head-tail pair, and *self-loops* which are from and to the same node.

For a given node $v \in V(G)$, we denote by $A^+(v)$ the set of arcs whose tail is v and define $d^+(v) = |A^+(v)|$. Let $N^+(v)$ be the set of out-neighbours of v and let $N^-(v)$ be the set of in-neighbours of v . By $d(v, w)$ we denote the number of arcs from v to w . For a directed multigraph G , let $V^+(G)$ be the set of nodes with positive out-degree and let $S_0(G)$ be the set of the *sinks*, that is, the nodes with out-degree 0. The *underlying undirected simple graph* of G is the undirected simple graph $\langle G \rangle$ with node set $V(G)$ which contains an edge $\{v, w\}$ whenever there is an arc between v and w . We call G *tree-like* (*path-like*) if $\langle G \rangle$ is a tree (path). A *rooted tree* is obtained from an undirected tree by designating one node as its *root* and orienting all edges away from the root.

2.2 Rotor-Routing

Let G be a directed multigraph. For any node $v \in V^+(G)$, a *rotor order* at v is a cyclic permutation of $A^+(v)$. A *rotor order* for G is a permutation θ of $A(G)$ such that, for each $v \in V^+(G)$, the restriction of θ to $A^+(v)$ is a rotor order at v . The directed multigraph G combined with a rotor order θ is a *rotor graph*.

A *rotor configuration* of G is a mapping from $\rho : V^+(G) \rightarrow A(G)$ such that $\rho(v) \in A^+(v)$ for each $v \in V^+(G)$. Let \mathcal{R}_G be the set of rotor configurations of G . A *particle configuration* of G is a mapping from $\sigma : V(G) \rightarrow \mathbb{Z}$. Let $\mathbb{Z}^{V(G)}$ be the set of particle configurations of G .

For node v , denote by $\mathbf{v} \in \mathbb{Z}^{V(G)}$ the function which is 1 at v , and 0 elsewhere. For example, $\sigma' = \sigma + 3\mathbf{v}$ means $\sigma'(v) = \sigma(v) + 3$ and $\sigma'(u) = \sigma(u)$ for $u \neq v$.

A *rotor-particle configuration* of G is a pair (ρ, σ) of a rotor configuration $\rho \in \mathcal{R}_G$ and a particle configuration $\sigma \in \mathbb{Z}^{V(G)}$. *Routing* at $v \in V^+(G)$ on (ρ, σ) is the operation of moving a particle from v along the arc $\rho(v)$ and then changing the rotor configuration at v to $\theta(\rho(v))$. The resulting rotor-particle configuration $(\rho', \sigma') = \text{rout}^v(\rho, \sigma)$ is then formally defined as

$$\rho'(u) = \begin{cases} \theta(\rho(u)) & u = v, \\ \rho(u) & u \neq v, \end{cases}$$

and $\sigma' = \sigma + \text{head}(\rho(v)) - \text{tail}(\rho(v))$. Routing decomposes into two operators. First, a *move* at v , denoted by $\phi(\rho; \mathbf{v}) = \text{head}(\rho(v)) - \text{tail}(\rho(v))$. Second, a *turn* at v , denoted by $\theta^{\mathbf{v}}$, where $\theta^{\mathbf{v}}(a) = \theta(a)$ for each $a \in A^+(v)$ and $\theta^{\mathbf{v}}(a) = a$ for each $a \notin A^+(v)$. We see that

$$\text{rout}^{\mathbf{v}}(\rho, \sigma) = (\theta^{\mathbf{v}} \circ \rho, \sigma + \phi(\rho; \mathbf{v})) = (\rho', \sigma') .$$

This kind of routing is referred to as *move-then-turn*, as opposed to *turn-then-move*. It is easy to see that both kinds of routing are isomorphic by advancing or reverting the rotor configuration. We will use this isomorphism in the proof of Proposition 5.

We define $\theta^{-\mathbf{v}} = (\theta^{\mathbf{v}})^{-1}$ and $\phi(\rho; -\mathbf{v}) = -\phi(\theta^{-\mathbf{v}} \circ \rho; \mathbf{v})$; the *inverse* of $\text{rout}^{\mathbf{v}}$ is defined as

$$\text{rout}^{-\mathbf{v}}(\rho', \sigma') = (\theta^{-\mathbf{v}} \circ \rho', \sigma' + \phi(\rho'; -\mathbf{v})) = (\rho, \sigma) .$$

Given distinct nodes $v \neq u$, observe that $\text{rout}^{\mathbf{v}}$ and $\text{rout}^{\mathbf{u}}$ commute, as the move and turn of v depend on and affect ρ only at component v . Hence, compositions of the routing operators are characterized by the number of times each node is routed.

A *routing vector* (or simply a *routing*) of G is a mapping from $r : V^+(G) \rightarrow \mathbb{Z}$. We denote by rout^r the operator resulting from composing all $(\text{rout}^{\mathbf{v}})^{r(v)}$ for $v \in V^+(G)$ in arbitrary order. Similarly, we let θ^r denote the composition of all $(\theta^{\mathbf{v}})^{r(v)}$ for $v \in V^+(G)$. Extending the move operator $\phi(\rho; \mathbf{v})$ is more involved. We do so by interpreting ϕ to be the *displacement* of particles when routing. As such, for $k \in \mathbb{N}$, we define $\phi(\rho; k\mathbf{v}) = \sum_{i=0}^{k-1} \phi(\theta^{i\mathbf{v}} \circ \rho; \mathbf{v})$ and $\phi(\rho; -k\mathbf{v}) = -\sum_{i=1}^k \phi(\theta^{-i\mathbf{v}} \circ \rho; \mathbf{v})$. Then we define $\phi(\rho; r) = \sum_{v \in V^+(G)} \phi(\rho; r(v)\mathbf{v})$. It is now straightforward to see that

$$\text{rout}^r(\rho, \sigma) = (\theta^r \circ \rho, \sigma + \phi(\rho; r)) .$$

For the sake of convenience, we extend these notations to sinks. Each routing $r \in \mathbb{Z}^{V^+(G)}$ is identified with the mapping in $\mathbb{Z}^{V(G)}$ such that $r(s) = 0$ for all $s \in S_0$. Furthermore, routing sinks has no effect, i.e., $\theta^{\mathbf{s}} = \text{id}$ and $\phi(\rho; \mathbf{s}) = \mathbf{0}$.

2.3 Arrival and Rotor-Routing Games

In the ARRIVAL game we start with a rotor-particle configuration (ρ, \mathbf{u}) where $u \in V^+(G)$ is the initial location of the particle. We then repeatedly route the node on which the particle is located. This makes the particle walk through the rotor graph until it ends up on a sink, at which point the game terminates. To ensure termination, throughout this article we require G to be *stopping*, meaning that every node has a directed path to some sink.

We now introduce the *rotor-routing game*, which generalizes ARRIVAL. We say that routing node v is *legal* on (ρ, σ) if $\sigma(v) > 0$. A *routing sequence* $(v_0, \dots, v_{k-1}) \in (V^+(G))^k$ is *legal* on (ρ_0, σ_0) if each routing of v_i is legal on (ρ_i, σ_i) where $(\rho_{i+1}, \sigma_{i+1}) = \text{rout}^{v_i}(\rho_i, \sigma_i)$. Finally, a non-negative routing r is *legal* if there is a *corresponding* legal routing sequence (v_0, \dots, v_{k-1}) , which means that $r(v) = |\{i \in \{0, \dots, k-1\} \mid v_i = v\}|$ for each $v \in V^+(G)$.

Such a legal routing r is *maximal* on (ρ, σ) if $\sigma' = \sigma + \phi(\rho; r)$ is non-positive on $V^+(G)$. That is, a routing r is maximal if no node can be legally routed after routing r .

In the rotor-routing game, at each step an arbitrary legal node is routed until all particles are on sinks. It is easy to see that in ARRIVAL, the legal routing sequence is unique, but in rotor-routing at any point there may be multiple nodes that can be legally routed.

An important consequence of routing a node behaving independent of the configuration of other nodes is that every legal routing sequence eventually terminates with the same corresponding maximal legal routing.

► **Lemma 3** ([9, Lemma 3.9]). *For all $(\rho, \sigma) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$ with $\sigma \geq \mathbf{0}$, there is a unique maximal legal routing r on (ρ, σ) .*

It follows that the maximal legal routing is an upper bound (component-wise) of all legal routings. We denote by $(\rho', \sigma') = \text{rout}_L^\infty(\rho, \sigma)$ the rotor-particle configuration resulting from the unique maximal legal routing on (ρ, σ) .

We are now ready to formally define the problems discussed in this paper through the notion of routings.

► **Problem** (ARRIVAL [3]). *Given $(\rho, \mathbf{u}) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$, compute σ' for $(\rho', \sigma') = \text{rout}_L^\infty(\rho, \mathbf{u})$.*

The following generalization of ARRIVAL was introduced by Hoang [8]:

► **Problem** (LEGAL MULTI-ARRIVAL [8]). *Given $(\rho, \sigma) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$ with $\sigma \geq \mathbf{0}$, compute σ' for $(\rho', \sigma') = \text{rout}_L^\infty(\rho, \sigma)$.*

An even more general scenario was introduced by Auger et al. [1]. Namely, if we do not require routings to be legal, for arbitrary σ any routing can be extended such that $\sigma'(v) = 0$ for all $v \in V^+(G)$. Such a rotor-particle configuration is called *fully routed*. We denote by $\text{rout}^\infty(\rho, \sigma)$ the set of fully routed rotor-particle configurations obtainable by routing (ρ, σ) .

► **Proposition 4** ([1, Theorem 2]). *For all $(\rho, \sigma) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$ there is a routing r such that with $(\rho', \sigma') = \text{rout}^r(\rho, \sigma)$ one has $\sigma'(v) = 0$ on all $v \in V^+(G)$. Furthermore, for any two such routings r_1 and r_2 , resulting in (ρ^1, σ^1) and (ρ^2, σ^2) , respectively, we have that $\sigma^1 = \sigma^2$.*

This leads to the following problem, which is our main concern in this paper:

► **Problem** (MULTI-ARRIVAL [1]). *Given $(\rho, \sigma) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$, compute σ' for arbitrary $(\rho', \sigma') \in \text{rout}^\infty(\rho, \sigma)$.*

For $\sigma \geq \mathbf{0}$, after applying the maximal legal routing it holds that $\sigma'(v) = 0$ for $v \in V^+(G)$. It follows that $\text{rout}_L^\infty(\rho, \sigma) \in \text{rout}^\infty(\rho, \sigma)$ and that LEGAL MULTI-ARRIVAL reduces to MULTI-ARRIVAL. Conversely, finding full routings reduces to finding a maximal legal routing in G and in $G^{(-1)}$, where $G^{(-1)}$ is obtained from G by reversing the rotor order i.e. using θ^{-1} .

► **Proposition 5.** *Let $(\rho, \sigma) \in \mathcal{R}_G \times \mathbb{Z}^{V(G)}$ and $\sigma = \sigma_1 - \sigma_2$ such that $\sigma_1, \sigma_2 \geq \mathbf{0}$. Then given the maximal legal routing r_1 on (ρ, σ_1) in G and the maximal legal routing r_2 on $(\theta^{r_1-1} \circ \rho, \sigma_2)$ in $G^{(-1)}$, we have that $r = r_1 - r_2$ is a full routing on (ρ, σ) in G .*

Proof. We define $g: \mathcal{R}_G \times \mathbb{Z}^{V(G)} \rightarrow \mathcal{R}_G \times \mathbb{Z}^{V(G)}$ by $g(\rho, \sigma) = (\theta^{-1} \circ \rho, -\sigma)$. The function g transforms rotor-particle configurations such that negative move-then-turn routings in G correspond to positive turn-then-move routings in $G^{(-1)}$. We let $\theta' = \theta^{-1}$ and ϕ' denote the move and displacement operators in $G^{(-1)}$.

▷ **Claim.** It holds that $\text{rout}_G^r(\rho, \sigma) = g^{-1}(\text{rout}_{G^{(-1)}}^{-r}(g(\rho, \sigma)))$.

Proof. We show that $\text{rout}_G^v(\rho, \sigma) = g^{-1}(\text{rout}_{G^{(-1)}}^{-v}(g(\rho, \sigma)))$, from which the claim follows. Recall that $\text{rout}_{G^{(-1)}}^{-v}(\rho, \sigma) = (\theta'^{-v} \circ \rho, \sigma - \phi(\theta'^{-v} \circ \rho; \mathbf{v}))$. First, for the turn operator we see that $\theta'^{-v} \circ \theta^{-1} = \theta^{-1} \circ \theta^v$. Then, for the move operator we derive that

$$-\phi(\theta'^{-v} \circ \theta^{-1} \circ \rho; \mathbf{v}) = -\phi(\rho; \mathbf{v}) .$$

Consequently, it holds that

$$\begin{aligned} g^{-1}(\text{rout}_{G^{(-1)}}^{-v}(g(\rho, \sigma))) &= g^{-1} \circ \text{rout}_{G^{(-1)}}^{-v}(\theta^{-1} \circ \rho, -\sigma) \\ &= g^{-1} \circ (\theta^{-1} \circ \theta^v \circ \rho, -\sigma - \phi(\rho; \mathbf{v})) \\ &= (\theta^v \circ \rho, \sigma + \phi(\rho; \mathbf{v})) = \text{rout}_G^v(\rho, \sigma) . \end{aligned}$$

This proves the claim. ◁

We next find that $\sigma + \phi(\rho; r_1 - r_2) = (\sigma_1 + \phi(\rho; r_1)) - (\sigma_2 + \phi'(\theta^{r_1-1} \circ \rho; r_2))$. Since r_1 is maximal on (ρ, σ_1) with $\sigma_1 \geq \mathbf{0}$ in G , it follows that $\sigma_1 + \phi(\rho; r_1)$ is 0 on $V^+(G)$. Similarly, as r_2 is maximal on $(\theta^{r_1-1} \circ \rho, \sigma_2)$, with $\sigma_2 \geq \mathbf{0}$ in $G^{(-1)}$ we have that $\sigma_2 + \phi'(\theta^{r_1-1} \circ \rho; r_2)$ is also 0 on $V^+(G)$. It follows that r is a full routing. ◀

Our approach finds maximal legal routings which enables us to solve both LEGAL MULTI-ARRIVAL and MULTI-ARRIVAL on tree-like rotor graphs.

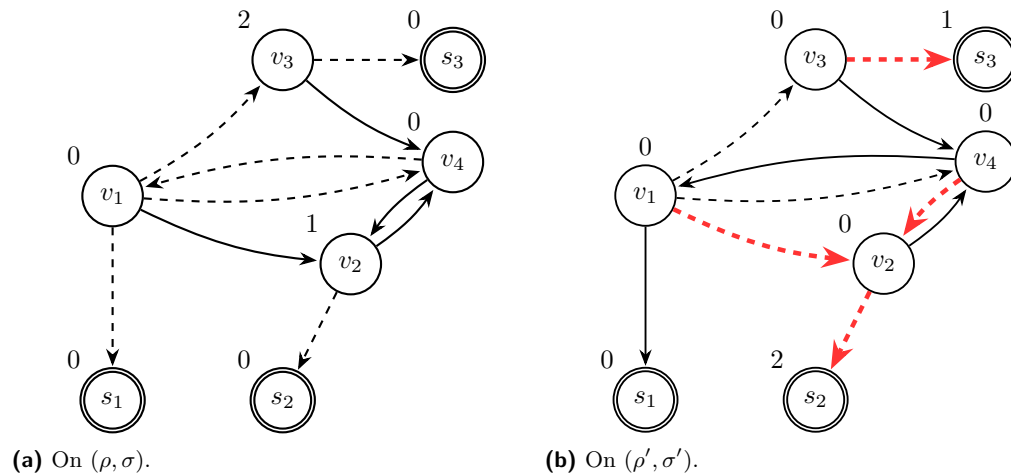
3 Routing Decompositions

In this section we show how to decompose maximal legal routings into legal routings whose most recently used arcs are directed towards sinks. We shall always denote the maximal legal routing by \hat{r} , assuming some (ρ, σ) to be fixed.

3.1 Destination Graphs

Given a rotor configuration ρ and routing $r \geq \mathbf{0}$, the *destination graph* $G[\rho; r]$ is a subgraph of G whose nodes consist of all those that either sent or received particles during routing r . For each node $v \in \text{supp}(r)$, that is $r(v) \neq 0$, the destination graph $G[\rho; r]$ contains only the outgoing arc that the last particle traversing v was sent over when routing r . That is, $G[\rho; r]$ is the subgraph of G induced by the arcs $\{\theta^{-1}(\rho'(v)) \mid v \in \text{supp}(r)\}$, where $\rho' = \theta^r \circ \rho$ is the rotor configuration after routing r . Each node $v \in \text{supp}(r)$ has one outgoing arc in $G[\rho; r]$ and every other node in $G[\rho; r]$ has no outgoing arcs.

Furthermore, the destination graph $G[\rho; \hat{r}]$ of the maximal legal routing is acyclic and the connected components are directed subtrees rooted at sinks. Figure 1 depicts a rotor graph with the rotor order being clockwise, and a corresponding destination graph.



■ **Figure 1** (a) A rotor graph G with a rotor-particle configuration (ρ, σ) , where solid lines represent ρ , labels on the nodes represent the particle distribution $\sigma = 1\mathbf{v}_2 + 2\mathbf{v}_3$. A possible maximal legal routing sequence is $v_2 \rightarrow v_4 \rightarrow v_2 \rightarrow s_2, v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_2 \rightarrow s_2$ and $v_3 \rightarrow s_3$, with corresponding maximal legal routing $\hat{r} = 1\mathbf{v}_1 + 4\mathbf{v}_2 + 2\mathbf{v}_3 + 3\mathbf{v}_4$. (b) Thick red lines are the arcs in the destination graph $G[\rho; \hat{r}]$ for the maximal legal routing \hat{r} . Note that $G[\rho; \hat{r}]$ does not contain s_1 .

3.2 Compensated Routings

Any legal routing r must satisfy the condition that, for each node v , the number of particles sent out of v during the routing process must not exceed the number of particles initially present at v and those sent into v during the routing. This establishes one of the main constraints for a routing to be legal. We call such a routing “compensated”.

Formally, we say that a routing r is *compensated* at v with respect to (ρ, σ) if

$$\sigma(v) + \sum_{u \in N^-(v)} \phi_v^u(\rho; r(u)) \geq \sum_{w \in N^+(v)} \phi_w^v(\rho; r(v)) = r(v),$$

where $\phi_w^v(\rho; j)$ denotes the number of particles sent to w when routing j particles out of v .

A routing is called *compensated* if it is compensated at each $v \in V^+(G)$. Equivalently, r is compensated if and only if $\sigma + \phi(\rho; r) \geq \mathbf{0}$. Note that compensated routings are not necessarily legal.

The following proposition, which is a special case of a theorem by Tóthmérész [11, Theorem 3.4], provides a characterization of compensated routings that are legal. For sake of completeness, we include a simplified proof.

► **Proposition 6.** *Given a routing $r \geq \mathbf{0}$ and rotor-particle configuration (ρ, σ) with $\sigma \geq \mathbf{0}$. Let $(\rho', \sigma') = \text{rout}^r(\rho, \sigma)$. The routing r is legal on (ρ, σ) if and only if, r is compensated on (ρ, σ) and each cycle in $G[\rho; r]$ contains some node v such that $\sigma'(v) > 0$. In particular, if $G[\rho; r]$ is acyclic, legality and compensation are equivalent.*

Proof. Suppose that r is legal. So it is compensated. If $G[\rho; r]$ contains a cycle C , then given some corresponding legal routing sequence, there is some node $v \in V(C)$ that is routed last in C . Let $w \in V(C)$ denote the successor of v in C . Then a particle is sent from v to w , but w is not routed afterwards. Hence, as the particle distribution is non-negative during the legal routing sequence, we have that $\sigma'(w) > 0$.

Conversely, assume that r is compensated on (ρ, σ) and each cycle in $G[\rho; r]$ contains some node v such that $\sigma'(v) > 0$. Let $r^* \leq r$ be a legal routing with $(\rho^*, \sigma^*) = \text{rout}^{r^*}(\rho, \sigma)$ obtained by routing until for each node v either $\sigma^*(v) = 0$ or $r^*(v) = r(v)$.

Since r^* is legal, it suffices to show that $r^* = r$. Assume, for the sake of contradiction, that $r^* \neq r$. We then consider the set $B = \{v \in V^+(G) \mid r^*(v) < r(v)\}$ of nodes that still need to be routed (if any).

It follows that $\sigma^*(v) = 0$ for each $v \in B$. In the remaining routing $r^* - r$, each particle sent out from a node in B must be replaced afterward; otherwise, we would end up with a node $v \in B$ such that $\sigma'(v) < 0$, which is impossible since r is compensated. This implies that during the routing $r^* - r$, no particle from B is sent outside of B ; otherwise, that particle could not be replaced in B as no node outside of B is routed in $r^* - r$. It follows that in the subgraph H of $G[\rho; r]$ induced by B , the nodes have out-degree of 1, implying that H contains a cycle. This contradicts the assumption that every cycle in $G[\rho; r]$ contains a node v such that $\sigma'(v) > 0$. ◀

3.3 s -Directed Routings

The compensated routings r are those for which each $v \in V^+(G)$ satisfies an inequality that relates $r(v)$ to $r(u)$ for all in-neighbours u of v . Enforcing this inequality is much simpler than enforcing the stricter property of legality. If restricted to routings with acyclic destination graph, these properties coincide. For finding the maximal legal routing \hat{r} , we may restrict the search to only those routings. But the destination graph being acyclic is also a non-trivial property. Instead, we consider an even more restricted case in which the destination graph is

oriented towards some fixed sink s . In tree-like rotor graphs, this implies a unique destination graph (up to parallel arcs), which can be enforced by an appropriate parametrization of routings. But first, we need to show that these restricted routings still allow us to recover \hat{r} .

Given a rotor configuration ρ and sink s , we say that node v is s -directed in a routing r if $v \in \text{supp}(r)$ and there is a path in $G[\rho; r]$ from v to s . Routing r is s -directed if every $v \in \text{supp}(r)$ is s -directed in r . Our aim, to be established in Theorem 9, is to decompose \hat{r} into a set of s -directed routings r_s for each sink s . The idea is that each such r_s need only match the number of particles sent into s by \hat{r} . We denote by $\phi_w(\rho; r)$ the number of particles sent to w when routing r .

► **Lemma 7.** *Let (ρ, σ) be a rotor-particle configuration with $\sigma \geq \mathbf{0}$ and $b \geq \mathbf{0}$ be a legal routing. Then, for each sink s , there is an s -directed legal routing $r \geq \mathbf{0}$ such that $r \leq b$ and $\phi_s(\rho; r) = \phi_s(\rho; b)$.*

Proof. We construct a series of legal routings $b = r_1 \geq r_2 \geq \dots$ with $A_1 \subset A_2 \subset \dots$ denoting the s -directed nodes of the respective routing, such that each r_{i+1} equals r_i on A_i and each $v \in A_i$ has the same in-flow in r_{i+1} as in r_i .

Suppose that we constructed r_i and that there is flow from $V^+(G) \setminus A_i$ to A_i when routing r_i . Consider an arbitrary legal routing sequence (v_0, \dots, v_{m-1}) corresponding to r_i with $(\rho_0, \sigma_0) = (\rho, \sigma)$ and $(\rho_{\ell+1}, \sigma_{\ell+1}) = \text{rout}^{v_\ell}(\rho_\ell, \sigma_\ell)$.

Let j be maximal such that $v_j \notin A_i$ and $\text{head}(\rho_j(v_j)) \in A_i$. That is, the j^{th} routing step is the last time a particle is sent into A_i from $V^+(G) \setminus A_i$. Now, we remove each routing step $\ell > j$ with $v_\ell \notin A_i$. Let r_{i+1} be the corresponding routing vector. Then r_{i+1} must be legal since for $\ell > j$ we have $v_\ell \in A_i$ and v_ℓ has in the ℓ -th routing step the same number of particles as before the removals. By construction r_{i+1} satisfies the required conditions and $A_i \cup \{v_j\} \subseteq A_{i+1}$ as v_j sent its last particle towards A_i in r_{i+1} .

Otherwise, suppose there is no flow from $V^+(G) \setminus A_i$ to A_i when routing r_i . We set r to be r_i on A_i , and zero on $V^+(G) \setminus A_i$. Then r is an s -directed legal routing with $r \leq r_1 = b$. Since $s \in A_1 \subseteq A_2 \subseteq \dots$ we have $\phi_s(\rho; r_1) = \phi_s(\rho; r_2) = \dots = \phi_s(\rho; r)$. This procedure is finite as it terminates once $\text{supp}(b) \subseteq A_i$ and each step increases the size of A_i . ◀

Next, we show that in the maximal legal routing \hat{r} the s -directed nodes already route only as much as is required to achieve the flow towards s . We therefore derive that a corresponding s -directed routing matches \hat{r} on those nodes.

► **Lemma 8.** *Let (ρ, σ) be a rotor-particle configuration with $\sigma \geq \mathbf{0}$. If $r \geq \mathbf{0}$ is an s -directed legal routing such that $\phi_s(\rho; r) = \phi_s(\rho; \hat{r})$, then $r(v) = \hat{r}(v)$ for each s -directed node of \hat{r} .*

Proof. Suppose u is s -directed in \hat{r} such that $r(u) \neq \hat{r}(u)$. We denote by v the successor on the path from u to s in $G[\rho; \hat{r}]$. As \hat{r} is maximal and r is legal, it must be that $r(u) < \hat{r}(u)$. Since the last particle sent from u in \hat{r} is towards v we have $\phi_v^u(\rho; r(u)) < \phi_v^u(\rho; \hat{r}(u))$. In particular, $\phi_v(\rho; r) < \phi_v(\rho; \hat{r})$. If $v \neq s$, then as \hat{r} routes v for each received particle and r receives less particles on v , it follows that $r(v) < \hat{r}(v)$. Now, v is also s -directed in \hat{r} with $r(v) \neq \hat{r}(v)$. We repeat the argument along the path from u to s in $G[\rho; \hat{r}]$ until we end up with $v = s$. Then $\phi_s(\rho; r) < \phi_s(\rho; \hat{r})$ contradicts the assumption that $\phi_s(\rho; r) = \phi_s(\rho; \hat{r})$. ◀

► **Theorem 9.** *Let (ρ, σ) be a rotor-particle configuration with $\sigma \geq \mathbf{0}$. For each $s \in S_0$, let r_s be an optimal solution of the following optimization problem:*

$$\text{maximize} \quad \phi_s(\rho; r) \quad \text{subject to} \quad r \text{ is an } s\text{-directed compensated routing.}$$

Then \hat{r} is given by $\hat{r}(v) = \max\{r_s(v) \mid s \in S_0\}$ for all $v \in V^+(G)$.

Proof. Let $s \in S_0$. As r_s is s -directed, it has an acyclic destination graph. Since it is also compensated, it follows from Proposition 6 that r_s is legal. In particular, as \hat{r} is maximal we have $r_s \leq \hat{r}$ and $\phi_s(\rho; r_s) \leq \phi_s(\rho; \hat{r})$. Now, as shown by Lemma 7 there is some legal s -directed routing r'_s such that $\phi_s(\rho; r'_s) = \phi_s(\rho; \hat{r})$. It follows from r_s maximizing $\phi_s(\rho; r)$ that $\phi_s(\rho; r_s) = \phi_s(\rho; \hat{r})$.

For each $v \notin \text{supp}(\hat{r})$ the claim holds as $0 \leq r_s(v) \leq \hat{r}(v) = 0$ for each $s \in S_0$. Let $v \in \text{supp}(\hat{r})$. Then as $G[\rho; \hat{r}]$ is acyclic and each node in $G[\rho; \hat{r}]$ has out-degree at most one, there is some u that the unique path starting at v ends on. If $u \notin S_0$, then $u \notin \text{supp}(\hat{r})$. But then u receives a particle in \hat{r} but is not routed, which contradicts the maximality of \hat{r} . Thus, $u \in S_0$ and v is u -directed in \hat{r} . Consequently, due to Lemma 8 we find that $\hat{r}(v) = r_u(v) = \max\{r_s(v) \mid s \in S_0\}$. ◀

4 Algorithms for Multi-Arrival on Tree-Like Rotor Graphs

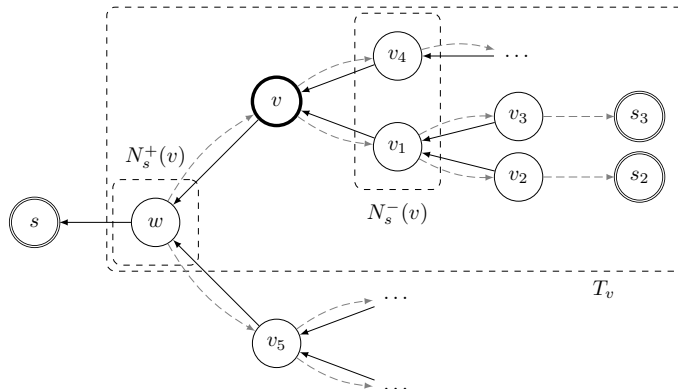
In the following, let T be a tree-like rotor graph. Henceforth assume, without loss of generality, that the induced subgraph of T on $V^+(T)$ is strongly connected, and each sink to have exactly one in-neighbour. Indeed, if T were not strongly connected, then we can solve MULTI-ARRIVAL in some ordering of the strongly connected components as each component has no particles flowing to the prior. Furthermore, if some sink had multiple in-neighbours, we split it into one sink per in-neighbor.

4.1 Relative Rotor Subgraphs

With the above assumptions on T , each $v \in V^+(T)$ has a unique successor on its paths to s in T . We denote by $N_s^+(v)$ the *successor* on the paths from v to s in T , and by $N_s^-(v)$ the set of *predecessors* whose successor is v on their respective paths towards s . When a routing is s -directed, its destination graph can be seen to be an s -rooted directed subtree of T .

Throughout this section, we fix a sink $s \in S_0$ and a rotor-particle configuration (ρ, σ) with $\sigma \geq \mathbf{0}$. For the sake of convenience, we leave this dependence on s and (ρ, σ) implicit in the introduced notation. Further, we simplify the notation and let $\phi(r)$ denote $\phi(\rho; r)$, and similarly extend this simplification to ϕ_w^v and ϕ_v .

We proceed to show that s -directed routings can be characterized by their restrictions to nested rotor subgraphs. For any node $v \in V^+(T)$ with $w = N_s^+(v)$ let T_v be the rotor graph obtained from T as follows. We remove the arcs between v and w , and then, remove the nodes not reachable from v . Finally, we add w again with only the original arcs from v to w (but not the reverse arcs). In T_v , node w is thus a sink. See Figure 2 for an example.



■ **Figure 2** Example of a relative subtree T_v . Grey dashed lines show the arcs oriented away from s . Note that, e.g., the rotor graph T_{v_2} contains v_1 as a sink, which therefore has no arcs from v_1 to v_2 .

For any integers $x, y \in \mathbb{N}$, we denote by $H_v(x, y)$ the set of w -directed compensated routings r in T_v on rotor-particle configuration $(\rho, \sigma + y\mathbf{v})$ such that $\phi_w^v(r(v)) = x$.

Furthermore, we define $h_v(y) = \max\{x \in \mathbb{N} \mid H_v(x, y) \neq \emptyset\}$.

Note that for a given y , from Lemma 7 it follows that $h_v(y)$ is the flow from v to w that is legally achievable in T_v when dispersing an additional y particles on v .

► **Lemma 10.** *Given $v \in V^+(T)$ and $x, y \in \mathbb{N}$ such that $H_v(x, y) \neq \emptyset$, we have $H_v(x', y) \neq \emptyset$ for any $0 \leq x' \leq x$. In particular, $H_v(x, y) \neq \emptyset$ if and only if $0 \leq x \leq h_v(y)$.*

Proof. Let $r \in H_v(x, y)$ and let b be some legal routing such that $x' = \phi_w(\rho; b) \leq \phi_w(\rho; r) = x$ in T_v , where $w = N_s^+(v)$. Note that such a b exists as we may truncate a legal routing sequence of r up to the x' -th routing towards w . Then, using Lemma 7 with b on T_v , we obtain an w -directed compensated routing r' with $x' = \phi_w(\rho; r')$. Hence, $r' \in H_v(x', y)$. As the zero routing is always compensated and s -directed, the second claim follows as well. ◀

Let $w = N_s^+(v)$ and $u \in N_s^-(v)$. We define the function $q_v: \mathbb{N} \rightarrow \mathbb{N}$ as the minimum number of routings required to send x particles from v to w . Formally, $q_v(x) = \min\{k \in \mathbb{N} \mid \phi_w^v(\rho; k) = x\}$. It is also convenient to specify the notation $q_u^v(x) = \phi_u^v(q_v(x))$, which is the number of particles that v sends to u as a by-product.

The following crucial lemma will enable us to recursively construct routings in $H_v(x, y)$.

► **Lemma 11.** *Let $v \in V^+(T)$ and $w = N_s^+(v)$. Consider a routing $r \geq \mathbf{0}$ on T_v and for $u \in N_s^-(v)$ let r_u denote the restriction of r on $V^+(T_u)$. Then, $H_v(0, y) = \{\mathbf{0}\}$, and for $x > 0$,*

$$r \in H_v(x, y)$$

if and only if

$$r(v) = q_v(x), \tag{1}$$

and there exist non-negative integers x_u , for $u \in N_s^-(v)$, such that

$$\sum_{u \in N_s^-(v)} x_u \geq r(v) - y - \sigma(v), \tag{2}$$

$$r_u \in H_u(x_u, \phi_u^v(r(v))), \text{ for each } u \in N_s^-(v) \quad . \tag{3}$$

In particular, $H_v(x, y) \neq \emptyset$ if and only if

$$\sum_{u \in N_s^-(v)} h_u(q_u^v(x)) \geq q_v(x) - y - \sigma(v) \quad . \tag{4}$$

Proof. Let $r \in H_v(0, y)$. Then $r(v) = 0$, as $\phi_w^v(r(v)) = 0$ and r is w -directed. It follows that v is not in $T_v[\rho; r]$, and since every path from $V^+(T_v)$ to w in T_v is through v , it must be that $\text{supp}(r) = \emptyset$.

Now, for the case $x > 0$, assume that $r \in H_v(x, y)$ and therefore $r(v) > 0$. Thus, the last routing of r at v should be towards w , which is equivalent to (1).

Set $x_u = \phi_u^v(r_u(u)) = \phi_u^v(r(u))$. Then we see that (2) is a rearrangement of r being compensated at v . For (3), consider $u \in N_s^-(v)$. As the nodes of T_u are w -directed in r and their paths towards w in $T_v[\rho; r]$ go through v , it follows that r_u is v -directed in T_u . It remains to show that r_u being compensated in T_v is equivalent to r_u being compensated

in T_u . With $y_u = \phi_u^v(r(v))$, note that r_u needs to be compensated on $(\rho, \sigma + y_u \mathbf{u})$ in T_u . Clearly, the only non-trivial case is the compensation at u , which follows from the in-flow being equal in both cases:

$$\phi_u(r) = \phi_u(r_u) + \phi_u^v(r(v)) + \sigma(u) = \phi_u(r_u) + \sigma(u) + y_u .$$

Consequently, $r_u \in H_u(x_u, y_u)$.

Now suppose that $r \geq \mathbf{0}$ is a routing that satisfies (1)–(3). Notably, as $x > 0$ we have $q_v(x) > 0$. As mentioned above, (1) is equivalent to v being w -directed in r . From (3) it follows that $x_u = \phi_v^u(r(u))$, which together with (2) implies that r is compensated at v . Also, r is w -directed and compensated on $V^+(T_v) \setminus \{v\}$, because by (3), each restriction r_u is v -directed and compensated. \blacktriangleleft

The following properties of the functions h_v are needed in our argument.

► **Lemma 12.** *For any $v \in V^+(T)$, define $f_v(x) = q_v(x) - \sigma(v) - \sum_{u \in N_s^-(v)} h_u(q_u^v(x))$ for all x . Then the following properties hold for the functions f_v and h_v :*

- (i) $h_v(y) = \max\{x \in \mathbb{N} \mid f_v(x) \leq y\}$,
- (ii) $h_v(y) \leq h_v(y+1) \leq h_v(y) + 1$, for $y \in \mathbb{N}$,
- (iii) $f_v(x+1) \geq f_v(x) + 1$, for $x \in \mathbb{N}$.

Proof. (i) We know that $h_v(y)$ equals the largest x with $H_v(x, y) \neq \emptyset$, which, by Lemma 11, is the largest x satisfying (4), that is, the largest x such that $f_v(x) \leq y$.

(ii) Let \hat{r}_y denote the maximal legal routing in T_v on $(\rho, \sigma + y\mathbf{v})$. Then, by definition of $h_v(y)$ and Lemma 7, we see that $h_v(y) = \phi_w(\rho; \hat{r}_y)$. Thus, with $h_v(y+1) = \phi_s(\rho; \hat{r}_{y+1})$, we see that $h_v(y+1) = h_v(y) + 1$ if that additional particle ends up in w and $h_v(y+1) = h_v(y)$ otherwise.

(iii) Being integer-valued function, it suffices to show that f_v is strictly increasing.

Observe that $q_v(x+1) - q_v(x)$ is the number of arcs until and including the next that has head w . But $q_u^v(x+1) - q_u^v(x)$ is then the number of those arcs that have head u . Consequently, $q_v(x+1) - q_v(x) > \sum_{u \in N_s^-(v)} (q_u^v(x+1) - q_u^v(x))$, where by (ii), the right-hand side is as large as $\sum_{u \in N_s^-(v)} (h(q_u^v(x+1)) - h(q_u^v(x)))$. It then follows that $f_v(x+1) > f_v(x)$. \blacktriangleleft

4.2 MULTI-ARRIVAL for Tree-Like Multigraphs

In this subsection, we prove the main result of the paper, Theorem 1. To do so, we first need to specify how a rotor graph is computationally represented.

Suppose we wanted to represent $A^+(v)$ and θ^v . We could take in a sequence of nodes w_1, \dots, w_k and let each index i identify an outgoing arc with head w_i . Arc with label i is taken to be before $i+1$ in θ^v with index $k+1$ wrapping around to 1. We can improve upon this by merging consecutive duplicates $w_i = w_{i+1}$ by associating with each w_i the number of times that entry is repeated. We denote by \hat{A} the reduced arc set of the rotor graph obtained by removing arcs in A that vanish due to this encoding of consecutive parallel arcs. This encoding is called *succinct* [4, 11]. Hence, $|\hat{A}|$ is polynomial in the input size, while $|A|$ may be exponential in the input size.

For a rooted simple tree R , its *contracted height* $\text{ch}(R)$ is defined as the minimum height over all trees which are obtained from R by contracting, for each node v of R , one of the arcs to its children.

► **Lemma 13.** *For any rooted simple tree R , it holds that $\text{ch}(R) \leq \log(1 + \ell(R))$, where $\ell(R)$ is the number of nodes of R with at least two children.*

Proof. We proceed by induction on the height of R ; recall that the height of a rooted tree is the maximum length of a path from its root to any of its other nodes. If the height of R is zero, the assertion holds trivially. Thus, assume now that the height of R is at least 1. Let R_1, \dots, R_k be the connected components of R after removing its root, with $\text{ch}(R_j) = \max\{\text{ch}(R_i) \mid i = 1, \dots, k\}$. If there is a unique such j , then $\text{ch}(R) = \text{ch}(R_j)$ since the edge to that subtree can be contracted, and we are thus done by the induction hypothesis.

Otherwise, there is another such j , say j' . We may assume that $\ell(R_j) \leq \ell(R_{j'})$. Then

$$\text{ch}(R) \leq 1 + \text{ch}(R_j) \leq \log(2 + 2\ell(R_j)) \leq \log(1 + \ell(R)) . \quad \blacktriangleleft$$

Let R'_u be the tree $\langle T[V^+] \rangle$ rooted at u . We define

$$k(T) = 1 + \max\{\text{ch}(R'_u) \mid u \in N^-(S_0)\} .$$

By Lemma 13, $k(T) \leq 1 + \log(n_{>2}(\langle T \rangle) + 1)$, where $n_{>2}(\langle T \rangle)$ denotes the number of nodes in $\langle T \rangle$ that have more than two neighbors.

Notice that $k(T) \leq \kappa(T)$, where $\kappa(T)$ is the upper bound on the contracted heights of $\langle T \rangle$ as mentioned in introduction for the sake of stating an informal version of the main result.

We can thus now state our main result formally.

► **Theorem 1.** MULTI-ARRIVAL can be solved on any tree-like rotor graph T with arc set A in time $\mathcal{O}(|S_0| |\hat{A}| \log^k(|A|))$, where $k = k(T) \leq 1 + \log(n_{>2}(\langle T \rangle) + 1)$. In particular, MULTI-ARRIVAL can be solved in time polynomial in $|A|$ if $k = \mathcal{O}\left(\frac{\log |A|}{\log(\log |A|)}\right)$. Furthermore, if $k = \mathcal{O}(1)$ (which includes path-like rotor graphs as then $k = 1$), MULTI-ARRIVAL can be solved in time polynomial in $|\hat{A}|$.

Furthermore, in the procedure, a full routing for MULTI-ARRIVAL is obtained.

When k is fixed, we have that $\log^k(|A|) \leq f(k)|A|$ for some computable function f , and thus the run time of our algorithm would be at most $f(k)|A|^3$. Therefore, we conclude:

► **Corollary 14.** MULTI-ARRIVAL on tree-like rotor graphs is fixed-parameter tractable with parameter k .

One of the main ingredients of our approach is an affine approximation for the function $h_v(y)$, which is provided by the following lemma:

► **Lemma 15.** For any $v \in V^+(T)$, there is an affine function $\bar{h}_v: \mathbb{R} \rightarrow \mathbb{R}$ such that

$$|\bar{h}_v(y) - h_v(y)| \leq 2|A(T_v)|, \quad y \in \mathbb{N} . \quad (5)$$

A set of these functions $\{\bar{h}_v \mid v \in V^+(T)\}$ can be determined in time $\mathcal{O}(|\hat{A}|)$.

Proof. We first observe that for $x > 0$, it holds that

$$\begin{aligned} q_v(x + d(v, w)) &= q_v(x) + d_+(v), \\ q_u^v(x + d(v, w)) &= q_u^v(x) + d(v, u) . \end{aligned}$$

These properties suggest the following approximations:

$$\begin{aligned} \bar{q}_v(x) &= (d_+(v) - q_v(d(v, w))) + \frac{xd_+(v)}{d(v, w)} \quad \text{to approximate } q_v(x), \text{ and} \\ \bar{q}_u^v(x) &= (d(v, u) - q_u^v(d(v, w))) + \frac{xd(v, u)}{d(v, w)} \quad \text{to approximate } q_u^v(x) . \end{aligned}$$

Let $x = \ell d(v, w) + t \geq d(v, w)$ with $0 \leq t < d(v, w)$. Then

$$|q_v(x) - \bar{q}_v(x)| = \left| (q_v(d(v, w) + t) - q_v(d(v, w))) - \frac{td_+(v)}{d(v, w)} \right| < d_+(v), \quad (6)$$

$$|q_u^v(x) - \bar{q}_u^v(x)| = \left| (q_u^v(d(v, w) + t) - q_u^v(d(v, w))) - \frac{td(v, u)}{d(v, w)} \right| < d(v, w). \quad (7)$$

Similarly, (6) and (7) hold for $x < d(v, w)$ as then $q_v(x)$ and $\bar{q}_v(x)$ lie within an interval of size $d_+(v)$, and $q_u^v(x)$ and $\bar{q}_u^v(x)$ lie within an interval of size $d(v, w)$. Now, define inductively

$$\begin{aligned} \bar{h}_v(y) &= \max\{x \in \mathbb{R} \mid \bar{f}_v(x) \leq y\}, \\ \bar{f}_v(x) &= \bar{q}_v(x) - \sigma(v) - \sum_{u \in N_s^-(v)} \bar{h}_u(\bar{q}_u^v(x)). \end{aligned}$$

▷ **Claim.** For any $x, y, \delta \in \mathbb{R}$ with $\delta > 0$, it holds that

$$\bar{f}_v(x + \delta) \geq \bar{f}_v(x) + \delta, \quad (8)$$

$$\bar{h}_v(y + \delta) \leq \bar{h}_v(y) + \delta. \quad (9)$$

Proof. We first observe that for each node v , (9) follows from (8). Let $\bar{h}_v(y) = x$. So $\bar{f}_v(x) = y$. Then $\bar{f}_v(x + \delta) \geq \bar{f}_v(x) + \delta = y + \delta$. As \bar{f}_v is strictly increasing, we must have $\bar{h}_v(y + \delta) \leq x + \delta = \bar{h}_v(y) + \delta$.

Now, we establish (8) by induction. We have that

$$\bar{f}_v(x + \delta) - \bar{f}_v(x) = \bar{q}_v(x + \delta) - \bar{q}_v(x) - \sum_{u \in N_s^-(v)} \left(\bar{h}_u(\bar{q}_u^v(x + \delta)) - \bar{h}_u(\bar{q}_u^v(x)) \right).$$

If $N_s^-(v) = \emptyset$, then $\bar{f}_v(x + \delta) - \bar{f}_v(x) \geq \frac{\delta d_+(v)}{d(v, w)} \geq \delta$. Next, assume that $N_s^-(v) \neq \emptyset$ and (8) holds for all $u \in N_s^-(v)$. Then by the above argument, (9) also holds for $u \in N_s^-(v)$, implying that

$$\bar{h}_u(\bar{q}_u^v(x + \delta)) - \bar{h}_u(\bar{q}_u^v(x)) = \bar{h}_u\left(\bar{q}_u^v(x) + \frac{\delta d(v, u)}{d(v, w)}\right) - \bar{h}_u(\bar{q}_u^v(x)) \leq \frac{\delta d(v, u)}{d(v, w)}.$$

It follows that $\bar{f}_v(x + \delta) - \bar{f}_v(x) \geq \frac{\delta d_+(v)}{d(v, w)} - \sum_{u \in N_s^-(v)} \frac{\delta d(v, u)}{d(v, w)} = \delta$, as required. ◁

▷ **Claim 16.** If for all $x \in \mathbb{Z}$, $|\bar{f}_v(x) - f_v(x)| \leq L$, then $|\bar{h}_v(y) - h_v(y)| \leq L + 1$ for all $y \in \mathbb{Z}$.

Proof. Suppose that $\bar{h}_v(y) = x$. Then $\bar{f}_v(\lfloor x \rfloor) \leq \bar{f}_v(x) \leq y$ and $\bar{f}_v(\lfloor x \rfloor + 1) > y$. Thus, $f_v(\lfloor x \rfloor + 1 + L) \geq f_v(\lfloor x \rfloor + 1) + L \geq \bar{f}_v(\lfloor x \rfloor + 1)$, that is $f_v(\lfloor x \rfloor + 1 + L) > y$. That means $h_v(y) \leq \lfloor x \rfloor + L$. On the other hand, $f_v(\lfloor x \rfloor - L) \leq f_v(\lfloor x \rfloor) - L \leq \bar{f}_v(\lfloor x \rfloor) \leq y$. So $h_v(y) \geq \lfloor x \rfloor - L$, and thus $|h_v(y) - \lfloor x \rfloor| \leq L$ which implies that $|h_v(y) - \bar{h}_v(y)| \leq L + 1$. ◁

Now, we complete the proof by induction. If $N_s^-(v) = \emptyset$, then $\bar{h}_v(y) = \frac{d(v, w)}{d_+(v)}(y + \sigma(v))$ is an affine function that satisfies (5), because of (6) and Claim 16.

So assume that $N_s^-(v) \neq \emptyset$ and the assertion holds for $\bar{h}_u(y)$ for every $u \in N_s^-(v)$. The property of being an affine function follows directly from the definition of $\bar{h}_v(y)$.

For an integer x , we have:

$$|h_u(q_u^v(x)) - \bar{h}_u(\bar{q}_u^v(x))| \leq |h_u(q_u^v(x)) - \bar{h}_u(q_u^v(x))| + |\bar{h}_u(q_u^v(x)) - \bar{h}_u(\bar{q}_u^v(x))|. \quad (10)$$

The induction hypothesis implies that

$$|h_u(q_u^v(x)) - \bar{h}_u(q_u^v(x))| \leq 2|A(T_u)| .$$

Combining (7) and (9), we see that

$$|\bar{h}_u(q_u^v(x)) - \bar{h}_u(\bar{q}_u^v(x))| \leq d(v, u) .$$

Thus, the right-hand side of (10) is at most $d(v, u) + 2|A(T_u)|$. It follows that for every $x \in \mathbb{Z}$, it holds that

$$|\bar{f}_v(x) - f_v(x)| \leq d_+(v) + \sum_{u \in N_s^-(v)} (d(v, u) + 2|A(T_u)|) .$$

Then applying Claim 16, for every $y \in \mathbb{Z}$ we have that

$$|\bar{h}_v(y) - h_v(y)| \leq 1 + d_+(v) + \sum_{u \in N_s^-(v)} (d(v, u) + 2|A(T_u)|) \leq 2|A(T_v)| .$$

The inductive constructing of \bar{h}_v for each $v \in V^+(T)$ can be done in time $\mathcal{O}(|\hat{A}^+(v)|)$. Therefore, the set $\{\bar{h}_v \mid v \in V^+(T)\}$ can be constructed in time $\mathcal{O}(|\hat{A}^+|)$. \blacktriangleleft

In following, we let $k_v = 1 + \text{ch}(T'_v)$ where T'_v is the rooted simple tree $\langle T_v \rangle - (S_0 \setminus \{s\})$ rooted at w .

► **Lemma 17.** *Let t be the unique in-neighbour of s . Then the maximum $\phi_s(r)$ over every s -directed compensated routing r can be obtained in time $\mathcal{O}(|\hat{A}| \log^{k_t}(|A|))$.*

Proof. We first determine the set of affine functions $\{\bar{h}_v \mid v \in V^+(T)\}$ in time $\mathcal{O}(|\hat{A}|)$ using Lemma 15. For given $x, y \in \mathbb{N}$, let \mathcal{T}_v be the time complexity required to query $H_v(x, y)$, that is to return a routing from the set or to confirm that it is empty.

We will prove that

$$\mathcal{T}_v = \mathcal{O}(|\hat{A}(T_v)| \log^{k_v-1}(|A(T_v)|)) . \quad (11)$$

Once we established this, for a given x we can determine if there is some $r \in H_t(x, 0)$, that is an s -directed compensated routing r_s such that $\phi_s(r_s) = x$, within time \mathcal{T}_t .

Moreover, by employing an exponential search using the approximation $\bar{h}_t(0)$, we only need to check the non-emptiness of $H_t(x, 0)$ for $\mathcal{O}(\log |A|)$ distinct values of x . This process enables us to compute

$$h_t(0) = \max\{\phi_s(r) \mid s\text{-directed compensated routing } r\}$$

in time $\mathcal{O}(\mathcal{T}_t \log |A|) = \mathcal{O}(|\hat{A}| \log^k(|A|))$, which completes the proof.

So it remains to prove (11). To do so, we first establish the following claim.

▷ **Claim 18.** For $v \in V^+(T)$ and any choice of $u^* \in N_s^-(v)$, it holds that

$$\mathcal{T}_v = \mathcal{O}\left(|\hat{A}^+(v)| + \mathcal{T}_{u^*} + \sum_{u \in N_s^-(v) \setminus \{u^*\}} \mathcal{T}_u \log |A(T_u)|\right) .$$

Proof. Given $x, y \in \mathbb{N}$, our objective is to either find a routing $r \in H_v(x, y)$ or confirm that the set is empty. We may assume $x > 0$ as otherwise we simply return $r = \mathbf{0}$.

For each $u \in N_s^-(v)$, let $y_u = q_u^v(x)$.

First, we determine $r(v) = q_v(x)$. Then, for each $u \in N_s^-(v) \setminus \{u^*\}$, we determine $\bar{h}_u(y_u)$.

Using the induced bound of that approximation on $h_u(y_u)$, we perform an exponential search to obtain $x_u = h_u(y_u)$ and $r_u \in H_u(x_u, y_u)$.

To perform this, due to Lemma 15, we require at most $1 + \log |A(T_u)|$ queries to H_u . As for x_{u^*} , we set

$$z = r(v) - \sigma(v) - y - \sum_{u \in N_s^-(v) \setminus \{u^*\}} x_u .$$

If $z \leq 0$, we let $x_{u^*} = 0$ and r_{u^*} be the zero routing which belongs to $H_{u^*}(0, y_{u^*})$. Otherwise, we let $x_{u^*} = z$ and query $r_{u^*} \in H_{u^*}(x_{u^*}, y_{u^*})$. If no such r_{u^*} exists, we terminate and assert $H_v(x, y) = \emptyset$.

Finally, we return r as given by $r(v)$ and whose restrictions to $V(T_u)$ is given by r_u for every $u \in N_s^-(v)$.

Now, the procedure takes the claimed time if $r(v)$ and $\{y_u \mid u \in N_s^-(v)\}$ can be calculated in time $\mathcal{O}(|\hat{A}^+(v)|)$. This can be done using a straightforward procedure employing two Euclidean divisions and two iterations through the encoding $\hat{A}^+(v)$.

As for correctness, if we failed to obtain r_{u^*} , we must have $x_{u^*} > h_{u^*}(y_{u^*})$. Hence:

$$0 > r(v) - \sigma(v) - y - \sum_{u \in N_s^-(v)} h_u(y_u),$$

which by Lemma 11 implies $H_v(x, y) = \emptyset$. Otherwise, r satisfies every condition of Lemma 11 and thus $r \in H_v(x, y)$. \triangleleft

To complete the proof of (11) we now proceed by induction. If $N_s^-(v) = \emptyset$, we have $\mathcal{T}_v = \mathcal{O}(|\hat{A}^+(v)|)$ which satisfies the hypothesis as $k_v = 1$. Otherwise, by the induction hypothesis, for each $u \in N_s^-(v)$, we have that $\mathcal{T}_u = \mathcal{O}(|\hat{A}(T_u)| \log^{k_u-1}(|A(T_u)|))$.

Now, using Claim 18, we obtain

$$\mathcal{T}_v = \mathcal{O}\left(|\hat{A}^+(v)| + |\hat{A}(T_{u^*})| \log^{k_{u^*}-1}(|A(T_{u^*})|) + \sum_{u \in N_s^-(v) \setminus \{u^*\}} |\hat{A}(T_u)| \log^{k_u}(|A(T_u)|)\right) .$$

We choose $u^* \in N_s^-(v)$ such that $k_{u^*} = \max\{k_u \mid u \in N_s^-(v)\}$. Then

$$k_v = \max\{k_{u^*}\} \cup \{k_u + 1 \mid u^* \neq u \in N_s^-(v)\},$$

from which (11) follows. \blacktriangleleft

Proof of Theorem 1. By Proposition 5, it suffices to show that LEGAL MULTI-ARRIVAL can be solved in the claimed time.

Our algorithm for Theorem 1 finds, for every $s \in S_0$, an s -directed compensated routing r_s which maximizes $\phi_s(r)$. This by Theorem 9 allows us to obtain the maximal legal routing \hat{r} given by $\hat{r}(v) = \max\{r_s(v) \mid s \in S_0\}$ for $v \in V^+(T)$.

Hence, for each $s \in S_0$ we need only show that r_s can be obtained in time $\mathcal{O}(|\hat{A}| \log^k(|A|))$, which is established by Lemma 17.

If for some constant $C > 0$, we have $k \leq \frac{C \log |A|}{\log(\log |A|)}$, then it is seen that $\log^k(|A|) \leq |A|^C$, and thus the run time of the algorithm is $\mathcal{O}\left(|A|^{\max(1, C)}\right)$.

Further, if for some constant $C > 0$, we have $k \leq C$, by the fact that $\log(|A|) \leq |\hat{A}|$, the run time is $\mathcal{O}(|\hat{A}|^{C+2})$. \blacktriangleleft

4.3 MULTI-ARRIVAL for Tree-like Simple Graphs

In this subsection, we focus on tree-like rotor graphs without parallel arcs, for which we prove that MULTI-ARRIVAL can be solved in polynomial time.

► **Theorem 2.** MULTI-ARRIVAL can be solved on tree-like rotor graphs T with arc set A , sink set S_0 and without parallel arcs in time $\mathcal{O}(|S_0||A|^3)$. Furthermore, in the procedure, a full routing is obtained.

We now employ a dynamic program that given bounds $b_1 \leq b_2$ that contain some unknown s -directed compensated routing r , constructs lower bounds of $\{h_v \mid v \in V^+(T)\}$. Then, as r is contained in those bounds, these lower bounds allow us to obtain an s -directed compensated routing $r' \geq r$ in time proportional to the size of these bounds.

► **Lemma 19.** Let T be a tree-like rotor graph, and let b_1, b_2 be two vectors such that there exists an s -directed compensated routing r with $b_1 \leq r \leq b_2$. Then we can construct an s -directed compensated routing r' such that $r' \geq r$ in time $\mathcal{O}(\Delta|\hat{A}|)$ where $\Delta = 1 + \max\{\phi_w^v(b_2(v)) - \phi_w^v(b_1(v)) \mid v \in V^+(T) \text{ and } w = N_s^+(v)\}$.

Proof. First, we inductively construct for each $v \in V^+(T)$ a strictly increasing function $\hat{f}_v: I_v \rightarrow \mathbb{N}$ where $I_v = \{\phi_w^v(b_1(v)), \dots, \phi_w^v(b_2(v))\}$ and a function $\hat{h}_v: \mathbb{N} \rightarrow \mathbb{N}$. The function \hat{h}_v will serve as a lower bound for h_v .

Suppose \hat{h}_u has been constructed for each $u \in N_s^-(v)$. We define \hat{f}_v and \hat{h}_v as follows:

$$\begin{aligned} \hat{f}_v(x) &= q_v(x) - \sigma(v) - \sum_{u \in N_s^-(v)} \hat{h}_u(q_u^v(x)), \quad \text{for } x \in I_v, \\ \hat{h}_v(y) &= \begin{cases} 0 & y < \min \hat{f}_v, \\ \max\{x \in I_v \mid \hat{f}_v(x) \leq y\} & \min \hat{f}_v \leq y. \end{cases} \end{aligned}$$

It is clear that \hat{f}_v is strictly increasing and \hat{h}_v is an increasing function by arguments analogous to that of Lemma 12. Similarly, as by induction we have $\hat{h}_u \leq h_u$ for all $u \in N_s^-(v)$, it follows that $\hat{h}_v \leq h_v$.

Next, we will show that

$$\hat{h}_v(\phi_w^v(r(w))) \geq \phi_w^v(r(v)) . \quad (12)$$

To this end, suppose that, by induction, $\hat{h}_u(\phi_u^v(r(v))) \geq \phi_v^u(r(u))$ for $u \in N_s^-(v)$. Then as r is a compensated routing, we have:

$$\phi_w^v(r(w)) \geq r(v) - \sigma(v) - \sum_{u \in N_s^-(v)} \phi_v^u(r(u)) .$$

Since the function $\phi_w^v(\cdot)$ is increasing and $b_1(v) \leq r(v) \leq b_2(v)$, we have $\phi_w^v(r(v)) \in I_v$. Also, $q_v(\phi_w^v(r(v))) = r(v)$, as r is s -directed.

Consequently,

$$\begin{aligned} \hat{f}_v(\phi_w^v(r(v))) &= r(v) - \sigma(v) - \sum_{u \in N_s^-(v)} \hat{h}_u(\phi_u^v(r(v))) \\ &\leq r(v) - \sigma(v) - \sum_{u \in N_s^-(v)} \phi_v^u(r(u)) \leq \phi_w^v(r(w)) . \end{aligned}$$

From this, by the definition of \hat{h}_v , (12) readily follows.

Next, we inductively construct an s -directed routing r' and begin by setting $r'(s) = r(s) = 0$. We assume by induction that $r'(w)$ is defined and satisfies $r'(w) \geq r(w)$.

Now, if $w \neq s$ and $r'(w) = r(w) = 0$, we set $r'(v) = 0$. From Lemma 11 we see that $r(v) = 0$. Otherwise, we define

$$r'(v) = q_v \left(\hat{h}_v(\phi_v^w(r'(w))) \right) . \quad (13)$$

From $r'(w) \geq r(w)$ and (12) it follows that $\hat{h}_v(\phi_v^w(r'(w))) \geq \hat{h}_v(\phi_v^w(r(w))) \geq \phi_v^v(r(w))$. Taking q_v from the left and right sides of this inequality, we get $r'(v) \geq r(v)$.

Since $r' \geq r$, it remains to show that r' is s -directed and compensated. We accomplish this by showing that the conditions of Lemma 11 are satisfied by r' . Let $v \in V^+(T)$ and assume for each $u \in N_s^-(v)$ we have $r'_u \in H_u(\phi_u^v(r'(u)), \phi_u^v(r'(v)))$ where r'_u denotes the restriction of r' to T_u . We will show that r'_v belongs to $H_v(\phi_v^v(r'(v)), \phi_v^w(r'(w)))$. By construction, $\phi_v^w(r'(w)) > 0$ if and only if $r'_v(v) > 0$. The case $r'_v(v) = 0$ is trivial as then we see that $r'_v = \mathbf{0}$. Otherwise, we need to show that (1)–(3) of Lemma 11 hold. By the assumption, (3) is already satisfied.

Taking $\phi_v^v(\cdot)$ from the both sides of (13), we obtain $\phi_v^v(r'(v)) = \hat{h}_v(\phi_v^w(r'(w)))$. So $r'(v)$ satisfies (1).

Finally, by definition of \hat{h}_v and r' , we have:

$$\begin{aligned} \phi_v^w(r'(w)) &\geq \hat{f}_v(\phi_v^v(r'(v))) \\ &= r'(v) - \sigma(v) - \sum_{u \in N_s^-(v)} \hat{h}_u(\phi_u^v(r'(v))) = r'(v) - \sigma(v) - \sum_{u \in N_s^-(v)} \phi_u^u(r'(u)) . \end{aligned}$$

Consequently, (2) holds as well.

Hence, the procedure returns the desired s -directed compensated routing $r' \geq r$.

Note that \hat{h}_v is increasing. Hence, it is straightforward to see that we can construct \hat{h}_v by iterating through $x \in \{\phi_w^v(b_1(v)), \dots, \phi_w^v(b_2(v))\}$ while keeping track of the index x_u for each $u \in N_s^-(v)$ that corresponds to the maximal $\hat{f}_u(x_u) \leq \phi_u^v(q_v(x))$. Hence, we can construct \hat{h}_v in time $\mathcal{O}(|\hat{A}^+(v)|\Delta + |N_s^-(v)|\Delta)$. In total, including the construction of r' , we thus take time at most $\mathcal{O}(\Delta|\hat{A}|)$. \blacktriangleleft

Proof of Theorem 2. We proceed analogously to the proof of Theorem 1. It suffices to show how to obtain r_s for each sink $s \in S_0$.

We will demonstrate that for the simple tree-like rotor graph T , we can obtain bounds $b_1 \leq b_2$ in time $\mathcal{O}(|A|)$, which contain an s -directed compensated routing r which maximizes $\phi_s(r)$. Additionally, we show that the parameter Δ , defined as in Lemma 19, satisfies $\Delta \leq \mathcal{O}(|A|^2)$. We then plug these into Lemma 19, which allows us to construct the desired routing r_s in time $\mathcal{O}(|A|^3)$. Note that, since T is simple, we have $|\hat{A}| = \mathcal{O}(|A|)$. The remainder of the proof is therefore devoted to finding the required vectors b_1 and b_2 .

Let $\{t\} = N_s^-(s)$ and let r denote the component-wise maximal routing in $H_t(h_t(0), 0)$. We first construct, for each $v \in V^+(T)$, the affine approximations \bar{h}_v given in Lemma 15. To this end, define $b_1(s) = b_2(s) = 0$. Now, for $v \in V^+(T)$ and $w = N_s^+(v)$, assume we have constructed $b_1(w) \leq r(w) \leq b_2(w)$. Hence:

$$h_v(\phi_v^w(b_1(w))) \leq h_v(\phi_v^w(r(w))) \leq h_v(\phi_v^w(b_2(w))) .$$

Since r is maximal we have that $\phi_w^v(r(v)) = h_v(\phi_v^w(r(w)))$. From r being s -directed it follows that

$$q_v(h_v(\phi_v^w(b_1(w)))) \leq r(v) \leq q_v(h_v(\phi_v^w(b_2(w)))) .$$

Hence, using \bar{h}_v with $C_v = 2|A(T_v)|$ denoting a bound on the error, we define:

$$b_1(v) = q_v \left(\bar{h}_v(\phi_v^w(b_1(w))) - C_v \right) \leq r(v) \leq q_v \left(\bar{h}_v(\phi_v^w(b_2(w))) + C_v \right) = b_2(v) .$$

Our next task, is to show that for each $v \in V^+(T)$, it holds that

$$|b_1(v) - b_2(v)| \leq 4d_+(v)|A| \operatorname{dist}(s, v) . \quad (14)$$

We prove this relation by induction starting at t . The base case is trivial, as

$$|b_1(t) - b_2(t)| = \left| q_t \left(\bar{h}_t(0) - C_t \right) - q_t \left(\bar{h}_t(0) + C_t \right) \right| \leq 4d_+(t)|A| .$$

Assume that (14) holds for v ; we now prove it for $u \in N_s^-(v)$. Since T is simple, we have that $q_u(x) - q_u(x') = d_+(u)(x - x')$ and

$$|\phi_u^v(x) - \phi_u^v(x')| \leq \left\lceil \frac{|x - x'|}{d_+(v)} \right\rceil .$$

Furthermore, as \bar{h}_v is a contraction (see (9)), we obtain:

$$\begin{aligned} |b_1(u) - b_2(u)| &= d_+(u) \left(\left| \bar{h}_v(\phi_u^v(b_1(v))) - \bar{h}_v(\phi_u^v(b_2(v))) \right| + 2C_u \right) \\ &\leq d_+(u) \left(|\phi_u^v(b_1(v)) - \phi_u^v(b_2(v))| + 2C_u \right) \\ &\leq d_+(u) \left(\frac{|b_1(v) - b_2(v)|}{|d_+(v)|} + 2C_u + 1 \right) \\ &\leq 4d_+(u) (\operatorname{dist}(s, v)|A| + |A(T_u)| + 1) \leq 4d_+(u) \operatorname{dist}(s, u)|A| . \end{aligned}$$

By (14) it holds that for each $v \in V^+(T)$, $|\phi_w^v(b_1(v)) - \phi_w^v(b_2(v))| = \mathcal{O}(|A|^2)$, implying that the error Δ , defined as in Lemma 19, satisfies $\Delta \leq \mathcal{O}(|A|^2)$. This completes the proof. \blacktriangleleft

5 Conclusion

Our main result is a quasi-polynomial time algorithm for MULTI-ARRIVAL on tree-like multigraphs. We established the polynomial-time solvability of MULTI-ARRIVAL on path-like multigraphs, and on tree-like multigraphs with bounded contracted height. Thereby, we extended the polynomial-time algorithm by Auger et al., which was restricted to path-like rotor graphs with certain uniform rotor order, to a much wider class of instances.

The notion of contracted height, as we introduce it in this paper, is related to several other classic notions of height on trees, such as topological height. Nonetheless, we have not found any direct reference of this concept in the literature. It may therefore be interesting to explore its relationship to those well-known concepts of height further.

The complexity of ARRIVAL in general graphs remains widely open. As a next step, it could be valuable to improve the quasi-polynomial time algorithms for MULTI-ARRIVAL on tree-like rotor graphs to a polynomial-time algorithm.

References

- 1 David Auger, Pierre Coucheney, and Loric Duhazé. Polynomial time algorithm for ARRIVAL on tree-like multigraphs. In *Proc. MFCS 2022*, volume 241 of *Leibniz Int. Proc. Inform.*, pages Art. No. 12, 14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.MFCS.2022.12.

- 2 David Auger, Pierre Coucheney, Loric Duhazé, and Kossi Roland Etse. Generalized ARRIVAL problem for rotor walks in path multigraphs. In *Proc. Reachability problems 2023*, volume 14235 of *Lecture Notes Comput. Sci.*, pages 183–198. Springer, 2023. doi:10.1007/978-3-031-45286-4_14.
- 3 Jérôme Dohrau, Bernd Gärtner, Manuel Kohler, Jiří Matoušek, and Emo Welzl. ARRIVAL: a zero-player graph game in $\text{NP} \cap \text{coNP}$. In *A journey through discrete mathematics*, pages 367–374. Springer, Cham, 2017. doi:10.1007/978-3-319-44479-6_14.
- 4 John Fearnley, Martin Gairing, Matthias Mnich, and Rahul Savani. Reachability switching games. *Log. Methods Comput. Sci.*, 17(2):Paper No. 10, 29, 2021. doi:10.23638/LMCS-17(2:10)2021.
- 5 Bernd Gärtner, Thomas Dueholm Hansen, Pavel Hubáček, Karel Král, Hagar Mosaad, and Veronika Slívová. ARRIVAL: Next Stop in CLS. In *Proc. ICALP 2018*, volume 107 of *Leibniz Int. Proc. Informatics*, pages 60:1–60:13, 2018. doi:10.4230/LIPICS.ICALP.2018.60.
- 6 Bernd Gärtner, Sebastian Haslebacher, and Hung P. Hoang. A subexponential algorithm for ARRIVAL. In *Proc. ICALP 2021*, volume 198 of *Leibniz Int. Proc. Inform.*, pages 69:1–69:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.69.
- 7 Giuliano Pezzolo Giacaglia, Lionel Levine, James Propp, and Linda Zayas-Palmer. Local-to-global principles for the hitting sequence of a rotor walk. *Electron. J. Combin.*, 19(1):Paper 5, 23, 2012. doi:10.37236/12.
- 8 Phuc Hung Hoang. *On Two Combinatorial Reconfiguration Problems: Reachability and Hamiltonicity*. PhD thesis, ETH Zurich, 2022.
- 9 Alexander E. Holroyd, Lionel Levine, Karola Mészáros, Yuval Peres, James Propp, and David B. Wilson. Chip-firing and rotor-routing on directed graphs. In *In and out of equilibrium. 2*, volume 60 of *Progr. Probab.*, pages 331–364. Birkhäuser, Basel, 2008. doi:10.1007/978-3-7643-8786-0_17.
- 10 Vyatcheslav B. Priezzhev, Deepak Dhar, Abhishek Dhar, and Supriya Krishnamurthy. Eulerian walkers as a model of self-organized criticality. *Phys. Rev. Lett.*, 77(25):5079, 1996. doi:10.1103/PhysRevLett.77.5079.
- 11 Lilla Tóthmérész. Rotor-routing reachability is easy, chip-firing reachability is hard. *European J. Combin.*, 101:Paper No. 103466, 9, 2022. doi:10.1016/j.ejc.2021.103466.

Identity-Preserving Lax Extensions and Where to Find Them

Sergey Goncharov   




University of Birmingham, UK

Dirk Hofmann   

CIDMA, University of Aveiro, Portugal

Pedro Nora  

Radboud Universiteit, Nijmegen, The Netherlands

Lutz Schröder   

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Paul Wild   

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Generic notions of bisimulation for various types of systems (nondeterministic, probabilistic, weighted etc.) rely on identity-preserving (*normal*) lax extensions of the functor encapsulating the system type, in the paradigm of universal coalgebra. It is known that preservation of weak pullbacks is a sufficient condition for a functor to admit a normal lax extension (the Barr extension, which in fact is then even strict); in the converse direction, nothing is currently known about necessary (weak) pullback preservation conditions for the existence of normal lax extensions. In the present work, we narrow this gap by showing on the one hand that functors admitting a normal lax extension preserve *1/4-iso* pullbacks, i.e. pullbacks in which at least one of the projections is an isomorphism. On the other hand, we give sufficient conditions, showing that a functor admits a normal lax extension if it weakly preserves either *1/4-iso* pullbacks and *4/4-epi* pullbacks (i.e. pullbacks in which all morphisms are epic) or inverse images. We apply these criteria to concrete examples, in particular to functors modelling neighbourhood systems and weighted systems.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Concurrency

Keywords and phrases (Bi-)simulations, lax extensions, modal logics, coalgebra

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.40

Related Version *Full Version:* <https://arxiv.org/abs/2410.14440>

Funding *Sergey Goncharov:* Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 501369690.

Dirk Hofmann: This work is supported by CIDMA under the FCT (Portuguese Foundation for Science and Technology) Multi-Annual Financing Program for R&D Units.

Lutz Schröder: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 531706730.

Paul Wild: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 434050016.

1 Introduction

Branching-time notions of behavioural equivalence of reactive systems are typically cast as notions of *bisimilarity*, which in turn are based on notions of *bisimulation*, the paradigmatic example being Park-Milner bisimilarity on labelled transition systems [32]. A key point about this setup is that while bisimilarity is an equivalence on states, individual bisimulations can



© Sergey Goncharov, Dirk Hofmann, Pedro Nora, Lutz Schröder, and Paul Wild; licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 40; pp. 40:1–40:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



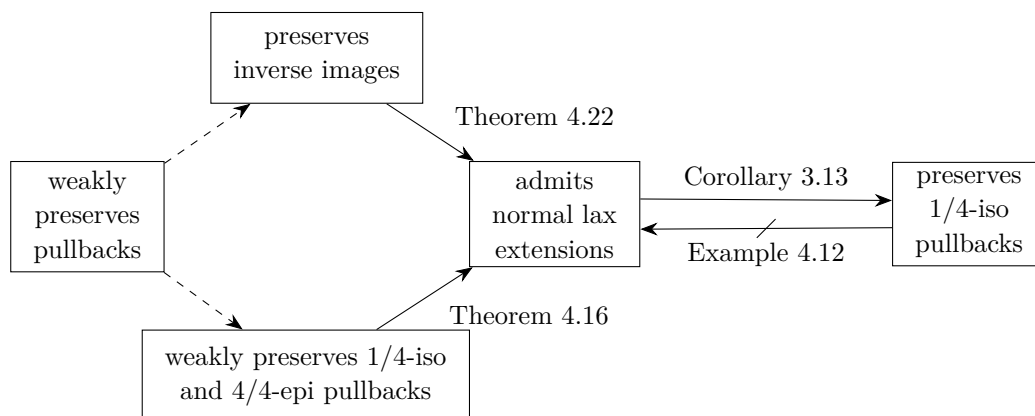
be much smaller than the full bisimilarity relation, and in particular need not themselves be equivalence relations. In a perspective where one views bisimulations as certificates for bisimilarity, this feature enables smaller certificates.

The concept of bisimilarity via bisimulations can be transferred to many system types beyond basic labelled transition systems, such as monotone neighbourhood systems [21], probabilistic transition systems, or weighted transition systems. In fact, such systems can be treated uniformly within the framework of universal coalgebra [38], in which the system type is encapsulated in the choice of a set functor (the powerset functor for non-deterministic branching, the distribution functor for probabilistic branching etc.). Coalgebraic notions of bisimulation were originally limited to functors that preserve weak pullbacks [38], equivalently admit a strictly functorial extension to the category of relations [5, 44]. They were later extended to functors admitting an *identity-preserving* or *normal lax extension* [30, 31] to the category of relations (this is essentially equivalent to notions of bisimilarity based on modal logic [15]). While there is currently no formal general definition of what a notion of bisimulation constitutes except via normal lax extensions, there is a reasonable claim [30, 31] that notions of bisimulation in the proper sense, in particular with bisimulations not required to be equivalence relations but stable under key operations such as relational composition, will not go beyond functors admitting a normal lax extension.

The *Barr extension* that underlies the original notion of coalgebraic bisimulation for weak-pullback-preserving functors [38] is, in particular, a normal lax extension; that is, preservation of weak pullbacks is sufficient for existence of a normal lax extension. However, this condition is far from being necessary; there are numerous functors that fail to preserve weak pullbacks but do admit a normal lax extension, such as the monotone neighbourhood functor [30, 31]. Using the latter fact, it has been shown that a finitary functor admits a normal lax extension if and only if it admits a separating set of finitary monotone modalities [30, 31], cast as monotone predicate liftings in the paradigm of coalgebraic logic [34, 39] (a similar result holds for unrestricted functors if one considers class-sized collections of infinitary modalities [14]). The latter condition amounts to existence of an expressive modal logic that has monotone modalities [34, 39], and as such admits μ -calculus-style fixpoint extensions [11]. In a nutshell, a system type admits a good notion of bisimulation if and only if it admits an expressive temporal logic. Both sides of this equivalence, however, need to be witnessed by the construction of a fairly complicated object; what is missing is a characterization via *properties* of the underlying functor, rather than via the existence of extra structure.

In the present work, we narrow the gap between weak pullback preservation as a sufficient condition for admitting a normal lax extension, and no known necessary pullback preservation condition. On the one hand, we establish a necessary preservation condition, showing that functors admitting a normal lax extension (weakly) preserve *1/4-iso pullbacks*, i.e. pullbacks in which at least one of the projections is isomorphic. (We often put “weakly” in brackets because for many of the pullback types we consider, notably for inverse images and 1/4-iso pullbacks, weak preservation coincides with preservation.) This is a quite natural condition: A key role in the field is played by *difunctional relations* [36], which may be thought of as relations obtained by chopping the domain of an equivalence in half; for instance, given labelled transition systems X, Y , the bisimilarity relation from X to Y is difunctional. In a nutshell, we show that a functor preserves 1/4-iso pullbacks iff it acts in a well-defined and monotone manner on difunctional relations. A first application of this necessary condition is a very quick proof of the known fact that the neighbourhood functor does not admit a normal lax extension [31].

We then go on to establish two separate sets of sufficient conditions: We show that a functor admits a normal lax extension if it (weakly) preserves either inverse images or both 1/4-iso pullbacks and 4/4-epi pullbacks, i.e. pullbacks in which all morphisms are epi (these are also known as *surjective pullbacks* [42], and weak preservation of 4/4-epi pullbacks is equivalent to weak preservation of kernel pairs [16]). These sufficient conditions are technically substantially more involved. As indicated above, they imply that finitary functors (weakly) preserving either inverse images or 1/4-iso pullbacks and 4/4-epi pullbacks admit a separating set of finitary monotone modalities; this generalizes a previous result showing the same for functors preserving all weak pullbacks [28]. We summarize our main contributions in Figure 1.



■ **Figure 1** Summary of main results. Solid arrows are present contributions, dashed arrows are trivial. All implications indicated by arrows are non-reversible; in particular, Example 4.12 shows this for Corollary 3.13.

As per the preceding discussion, these necessary and sufficient criteria essentially determine (when applicable) whether or not a given type of systems admits a good notion of bisimulation.

The criterion of weak preservation of 1/4-iso pullbacks and 4/4-epi pullbacks is satisfied by the monotone neighbourhood functor and generalizations thereof (e.g. [42]), and thus in particular reproves the above-mentioned known fact that functors admitting separating sets of monotone modalities have normal lax extensions. The criterion of (weak) preservation of inverse images, in connection with the necessary criterion, implies that a monoid-valued functor for a commutative monoid M (whose coalgebras are M -weighted transition systems) admits a normal lax extension if and only if M is positive (which in turn is equivalent to the functor preserving inverse images [18]).

Related work. With variations in the axiomatics and terminology, lax extensions go back to an extended strand of work on relation liftings (e.g. [3, 43, 22, 29, 41, 40]). We have already mentioned work by Marti and Venema relating lax extensions to modal logic [30, 31]; at the same time, Marti and Venema prove that the notion of bisimulation induced by a normal lax extension captures the standard notion of behavioural equivalence. *Lax relation liftings*, constructed for functors carrying a coherent order structure [24], also serve the study of coalgebraic simulation but obey a different axiomatics than lax extensions [31, Remark 4]). Strictly functorial (and converse-preserving) extensions of set functors to the category of sets and relations are known to be unique when they exist, and exist if and only if the functor preserves weak pullbacks [7, 44]; this has been extended to other base

categories [3, 8]. There has been both longstanding and recent interest in quantitative notions of relation liftings and lax extensions that act on relations taking values in a quantale, such as the unit interval, in particular with a view to obtaining notions of quantitative bisimulation [37, 47, 23, 13, 45, 46, 14] that witness low behavioural distance (the latter having first been treated in coalgebraic generality by Baldan et al. [4]). The correspondence between normal lax extensions and separating sets of modalities generalizes to the quantitative setting [45, 46, 14].

Organization. We review material on relations, in particular difunctional relations, and lax extensions in Section 2. In Section 3, we introduce our necessary pullback preservation condition and show that it characterizes well-definedness of the natural functor action on difunctional relations. We prove our main results in Section 4. In Subsection 4.1 we show that a functor that weakly preserves 1/4-iso pullbacks and 4/4-epi pullbacks admits a normal lax extension, and in Subsection 4.2 we show the same for functors that preserve 1/4-mono pullbacks.

2 Preliminaries: Relations and Lax Extensions

We work in the category \mathbf{Set} of sets and functions throughout. We assume basic familiarity with category theory (e.g. [2]). A central role in the development is played by (weak) pullbacks: A commutative square $f \cdot p = g \cdot q$ is a pullback (of f, g) if for every competing square $f \cdot p' = g \cdot q'$, there exists a unique morphism k such that $p \cdot k = p'$ and $q \cdot k = q'$; the notion of weak pullback is defined in the same way except that k is not required to be unique. A functor F *weakly preserves* a given pullback if it maps the pullback to a weak pullback; it is known that weak preservation of pullbacks of a given type is equivalent to preservation of weak pullbacks of the same type [17, Corollary 4.4]. Our interest in functors $F: \mathbf{Set} \rightarrow \mathbf{Set}$ is driven mainly by their role as encapsulating types of transition systems in the paradigm of *universal coalgebra* [38]: An **F-coalgebra** (X, α) consists of a set X of **states** and a **transition map** $\alpha: X \rightarrow FX$ assigning to each state $x \in X$ a collection $\alpha(x)$ of successors, structured according to F . For instance, coalgebras for the **powerset functor** \mathcal{P} assign to each state a *set* of successors, and hence are just standard relational transition systems, while coalgebras for the **distribution functor** \mathcal{D} (which maps a set X to the set of discrete probability distributions on X) assign to each state a distribution on successor states, and are thus probabilistic transition systems.

A **morphism** $f: (X, \alpha) \rightarrow (Y, \beta)$ of F -coalgebras is a map $f: X \rightarrow Y$ for which $\beta \cdot f = Ff \cdot \alpha$. Such morphisms are thought of as preserving the behaviour of states, and correspondingly, states x and y in coalgebras (X, α) and (Y, β) , respectively, are **behaviourally equivalent** if there exist a coalgebra (Z, γ) and morphisms $f: (X, \alpha) \rightarrow (Z, \gamma)$, $g: (Y, \beta) \rightarrow (Z, \gamma)$ such that $f(x) = g(y)$.

► **Example 2.1.** On relational transition systems, i.e. coalgebras for the powerset functor \mathcal{P} , behavioural equivalence instantiates to the usual notion of bisimilarity. More generally, labelled transition systems with labels taken from a set \mathcal{A} are coalgebras for the functor $\mathcal{P}(\mathcal{A} \times (-))$, and behavioural equivalence instantiates to Park-Milner bisimilarity [1]. On Markov chains, understood as \mathcal{D} -coalgebras, all states are behaviourally equivalent, as all states are identified in the final coalgebra $1 \rightarrow \mathcal{D}1 \cong 1$. This triviality is removed in various forms of probabilistic *labelled* transition systems, for instance in $\mathcal{D}(\mathcal{A} \times (-))$ -coalgebras, on which behavioural equivalence instantiates to standard notions of probabilistic bisimilarity [26].

One is then interested in notions of bisimulation relation that characterize behavioural equivalence in the sense that two states are behaviourally equivalent iff they are related by some bisimulation [38, 31]; this motivates the detailed study of relations and of extensions of F that act on relations. We write $r: X \rightarrow Y$ to indicate that r is a relation from the set X to the set Y (i.e. $r \subseteq X \times Y$), and we write $x r y$ when $(x, y) \in r$. Both for functions and for relations, we use *applicative* composition, i.e. given $r: X \rightarrow Y$ and $s: Y \rightarrow Z$, their composite is $s \cdot r: X \rightarrow Z$ (defined as $s \cdot r = \{(x, z) \mid \exists y \in Y. x r y s z\}$). We say that r, s of type $r: X \rightarrow Y$ and $s: Y \rightarrow Z$ are **composable**, and we extend this terminology to sequences of relations in the obvious manner. Relations between the same sets are ordered by inclusion, that is $r \leq r' \iff r \subseteq r'$. We denote by $1_X: X \rightarrow X$ the identity map (hence relation) on X , and we say that a relation $r: X \rightarrow X$ is a **subidentity** if $r \leq 1_X$. Given a relation $r: X \rightarrow Y$, $r^\circ: Y \rightarrow X$ denotes the corresponding converse relation; in particular, if $f: X \rightarrow Y$ is a function, then $f^\circ: Y \rightarrow X$ denotes the converse of the corresponding relation. For a relation $r: X \rightarrow Y$, we denote by $\text{dom } r \subseteq X$ and $\text{cod } r \subseteq Y$ the respective domain and codomain (i.e. $\text{dom } r = \{x \in X \mid \exists y \in Y. x r y\}$ and $\text{cod } r = \{y \in Y \mid \exists x \in X. x r y\}$). A special class of relations of interest are **difunctional relations** [36], which are relations factorizable as $g^\circ \cdot f$ for some functions $f: X \rightarrow Z$ and $g: Y \rightarrow Z$, i.e. $x r y$ iff $f(x) = g(y)$. In the following we record some folklore facts about difunctional relations.

► **Lemma 2.2.** *Let $r: X \rightarrow Y$ be a relation. Then the following are equivalent:*

1. r is difunctional;
2. for all x_1, x_2 in X and $y_1, y_2 \in Y$, if $x_1 r y_1$ and $x_2 r y_2$, then $x_1 r y_2$.
3. for every span $X \xleftarrow{\pi_1} R \xrightarrow{\pi_2} Y$ such that $r = \pi_2 \cdot \pi_1^\circ$, the pushout square

$$\begin{array}{ccc} R & \xrightarrow{\pi_2} & Y \\ \pi_1 \downarrow & \lrcorner & \downarrow p_2 \\ X & \xrightarrow{p_1} & O \end{array}$$

is a weak pullback.

As we can see in Lemma 2.2(3) above, difunctional relations are characterized as weak pullbacks, and in this regard we recall that generally, a commutative square $f \cdot p = g \cdot q$ is a weak pullback iff $q \cdot p^\circ = g^\circ \cdot f$, equivalently $p \cdot q^\circ = f^\circ \cdot g$.

The **difunctional closure** of a relation $r: X \rightarrow Y$ is the least difunctional relation $\hat{r}: X \rightarrow Y$ greater than or equal to r . It follows from Lemma 2.2 that the difunctional closure of a relation $r: X \rightarrow Y$ given by a span $X \xleftarrow{\pi_1} R \xrightarrow{\pi_2} Y$ is obtained by computing its pushout $X \xrightarrow{p_1} O \xleftarrow{p_2} Y$; i.e., the difunctional closure \hat{r} of r is the relation $p_2^\circ \cdot p_1$. More explicitly, $\hat{r} = \bigvee_{n \in \mathbb{N}} r \cdot (r^\circ \cdot r)^n$ (e.g. [36, 20]).

A **lax extension** L of an endofunctor $F: \text{Set} \rightarrow \text{Set}$ is a mapping that sends any relation $r: X \rightarrow Y$ to a relation $Lr: FX \rightarrow FY$ in such a way that

- (L1) $r \leq r' \implies Lr \leq Lr'$,
- (L2) $Ls \cdot Lr \leq L(s \cdot r)$,
- (L3) $Ff \leq Lf$ and $(Ff)^\circ \leq L(f^\circ)$,

for all $r: X \rightarrow Y$, $s: Y \rightarrow Z$ and $f: X \rightarrow Y$. We define **relax extensions** in the same way, without however requiring property (L2). We call a (re)lax extension **identity-preserving**, or **normal**, if $L1_X = 1_{FX}$ for every set X , and we say that a (re)lax extension **preserves converses** if $L(r^\circ) = (Lr)^\circ$.

A tactical advantage of using the term “relax extension” is that we can thus refer to constructions that produce lax extensions most of the time, except for some cases when (L2) may fail. A prototypical example of this sort is the **Barr extension** \bar{F} [6], which for weak-pullback-preserving F is even a strict extension, and is defined as follows. Given a relation $r: X \rightarrow Y$, choose a factorization $\pi_2 \cdot \pi_1^\circ$ for some span $X \xleftarrow{\pi_1} R \xrightarrow{\pi_2} Y$ and put $\bar{F}r = F\pi_2 \cdot (F\pi_1)^\circ$. This assignment is independent of the factorization of r , and r admits a **canonical factorization** which is given by projecting into X and Y the subset of $X \times Y$ of pairs of elements related by r . It is well-known that for every Set -functor, the Barr extension is a normal relax extension, but it is a lax extension precisely when F preserves weak pullbacks [27]. In this case, the Barr extension is also the least lax extension of F , for it follows from (L1)–(L3) that $F\pi_2 \cdot (F\pi_1)^\circ \leq Lr$ for every lax extension L .

Lax extensions have been used extensively to treat the notion of bisimulation coalgebraically (e.g. [22, 29, 31]). Given a lax extension $L: \text{Rel} \rightarrow \text{Rel}$ of a functor $F: \text{Set} \rightarrow \text{Set}$, an **L -simulation** between F -coalgebras (X, α) and (Y, β) is a relation $s: X \rightarrow Y$ such that $\beta \cdot s \leq Ls \cdot \alpha$, that is, whenever $x r y$, then $\alpha(x) Lr \beta(y)$. If L preserves converses, then L -simulations are more suitably called **L -bisimulations**. Between two given coalgebras, there is a greatest L -(bi)simulation, which is termed **L -(bi)similarity**. It has been shown [31] that if L is normal and preserves converses, then L -bisimilarity coincides with coalgebraic behavioural equivalence as recalled above. The axioms of lax extensions guarantee that L -bisimulations are closed under converse and composition and that coalgebra morphisms are (functional) L -bisimulations, so that L -bisimilarity includes behavioural equivalence; that is, L -bisimilarity is *complete* for behavioural equivalence. Normality of lax extensions ensures that L -bisimulations are *sound* for behavioural equivalence, i.e. L -bisimilarity is included in behavioural equivalence.

► **Example 2.3.**

1. For relational transition systems, understood as \mathcal{P} -coalgebras, we have a normal lax extension L of \mathcal{P} given by the standard Barr extension, which in turn coincides with the well-known Egli-Milner extension: Given $r: X \rightarrow Y$, $S \in \mathcal{P}X$, and $T \in \mathcal{P}Y$, we have $S Lr T$ iff for all $x \in S$, there exists $y \in T$ such that $x r y$, and symmetrically. An L -bisimulation is then just a bisimulation in the standard sense.
2. On $F = \mathcal{D}(\mathcal{A} \times (-))$, we have a normal lax extension L given for $r: X \rightarrow Y$, $\mu \in \mathcal{D}(\mathcal{A} \times X)$, $\nu \in \mathcal{D}(\mathcal{A} \times Y)$ by $\mu Lr \nu$ iff for all $l \in \mathcal{A}$, $A \in \mathcal{P}X$, we have $\nu(\{l\} \times r[A]) \geq \mu(\{l\} \times A)$, and symmetrically [15]. The arising notion of L -bisimulation is sound and complete for probabilistic bisimilarity on probabilistic labelled transition systems.

► **Remark 2.4.** As mentioned in the introduction, a functor F admits a normal lax extension iff F admits a separating class of monotone predicate liftings [31, 14]. For readability, we discuss only the case where both the functor and the predicate liftings are finitary [31]. An *n -ary predicate lifting* λ for F is a natural transformation of type $\lambda: \mathcal{Q}^n \rightarrow \mathcal{Q} \cdot F^{\text{op}}$ where \mathcal{Q} denotes the contravariant powerset functor (given by $\mathcal{Q}X$ being the powerset of a set X , and $\mathcal{Q}f(B) = f^{-1}[B]$ for $f: X \rightarrow Y$ and $B \in \mathcal{Q}Y$); that is, for a set X , λ_X lifts n predicates on X to a predicate on FX . Predicate liftings determine modalities in coalgebraic modal logic [34, 39]; a basic example is the unary predicate lifting λ for the (covariant) powerset functor \mathcal{P} given by $\lambda_X(A) = \{B \in \mathcal{P}X \mid B \subseteq A\}$ for a predicate $A \subseteq X$, which determines the standard box modality on \mathcal{P} -coalgebras, i.e. on standard relational transition systems. A set of predicate liftings is *separating* if distinct elements of FX can be separated by lifted predicates; this condition ensures that the associated instance of coalgebraic modal logic is

expressive, i.e. separates behaviourally inequivalent states [34, 39]. Monotonicity of predicate liftings allows the definition of modal fixpoint logics for temporal specification [11]. In the mentioned correspondence between lax extensions and predicate liftings, the construction of predicate liftings from a lax extension L roughly speaking involves application of L to the elementhood relation.

3 Functor Actions on Difunctional Relations

Our pullback preservation criterion for existence of normal lax extensions grows from an analysis of how functors act on difunctional relations. To start off, it is well-known that normal lax extensions of a given Set -functor are given on difunctional relations by the action of the functor (e.g. [31, 23]):

► **Proposition 3.1.** *Let L be an assignment of relations $Lr: FX \leftrightarrow FY$ to relations $r: X \leftrightarrow Y$ that satisfies (L1), (L2) as well as $1_{FX} \leq L1_X$ for all $X \in \text{Set}$. Then L is a lax extension of F iff for all functions $f: W \rightarrow X$, $g: Z \rightarrow Y$ and relations $r: X \leftrightarrow Y$, $L(g^\circ \cdot r \cdot f) = (Fg)^\circ \cdot Lr \cdot Ff$.*

► **Corollary 3.2.** *All normal lax extensions of a given Set -functor coincide on difunctional relations. Specifically, for every normal lax extension L of $F: \text{Set} \rightarrow \text{Set}$, $L(g^\circ \cdot f) = Fg^\circ \cdot Ff$ for all $f: X \rightarrow A$ and $g: Y \rightarrow A$.*

Therefore, a functor $F: \text{Set} \rightarrow \text{Set}$ that admits at least one normal lax extension must be **monotone on difunctional relations** in the following sense: for all difunctional relations $g^\circ \cdot f: X \leftrightarrow Y$ and $g'^\circ \cdot f': X \leftrightarrow Y$, if $g^\circ \cdot f \leq g'^\circ \cdot f'$ then $(Fg)^\circ \cdot Ff \leq (Fg')^\circ \cdot Ff'$. This property no longer mentions lax extensions, and implies that the functor is **well-defined on difunctional relations**, i.e. that F sends cospans that determine the same difunctional relation to cospans that determine the same difunctional relation. In this section, we show that being monotone on difunctional relations is equivalent to preserving 1/4-iso (2/4-mono) pullbacks in the sense defined next; as indicated in the introduction, this allows for a quick proof of the fact that the neighbourhood functor fails to admit a normal lax extension [31]. On this occasion, we also discuss various types of pullbacks and their (weak) preservation in some more breadth for later use in our sufficient criteria (Section 4).

► **Definition 3.3.** *We say that a functor $F: \text{Set} \rightarrow \text{Set}$ preserves 1/4-iso 2/4-mono pullbacks, 1/4-iso pullbacks, 1/4-mono pullbacks and inverse images if it sends pullbacks of the following forms, respectively, to pullbacks, with arrows \mapsto and $\xrightarrow{\cong}$ indicating injectivity and bijectivity correspondingly.*

$$\begin{array}{cccc}
 P \xrightarrow{\cong} B & P \xrightarrow{\cong} B & P \longrightarrow B & P \longrightarrow B \\
 \downarrow \lrcorner \downarrow & \downarrow \lrcorner \downarrow & \downarrow \lrcorner \downarrow & \downarrow \lrcorner \downarrow \\
 X \longrightarrow Y & X \longrightarrow Y & X \longrightarrow Y & X \longrightarrow Y
 \end{array}$$

► **Remark 3.4.** 1/4-Iso 2/4-mono pullbacks are special inverse images, characterized by the property that the fibre over every element in the image of the function $B \mapsto Y$ is a singleton. In particular, the inverse image of the empty subset is a 1/4-iso 2/4-mono pullback.

Due to the following proposition, for consistency, we tend to use “preservation of 1/4-mono pullbacks” instead of “preservation of inverse images”.

► **Proposition 3.5.** *A Set -functor preserves 1/4-mono pullbacks iff it preserves inverse images.*

40:8 Identity-Preserving Lax Extensions and Where to Find Them

Similarly, we will see in Theorem 3.12 that preservation of 1/4-iso pullbacks is equivalent to preservation of 1/4-iso 2/4-mono pullbacks. We thus tend to use the terms “1/4-iso 2/4-mono pullback preserving” and “1/4-iso pullback preserving” interchangeably. Furthermore, in Example 3.10 we will see that preservation of 1/4-mono pullbacks is properly stronger than preservation of 1/4-iso pullbacks.

Each of the preservation properties introduced in Definition 3.3 implies preservation of monomorphisms, even if we only require that the corresponding pullbacks are weakly preserved. Hence, as at least one of the projections of the pullbacks is monic, preserving the pullbacks mentioned is equivalent to weakly preserving them, and, therefore, each of the properties is implied by weakly preserving pullbacks. Also, note that weakly preserving limits of a given shape is equivalent to preserving weak limits of that shape (e.g. [17, Corollary 4.4]). Furthermore, weakly preserving pullbacks is known to be sufficient for the existence of a normal lax extension – the Barr extension – and this condition can be decomposed as follows:

► **Theorem 3.6** ([19, Theorem 2.7]). *A Set-functor weakly preserves pullbacks iff it weakly preserves inverse images and kernel pairs.*

It turns out that weakly preserving kernel pairs is equivalent to weakly preserving 4/4-epi pullbacks as defined next.

► **Definition 3.7.** *We say that a functor $F: \text{Set} \rightarrow \text{Set}$ weakly preserves 4/4-epi pullbacks, if it sends pullbacks of the form*

$$\begin{array}{ccc} P & \twoheadrightarrow & B \\ \downarrow & \lrcorner & \downarrow \\ X & \twoheadrightarrow & Y, \end{array}$$

with arrows \twoheadrightarrow indicating surjectivity, to weak pullbacks (necessarily of surjections).

► **Theorem 3.8** ([16, Corollary 5]). *A Set-functor weakly preserves kernel pairs iff it weakly preserves 4/4-epi pullbacks.*

Therefore, the condition of weakly preserving pullbacks can be decomposed as:

► **Corollary 3.9.** *A Set-functor weakly preserves pullbacks iff it weakly preserves 1/4-mono pullbacks and 4/4-epi pullbacks.*

In Section 4, we will show that either preserving 1/4-mono pullbacks or weakly preserving 1/4-iso pullbacks and 4/4-epi pullbacks is sufficient for the existence of a normal lax extension.

► **Example 3.10.**

1. The subfunctor $(-)_2^3: \text{Set} \rightarrow \text{Set}$ of the functor $(-)^3: \text{Set} \rightarrow \text{Set}$ that sends a set X to the set of triples of elements of X consisting of at most two distinct elements does not preserve pullbacks weakly [1] but it preserves inverse images.
2. The neighbourhood functor $\mathcal{N}: \text{Set} \rightarrow \text{Set}$ (whose coalgebras are neighbourhood frames [10]) sends a set X to the set $\mathcal{N}X = \mathcal{P}\mathcal{P}X$ of neighbourhood systems over X , and a function $f: X \rightarrow Y$ to the function $\mathcal{N}f: \mathcal{N}X \rightarrow \mathcal{N}Y$ that assigns to every element $\mathcal{A} \in \mathcal{N}X$ the set $\{B \subseteq Y \mid f^{-1}[B] \in \mathcal{A}\}$. The monotone neighbourhood functor $\mathcal{M}: \text{Set} \rightarrow \text{Set}$ is the subfunctor of the neighbourhood functor that sends a set X to the set of upward-closed subsets of $(\mathcal{P}X, \subseteq)$. Its coalgebras are monotone neighbourhood frames, which feature, e.g., in the semantics of game logic [33] and concurrent dynamic logic [35]. A closely related functor is the clique functor $\mathcal{C}: \text{Set} \rightarrow \text{Set}$, which is the

subfunctor of \mathcal{M} given by $\mathcal{C}X = \{\alpha \in \mathcal{M}X \mid \forall A, B \in \alpha. A \cap B \neq \emptyset\}$. The functors \mathcal{M} and \mathcal{C} do not preserve inverse images: Consider the sets $3 = \{0, 1, 2\}$ and $2 = \{a, b\}$. Let $e: 3 \rightarrow 2$ be the function that sends $0, 1$ to a and 2 to b , and $B = \{a\}$. Then $\mathcal{M}e(\uparrow\{0, 1\} \cup \uparrow\{1, 2\}) = \uparrow\{a\}$, where \uparrow denotes upwards closure, but $e^{-1}[B] = \{0, 1\}$ and $\uparrow\{0, 1\} \cup \uparrow\{1, 2\}$ does not belong to the image of the function $\mathcal{M}i: \mathcal{M}\{0, 1\} \rightarrow \mathcal{M}3$, where $i: \{0, 1\} \rightarrow 3$ denotes the corresponding inclusion. However, routine calculations show that these functors do preserve 1/4-iso (2/4-mono) pullbacks and weakly preserve 4/4-epi pullbacks (for the first functor, see [42, Proposition 4.4]).

3. Given a commutative monoid $(M, +, 0)$ (or just M), the *monoid-valued functor* $M^{(-)}$ maps a set X to the set $M^{(X)}$ of functions $\mu: X \rightarrow M$ with *finite support*, i.e. $\mu(x) \neq 0$ for only finitely many x . The coalgebras of $M^{(-)}$ are M -weighted transition systems. It is known that $M^{(-)}$ preserves inverse images iff M is *positive*, i.e. does not have non-zero invertible elements [18, Theorem 5.13] (the cited theorem shows the equivalence for non-empty inverse images; it is easy to check that in case M is positive, $M^{(-)}$ preserves empty pullbacks). Moreover, $M^{(-)}$ preserves weak pullbacks iff M is positive and *refinable*, i.e. whenever $m_1 + m_2 = n_1 + n_2$ for $m_1, m_2, n_1, n_2 \in M$, then there exists a 2×2 -matrix with entries in M whose i -th column sums up to m_i and whose j -th row sums up to n_j , for $i, j \in \{1, 2\}$ [18, Theorem 5.13]. Monoids that are positive but not refinable are fairly common [12]; the simplest example is the additive monoid $\{0, 1, 2\}$ where $2 + 1 = 2$. The functor $M^{(-)}$ preserves 1/4-iso (2/4-mono) pullbacks iff it preserves inverse images iff M is positive. Indeed, suppose that M is not positive. Consider the functions $!_2: 2 \rightarrow 1$, $!_\emptyset: \emptyset \rightarrow 1$. Then, for mutually inverse non-zero elements u and v of M , the function $M^{(!_2)}$ sends both the pair $(0, 0)$ and the pair (u, v) to $0 \in M^{(1)}$, which is in the image of $M^{(!_\emptyset)}: M^{(\emptyset)} \rightarrow M^{(1)}$. Therefore, the functor $M^{(-)}$ does not preserve the (1/4-iso) pullback of $(!_2, !_\emptyset)$: This pullback has vertex \emptyset , and $M^{(\emptyset)}$ has only one element.
4. In recent work [16], it has been shown that the functor of a monad induced by a variety of algebras preserves inverse images iff whenever a variable x is canceled from a term when identified with other variables, then the term does not actually depend on x . This provides a large reservoir of functors that preserve inverse images but do not always have easily guessable normal lax extensions (whose existence will however be guaranteed by our main results). One example is the functor that maps a set X to the free semigroup over X quotiented by the equation $xxx = xx$, as neither this equation nor associativity cancel any variables. Notice that this functor does not preserve 4/4-epi pullbacks.

Finally, we show that being monotone on difunctional relations is equivalent to preserving 1/4-iso (2/4-mono) pullbacks. The next lemma connects the order on difunctional relations and pullbacks of such type.

► **Lemma 3.11.** *Let $X \xrightarrow{f} A \xleftarrow{g} Y$ and $X \xrightarrow{f'} A' \xleftarrow{g'} Y$ be cospans for which there is a map $h: A \rightarrow A'$ such that $f' = h \cdot f$ and $g' = h \cdot g$. Moreover, consider the commutative square*

$$\begin{array}{ccc} f[X] \cap g[Y] & \xrightarrow{h'} & f'[X] \cap g'[Y] \\ \downarrow & & \downarrow \\ A & \xrightarrow{h} & A' \end{array} \quad (3.i)$$

where $h': f[X] \cap g[Y] \rightarrow f'[X] \cap g'[Y]$ is the restriction of h to $f[X] \cap g[Y]$ and the vertical arrows denote subset inclusions.

1. If $g^\circ \cdot f \geq g'^\circ \cdot f'$, then h' is a bijection.
2. If $g^\circ \cdot f \geq g'^\circ \cdot f'$ and the cospan (f, g) is epi, then (3.i) is a pullback.
3. If h' is a bijection and (3.i) is a pullback, then $g^\circ \cdot f \geq g'^\circ \cdot f'$.

40:10 Identity-Preserving Lax Extensions and Where to Find Them

(Notice in particular that if h' is a bijection and (3.i) is a pullback, then (3.i) is a 1/4-iso pullback.) Using Lemma 3.11, one proves the announced characterization:

► **Theorem 3.12.** *The following clauses are equivalent for a functor $F: \mathbf{Set} \rightarrow \mathbf{Set}$:*

1. F preserves 1/4-iso 2/4-mono pullbacks.
2. F is well-defined on difunctional relations.
3. F is monotone on difunctional relations.
4. F preserves 1/4-iso pullbacks.

► **Corollary 3.13.** *If a Set-functor admits a normal lax extension, then it preserves 1/4-iso pullbacks.*

Therefore, the following functors do not admit a normal lax extension.

► **Example 3.14.**

1. The neighbourhood functor $\mathcal{N}: \mathbf{Set} \rightarrow \mathbf{Set}$ (cf. Example 3.10(2)) does not preserve 1/4-iso pullbacks: the element $\mathcal{P}1 \in \mathcal{N}1$ belongs to the image of the function $\mathcal{N}!_{\emptyset}$, with $!_{\emptyset}: \emptyset \rightarrow 1$, however, its fiber w.r.t. $\mathcal{N}!_2$, with $!_2: 2 \rightarrow 1$, is not a singleton.
2. For every non-positive commutative monoid, the monoid valued functor $M^{(-)}: \mathbf{Set} \rightarrow \mathbf{Set}$ does not preserve 1/4-iso pullbacks (Example 3.10(3)).
3. More generally, by (the proof of) [12, Proposition 4.4], the functor $F: \mathbf{Set} \rightarrow \mathbf{Set}$ of the monad induced by a variety of algebras that admits a weak form of subtraction (for instance, groups, rings, vector spaces) does not preserve 1/4-iso pullbacks.
4. For every set A with at least two elements, consider the functor $\mathbf{Set}(A, -)/\sim$ that maps a set X to the quotient of the set $\mathbf{Set}(A, X)$ by the equivalence relation \sim that identifies exactly all non-injective maps, and maps a function $f: X \rightarrow Y$ to the one sending the equivalence class of $g: A \rightarrow X$ to that of $f \cdot g$. This functor does not preserve 1/4-iso pullbacks. For instance, for $A = \{0, 1\}$, consider the sets $3 = \{a, b, c\}$ and $B = \{0\}$. Then, the fibre of each element of $B \subseteq A$ w.r.t. the function $f: 3 \rightarrow A$ that sends a to 0 and b, c to 1 is a singleton; however, the fibre of the equivalence class of the constant map into 0 w.r.t. $\mathbf{Set}(A, f)_{\sim}$ is not a singleton. Similar counterexamples can be constructed for arbitrary A with at least two elements.

► **Remark 3.15.** Every coalgebra can be quotiented by behavioural equivalence (e.g. [25]). Such a quotient can be described by a *cocongruence* on a given coalgebra, i.e. an equivalence relation that is compatible with the coalgebra structure, and, of course, a cocongruence can be specified by a generating relation that need not itself be an equivalence. For instance, cocongruences have been studied in the context of linear weighted automata [9] (where they are in fact termed bisimulations), and even on neighbourhood frames, one obtains such a notion of equivalence-witnessing relation from the standard Barr extension [31]. All this does not contradict the moral claim that, by Example 3.14, there are no “good” notions of bisimulation for, e.g., neighbourhood frames or integer-weighted transition systems, as (generating relations of) cocongruences are missing some of the features that we include in the wish list for bisimulations and that L -bisimulations do provide (cf. Section 2). Notably, cocongruences work only on a single coalgebra (while we expect bisimulations to connect two possibly different coalgebras), and they fail to be closed under relational composition.

4 Existence of Normal Lax Extensions

We proceed to present the main results of the paper: a Set-functor that weakly preserves 1/4-iso pullbacks and 4/4-epi pullbacks, or that preserves 1/4-mono pullbacks admits a normal lax extension. In view of the facts recalled in Section 2, this means that these functors

admit a notion of bisimulation that captures behavioural equivalence, or equivalently, that they admit a separating class of monotone predicate liftings.

We begin by showing that the smallest lax extension of a **Set**-functor is obtained by “closing its Barr relax extension under composition”. As a consequence, in Corollary 4.5 we obtain a criterion to determine if a **Set**-functor admits a normal lax extension.

Consider the partially ordered classes $\text{Lax}(F)$ and $\text{ReLax}(F)$ of lax and relax extensions of F , respectively, ordered pointwise. With the following result we can construct lax extensions from relax extensions in a universal way.

► **Proposition 4.1.** *Let $F: \text{Set} \rightarrow \text{Set}$ be a functor. The inclusion $\text{Lax}(F) \hookrightarrow \text{ReLax}(F)$ has a left adjoint $(-)^{\bullet}: \text{ReLax}(F) \rightarrow \text{Lax}(F)$ that sends a relax extension $R: \text{Rel} \rightarrow \text{Rel}$ of F to its laxification $R^{\bullet}: \text{Rel} \rightarrow \text{Rel}$, which is defined on $r: X \rightarrow Y$ by*

$$R^{\bullet}r = \bigvee_{\substack{r_1, \dots, r_n: \\ r_n \cdot \dots \cdot r_1 \leq r}} Rr_n \cdot \dots \cdot Rr_1. \quad (4.i)$$

Furthermore, if a relax extension $R: \text{Set} \rightarrow \text{Set}$ preserves converses, then so does its laxification.

Since every lax extension of a functor is greater than or equal to the Barr relax extension (cf. Section 2), we thus have:

► **Corollary 4.2.** *The smallest lax extension of a functor is given by the laxification of its Barr relax extension.*

For the Barr relax extension of a **Set**-functor, the supremum in the formula (4.i) can be restricted as follows.

► **Lemma 4.3.** *For every composable sequence r_1, \dots, r_n such that $r_n \cdot \dots \cdot r_1 \leq r$, for some relation r , there is a composable sequence r'_1, \dots, r'_n such that $r'_n \cdot \dots \cdot r'_1 = r$ and $\bar{F}r_n \cdot \dots \cdot \bar{F}r_1 \leq \bar{F}r'_n \cdot \dots \cdot \bar{F}r'_1$.*

► **Corollary 4.4.** *Let $F: \text{Set} \rightarrow \text{Set}$ be a functor. For every relation $r: X \rightarrow Y$,*

$$(\bar{F})^{\bullet}r = \bigvee_{\substack{r_1, \dots, r_n: \\ r_n \cdot \dots \cdot r_1 = r}} \bar{F}r_n \cdot \dots \cdot \bar{F}r_1.$$

Therefore, as normality of a lax extension also implies normality of any lax extension below it, we have

► **Corollary 4.5.** *A functor $F: \text{Set} \rightarrow \text{Set}$ admits a normal lax extension iff the laxification of its Barr relax extension is normal. More concretely, a functor $F: \text{Set} \rightarrow \text{Set}$ admits a normal lax extension iff for every set X and every composable sequence of relations r_1, \dots, r_n , whenever $r_n \cdot \dots \cdot r_1 = 1_X$, then $\bar{F}r_n \cdot \dots \cdot \bar{F}r_1 \leq 1_{FX}$.*

► **Remark 4.6.** It is well-known [6] that for every functor $F: \text{Set} \rightarrow \text{Set}$ and all relations $r: X \rightarrow Y$ and $s: Y \rightarrow Z$, $\bar{F}(s \cdot r) \leq \bar{F}s \cdot \bar{F}r$. Hence, once we show the inequality of Corollary 4.5 we actually have equality.

In general terms, our main results follow by showing that in Corollary 4.5, under certain conditions on **Set**-functors, it suffices to consider composable sequences of relations that satisfy nice properties. In this regard, it is convenient to introduce the following notion.

► **Definition 4.7.** *Let r_1, \dots, r_n be a composable sequence of relations. A composable sequence s_1, \dots, s_k is said to be a **Barr upper bound** of the sequence r_1, \dots, r_n if $r_n \cdot \dots \cdot r_1 = s_k \cdot \dots \cdot s_1$ and $\bar{F}r_n \cdot \dots \cdot \bar{F}r_1 \leq \bar{F}s_k \cdot \dots \cdot \bar{F}s_1$.*

40:12 Identity-Preserving Lax Extensions and Where to Find Them

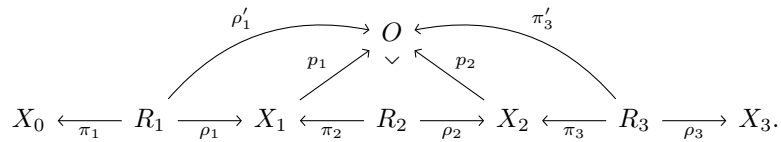
In Section 3 we have seen that every Set -functor that admits a normal lax extension preserves 1/4-iso pullbacks, or equivalently, it is monotone on difunctional relations (Theorem 3.12). As we show next, the latter condition is also equivalent to satisfying the criterion of Corollary 4.5 for pairs of composable relations.

► **Proposition 4.8.** *Let $F: \text{Set} \rightarrow \text{Set}$ be a functor. The following clauses are equivalent:*

- (i) *The functor $F: \text{Set} \rightarrow \text{Set}$ preserves 1/4-iso pullbacks.*
- (ii) *For all relations $r_1: X \rightarrow Y, r_2: Y \rightarrow X$ such that $r_2 \cdot r_1 \leq 1_X, \bar{F}r_2 \cdot \bar{F}r_1 \leq 1_{FX}$.*
- (iii) *For all relations $r_1: X \rightarrow Y, r_2: Y \rightarrow X$ such that $r_2 \cdot r_1 = 1_X, \bar{F}r_2 \cdot \bar{F}r_1 \leq 1_{FX}$.*

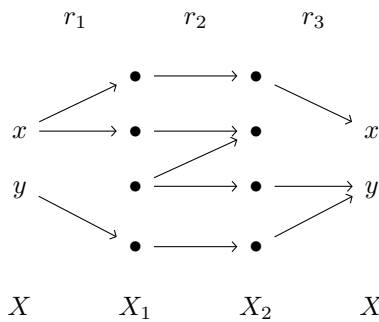
Now, suppose that we want to extend the previous result in inductive style to composable triples of relations. Due to the next lemma, a simple idea to reduce the case of composable triples to the case of composable pairs of relations is to take the difunctional closure of the second relation in the sequence.

► **Lemma 4.9.** *Let $r_1: X_0 \rightarrow X_1, r_2: X_1 \rightarrow X_2$ and $r_3: X_2 \rightarrow X_3$ be relations given by spans that form the base of the commutative diagram*

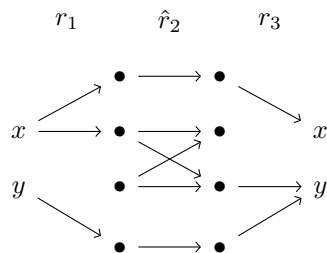


Then, with $r'_1: X \rightarrow O$ and $r'_3: O \rightarrow X_3$ defined by the spans $X_0 \xleftarrow{\pi_1} R_1 \xrightarrow{\rho_1} O$ and $O \xleftarrow{\pi_3} R_3 \xrightarrow{\rho_3} X_3$, respectively, $\bar{F}r_3 \cdot \bar{F}r_2 \cdot \bar{F}r_1 \leq \bar{F}r_3 \cdot \bar{F}\hat{r}_2 \cdot \bar{F}r_1 \leq \bar{F}r'_3 \cdot \bar{F}r'_1$.

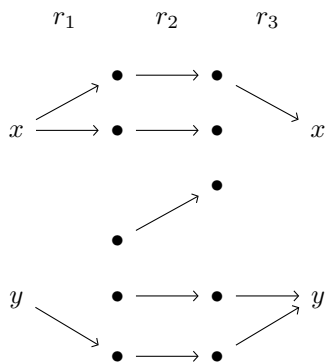
Indeed, let $r_1: X \rightarrow X_1, r_2: X_1 \rightarrow X_2$ and $r_3: X_2 \rightarrow X$ be relations such that $r_3 \cdot r_2 \cdot r_1 = 1_X$. Then, by Proposition 4.8 and Lemma 4.9, we conclude that $\bar{F}r_3 \cdot \bar{F}r_2 \cdot \bar{F}r_1 \leq 1_{FX}$ once we show that $r'_3 \cdot r'_1 = 1_X$. Of course, in general, this does not hold. Consider the following example where the arrows depict pairs of related elements.



By taking the difunctional closure \hat{r}_2 of r_2 we get



So, $r_3 \cdot \hat{r}_2 \cdot r_1 = r'_3 \cdot r'_1$ is not a subidentity. Now the property of preserving 1/4-iso pullbacks is helpful again. As we will see in Lemma 4.10, under this condition, the sequence below is a Barr upper bound of the first one and it is obtained from it by “splitting” where necessary the elements of X_1 that do not belong to the codomain of r_1 and the elements of X_2 that do not belong to the domain of r_3 .



In this situation we can apply the difunctional closure to r_2 (which in this particular example is already difunctional) to reduce the number of relations as discussed in Lemma 4.9.

► **Lemma 4.10.** *Let $F: \mathbf{Set} \rightarrow \mathbf{Set}$ be a functor that preserves 1/4-iso pullbacks, and let $r_1: X \rightarrow Y$, $r_2: Y \rightarrow Z$ and $r_3: Z \rightarrow W$ be relations. Then, there are relations $s_1: X \rightarrow Y'$, $s_2: Y' \rightarrow Z'$ and $s_3: Z' \rightarrow W$ such that s_1, s_2, s_3 is a Barr upper bound of r_1, r_2, r_3 and*

1. *for all $y, y' \in Y'$ and all $z \in Z'$, if $y \neq y'$, $y s_2 z$ and $y' s_2 z$, then $z \in \text{dom}(s_3)$;*
2. *for all $y \in Y'$ and $z, z' \in Z'$, if $z \neq z'$, $y s_2 z$ and $y s_2 z'$, then $y \in \text{cod}(s_1)$.*

The previous lemma essentially closes the argument that we have been crafting so far.

► **Theorem 4.11.** *Let $F: \mathbf{Set} \rightarrow \mathbf{Set}$ be a functor. The following clauses are equivalent:*

- (i) *The functor $F: \mathbf{Set} \rightarrow \mathbf{Set}$ preserves 1/4-iso pullbacks.*
- (ii) *For all relations $r_1: X \rightarrow Y$, $r_2: Y \rightarrow Z$ and $r_3: Z \rightarrow X$ such that $r_3 \cdot r_2 \cdot r_1 \leq 1_X$, $\bar{F}r_3 \cdot \bar{F}r_2 \cdot \bar{F}r_1 \leq 1_{FX}$.*
- (iii) *For all relations $r_1: X \rightarrow Y$, $r_2: Y \rightarrow Z$ and $r_3: Z \rightarrow X$ such that $r_3 \cdot r_2 \cdot r_1 = 1_X$, $\bar{F}r_3 \cdot \bar{F}r_2 \cdot \bar{F}r_1 \leq 1_{FX}$.*

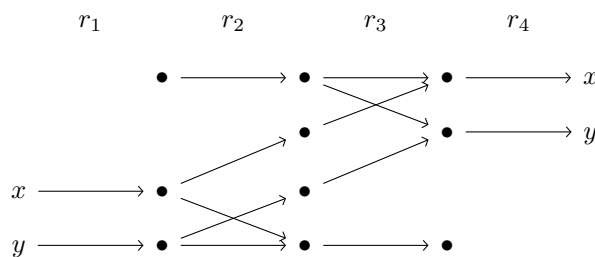
However, as we see next, Theorem 4.11 is as far as we can go under the assumption of 1/4-iso pullbacks preservation. In other words, the fact that a **Set**-functor preserves 1/4-iso pullbacks is *not* sufficient to conclude that it admits a normal lax extension.

► **Example 4.12.** Let us define a functor $F: \mathbf{Set} \rightarrow \mathbf{Set}$ as a quotient of $\coprod_{n \in \{f,g\}} \{n\} \times X^5 \cong X^5 + X^5$ under the equivalence defined by the clauses:

$$\begin{array}{ll}
 f(y, x, z, x, t) \sim f(y', x, z', x, t') & f(t, x, x, y, y) \sim f(t', x, x, y, y) \\
 g(y, x, z, x, t) \sim g(y', x, z', x, t') & g(x, x, y, y, t) \sim g(x, x, y, y, t') \\
 f(y, x, z, x, t) \sim g(y', x, z', x, t') & f(t, x, z, y, z) \sim g(t, x, t, y, z)
 \end{array}$$

where $f(x_1, \dots, x_5)$ and $g(x_1, \dots, x_5)$ denote the corresponding elements (f, x_1, \dots, x_5) , $(g, x_1, \dots, x_5) \in \coprod_{n \in \{f,g\}} \{n\} \times X^5$. Let $2 = \{x, y\}$ and consider the composable sequence of relations depicted below.

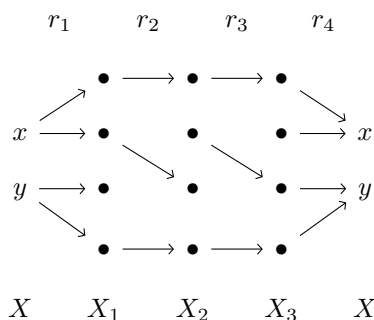
40:14 Identity-Preserving Lax Extensions and Where to Find Them



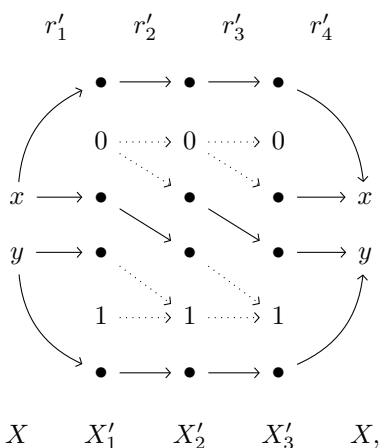
Then, F preserves 1/4-iso pullbacks and $r_4 \cdot r_3 \cdot r_2 \cdot r_1 = 1_2$, however, $\bar{F}r_4 \cdot \bar{F}r_3 \cdot \bar{F}r_2 \cdot \bar{F}r_1 \not\leq 1_{F2}$.

4.1 The case of functors that weakly preserve 4/4-epi pullbacks

From Theorem 4.11 it basically follows that a functor that weakly preserves 1/4-iso pullbacks and 4/4-epi pullbacks admits a normal lax extension. But to see this, first we need to sharpen Corollary 4.5. The goal is to show that it suffices to consider composable sequences of relations where all relations other than the first and the last are total and surjective. To illustrate how we achieve this, let us consider the sequence of relations depicted below.

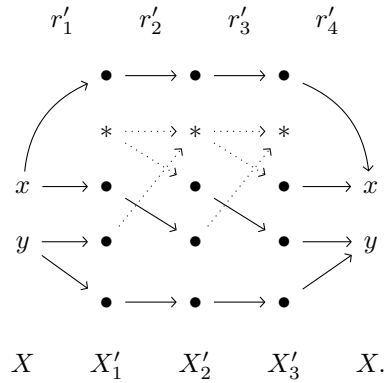


Then, by adding new elements 0 and 1 to X_1, X_2 and X_3 we can extend this sequence to the sequence



where the dotted arrows indicate pairs of elements that were added to the corresponding relation as follows: for $i = 2, 3$, r'_i relates $0 \in X'_{i-1}$ to every element of $X_i \cup \{0\}$ that does not belong to the codomain of r_i and relates every element of $X_{i-1} \cup \{1\}$ that does not belong to the domain of r_i to $1 \in X'_i$. In this way, we guarantee that r'_2 and r'_3 are total and surjective and that $r'_4 \cdot r'_3 \cdot r'_2 \cdot r'_1 = r_4 \cdot r_3 \cdot r_2 \cdot r_1 = 1_X$. We could have extended r_2 and r_3

to total and surjective relations by adding just a single element $*$ to X_1 , X_2 and X_3 that would simultaneously take the role of 0 and 1. However, composing the resulting sequence of relations would not yield the identity relation:



In other words, by splitting $*$ in two elements 0 and 1, the former to make the relations r_2 and r_3 surjective and the latter to make them total, we obtain a subidentity because we never create paths between elements of X_1 that are not part of the domain of r_2 and elements of X_3 that are not part of the codomain of r_3 . In the next lemma we formalize this procedure for arbitrary composable sequences of relations and show that it yields Barr upper bounds.

► **Lemma 4.13.** *A functor $F: \text{Set} \rightarrow \text{Set}$ that preserves 1/4-iso pullbacks admits a normal lax extension iff for every composable sequence of relations r_1, \dots, r_n such that $n \geq 4$ and r_2, \dots, r_{n-1} are total and surjective, whenever $r_n \cdot \dots \cdot r_1 = 1_X$, for some set X , then $\bar{F}r_n \cdot \dots \cdot \bar{F}r_1 \leq 1_{FX}$.*

► **Remark 4.14.** In a composable sequence of relations that satisfies the conditions of Lemma 4.13 the first relation is necessarily total while the last one is necessarily surjective.

Now, our first main result follows straightforwardly. Since the composite of total and surjective relations is total and surjective, due to the following fact, every composable sequence of relations where all relations other than the first and the last are total and surjective admits a Barr upper bound consisting of three relations.

► **Proposition 4.15.** *A functor $F: \text{Set} \rightarrow \text{Set}$ weakly preserves 4/4-epi pullbacks iff for all relations $r: X \rightarrow Y$ and $s: Y \rightarrow Z$, whenever r is surjective and s is total, $\bar{F}s \cdot \bar{F}r = \bar{F}(s \cdot r)$.*

► **Theorem 4.16.** *A Set-functor that weakly preserves 1/4-iso pullbacks and 4/4-epi weak pullbacks admits a normal lax extension.*

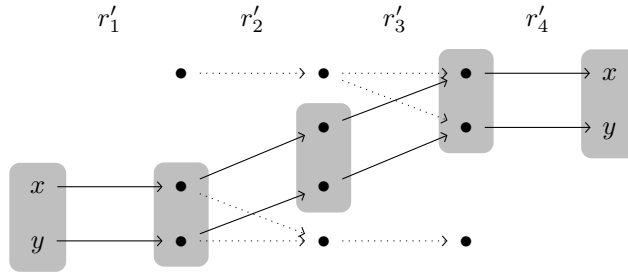
► **Remark 4.17.** Preservation of 4/4-epi pullbacks plays a role in the analysis of interpolation in coalgebraic logic [42]. In particular, this analysis implies that given a separating set Λ of monotone predicate liftings for a finite-set-preserving functor F , which induces an expressive modal logic $\mathcal{L}(\Lambda)$ for F -coalgebras, the logic $\mathcal{L}(\Lambda)$ has interpolation iff F weakly preserves 4/4-epi pullbacks [42, Theorem 37]. In connection with the fact that a functor has a normal lax extension iff it has a separating set of monotone predicate liftings [31], we obtain the following application of Theorem 4.16 and Corollary 3.13: A finite-set preserving functor F has a separating set of monotone predicate liftings such that the associated modal logic has uniform interpolation iff F weakly preserves 1/4-iso pullbacks and 4/4-epi pullbacks.

4.2 The case of functors that preserve 1/4-mono pullbacks

To obtain Theorem 4.16, we refined Corollary 4.5 to composable sequences of relations where all relations *other than* the first and the last are total and surjective. And to achieve this in Lemma 4.13, given a composable sequence of relations, we *added* pairs of related elements to the relations in the sequence. In the sequel, we will show that every functor that preserves 1/4-mono pullbacks admits a normal lax extension. We will see that for these functors it is even possible to refine Corollary 4.5 to composable sequences of relations where *all* relations are total and surjective. However, we will achieve this in Lemma 4.19 below by, given a composable sequence of relations, *removing* pairs of related elements from the relations in the sequence. Our proof strategy is justified by the next fact.

► **Proposition 4.18.** *A functor $F: \text{Set} \rightarrow \text{Set}$ preserves 1/4-mono pullbacks iff for all relations $r: X \rightarrow Y$ and $s: Y \rightarrow Z$, whenever r is the converse of a partial function or s is a partial function, $\overline{F}s \cdot \overline{F}r = \overline{F}(s \cdot r)$.*

This result enables a “look ahead and behind” strategy for Corollary 4.5. The idea is that, given a composable sequence of relations r_1, \dots, r_n such that $r_n \cdot \dots \cdot r_1 = 1_X$, then, with $r_i: X_{i-1} \rightarrow X_i$ being a relation in the sequence, removing the elements of X_i that do not belong to the codomain of $r_i \cdot \dots \cdot r_1$ or do not belong to the domain of $r_n \cdot \dots \cdot r_{i+1}$ yields a Barr upper bound of our original sequence. For instance, consider the composable sequence of relations depicted in Example 4.12, which we used to show that there are functors that preserve 1/4-iso pullbacks but do not admit a normal lax extension. In the next lemma, in particular, we show that for functors that preserve 1/4-mono pullbacks the sequence below of total and surjective relations is a Barr upper bound of this one. The dotted arrows represent pairs of related elements that were removed, and the grey boxes represent the elements of each set that are *not* removed.



► **Lemma 4.19.** *A functor $F: \text{Set} \rightarrow \text{Set}$ that preserves 1/4-mono pullbacks admits a normal lax extension if for every composable sequence of total and surjective relations r_1, \dots, r_n , whenever $r_n \cdot \dots \cdot r_1 = 1_X$ for some set X , then $\overline{F}r_n \cdot \dots \cdot \overline{F}r_1 \leq 1_{FX}$.*

It turns out that the sufficient condition of the previous lemma is actually satisfied by every Set-functor that preserves 1/4-iso pullbacks. Indeed, due to the next result, Lemma 4.9 and the fact that surjections are stable under pushouts, every composable sequence of total and surjective relations whose composite is an identity admits a Barr upper bound consisting of three relations.

► **Lemma 4.20.** *Let $r_1: X \rightarrow X_1$, $r_2: X_1 \rightarrow X_2$ and $r_3: X_2 \rightarrow X$ be a composable sequence of total and surjective relations, and let $\hat{r}_2: X_1 \rightarrow X_2$ be the difunctional closure of r_2 . If $r_3 \cdot r_2 \cdot r_1 = 1_X$, then $r_3 \cdot \hat{r}_2 \cdot r_1 = 1_X$.*

► **Proposition 4.21.** *Let $F: \text{Set} \rightarrow \text{Set}$ be a functor that preserves 1/4-iso pullbacks, and let r_1, \dots, r_n be a composable sequence of total and surjective relations. If $r_n \cdot \dots \cdot r_1 = 1_X$ for some set X , then $\overline{F}r_n \cdot \dots \cdot \overline{F}r_1 \leq 1_{FX}$.*

Therefore,

► **Theorem 4.22.** *Every Set-functor that preserves 1/4-mono pullbacks admits a normal lax extension.*

In particular, since in Example 3.10(3) we have seen that for (commutative) monoid-valued functors preserving 1/4-mono pullbacks is equivalent to preserving 1/4-iso pullbacks, as a consequence of Theorem 4.22 and Corollary 3.13 we obtain:

► **Corollary 4.23.** *A (commutative) monoid-valued functor admits a normal lax extension iff the monoid is positive.*

► **Remark 4.24.** The above result may be equivalently stated as saying that a monoid-valued functor has a separating set of monotone predicate liftings iff the monoid is positive. In this formulation, it improves on a previous result effectively stating the same equivalence under the additional assumption that the monoid is refinable [42, Proposition 22]. For every monoid M , one has a preorder on M given by $m \geq n$ iff $\exists k \in M. m = n + k$, which is a partial order whenever the monoid is cancellative and positive. It is then clear that one has a separating set of monotone predicate liftings \diamond_m , for $m \in M$, defined by $\diamond_m(A) = \{\mu \in M^{(X)} \mid \mu(A) \geq m\}$ where we write $\mu(A) = \sum_{x \in A} \mu(x)$. The arising normal lax extension is given for $r: X \rightarrow Y$, $\mu \in M^{(X)}$, $\nu \in M^{(Y)}$ by $\mu \llcorner r \nu$ iff $\nu(r[A]) \geq \mu(A)$ for all $A \subseteq X$ and symmetrically, much like for probabilistic transition systems (Example 2.3(2)). For non-cancellative positive monoids, the description of the normal lax extension and the separating set of monotone predicate liftings whose existence are guaranteed by Corollary 4.23 is in general more involved. In particular, the predicate liftings \diamond_m described above may fail to be separating, as witnessed, for instance, by the commutative additive monoid $\{0, 1, 2\}$ with $1 + 2 = 1$. Specifically, $\mu, \nu \in M^{\{\star\}}$ given by $\mu(\star) = 1$ and $\nu(\star) = 2$ cannot be distinguished.

The class of Set-functors that admit a normal lax extension is closed under subfunctors and several natural constructions such as the sum of functors. This makes it easy to extend the reach of our sufficient conditions, but it also shows that it is easy to provide examples of functors that admit a normal lax extension and do not weakly preserve 1/4-mono pullbacks nor 4/4-epi pullbacks. A quick example is the functor given by the sum of the functor $(-)_2^3$ and the monotone neighbourhood functor. To conclude this section, we present a less obvious example that is constructed analogously to Example 4.12. Notice that, as we have seen in Example 3.14(4), the class of functors that admit a normal lax extension is not closed under quotients.

► **Example 4.25.** For any set X , let $\mathbb{F}X$ be the quotient of X^3 under the equivalence relation \sim defined by the clauses $(x, x, y) \sim (x, x, x) \sim (y, x, x)$. This yields a functor $\mathbb{F}: \text{Set} \rightarrow \text{Set}$ that neither weakly preserves 1/4-mono pullbacks nor 4/4-epi pullbacks, however, \mathbb{F} admits a normal lax extension.

5 Conclusions

Normal lax extensions of functors play a dual role in the coalgebraic modelling of reactive systems, on the one hand allowing for good notions of bisimulations on functor coalgebras and on the other hand guaranteeing the existence of expressive temporal logics. We have shown on the one hand that every functor admitting a lax extension preserves 1/4-iso pullbacks, and on the other hand that a functor admits a normal lax extension if it weakly preserves either 1/4-iso pullbacks and 4/4-epi pullbacks or inverse images. These results improve on

previous results [28, 30, 31], which combine to imply that weak-pullback-preserving functors admit normal lax extensions. One application of our results implies, roughly, that a given type of monoid-weighted transition systems admits a good notion of bisimulation iff the monoid is positive.

The most obvious issue for future work is to close the remaining gap, i.e. to give a necessary and sufficient criterion for the existence of normal lax extensions in terms of limit preservation. Additionally, the structure of the lattice of normal lax extensions of a functor merits attention, in the sense that larger lax extensions induce more permissive notions of bisimulation.

References

- 1 Peter Aczel and Nax Paul Mendler. A final coalgebra theorem. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné, editors, *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989. doi:10.1007/BFB0018361.
- 2 Jiří Adámek, Horst Herrlich, and George E. Strecker. *Abstract and concrete categories: The joy of cats*. John Wiley & Sons Inc., 1990. Republished in: Reprints in Theory and Applications of Categories, No. 17 (2006) pp. 1–507. URL: <http://tac.mta.ca/tac/reprints/articles/17/tr17abs.html>.
- 3 Roland Carl Backhouse, Peter J. de Bruin, Paul F. Hoogendijk, Grant Malcolm, Ed Voermans, and Jaap van der Woude. Polynomial relators (extended abstract). In Maurice Nivat, Charles Rattray, Teodor Rus, and Giuseppe Scollo, editors, *Algebraic Methodology and Software Technology, AMAST 1991*, Workshops in Computing, pages 303–326. Springer, 1991.
- 4 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Coalgebraic behavioral metrics. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:20)2018.
- 5 Michael Barr. Relational algebras. In *Reports of the Midwest Category Seminar IV*, number 137 in *Lect. Notes Math.*, pages 39–55. Springer, 1970. doi:10.1007/BFb0060439.
- 6 Michael Barr. Relational algebras. In *Reports of the Midwest Category Seminar IV*, pages 39–55. Springer, 1970. doi:10.1007/bfb0060439.
- 7 Michael Barr. Terminal coalgebras in well-founded set theory. *Theor. Comput. Sci.*, 114(2):299–315, 1993. doi:10.1016/0304-3975(93)90076-6.
- 8 Richard S. Bird and Oege de Moor. *Algebra of programming*. Prentice Hall International series in computer science. Prentice Hall, 1997.
- 9 Filippo Bonchi, Marcello M. Bonsangue, Michele Boreale, Jan J. M. M. Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Inf. Comput.*, 211:77–105, 2012. doi:10.1016/J.IC.2011.12.002.
- 10 Brian F. Chellas. *Modal Logic – An Introduction*. Cambridge University Press, 1980. doi:10.1017/CB09780511621192.
- 11 Corina Cirstea, Clemens Kupke, and Dirk Pattinson. EXPTIME tableaux for the coalgebraic mu-calculus. *Log. Methods Comput. Sci.*, 7(3), 2011. doi:10.2168/LMCS-7(3:3)2011.
- 12 Maria Manuel Clementino, Dirk Hofmann, and George Janelidze. The monads of classical algebra are seldom weakly cartesian. *J. Homotopy and Related Structures*, 9(1):175–197, 2014.
- 13 Francesco Gavazzo. Quantitative behavioural reasoning for higher-order effectful programs: Applicative distances. In Anuj Dawar and Erich Grädel, editors, *Logic in Computer Science, LICS 2018*, pages 452–461. ACM, 2018. doi:10.1145/3209108.3209149.
- 14 Sergey Goncharov, Dirk Hofmann, Pedro Nora, Lutz Schröder, and Paul Wild. A point-free perspective on lax extensions and predicate liftings. *Mathematical Structures in Computer Science*, pages 1–30, 2023. doi:10.1017/S096012952300035X.
- 15 Daniel Gorín and Lutz Schröder. Simulations and bisimulations for coalgebraic modal logics. In Reiko Heckel and Stefan Milius, editors, *Algebra and Coalgebra in Computer Science, CALCO 2013*, volume 8089 of *LNCS*, pages 253–266. Springer, 2013. doi:10.1007/978-3-642-40206-7_19.

- 16 H. Peter Gumm. Free-algebra functors from a coalgebraic perspective. In Daniela Petrisan and Jurriaan Rot, editors, *Coalgebraic Methods in Computer Science, CMCS 2020*, volume 12094 of *LNCS*, pages 55–67. Springer, 2020. doi:10.1007/978-3-030-57201-3_4.
- 17 H. Peter Gumm and Tobias Schröder. Coalgebraic structure from weak limit preserving functors. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science, CMCS 2000, Berlin, Germany, March 25-26, 2000*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 111–131. Elsevier, 2000. doi:10.1016/S1571-0661(05)80346-9.
- 18 H. Peter Gumm and Tobias Schröder. Monoid-labeled transition systems. In Andrea Corradini, Marina Lenisa, and Ugo Montanari, editors, *Coalgebraic Methods in Computer Science, CMCS 2001*, volume 44(1) of *ENTCS*, pages 185–204. Elsevier, 2001. doi:10.1016/S1571-0661(04)80908-3.
- 19 H. Peter Gumm and Tobias Schröder. Types and coalgebraic structure. *Algebra universalis*, 53(2–3):229–252, August 2005. doi:10.1007/s00012-005-1888-2.
- 20 H. Peter Gumm and Mehdi Zarrad. Coalgebraic simulations and congruences. In Marcello M. Bonsangue, editor, *Coalgebraic Methods in Computer Science - 12th IFIP WG 1.3 International Workshop, CMCS 2014, Colocated with ETAPS 2014, Grenoble, France, April 5-6, 2014, Revised Selected Papers*, volume 8446 of *Lecture Notes in Computer Science*, pages 118–134. Springer, 2014. doi:10.1007/978-3-662-44124-4_7.
- 21 Helle Hvid Hansen and Clemens Kupke. A coalgebraic perspective on monotone modal logic. In Jirí Adámek and Stefan Milius, editors, *Coalgebraic Methods in Computer Science, CMCS 2004*, volume 106 of *ENTCS*, pages 121–143. Elsevier, 2004. doi:10.1016/j.entcs.2004.02.028.
- 22 Wim H. Hesselink and Albert Thijs. Fixpoint semantics and simulation. *Theor. Comput. Sci.*, 238(1-2):275–311, 2000. doi:10.1016/S0304-3975(98)00176-5.
- 23 Dirk Hofmann, Gavin J. Seal, and Walter Tholen, editors. *Monoidal Topology. A Categorical Approach to Order, Metric, and Topology*, volume 153 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, July 2014. Authors: Maria Manuel Clementino, Eva Colebunders, Dirk Hofmann, Robert Lowen, Rory Lucyshyn-Wright, Gavin J. Seal and Walter Tholen. doi:10.1017/cbo9781107517288.
- 24 Jesse Hughes and Bart Jacobs. Simulations in coalgebra. *Theor. Comput. Sci.*, 327(1-2):71–108, 2004. doi:10.1016/J.TCS.2004.07.022.
- 25 Thomas Ihringer. *Allgemeine Algebra. Mit einem Anhang über Universelle Coalgebra von H. P. Gumm*, volume 10 of *Berliner Studienreihe zur Mathematik*. Heldermann Verlag, 2003.
- 26 Bartek Klin. Structural operational semantics for weighted transition systems. In Jens Palsberg, editor, *Semantics and Algebraic Specification, Essays Dedicated to Peter D. Mosses on the Occasion of His 60th Birthday*, volume 5700 of *LNCS*, pages 121–139. Springer, 2009. doi:10.1007/978-3-642-04164-8_7.
- 27 Clemens Kupke, Alexander Kurz, and Yde Venema. Completeness for the coalgebraic cover modality. *Log. Methods Comput. Sci.*, 8(3), 2012. doi:10.2168/LMCS-8(3:2)2012.
- 28 Alexander Kurz and Raul Andres Leal. Modalities in the stone age: A comparison of coalgebraic logics. *Theor. Comput. Sci.*, 430:88–116, 2012. doi:10.1016/J.TCS.2012.03.027.
- 29 Paul Blain Levy. Similarity quotients as final coalgebras. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures, FOSSACS 2011*, volume 6604 of *LNCS*, pages 27–41. Springer, 2011. doi:10.1007/978-3-642-19805-2_3.
- 30 Johannes Marti and Yde Venema. Lax extensions of coalgebra functors. In Dirk Pattinson and Lutz Schröder, editors, *Coalgebraic Methods in Computer Science, CMCS 2021*, volume 7399 of *LNCS*, pages 150–169. Springer, 2012. doi:10.1007/978-3-642-32784-1_9.
- 31 Johannes Marti and Yde Venema. Lax extensions of coalgebra functors and their logic. *Journal of Computer and System Sciences*, 81(5):880–900, 2015. doi:10.1016/j.jcss.2014.12.006.
- 32 Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.
- 33 Rohit Parikh. Propositional game logic. In *Foundations of Computer Science, FOCS 1983*, pages 195–200. IEEE Computer Society, 1983. doi:10.1109/SFCS.1983.47.

- 34 Dirk Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame J. Formal Log.*, 45(1):19–33, 2004. doi:10.1305/ndjfl/1094155277.
- 35 David Peleg. Concurrent dynamic logic (extended abstract). In Robert Sedgewick, editor, *Symposium on Theory of Computing, STOC 1985*, pages 232–239. ACM, 1985. doi:10.1145/22145.22172.
- 36 Jacques Riguet. Relations binaires, fermetures, correspondances de Galois. *Bulletin de la Société Mathématique de France*, 76:114–155, 1948. doi:10.24033/bsmf.1401.
- 37 Jan J. M. M. Rutten. Relators and metric bisimulations. In Bart Jacobs, Larry Moss, Horst Reichel, and Jan J. M. M. Rutten, editors, *Coalgebraic Methods in Computer Science, CMCS 1998*, volume 11 of *ENTCS*, pages 252–258. Elsevier, 1998. doi:10.1016/S1571-0661(04)00063-5.
- 38 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.
- 39 Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390(2-3):230–247, January 2008. doi:10.1016/j.tcs.2007.09.023.
- 40 Christoph Schubert and Gavin J. Seal. Extensions in the theory of lax algebras. *Theory and Applications of Categories*, 21(7):118–151, 2008.
- 41 Gavin J. Seal. Canonical and op-canonical lax algebras. *Theory and Applications of Categories*, 14(10):221–243, 2005. URL: <http://www.tac.mta.ca/tac/volumes/14/10/14-10abs.html>.
- 42 Fatemeh Seifan, Lutz Schröder, and Dirk Pattinson. Uniform interpolation in coalgebraic modal logic. In Filippo Bonchi and Barbara König, editors, *Algebra and Coalgebra in Computer Science, CALCO 2017*, volume 72 of *LIPICs*, pages 21:1–21:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.CALCO.2017.21.
- 43 Albert Thijs. *Simulation and fixpoint semantics*. PhD thesis, University of Groningen, 1996.
- 44 Věra Trnková. General theory of relational automata. *Fund. Inform.*, 3(2):189–233, 1980. doi:10.3233/FI-1980-3208.
- 45 Paul Wild and Lutz Schröder. Characteristic logics for behavioural metrics via fuzzy lax extensions. In Igor Konnov and Laura Kovács, editors, *Concurrency Theory, CONCUR 2020*, volume 171 of *LIPICs*, pages 27:1–27:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CONCUR.2020.27.
- 46 Paul Wild and Lutz Schröder. Characteristic logics for behavioural hemimetrics via fuzzy lax extensions. *Log. Methods Comput. Sci.*, 18(2), 2022. doi:10.46298/LMCS-18(2:19)2022.
- 47 James Worrell. Coinduction for recursive data types: partial orders, metric spaces and Ω -categories. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science, CMCS 2000*, volume 33 of *ENTCS*, pages 337–356. Elsevier, 2000. doi:10.1016/S1571-0661(05)80356-1.

Residue Domination in Bounded-Treewidth Graphs

Jakob Greilhuber  

TU Wien, Austria

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Philipp Schepper 

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Philip Wellnitz  

National Institute of Informatics, Tokyo, Japan

The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan

Abstract

For the vertex selection problem (σ, ρ) -DOMSET one is given two fixed sets σ and ρ of integers and the task is to decide whether we can select vertices of the input graph such that, for every selected vertex, the number of selected neighbors is in σ and, for every unselected vertex, the number of selected neighbors is in ρ [Telle, Nord. J. Comp. 1994]. This framework covers many fundamental graph problems such as INDEPENDENT SET and DOMINATING SET.

We significantly extend the recent result by Focke et al. [SODA 2023] to investigate the case when σ and ρ are two (potentially different) residue classes modulo $m \geq 2$. We study the problem parameterized by treewidth and present an algorithm that solves in time $m^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ the decision, minimization and maximization version of the problem. This significantly improves upon the known algorithms where for the case $m \geq 3$ not even an explicit running time is known. We complement our algorithm by providing matching lower bounds which state that there is no $(m - \varepsilon)^{\text{pw}} \cdot n^{\mathcal{O}(1)}$ -time algorithm parameterized by pathwidth pw , unless SETH fails. For $m = 2$, we extend these bounds to the minimization version as the decision version is efficiently solvable.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Treewidth, Generalized Dominating Set, Strong Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.41

Related Version The Master's Thesis of Jakob Greilhuber [36] is based on the lower bound results of this work.

Full Version: <https://arxiv.org/abs/2403.07524>

Funding *Jakob Greilhuber:* Part of Saarbrücken Graduate School of Computer Science, Germany.

Philipp Schepper: Part of Saarbrücken Graduate School of Computer Science, Germany.

Acknowledgements The work of Jakob Greilhuber has been carried out mostly during a summer internship at the Max Planck Institute for Informatics, Saarbrücken, Germany. The work of Philip Wellnitz was partially carried out at the Max Planck Institute for Informatics.

1 Introduction

Classical graph problems such as Dominating Set or Independent Set are ubiquitous in computer science. These problems are not only of theoretical interest but also have many practical applications; including facility location, coding theory, modeling communication networks, map labeling, or even similarity measures on molecules [3, 4, 16, 31, 38, 39]. Therefore, these problems are extensively studied on plenty of graph classes and several generalizations and variations have been formulated and considered [7, 12, 22, 30, 32, 41, 50,



© Jakob Greilhuber, Philipp Schepper, and Philip Wellnitz;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 41; pp. 41:1–41:20



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



52, 54, 59]. Moreover, the problems seem to come with a significant complexity but also sufficient structural properties to serve as a testing point for new techniques which frequently result in faster algorithms for the problems [20, 28, 61].

In 1993, Telle and Proskurowski introduced the general class of (σ, ρ) -DOMSET problems which capture several well-known vertex selection problems for appropriately chosen sets $\sigma, \rho \subseteq \mathbb{Z}_{\geq 0}$ [57, 58]. In this problem the input is an undirected graph and the task is to decide if we can select vertices such that (1), for every selected vertex, the number of selected neighbors is contained in the set σ and (2), for every unselected vertex, the number of selected neighbors is contained in the set ρ . Formally, for a graph G , decide if there exists a vertex set $S \subseteq V(G)$ such that, for all $v \in S$, we have $|N(v) \cap S| \in \sigma$, and, for all $v \notin S$, we have $|N(v) \cap S| \in \rho$. Such a set S is called a (σ, ρ) -set.

It is easy to see that (σ, ρ) -DOMSET captures classical Dominating Set when we set $\sigma = \mathbb{Z}_{\geq 0}$ and $\rho = \mathbb{Z}_{\geq 0} \setminus \{0\}$ and ask for a selection of bounded size. Moreover, with different requirements imposed on the size of the selection, we can also reformulate other problems such as Independent Set ($\sigma = \{0\}$ and $\rho = \mathbb{Z}_{\geq 0}$), Perfect Code ($\sigma = \{0\}$ and $\rho = \{1\}$), Induced q -regular Subgraph ($\sigma = \{q\}$ and $\rho = \mathbb{Z}_{\geq 0}$), Odd Domination ($\sigma = \{0, 2, \dots\}$ and $\rho = \{1, 3, \dots\}$), and many more. We refer to [8, 57] for a longer list of problems that can be described as (σ, ρ) -DOMSET.

Since (σ, ρ) -DOMSET generalizes many fundamental graph problems, the ultimate goal is to settle the complexity of (σ, ρ) -DOMSET for *all* (decidable) sets σ and ρ . We know that for many choices of σ and ρ the problem is NP-hard. Hence, we frequently either restrict the input to special graph classes or parameterize by some (structural) measure of the input (for example, the solution size).

One of the best explored structural parameters is treewidth [6, 18, 20, 26, 42, 45, 46, 51, 61] which measures how similar a graph is to a tree (see [19, Chapter 7] for a more thorough introduction). Many problems admit efficient algorithms on trees with a simple dynamic program. With treewidth as parameter, we can lift these programs to more general graphs and obtain fast algorithms especially compared to the running time obtained from Courcelle's Theorem [17]. For most of these problems the goal is to find the smallest constant c such that the respective problem can be solved in time $c^{\text{tw}} \cdot n^{\mathcal{O}(1)}$.

For problems parameterized by treewidth a perpetual improvement of the algorithm seems unlikely. After a few iterations, frequently conceptually new ideas seem to be necessary to obtain further improvements. Lokshantov, Marx, and Saurabh initiated a line of research that proves that such limitations are often not a shortcoming of the techniques at hand, but rather an inherent property of the problem itself [45]. For example, they prove that the known algorithm for Dominating Set [61] which takes time $3^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ cannot be improved further unless the Strong Exponential-Time Hypothesis (SETH) [9, 40] fails.

Hence, the ultimate goal for (σ, ρ) -DOMSET is to show the following result (or to prove that no such constant exists in the respective setting):

For all sets σ and ρ , determine the constant $c_{\sigma, \rho}$ such that (σ, ρ) -DOMSET
can be solved in time $c_{\sigma, \rho}^{\text{tw}} \cdot n^{\mathcal{O}(1)}$
but not in time $(c_{\sigma, \rho} - \varepsilon)^{\text{tw}} \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$, unless SETH fails.

For certain choices of the sets σ and ρ , some (partial) results of this form are already known [49]. A broad class of algorithms was given by van Rooij, Bodlaender, and Rossmanith [61] for the case of finite and cofinite sets. These algorithms were later improved by van Rooij [60]. Focke et al. [26, 24] recently introduced highly non-trivial techniques to improve these algorithms further for an infinite class of choices for σ and ρ . Moreover, Focke et al. additionally provide matching lower bounds for these new algorithms that rule out additional improvements [26, 25].

Beyond Finite and Cofinite Sets. Although the known results already capture large classes of problems, they are limited to finite and cofinite sets. This leaves open the entire range of infinite sets (with infinite complements), which contains not only “unstructured” sets like the set of prime numbers, for example, but also easy to describe and frequently used sets like the even or odd numbers, and arithmetic progressions in general.

One important example for families that are neither finite nor cofinite are *residue classes*. We say that a set $\tau \subseteq \mathbb{Z}_{\geq 0}$ is a residue class modulo m if there are two integers a and m such that $\tau = \{n \in \mathbb{Z}_{\geq 0} \mid n \equiv_m a\}$; we usually require that $0 \leq a < m$ for the canonical representation. Again, the two most natural residue classes are the even and odd numbers.

Surprisingly, for such infinite sets the complexity of (σ, ρ) -DOMSET is significantly underexplored and heavily fragmented even in the classical, non-parameterized, setting. This is especially surprising as this variant of the problem has direct applications in other fields like coding theory [13, 38].

Halldórsson, Kratochvíl, and Telle consider a variation of Independent Set where the unselected vertices have parity constraints [37]. They provide a complete dichotomy between polynomial-time solvable cases and NP-hard cases. Later the same group of authors considered the case where each of the sets comprises either the even or odd integers and proved similar hardness results [38]. Caro, Klostermeyer, and Goldwasser consider a variant of (σ, ρ) -DOMSET with residue classes as sets where they restrict the *closed* neighborhood of a vertex [11]. In this setting they prove new upper bounds for specific graphs classes including complements of powers of cycles and grid graphs. Fomin, Golovach, Kratochvíl, Kratsch, and Liedloff consider general graphs and provide *exponential-time* algorithms for general residue classes [27].

Although for the case of general sets some more results are known in the parameterized setting [1, 8, 30, 35], surprisingly few involve sets that are neither finite nor cofinite and the parameter treewidth. Gassner and Hatzl [33] consider the problem of residue domination where the sets are either all even or all odd numbers. They provide an algorithm for the problem with running time $2^{3\text{tw}} \cdot n^{\mathcal{O}(1)}$. Gassner and Hatzl also conjecture that their algorithm works when the sets have a larger modulus but unfortunately do not state the expected running time or the actual algorithm.

Chapelle in contrast provides a general algorithm for (σ, ρ) -DOMSET which covers *all* ultimately periodic sets¹ but, unfortunately, does not provide an explicit running time of the algorithm [14, 15].

In this work, we answer the main question from above and settle the complexity of (σ, ρ) -DOMSET for another large class of sets, namely for the residue classes.

► **Main Theorem 1.** *Write $\sigma, \rho \subseteq \mathbb{Z}_{\geq 0}$ for two residue classes modulo $m \geq 2$.*

Then, in time $m^{\text{tw}} \cdot |G|^{\mathcal{O}(1)}$ we can decide simultaneously for all s if the given graph G has a (σ, ρ) -set of size s when a tree decomposition of width tw is given with the input.

We remark that our algorithm does not only solve the decision version but also the maximization, minimization and exact version of (σ, ρ) -DOMSET.

Despite the fact that there are some pairs for which the decision version of (σ, ρ) -DOMSET is efficiently solvable (for example, the empty set is a trivial minimum solution if $0 \in \rho$), we prove that for all other “difficult” cases our algorithm is optimal even for the decision version and cannot be improved unless SETH fails. We refer to Definition 2.6 for a complete list of the “easy” pairs for which the decision version can be solved in polynomial time; to all other pairs we refer as “difficult”.

¹ A set τ is ultimately periodic if there is a finite automaton (over a unary alphabet) such that the length of the accepted words is precisely described by the set τ .

► **Main Theorem 2.** Write $\sigma, \rho \subseteq \mathbb{Z}_{\geq 0}$ for difficult residue classes modulo $m \geq 2$.

Unless *SETH* fails, for all $\varepsilon > 0$, there is no algorithm that can decide in time $(m - \varepsilon)^{pw} \cdot |G|^{\mathcal{O}(1)}$ whether the input graph G has a (σ, ρ) -set, when a path decomposition of width pw is given with the input.

Observe that our lower bound is for the larger parameter pathwidth, which immediately implies the result for the smaller parameter treewidth.

Our Contribution. Before we outline the formal ideas behind our results, we first highlight why these bounds are more surprising than they seem to be at the first glance.

To that end, let us take a deeper look at the algorithms of [24, 60]. Typically, the limiting factor for faster algorithms parameterized by treewidth is the number of states that have to be considered for each bag of the tree decomposition. For vertex selection problems, the state of a vertex is defined by two values: (1) whether it is selected and (2) how many selected neighbors it gets in some (partial) solution. To bound this latter number, we identify the largest “reasonable” state a vertex can have when it is selected and when it is unselected.

For finite sets σ and ρ this largest reasonable state is simply determined by the maximum of the respective sets, that is, we set $s_{\text{top}} = \max \sigma$ and $r_{\text{top}} = \max \rho$ as the largest reasonable number of neighbors, respectively.

Then, for a selected vertex, the allowed number of selected neighbors ranges from 0 to s_{top} , yielding $s_{\text{top}} + 1$ states for selected vertices. Similarly, we need to consider $r_{\text{top}} + 1$ states for unselected vertices. Combining the two cases, for each bag of the tree decomposition there are at most $(s_{\text{top}} + r_{\text{top}} + 2)^{tw+1}$ states to consider. Surprisingly, Focke et al. proved that, for an infinite number of finite sets, even at most $(t_{\text{top}} + 1)^{tw+1}$ of said states suffice, where $t_{\text{top}} = \max(s_{\text{top}}, r_{\text{top}})$ [26].² When $s_{\text{top}} = r_{\text{top}}$ this improves the algorithm by a factor of 2^{tw+1} .

Similarly, for the case of residue classes with modulus m , the number of selected neighbors effectively ranges from 0 to $m - 1$ as for all larger values the behavior is equivalent to some smaller value. This gives us m states if a vertex is unselected and m states if a vertex is selected. Hence, the straight-forward bound for the number of states is $(2m)^{tw+1}$ which is a factor of 2^{tw+1} worse than the bound from the running time of our algorithm.

We remark that the most naive approach, for which we remove all integers from the sets σ and ρ that are larger than the number of vertices of the input graph and then apply the improved result by Focke et al. for finite sets, fails miserably. This approach would merely give an XP-algorithm as the size of the sets now depends on the size of the input graph.

Hence, it is far from trivial to obtain the claimed running time since the classical approach would not give something better than $(2m)^{tw} \cdot n^{\mathcal{O}(1)}$.

Our Techniques. Although we use the algorithmic result by Focke et al. as a basis for our upper bounds, our algorithm does not follow as an immediate corollary. There are two main challenges that we need to overcome to obtain the fast running time.

First, all previous results with *tight* bounds considered finite or cofinite sets. In this setting all integers (starting from some threshold) are somewhat equivalent in the sense that they are either all contained in the set or all not contained in the set. This makes defining a largest reasonable state quite convenient. For the residue classes this is not as easily possible as the integers change between membership and non-membership. Hence, we need an even more careful construction and analysis when improving upon the naive bound for the number of states.

² To keep notation simple, we omit the special case where the bound is $(t_{\text{top}} + 2)^{tw+1}$.

Second, it does not suffice to bound the number of states which the dynamic program considers, but we also need to be able to combine these states efficiently at the join nodes. Although this is a general issue when parameterizing by treewidth, until today there does not seem to be one solution which works in a black-box manner in all settings. Instead, we need to carefully design a new approach that takes care of the particular setting we consider which is based on but differs from the existing results for finite and cofinite sets.

For the lower bound result we are at a similar situation as for the upper bound; the setting is similar to what is known but still different. There are several known lower bounds for Dominating Set and its related problems, but these reductions are usually quite tailored to the specific problem and lack a modular construction – it is difficult to reuse existing results. The proofs are usually based on a direct reduction from k -SAT which introduces an additional overhead that obfuscates the high-level idea by technicalities (SAT has a running time bound of the form 2^n but we want a bound of the form c^{pw}).

We avoid this overhead by reducing from an appropriate Constraint Satisfaction Problem introduced by Lampis for precisely such settings [44]. Our results can be seen as one of the first applications of this new approach (outside the original setting) that can potentially also serve as a blue-print to simplify many other reductions and lower bound proofs or to directly obtain simpler results from scratch.

The Special Case of Parity Domination. When taking a closer look at the precise statement of our lower bound (and Definition 2.6), we are reminded that both results do not apply for the same set of pairs. Especially for the case of residue classes with modulus $m = 2$, our algorithm solves the minimization version, but Main Theorem 2 provides no matching lower bound. As we may solve the decision problem for these cases in polynomial-time via Gaussian elimination (see, for example, [2, 21, 34, 38, 56]), our lower bound explicitly excludes these cases by referring to them as “easy”.

Surprisingly, exactly these easy cases can be related to a single-player game called *Lights Out* that was published 1995. In this game the unassuming player is presented with a 5×5 grid of switches and lamps, some or all of them initially turned on, and the task is to turn off all lamps by pressing the switches. The catch is that every switch flips not only the state of its corresponding lamp (from “on” to “off” or vice-versa), but also the states of the neighboring lamps in the grid [5, 23].³

Since the order in which the buttons are pressed does not matter and every button has to be pressed at most once as a second press would undo the first operation, we can describe a *solution* to an initial configuration as a set of switches that need to be flipped to turn all lights off.

When we assume that initially all lights are turned on, then we can directly treat *Lights Out* as a variant of (σ, ρ) -DOMSET with $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 0\}$ and $\rho = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$. We also refer to this problem, where the input is an arbitrary graph, as REFLEXIVE-ALLOFF since we assume that each switch triggers the corresponding lamp. When this is not the case but still all lights are initially turned on, we have $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\} = \rho$ and refer to the problem as ALLOFF as the corresponding switch does not trigger the associated lamp.

Despite the fact that it is easy to find *some* solution for these two problems if one exists, the minimization versions do not have such a trivial answer and are known to be NP-complete [11, 38, 55]. Hence, we investigate the minimization versions for these two problems and complement the algorithmic result from Main Theorem 1 as follows.

³ Similar games have also been released under the names *Merlin* and *Orbit* [23].

► **Main Theorem 3.** *Unless SETH fails, for all $\varepsilon > 0$, there is no algorithm for each of the problems REFLEXIVE-ALLOFF and ALLOFF that can decide in time $(2 - \varepsilon)^{pw} \cdot |G|^{\mathcal{O}(1)}$ whether there exists a solution of arbitrary size (the size is given as input) for a graph G that is given with a path decomposition of width pw .*

Together with the lower bound for the general case, we conclude that our algorithm is the best possible, unless a major breakthrough for solving SAT happens.

Further Directions. When taking a step back, the results of this work serve two purposes which can be seen as starting points for further investigations and improvements.

- First, we settle the complexity of (σ, ρ) -DOMSET conclusively for the case of residue classes by providing matching upper and lower bounds.

We later list some candidates that might allow similar improvements. Such improvements would, similar to our results, extend the list of problems started by Focke et al. [26] where significant improvements for the supposedly optimal algorithms are possible.

- Second, in comparison to the fairly complicated results for the case of finite sets in [26], this work can be seen as a significantly simpler introduction to those techniques that are relevant to obtain faster algorithms by exploiting the structural properties of the sets. We believe that for many other (parameterized) problems – including but not limited to (σ, ρ) -DOMSET – the algorithms can be improved exponentially by using these new techniques.

In the following we list several possible directions that could serve as a next step on the route to a complete picture of the complexity of (σ, ρ) -DOMSET.

- A first natural case could be pairs of two residue classes with *different* moduli m_σ and m_ρ . Then, the natural structural parameter m (which is the modulus in our case) is the greatest common divisor of m_σ and m_ρ . In this setting the case $m = 1$ is also relevant as this does not directly imply that the sets contain all natural numbers.
- A different direction considers the combination of a residue class with a finite or cofinite set. Focke et al. show that representative sets [29, 43, 48, 53] can be used to speed up the algorithm even further for the case of cofinite sets [24]. Independently of finding the optimal algorithm to handle the join operation for representative sets, it is not even clear what the optimal running time should be in such a case.
- Caro and Jacobson [10] introduced the problem NON- $z(\bmod k)$ DOMINATING SET which can also be described as a (σ, ρ) -DOMSET problem where the sets are complements of residue classes, which is equivalent to a finite union of residue classes. For example, for $z = 0$ and $k = 3$, we set $\sigma = \{0, 1, 3, 4, \dots\} = \{0, 3, 6, \dots\} \cup \{1, 4, 7, \dots\}$ and $\rho = \{1, 2, 4, 5, 7, 8, \dots\} = \{1, 4, 7, \dots\} \cup \{2, 5, 8, \dots\}$. What is the optimal running time in this case?
- The general algorithm by Chapelle for the case when both sets are ultimately periodic has a running time single-exponential in treewidth despite being stated implicitly only [14, 15]. What is the best running time for an algorithm solving *all* cases of (σ, ρ) -DOMSET that are currently known to be fixed-parameter tractable?
- Are there more classes of sets for which there is an fpt algorithm parameterized by treewidth? Chapelle showed that once there are large gaps in the set, the problem becomes significantly harder [14, 15].

► **Theorem 1.1** ([14, Theorem 1] and [15, Théorème 3.3.1]). *Write σ for a set with arbitrarily large gaps between two consecutive elements (such that a gap of length t is at distance $\text{poly}(t)$ in σ), and write ρ for a cofinite set with $\min \sigma \geq 1$ and $\min \rho \geq 2$. Then, the problem (σ, ρ) -DOMSET is W[1]-hard when parameterized by the treewidth of the input graph.*

Examples are the two natural sets where $\sigma = \{2^i \mid i \in \mathbb{Z}_{\geq 0}\}$ or when σ is the set of all Fibonacci numbers [14]. We observe that this is one of the rare cases where a problem is W[1]-hard even when parameterizing by treewidth.

The classification by Chapelle is not a dichotomy result in the sense that it provides a full classification between the fpt cases and the ones that are W[1]-hard. For instance, what is the complexity for sets like $\sigma = \mathbb{Z}_{\geq 0} \setminus \{2^i \mid i \in \mathbb{Z}_{\geq 0}\}$ which have gaps of constant size only but are not ultimately periodic?

- With our results, there are improved algorithms for the case when the sets are finite, cofinite or residue classes. Nevertheless, the description of the exact running time is highly non-uniform, that is, the exact complexity explicitly depends on the underlying set. Can we describe the complexity of optimal algorithms in a compact form as, for example, done by Chapelle for the general algorithm via finite automata [14, 15]? This notation suffices to describe the state of a single vertex, but the representation of the structural insights leading to fewer states and faster algorithms remains open.
- Lastly, for which other problems besides (σ, ρ) -DOMSET can the techniques from our upper bounds (sparse languages and compression of vectors) be used to obtain faster algorithms? As the high-level idea of our lower bounds is quite modular, it should also be possible to use these concepts as blue-prints to achieve matching bounds for other problems as well.

2 Technical Overview

In this section we give a high-level overview of the results in this paper and outline the main technical contributions we use. We start by rigorously defining the main problem considered in this work and the property of a set being m -structured.

► **Definition 2.1** ((σ, ρ) -sets, (σ, ρ) -DOMSET). *Fix two non-empty sets σ and ρ of non-negative integers.*

For a graph G , a set $S \subseteq V(G)$ is a (σ, ρ) -set for G if and only if (1) for all $v \in S$, we have $|N(v) \cap S| \in \sigma$, and (2), for all $v \in V(G) \setminus S$, we have $|N(v) \cap S| \in \rho$.

The problem (σ, ρ) -DOMSET asks for a given graph G , whether there is a (σ, ρ) -set S or not.

We also refer to the problem above as the *decision* version. The problem naturally also admits related problems such as asking for a solution of a specific size, or for the smallest or largest solution, that is, the *minimization* and *maximization* version.

For the case of finite and cofinite sets, Focke et al. [24, 25] realized that the complexity of (σ, ρ) -DOMSET significantly changes (and allows faster algorithms) when σ and ρ exhibit a specific structure, which they refer to as *m-structured*.

► **Definition 2.2** (m -structured sets [24, Definition 3.2]). *Fix an integer $m \geq 1$. A set $\tau \subseteq \mathbb{Z}_{\geq 0}$ is m -structured if all numbers in τ are in the same residue class modulo m , that is, if there is an integer c^* such that $c \equiv_m c^*$ for all $c \in \tau$.*

Observe that every set τ is m -structured for $m = 1$. Therefore, one is usually interested in the largest m such that a set is m -structured. When considering two sets σ and ρ , we say that this pair is m -structured if each of the two sets is m -structured. More formally, assume that σ is m_σ -structured and ρ is m_ρ -structured. In this case the pair (σ, ρ) is m -structured where m is the greatest common divisor of m_σ and m_ρ . As in our case the sets σ and ρ are residue classes modulo $m \geq 2$, the sets are always m -structured.

In the following we first present the algorithmic result, which outlines the proof of Main Theorem 1. Afterward we move to the lower bounds where we consider Main Theorem 2 and finally we focus on the special case of Lights Out from Main Theorem 3.

2.1 Upper Bounds

The basic idea to prove the upper bound is to provide a dynamic programming algorithm that operates on a tree decomposition of the given graph. For each node of this decomposition we store all valid states, where each such state describes how a possible solution, i.e., a set of selected vertices, interacts with the bags of the corresponding node. We formalize this by the notion of a partial solution.

For a node t with associated bag X_t , we denote by V_t the set of vertices introduced in the subtree rooted at t and by G_t the graph induced by these vertices. We say that a set $S \subseteq V_t$ is a partial solution (for G_t) if

- for each vertex $v \in S \setminus X_t$, we have $|N(v) \cap S| \in \sigma$, and
- for each vertex $v \in V_t \setminus (S \cup X_t)$, we have $|N(v) \cap S| \in \rho$.

The solution is partial in the sense that there are no constraints imposed on the number of neighbors of the vertices in X_t , that is, only the vertices in $V_t \setminus X_t$ must have a valid number of neighbors.

We characterize the partial solutions by the states of the vertices in the bag. When σ and ρ are finite or cofinite sets, the largest reasonable state is included in the respective set, which is not necessarily the case for residue classes. Consider two fixed, residue classes σ and ρ modulo $m \geq 2$. Every selected vertex can have up to m different states and similarly, every unselected vertex can have m different states. Hence, for each bag, the number of relevant different partial solutions is bounded by $(2m)^{|X_t|}$.

High-level Idea. The crucial step to fast and efficient algorithms is to provide a better bound on the number of states for each bag when the sets σ and ρ are residue classes modulo $m \geq 2$. We denote by \mathbb{A} the set of all possible states a vertex might have in a valid solution. Then, let $L \subseteq \mathbb{A}^{X_t}$ be the set of all possible state-vectors corresponding to partial solutions for G_t . Our first goal is to show that $|L| \leq m^{|X_t|}$, which means that not all theoretically possible combinations of states can actually have a corresponding partial solution in the graph.

Moreover, we also need to be able to combine two partial solutions at the join nodes of the tree decomposition. For a fast join operation, it does not suffice to bound the size of L . This follows from the observation that the convolution algorithm used to handle the join operation does not depend on L but on the *space* where the states come from. In our case, the size of the space where L comes from is still $(2m)^{|X_t|}$, which is too large. To decrease this size, we observe that a significant amount of information about the states of the vertices can be inferred from other positions, that is, we can *compress* the vectors.

As a last step it remains to combine the states significantly faster than a naive algorithm. To efficiently compute the join, we use an approach based on the fast convolution techniques by van Rooij which was already used for the finite case [60]. However, we have to ensure that the compression of the vectors is actually compatible with the join operation, that

is, while designing the compression we already have to take in mind that we later join two (compressed) partial solutions together. Since the compressed vectors are significantly simpler, these states can now be combined much faster.

Bounding the Size of a Single Language. Recall that every partial solution S can be described by a state-vector $x \in \mathbb{A}^n$ where we abuse notation and set $n := |X_t|$. When x describes the partial solution S , we also say that S is a witness for x . We denote the set of the state-vectors of all partial solutions for G_t as L . We later refer to the set L as the realized language of (G_t, X_t) . To provide the improved bound on the size of L , we decompose each state-vector x into two vectors: The *selection-vector* of x , also called the σ -vector and denoted by $\vec{\sigma}(x)$, indicates whether each vertex in X_t is selected or not. The *weight-vector* of x , denoted by $\vec{w}(x)$, contains the number of selected neighbors of the vertices in X_t .

The key insight into the improved bound is that for two partial solutions of similar size (with regard to modulo m), the σ -vectors and the weight-vectors of these two solutions are orthogonal. This observation was already used to prove the improved bound when σ and ρ are finite [24]. We extend this result to the case of residue classes.

To formally state the key insight, we define the notion of a graph with portals.

► **Definition 2.3** (Graph with Portals; compare [24, Section 3]). *A graph with portals G is a pair (G', U) , where G' is a graph and $U \subseteq V(G')$. If $U = \{u_1, \dots, u_k\}$, then we also write (G', u_1, \dots, u_k) instead of (G', U) .*

If it is clear from the context, we also refer to a graph with portals simply as a graph.

Intuitively, one can think of G' being the graph G_t , and U the set X_t for a node t of a tree decomposition.

► **Lemma 2.4** (Compare [24, Lemma 4.3]). *Let σ and ρ denote two residue classes modulo $m \geq 2$. Let (G, U) be a graph with portals and let $L := L(G, U) \subseteq \mathbb{A}^U$ denote its realized language. Consider two strings $x, y \in L$ with witnesses $S_x, S_y \subseteq V(G)$ such that $|S_x \setminus U| \equiv_m |S_y \setminus U|$. Then, $\vec{\sigma}(x) \cdot \vec{w}(y) \equiv_m \vec{\sigma}(y) \cdot \vec{w}(x)$.*

To prove this result, we count edges between the vertices in S_x and the vertices in S_y in two different ways. We first count the edges based on their endpoint in S_x . These vertices can be partitioned into three groups: (1) the vertices contained in U , (2) the vertices outside U which are not in S_y , and (3) the vertices outside U which are in S_y . Then, the number of edges $|E(S_x \rightarrow S_y)|$ from S_x to S_y satisfies

$$|E(S_x \rightarrow S_y)| \equiv_m \min \rho \cdot |S_x \setminus (S_y \cup U)| + \min \sigma \cdot |(S_x \cap S_y) \setminus U| + \vec{\sigma}(x) \cdot \vec{w}(y)$$

because the sets σ and ρ are residue classes modulo m . When counting the edges based on their endpoint in S_y , the positions of x and y flip and the result follows. As this property enables us to prove that the size of L is small, we refer to this property as *sparse*.

Even though intuitively this orthogonality provides a reason why the size of the language is not too large, this does not result in a formal proof. However, when fixing which vertices are selected, that is, when fixing a σ -vector \vec{s} , then there is an even stronger restriction on the values of the weight-vectors. Instead of restricting the entire vector, it actually suffices to fix the vector on a certain number of positions which are described by some set S to which we refer as σ -defining set. If two σ -vectors of strings of the language agree on these positions from S , then *all* remaining positions of the two σ -vectors must be identical as well.

With the sparseness property we then show that it suffices to fix the σ -vectors on the positions from S (which then determines the values on \bar{S}), and the weight-vector on the positions from \bar{S} (which then determines the weight-vector on the positions from S). Formally, we prove Lemma 2.5 which mirrors [24, Lemma 4.9] in the case of residue classes.

► **Lemma 2.5** (Compare [24, Lemma 4.9]). *Let σ and ρ denote two residue classes modulo $m \geq 2$. Let $L \subseteq \mathbb{A}^n$ be a sparse language with a σ -defining set S for $\{\vec{\sigma}(x) \mid x \in L\}$. Then, for any two strings $x, y \in L$ with $\vec{\sigma}(x) = \vec{\sigma}(y)$, the positions \bar{S} uniquely characterize the weight vectors of x and y , that is, we have*

$$\vec{w}(x)[\bar{S}] = \vec{w}(y)[\bar{S}] \quad \text{implies} \quad \vec{w}(x) = \vec{w}(y).$$

With this result it is straight-forward to bound the size of a sparse language of dimension n . Our goal is to bound the number of weight-vectors that can be combined with a fixed σ -vector to form a valid type. Assume we fixed a σ -vector \vec{s} on the positions from S . Since this already determines the remaining positions of the σ -vector (even if we do not know the values a priori), the number of possible σ -vectors is at most $2^{|\bar{S}|}$. For the weight-vector there are m choices for each of the positions from \bar{S} . Then, the values for the positions from S are uniquely determined by those on \bar{S} because of the previous result. Using $m \geq 2$ this allows us to bound the size of a sparse language by

$$m^{|\bar{S}|} \cdot 2^{|\bar{S}|} \leq m^n.$$

Compressing Weight-Vectors. Based on the previous observations and results, we focus on the analysis for a fixed σ -vector \vec{s} . Though we could iterate over all at most $2^{|\bar{S}|}$ possible σ -vectors without dominating the running time, the final algorithm only considers the σ -vectors resulting from the underlying set L . Hence, we assume that all vectors in L share the same σ -vector \vec{s} .

When looking again at the bound for the size of L , it already becomes apparent how we can compress the weight-vectors. Recall that once we have fixed the entries of a weight-vector of some vector $x \in L$ at the positions of \bar{S} , the entries of the weight-vector on S are predetermined by Lemma 2.5. Hence, for the compressed vector we simply omit the entries on the positions in S , that is, the compressed weight-vector is the projection of the original weight-vector to the dimensions from \bar{S} .

It remains to recover the original vectors from their compression. As the implication from Lemma 2.5 actually does not provide the values for the positions on S , it seems tempting to store a single representative, which we refer to as *origin*-vector o to recover the omitted values for all compressed vectors. Unfortunately, this is not (yet) sufficient.

Observe that Lemma 2.5, which serves as basis for the compression, requires that the weight-vector u and the origin-vector o agree on the coordinates from \bar{S} . Therefore, it would be necessary to store one origin-vector for each possible choice of values on \bar{S} , which would not yield any improvement in the end.

In order to recover the values of the compressed weight-vector, we use our structural property from Lemma 2.4 once more. Intuitively, we use that changing the weight-vector at one position (from \bar{S} , in our case), has an effect on the value at some other position (from S , in our case). Based on this idea we define an auxiliary vector, which we refer to as *remainder*-vector. Intuitively, the entries of this vector capture the difference of the weight-vector u and the origin-vector o on the positions in \bar{S} . By the previous observation this also encodes how much these two vectors u and o differ on the positions from S . This remainder-vector then allows us to efficiently decompress the compressed weight-vectors again. In consequence, the final compression reduces the size of the space where the weight-vectors are chosen from, which is a prerequisite for the last part of the algorithm.

Faster Join Operations. To obtain the fast join operation, we apply the known convolution techniques by van Rooij [60]. As the convolution requires that all operations are done modulo some small number, we can directly apply it as every coordinate of the compressed vector is computed modulo m . As the convolution operates in the time of the space where the vectors are from, we obtain an overall running time of $m^{|X_e|}$ for the join operation.

The final algorithm is then a dynamic program where the procedures for all nodes except the join node follow the standard procedure. For the join node, we iterate over all potential σ -vectors of the combined language, then join the compressed weight-vectors, and finally output the union of their decompositions.

By designing the algorithm such that we consider solutions of a certain size, we achieve that the considered languages are sparse and thus, the established machinery provides the optimal bound for the running time. In total, we obtain Main Theorem 1.

► **Main Theorem 1.** *Write $\sigma, \rho \subseteq \mathbb{Z}_{\geq 0}$ for two residue classes modulo $m \geq 2$.*

Then, in time $m^{\text{tw}} \cdot |G|^{\mathcal{O}(1)}$ we can decide simultaneously for all s if the given graph G has a (σ, ρ) -set of size s when a tree decomposition of width tw is given with the input.

2.2 Lower Bounds

After establishing the upper bounds, we focus on proving matching lower bounds, that is, we prove the previous algorithm to be optimal under SETH. For all difficult cases, we provide a general lower bound and for the easy cases that are solvable in polynomial time but are non-trivial, we prove a lower bound for the minimization version by a separate reduction. In the following we first focus on the difficult cases.

► **Definition 2.6 (Easy and Difficult Cases).** *Let σ and ρ be two residue classes. We say that this pair is easy if $0 \in \rho$ or*

- $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 0\}$ and $\rho = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$, or
- $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$ and $\rho = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$.

Otherwise, we say that the pair is difficult.

Clearly the case $0 \in \rho$ is trivial since the empty set is a valid solution. For the other two cases we can formulate the problem as a system of linear equations over \mathbb{F}_2 : for each vertex we create a variable indicating the selection status and introduce one appropriately chosen constraint involving the neighboring vertices. Then, Gaussian elimination provides a solution in polynomial time. We refer to [2, 34, 38, 56] for a formal proof. Thus, unless mentioned otherwise, we henceforth focus on the difficult cases.

Starting from the first SETH-based lower bounds when parameterizing by treewidth by Lokshtanov, Marx and Saurabh [45] (see also references in [44] for other applications) many reductions suffered from the following obstacle: SETH provides a lower bound of the form $(2 - \varepsilon)^n$ whereas for most problems a lower bound of the form $(c - \varepsilon)^{\text{tw}}$ is needed for some integer $c > 2$. To bridge this gap, several technicalities are needed to obtain the bound with the correct base. Lampis introduced the problem (family) q -CSP- B , which hides these technicalities and allows for cleaner reductions. This problem generalizes q -SAT such that every variable can now take B different values, that is, for $B = 2$ this is the classical q -SAT problem. Formally q -CSP- B is defined as follows.

► **Definition 2.7 (q -CSP- B [44]).** *Fix two numbers $q, B \geq 2$. An instance of q -CSP- B is a tuple (X, \mathcal{C}) that consists of a set X of n variables having the domain $D = [1..B]$ each, and a set \mathcal{C} of constraints on X . A constraint C is a pair $(\text{scp}(C), \text{acc}(C))$ where $\text{scp}(C) \in X^q$ is the scope of C and $\text{acc}(C) \subseteq D^q$ is the set of accepted states.*

41:12 Residue Domination in Bounded-Treewidth Graphs

The task of the problem is to decide if (X, \mathcal{C}) is satisfiable, that is, decide if there exists an assignment $\pi: X \rightarrow D$ such that, for all constraints C with $\text{scp}(C) = (v_{\lambda_1}, \dots, v_{\lambda_q})$ it holds that $(\pi(v_{\lambda_1}), \dots, \pi(v_{\lambda_q})) \in \text{acc}(C)$.

In other words, the constraints specify valid assignments for the variables, and we are looking for a variable assignment that satisfies all constraints.

Apart from introducing this problem, Lampis also proved a conditional lower bound based on SETH which allows us to base our reduction on this special type of CSP.

► **Theorem 2.8** ([44, Theorem 3.1]). *For any $B \geq 2, \varepsilon > 0$ we have the following: assuming SETH, there is a q such that n -variable q -CSP- B with ℓ constraints cannot be solved in time $(B - \varepsilon)^n \cdot (n + \ell)^{\mathcal{O}(1)}$.*

To obtain the correct lower bound the most suitable version of q -CSP- B can be used, which then hides the unwanted technicalities.

In our case we cover numerous (actually infinitely many) problems. This creates many positions in the potential proof where (unwanted) properties of the sets σ and ρ have to be circumvented or exploited. In order to minimize these places and to make use of the special starting problem, we split the proof in two parts. This concept of splitting the reduction has already proven to be successful for several other problems [18, 24, 47, 48].

As synchronizing point, we generalize the known (σ, ρ) -DOMSET problem where we additionally allow that *relations* are added to the graph. Therefore, we refer to this problem as (σ, ρ) -DOMSET^{REL}. Intuitively one can think of these relations as constraints that observe a predefined set of vertices, which we refer to as *scope*, and enforce that only certain ways of selecting these vertices are allowed in a valid solution. To formally state this intermediate problem, we first define the notion of a *graph with relations*.

► **Definition 2.9** (Graph with Relations [25, Definition 4.1]). *We define a graph with relations as a tuple $G = (V, E, \mathcal{C})$, where V is a set of vertices, E is a set of edges on V , and \mathcal{C} is a set of relational constraints, that is, each $C \in \mathcal{C}$ is in itself a tuple $(\text{scp}(C), \text{acc}(C))$. Here the scope $\text{scp}(C)$ of C is an unordered tuple of $|\text{scp}(C)|$ vertices from V . Then, $\text{acc}(C) \subseteq 2^{\text{scp}(C)}$ is a $|\text{scp}(C)|$ -ary relation specifying possible selections within $\text{scp}(C)$. We also say that C observes $\text{scp}(C)$.*

The size of G is $|G| := |V| + \sum_{C \in \mathcal{C}} |\text{acc}(C)|$. Slightly abusing notation, we usually do not distinguish between G and its underlying graph (V, E) . We use G to refer to both objects depending on the context.

We define the treewidth and pathwidth of a graph with relations as the corresponding measure of the modified graph that is obtained by replacing all relations by a clique on the vertices from the scope.

We lift the notion of (σ, ρ) -set from Definition 2.1 to graphs with relations by requiring that every relation has to be satisfied as well. Hence, the definition of (σ, ρ) -DOMSET^{REL} follows naturally. These definitions are a reformulation of [25, Definition 4.3 and 4.8].

► **Definition 2.10** ((σ, ρ) -Sets of a Graph with Relations, (σ, ρ) -DOMSET^{REL}). *Fix two non-empty sets σ and ρ of non-negative integers.*

For a graph with relations $G = (V, E, \mathcal{C})$, a set $S \subseteq V$ is a (σ, ρ) -set of G if and only if (1) S is a (σ, ρ) -set of the underlying graph (V, E) and (2) for every $C \in \mathcal{C}$, the set S satisfies $S \cap \text{scp}(C) \in \text{acc}(C)$. We use $|G|$ as the size of the graph and say that the arity of G is the maximum arity of a relation of G .

The problem (σ, ρ) -DOMSET^{REL} asks for a given graph with relations $G = (V, C, \mathcal{C})$, whether there is such a (σ, ρ) -set or not.

With this intermediate problem, we can now formally state the two parts of our lower bound proof. The first step embeds the q -CSP- B problem (for appropriately chosen B) into the graph problem (σ, ρ) -DOMSET^{REL}. We design the reduction in such a way that the resulting instance has a small pathwidth (namely, roughly equal to the number of variables). Combined with the conditional lower bound for q -CSP- B based on SETH from Theorem 2.8, we prove the following intermediate lower bound.

► **Lemma 2.11.** *Let σ and ρ be two residue classes modulo $m \geq 2$.*

Then, for all $\varepsilon > 0$, there is a constant d such that (σ, ρ) -DOMSET^{REL} on instances of arity at most d cannot be solved in time $(m - \varepsilon)^{\text{pw}} \cdot |G|^{\mathcal{O}(1)}$, where pw is the width of a path decomposition provided with the input instance G , unless SETH fails.

For the second step, we then remove the relations from the constructed (σ, ρ) -DOMSET^{REL} instance, to obtain a reduction to the (σ, ρ) -DOMSET problem. Observe that the construction from Lemma 2.11 works for the general case (even when $0 \in \rho$ is allowed). Hence, our second step now exploits that the sets are difficult.

► **Lemma 2.12.** *Let σ and ρ be two difficult residue classes modulo m . For all constants d , there is a polynomial-time reduction from (σ, ρ) -DOMSET^{REL} on instances with arity d given with a path decomposition of width pw to (σ, ρ) -DOMSET on instances given with a path decomposition of width $\text{pw} + \mathcal{O}(2^d)$.*

Combining these two intermediate results directly leads to the proof of Main Theorem 2.

► **Main Theorem 2.** *Write $\sigma, \rho \subseteq \mathbb{Z}_{\geq 0}$ for difficult residue classes modulo $m \geq 2$.*

Unless SETH fails, for all $\varepsilon > 0$, there is no algorithm that can decide in time $(m - \varepsilon)^{\text{pw}} \cdot |G|^{\mathcal{O}(1)}$ whether the input graph G has a (σ, ρ) -set, when a path decomposition of width pw is given with the input.

Proof. Assume we are given a faster algorithm for (σ, ρ) -DOMSET for some $\varepsilon > 0$. Let d be the constant from Lemma 2.11 such that there is no algorithm solving (σ, ρ) -DOMSET^{REL} in time $(m - \varepsilon)^{\text{pw}} \cdot |G|^{\mathcal{O}(1)}$ when the input instance G is given with a path decomposition of width pw .

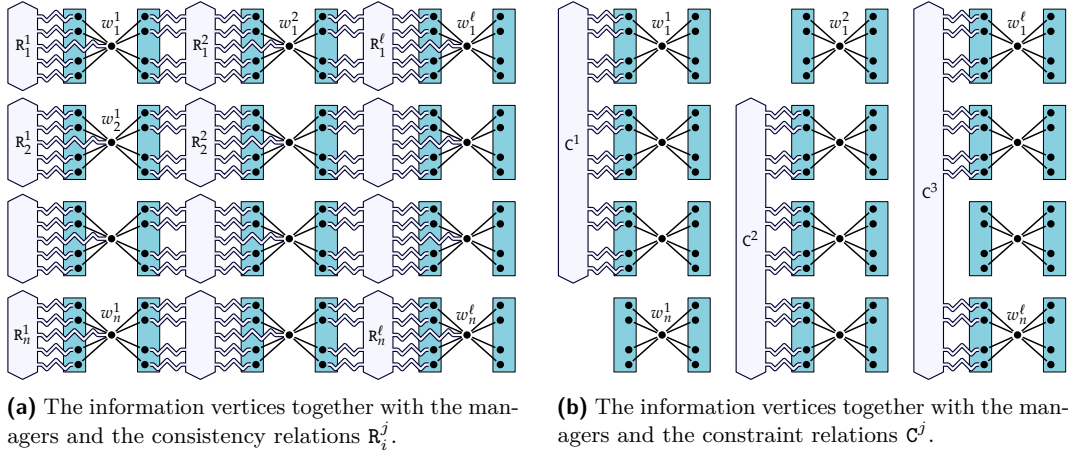
Consider an instance G of (σ, ρ) -DOMSET^{REL} with arity d along with a path decomposition of width $\text{pw}(G)$. We use Lemma 2.12 to transform this instance into an instance G' of (σ, ρ) -DOMSET with a path decomposition of width $\text{pw}(G') = \text{pw}(G) + \mathcal{O}(2^d)$.

We apply the fast algorithm for (σ, ρ) -DOMSET to the instance G' which correctly outputs the answer for the original instance G of (σ, ρ) -DOMSET^{REL}. The running time of this entire procedure is

$$|G|^{\mathcal{O}(1)} + (m - \varepsilon)^{\text{pw}(G')} \cdot |G'|^{\mathcal{O}(1)} = (m - \varepsilon)^{\text{pw}(G) + \mathcal{O}(2^d)} \cdot |G|^{\mathcal{O}(1)} = (m - \varepsilon)^{\text{pw}(G)} \cdot |G|^{\mathcal{O}(1)}$$

since d is a constant only depending on ε . This contradicts SETH and concludes the proof. ◀

The following highlights the main technical contributions leading to the results from Lemmas 2.11 and 2.12.



■ **Figure 1** A depiction of the construction from the lower bound where $m = 5$, $n = 4$, and $\ell = 3$.

Step 1: Encoding the CSP as a Graph Problem

Focke et al. already established the corresponding intermediate result when σ and ρ are finite [25]. Hence, we could try to reuse their lower bound for $(\hat{\sigma}, \hat{\rho})\text{-DOMSET}^{\text{REL}}$ for two finite sets $\hat{\sigma} \subseteq \sigma$ and $\hat{\rho} \subseteq \rho$. However, since σ and ρ are residue classes, several solutions could be indistinguishable from each other (not globally but locally from the perspective of a single vertex) which would result in unpredictable behavior of the construction. Thus, we need to come up with a new intermediate lower bound.

To prove this lower bound for $(\sigma, \rho)\text{-DOMSET}^{\text{REL}}$, we provide a reduction from $q\text{-CSP-}B$ where $B = m$ but reuse some ideas from the known lower bounds in [18, 25, 47, 48]. This allows for a much cleaner reduction (especially compared to the one from [25]) that focuses on the conversion of a constraint satisfaction problem into a vertex selection problem without having to deal with technicalities. Consult Figure 1 for an illustration of the high-level idea of the construction.

Consider a $q\text{-CSP-}m$ instance I with n variables and ℓ constraints. To achieve a low treewidth (or actually pathwidth), we construct a graph with $n \cdot \ell$ vertices, which we refer to as *information vertices*, that are arranged as an n times ℓ grid; rows corresponding to variables and columns corresponding to constraints. We refer to the information vertex from row i and column j as w_i^j . We encode the m different values of each variable by the states of the information vertices in the graph.

To provide sufficiently many neighbors to these information vertices, we introduce *managers*. In our case a manager consists of $2n$ blocks, n left blocks and n right blocks, and each block can provide up to $m - 1$ neighbors to a single vertex. We create one manager for each column (i.e., constraint) and assign one left block and one right block to each information vertex. Then, we make each information vertex adjacent to the two associated blocks by $m - 1$ edges each.

We use the number of selected neighbors from the left block to determine the state of an information vertex (though the vertex might have selected neighbors in the right block too). This directly relates the states of the information vertices to the variable assignments.

Recall that we create a separate manager for each column and that the managers are not connected to each other. Thus, despite the mentioned correspondence, even for a single row the information vertices can have different states. Phrased differently, the encoded

assignment is not necessarily consistent. To keep treewidth low, we cannot simply add a single big relation for each row enforcing the intended behavior. Instead, for each row i , we add a small *consistency relation* R_i^j between every two consecutive columns j and $j + 1$.

The relation R_i^j ensures the consistency between the information vertices w_i^j and w_i^{j+1} , and thus, additionally observes the right block of w_i^j and the left block of w_i^{j+1} . First, R_i^j ensures that information vertex w_i^j is unselected. Now assume that w_i^j has b_1 selected neighbors in its right block and w_i^{j+1} has b_2 selected neighbors in its left block. Then, relation R_i^j ensures that b_2 complements b_1 in the sense that $b_2 = \min \rho - b_1 \pmod m$, that is, b_2 is the smallest number such that $b_1 + b_2 \equiv_m \min \rho$.

It remains to analyze the influence of the information vertices themselves on the consistency of the encoded assignment. By our construction, information vertex w_i^j receives b_0 neighbors from its left block of the manager, receives b_1 neighbors from its right block of the manager, and receives no other neighbors. Since we consider a solution, vertex w_i^j must have a valid number of neighbors, that is, the solution must satisfy $b_0 + b_1 \in \rho$. Since ρ is a residue class modulo m , we get $b_0 + b_1 \equiv_m \min \rho$ which implies that $b_1 = \min \rho - b_0 \pmod m$. When combining this with the observation from the previous paragraph, where we consider two different information vertices, we get $b_2 = \min \rho - (\min \rho - b_0 \pmod m) \pmod m$ and hence, $b_2 = b_0 \pmod m$ which implies that all information vertices of one row have the same state.

As a last step we encode the constraints of the CSP instance. For each constraint C_j we add one *constraint relation* C^j which observes, for each variable appearing in C_j , the neighbors of the corresponding information vertex in the left block of the manager (they are needed to infer the state of the information vertices). The relation C^j then accepts a selection of these vertices if and only if it corresponds to a satisfying assignment.

This concludes the lower bound for the intermediate problem (σ, ρ) -DOMSET^{REL}. Next, we remove the relations and replace them by appropriate gadgets to lift the result to (σ, ρ) -DOMSET.

Step 2: Realizing the Relations

Formally, the second step is a reduction from (σ, ρ) -DOMSET^{REL} to (σ, ρ) -DOMSET. We replace each relation by a suitable gadget that precisely mimics the behavior of the original relation, that is, we *realize* the relation. The realization gadget accepts a selection of vertices if and only if the original relation also accepted this selection. Moreover, such a gadget must not add any selected neighbors to a vertex from the scope, as that could affect the existence of a solution (in the positive but also in the negative).

Curticapean and Marx [18] show that just two types of relations suffice to realize arbitrary relations. Focke et al. prove that for (σ, ρ) -DOMSET only $\text{HW}_{=1}$ relations are needed [25, Corollary 8.8], that is, once we can realize such $\text{HW}_{=1}$ relations, then every relation can be realized. The $\text{HW}_{=1}$ relation accepts if exactly one vertex from the scope of the relation is selected, that is, if the Hamming weight of the σ -vector is exactly one. We strengthen this result further by using an observation from [47] such that only realizations of $\text{HW}_{=1}$ with arity one, two or three are needed.

To realize these relations, we use an auxiliary relation. For some set τ , the relation $\text{HW}_{\in \tau}$ accepts, if and only if the number of selected vertices from the scope of the relation, i.e., the Hamming weight of the σ -vector, is contained in τ . Once we set $\tau - k := \{t - k \mid t \in \tau\}$ to simplify notation, our main results for realizing relations reads as follows.

► **Lemma 2.13.** *Let σ and ρ be two difficult residue classes modulo m . Then, the relation $\text{HW}_{\in \rho - \min \rho + 1}$ can be realized.*

When σ and ρ are difficult we have $m \geq 3$, and hence this gadget directly gives $\text{HW}_{=1}$ when restricting to arity at most 3 as $\min \rho + 1, \min \rho + 2 \notin \rho$. Thus, together with the previous intermediate lower bound this concludes the proof of the lower bound.

For $m = 2$ the decision version is easy, so we cannot expect to realize the $\text{HW}_{=1}$ relation. So, we consider the *minimization version* instead and focus on the two non-trivial cases; for $\rho = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$, we consider $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 1\}$ and $\sigma = \{x \in \mathbb{Z}_{\geq 0} \mid x \equiv_2 0\}$. For the lower bounds from Main Theorem 3, we modify the known NP-hardness reductions by Sutner [55] to keep pathwidth low and to obtain the matching bounds.

References

- 1 Jochen Alber, Hans L. Bodlaender, Henning Fernau, Ton Kloks, and Rolf Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. doi:10.1007/S00453-001-0116-5.
- 2 Marlow Anderson and Todd Feil. Turning lights out with linear algebra. *Mathematics Magazine*, 71(4):300–303, 1998. doi:10.1080/0025570X.1998.11996658.
- 3 Ferhat Ay, Manolis Kellis, and Tamer Kahveci. Submap: Aligning metabolic pathways with subnetwork mappings. *J. Comput. Biol.*, 18(3):219–235, 2011. PMID: 21385030. doi:10.1089/cmb.2010.0280.
- 4 Lukas Barth, Benjamin Niedermann, Martin Nöllenburg, and Darren Strash. Temporal map labeling: a new unified framework with experiments. In Siva Ravada, Mohammed Eunus Ali, Shawn D. Newsam, Matthias Renz, and Goce Trajcevski, editors, *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*, pages 23:1–23:10. ACM, 2016. doi:10.1145/2996913.2996957.
- 5 Abraham Berman, Franziska Borer, and Norbert Hungerbühler. Lights out on graphs. *Mathematische Semesterberichte*, 68(2):237–255, 2021. doi:10.1007/s00591-021-00297-5.
- 6 Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 226–234. ACM, 1993. doi:10.1145/167088.167161.
- 7 Glencora Borradaile and Hung Le. Optimal dynamic program for r-domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPICs*, pages 8:1–8:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.IPEC.2016.8.
- 8 Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.*, 511:66–76, 2013. doi:10.1016/J.TCS.2013.01.009.
- 9 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.
- 10 Yair Caro and Michael S. Jacobson. On non- $z \pmod k$ dominating sets. *Discuss. Math. Graph Theory*, 23(1):189–199, 2003. doi:10.7151/dmgt.1195.
- 11 Yair Caro, William F. Klostermeyer, and John L. Goldwasser. Odd and residue domination numbers of a graph. *Discuss. Math. Graph Theory*, 21(1):119–136, 2001. doi:10.7151/dmgt.1137.
- 12 David Cattanéo and Simon Perdrix. Parameterized complexity of weak odd domination problems. In Leszek Gasieniec and Frank Wolter, editors, *Fundamentals of Computation Theory - 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013. Proceedings*, volume 8070 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 2013. doi:10.1007/978-3-642-40164-0_13.


- 13 David Cattanéo and Simon Perdrix. The parameterized complexity of domination-type problems and application to linear codes. In T. V. Gopal, Manindra Agrawal, Angsheng Li, and S. Barry Cooper, editors, *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, volume 8402 of *Lecture Notes in Computer Science*, pages 86–103. Springer, 2014. doi:10.1007/978-3-319-06089-7_7.
- 14 Mathieu Chapelle. Parameterized complexity of generalized domination problems on bounded tree-width graphs. *CoRR*, abs/1004.2642v5, 2010. doi:10.48550/arxiv.1004.2642.
- 15 Mathieu Chapelle. *Décompositions de graphes : quelques limites et obstructions. (Graphs decompositions: some limits and obstructions)*. PhD thesis, University of Orléans, France, 2011. URL: <https://tel.archives-ouvertes.fr/tel-00659666>.
- 16 E. Cockayne and S. Hedetniemi. Optimal domination in graphs. *IEEE Transactions on Circuits and Systems*, 22(11):855–857, 1975. doi:10.1109/TCS.1975.1083994.
- 17 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.
- 18 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. 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 1650–1669. SIAM, 2016. doi:10.1137/1.9781611974331.CH113.
- 19 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.
- 20 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.
- 21 Yevgeniy Dodis and Peter Winkler. Universal configurations in light-flipping games. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 926–927. ACM/SIAM, 2001. URL: <http://dl.acm.org/citation.cfm?id=365411.365812>.
- 22 Devdatt P. Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 717–724. ACM/SIAM, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644226>.
- 23 Rudolf Fleischer and Jiajin Yu. A survey of the game “Lights Out!”. In Andrej Brodnik, Alejandro López-Ortiz, Venkatesh Raman, and Alfredo Viola, editors, *Space-Efficient Data Structures, Streams, and Algorithms - Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, volume 8066 of *Lecture Notes in Computer Science*, pages 176–198. Springer, 2013. doi:10.1007/978-3-642-40273-9_13.
- 24 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs Part I: Algorithmic results. *CoRR*, abs/2211.04278v2, 2023. doi:10.48550/arXiv.2211.04278.
- 25 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs Part II: Hardness results. *CoRR*, abs/2306.03640, 2023. doi:10.48550/arXiv.2306.03640.
- 26 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight complexity bounds for counting generalized dominating sets in bounded-treewidth graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3664–3683. SIAM, 2023. doi:10.1137/1.9781611977554.ch140.

- 27 Fedor V. Fomin, Petr A. Golovach, Jan Kratochvíl, Dieter Kratsch, and Mathieu Liedloff. Sort and search: Exact algorithms for generalized domination. *Inf. Process. Lett.*, 109(14):795–798, 2009. doi:10.1016/J.IPL.2009.03.023.
- 28 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, 2009. doi:10.1145/1552285.1552286.
- 29 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.
- 30 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.
- 31 Andrei Gagarin and Pdraig Corcoran. Multiple domination models for placement of electric vehicle charging stations in road networks. *Comput. Oper. Res.*, 96:69–79, 2018. doi:10.1016/j.cor.2018.03.014.
- 32 Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Ioan Todinca. Exponential time algorithms for the minimum dominating set problem on some graph classes. *ACM Trans. Algorithms*, 6(1):9:1–9:21, 2009. doi:10.1145/1644015.1644024.
- 33 Elisabeth Gassner and Johannes Hatzl. A parity domination problem in graphs with bounded treewidth and distance-hereditary graphs. *Computing*, 82(2-3):171–187, July 2008. doi:10.1007/s00607-008-0005-8.
- 34 John Goldwasser, William Klostermeyer, and George Trapp. Characterizing switch-setting problems. *Linear and Multilinear Algebra*, 43(1-3):121–135, 1997. doi:10.1080/03081089708818520.
- 35 Petr A. Golovach, Jan Kratochvíl, and Ondrej Suchý. Parameterized complexity of generalized domination problems. *Discret. Appl. Math.*, 160(6):780–792, 2012. doi:10.1016/J.DAM.2010.11.012.
- 36 Jakob Greilhuber. Shining light on periodic dominating sets in bounded-treewidth graphs. Master’s thesis, TU Wien, 2024. doi:10.34726/hss.2024.120579.
- 37 Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Independent sets with domination constraints. *Discret. Appl. Math.*, 99(1-3):39–54, 2000. doi:10.1016/S0166-218X(99)00124-9.
- 38 Magnús M. Halldórsson, Jan Kratochvíl, and Jan Arne Telle. Mod-2 independence and domination in graphs. *Int. J. Found. Comput. Sci.*, 11(3):355–363, 2000. doi:10.1142/S0129054100000272.
- 39 Stephen T. Hedetniemi and Renu C. Laskar. Bibliography on domination in graphs and some basic definitions of domination parameters. *Discret. Math.*, 86(1-3):257–277, 1990. doi:10.1016/0012-365X(90)90365-0.
- 40 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 41 Samir Khuller, Manish Purohit, and Kanthi K. Sarpatwar. Analyzing the optimal neighborhood: Algorithms for budgeted and partial connected dominating set problems. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1702–1713. SIAM, 2014. doi:10.1137/1.9781611973402.123.
- 42 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.
- 43 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.
- 44 Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.

- 45 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, April 2018. doi:10.1145/3170442.
- 46 Dániel Marx. Four shorts stories on surprising algorithmic uses of treewidth. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2020. doi:10.1007/978-3-030-42071-0_10.
- 47 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Degrees and gaps: Tight complexity results of general factor problems parameterized by treewidth and cutwidth. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 95:1–95:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.95.
- 48 Dániel Marx, Govind S. Sankar, and Philipp Schepper. Anti-factor is FPT parameterized by treewidth and list size (but counting is hard). In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of *LIPICs*, pages 22:1–22:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.22.
- 49 Mohsen Alambardar Meybodi, Fedor V. Fomin, Amer E. Mouawad, and Fahad Panolan. On the parameterized complexity of $[1, j]$ -domination problems. *Theor. Comput. Sci.*, 804:207–218, 2020. doi:10.1016/j.tcs.2019.11.032.
- 50 Neeldhara Misra and Piyush Rathi. The parameterized complexity of dominating set and friends revisited for structured graphs. In René van Bevern and Gregory Kucherov, editors, *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2019. doi:10.1007/978-3-030-19955-5_26.
- 51 Karolina Okrasa and Pawel Rzazewski. Fine-grained complexity of graph homomorphism problem for bounded-treewidth 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 1578–1590. SIAM, 2020. doi:10.1137/1.9781611975994.97.
- 52 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Solving dominating set in larger classes of graphs: FPT algorithms and polynomial kernels. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 694–705. Springer, 2009. doi:10.1007/978-3-642-04128-0_62.
- 53 Hadas Shachnai and Meirav Zehavi. Representative families: A unified tradeoff-based approach. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 786–797. Springer, 2014. doi:10.1007/978-3-662-44777-2_65.
- 54 Shay Solomon and Amitai Uzrad. Dynamic $((1+\epsilon) \ln n)$ -approximation algorithms for minimum set cover and dominating set. 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 1187–1200. ACM, 2023. doi:10.1145/3564246.3585211.
- 55 Klaus Sutner. Additive automata on graphs. *Complex Syst.*, 2(6), 1988. URL: http://www.complex-systems.com/abstracts/v02_i06_a03.html.
- 56 Klaus Sutner. Linear cellular automata and the garden-of-eden. *The Mathematical Intelligencer*, 11(2):49–53, 1989. doi:10.1007/BF03023823.
- 57 Jan Arne Telle. Complexity of domination-type problems in graphs. *Nord. J. Comput.*, 1(1):157–171, 1994.

- 58 Jan Arne Telle and Andrzej Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings*, volume 709 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 1993. doi:10.1007/3-540-57155-8_284.
- 59 Kuo-Hui Tsai and Wen-Lian Hsu. Fast algorithms for the dominating set problem on permutation graphs. In Tetsuo Asano, Toshihide Ibaraki, Hiroshi Imai, and Takao Nishizeki, editors, *Algorithms, International Symposium SIGAL '90, Tokyo, Japan, August 16-18, 1990, Proceedings*, volume 450 of *Lecture Notes in Computer Science*, pages 109–117. Springer, 1990. doi:10.1007/3-540-52921-7_60.
- 60 Johan M. M. van Rooij. Fast algorithms for join operations on tree decompositions. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 262–297. Springer, 2020. doi:10.1007/978-3-030-42071-0_18.
- 61 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

Local Enumeration: The Not-All-Equal Case

Mohit Gurumukhani ✉ 🏠 

Cornell University, Ithaca, NY, USA

Ramamohan Paturi ✉

Department of Computer Science and Engineering,
University of California San Diego, La Jolla, CA, USA

Michael Saks ✉

Department of Mathematics, Rutgers University, Piscataway, NJ, USA

Navid Talebanfard ✉ 🏠 

University of Sheffield, UK

Abstract

Gurumukhani et al. (CCC'24) proposed the *local enumeration* problem $\text{ENUM}(k, t)$ as an approach to break the Super Strong Exponential Time Hypothesis (SSETH): for a natural number k and a parameter t , given an n -variate k -CNF with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all satisfying assignments of Hamming weight exactly $t(n)$. Furthermore, they gave a randomized algorithm for $\text{ENUM}(k, t)$ and employed new ideas to analyze the first non-trivial case, namely $k = 3$. In particular, they solved $\text{ENUM}(3, \frac{n}{2})$ in expected 1.598^n time. A simple construction shows a lower bound of $6^{\frac{n}{4}} \approx 1.565^n$.

In this paper, we show that to break SSETH, it is sufficient to consider a *simpler* local enumeration problem $\text{NAE-ENUM}(k, t)$: for a natural number k and a parameter t , given an n -variate k -CNF with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all Not-All-Equal (NAE) solutions of Hamming weight exactly $t(n)$, i.e., those that satisfy and falsify some literal in every clause. We refine the algorithm of Gurumukhani et al. and show that it *optimally* solves $\text{NAE-ENUM}(3, \frac{n}{2})$, namely, in expected time $\text{poly}(n) \cdot 6^{\frac{n}{4}}$.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases Depth 3 circuits, k -CNF satisfiability, Circuit lower bounds, Majority function

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.42

Related Version *Full Version*: <https://arxiv.org/abs/2501.02886>

Funding *Mohit Gurumukhani*: Supported by NSF CAREER Award 2045576 and a Sloan Research Fellowship.

Ramamohan Paturi: Partially supported by NSF grant 2212136.

Acknowledgements We want to thank Pavel Pudlák for helpful discussions.

1 Introduction

Four decades of research on the exact complexity of k -SAT has given rise to a handful of non-trivial exponential time algorithms, i.e., algorithms running in time $2^{(1-\epsilon)n}$ with non-trivial *savings* $\epsilon > 0$ [10, 12, 15, 2, 11, 8, 6, 13]. Despite extensive effort, PPSZ [11] remains essentially the fastest known k -SAT algorithm. It is also known that its analysis cannot be substantially improved [14]. The *Super Strong Exponential Time Hypothesis* (SSETH) formalizes the lack of progress in improving the exact complexity of k -SAT, and states that it cannot be solved in time $2^{(1-\omega(1/k))n}$ [16]. Gurumukhani et al. [5] recently proposed the *local enumeration* problem as a new approach to refute SSETH. More precisely, $\text{ENUM}(k, t)$ is defined as follows: for natural number k and a parameter t , given an n -variate k -CNF F with no satisfying assignment of Hamming weight less than $t(n)$, enumerate



© Mohit Gurumukhani, Ramamohan Paturi, Michael Saks, and Navid Talebanfard;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 42; pp. 42:1–42:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



all satisfying assignments of Hamming weight exactly $t(n)$. They provided a randomized branching algorithm and presented novel ideas to analyze it for the first non-trivial case, $k = 3$.

In this paper, we argue that fast algorithms for an easier local enumeration problem would also refute SSETH. To be precise, we consider the NAE-ENUM(k, t) problem which is formally defined as follows: for a natural number k and a parameter t , given an n -variate k -CNF F with no satisfying assignment of Hamming weight less than $t(n)$, enumerate all Not-All-Equal (NAE) solutions of Hamming weight exactly $t(n)$ ¹. Recall that a NAE solution to a CNF is one which satisfies and falsifies a literal in every clause. We will show that a refinement of the algorithm of [5] can *optimally* solve NAE-ENUM($k, \frac{n}{2}$) for $k = 3$. For this algorithm $t = \frac{n}{2}$ is the hardest case, and we therefore have an algorithm for all $t \leq \frac{n}{2}$. Our proof utilizes a new technique for analyzing a transversal tree (a tree of satisfying solutions with Hamming weight exactly $t(n)$ constructed using the clauses of the k -CNF).

It is easy to see that k -SAT can be reduced to $(k + 1)$ -NAE-SAT by the following folklore reduction: Given a k -CNF F , define a $(k + 1)$ -CNF F' as follows. Let z be a new variable. For each clause $C \in F$ we include the clause $C \vee z$ in F' . It is clear that F is satisfiable iff F' has a NAE solution². Furthermore, if we can solve k -NAE-SAT in time $2^{(1-\mu_k)n}$, then we can solve k -SAT in time $O(2^{(1-\mu_{k-1})n})$. In the other direction, the k -NAE-SAT problem has a trivial reduction to k -SAT: Given a k -CNF F , construct the formula F' by adding, for each clause of F the clause consisting of the negations of its literals, then F has a NAE solution if and only if F' is satisfiable. Therefore, for large k , k -SAT can be solved with asymptotically the same *savings* as that of k -NAE-SAT.

It is noted in [5] that upper bounds on ENUM(k, t) for all $t \leq \frac{n}{2}$ imply k -SAT upper bounds. It is easy to see that the same holds for NAE-ENUM(k, t) and k -NAE-SAT. Therefore, by the discussion above, as far as k -SAT savings are concerned, we may only focus on NAE-ENUM(k, t) for all $t \leq \frac{n}{2}$.

Lower bounds for local enumeration

Define k -CNF $\text{Maj}_{n,k}$ as follows. Divide the n variables into blocks of size $2k - 2$, and for each block include all positive clauses of size k . Every satisfying assignment of $\text{Maj}_{n,k}$ sets at least $k - 1$ variables in each block to 1, and thus has Hamming weight at least $\frac{n}{2}$. Notably, all minimum-weight satisfying assignments are NAE. Furthermore, the number of minimum-weight satisfying assignments of $\text{Maj}_{n,k}$ is $2^{(1-O(\log(k)/k))n}$. This, in particular, is a lower bound on the complexity of both ENUM($k, \frac{n}{2}$) and NAE-ENUM($k, \frac{n}{2}$), and thus if we can show that any of these bounds is tight for all $t \leq \frac{n}{2}$, we break SSETH. We reiterate that it is wide open to obtain even a combinatorial (non-algorithmic) upper bound of this kind; while this is not good enough to break SSETH, it is necessary and will be very interesting to obtain such a bound.

1.1 Our Contributions

Gurumukhani et al. [5] showed that their algorithm solves ENUM(k, t) for all $t \leq \frac{n}{2}$, and in particular, it solves ENUM($3, \frac{n}{2}$) in expected 1.598^n time. Compare this with the lower bound of $6^{\frac{n}{4}} \approx 1.565^n$ which follows from $\text{Maj}_{n,3}$. With respect to this algorithm, the complexity of

¹ we emphasize that we require F to have no satisfying assignment, not only no NAE-satisfying assignment of weight less than $t(n)$

² A satisfying assignment of F can be extended to a NAE solution of F' by setting $z = 0$. Conversely, a NAE solution of F' is a satisfying assignment of F projected on the first n variables if $z = 0$, and if $z = 1$, the negation of the remaining variables is a satisfying assignment of F .

ENUM(k, t) increases as t grows. Therefore $t = \frac{n}{2}$ is the hardest instance for this algorithm. Here we refine their algorithm, and show that it optimally solves NAE-ENUM($3, \frac{n}{2}$), which is also the hardest instance for this algorithm.

► **Theorem 1.** *For $n \geq 0, t \leq n/2$, let F be an arbitrary n -variate 3-CNF where every satisfying assignment has Hamming weight at least t . Then, the number of NAE satisfying assignments of F of Hamming weight exactly t is at most $6^{\frac{n}{4}}$. Furthermore, we can enumerate these solutions in expected $\text{poly}(n) \cdot 6^{\frac{n}{4}}$ time.*

Theorem 1 will follow from Theorem 21 and Theorem 38. The algorithm of [5] is a randomized variant of the seminal method of Monien and Speckenmeyer [10]: take a positive clause $C = x_1 \vee \dots \vee x_k$ and for a random ordering π of its variables, recursively solve the problem under the restriction $x_{\pi(1)} = \dots = x_{\pi(i-1)} = 0, x_{\pi(i)} = 1$, for every $i \in [k]$. While the analysis in [5] is essentially local (only depends on the information along root-leaf paths), we had to develop a global technique to get the optimal bound where we use information from one subtree to analyze the performance in an entirely different subtree. We believe that the elements of our analysis have the potential to generalize for $k > 3$.

Depth-3 complexity of Majority function

Majority is a natural candidate function for beating the state-of-the-art depth-3 circuit lower bounds [7, 9, 1]. The local enumeration paradigm of [5] gives a promising paradigm to establish this, and their result yields new lower bounds for Σ_3^3 circuits, i.e., OR-AND-OR circuits with bottom fan-in at most 3. Our result can also be interpreted as an optimal lower bound Majority for Σ_3^3 circuits in which every Hamming weight $\frac{n}{2}$ input is a NAE solution of some depth-2 subcircuit.

Hypergraph Turán problems

Following [3, 5, 4], by restricting ourselves to monotone formulas, our result has a natural interpretation for hypergraphs. Let $H = (V, E)$ be a 3-uniform n -vertex hypergraph with no transversal of size less than $\frac{n}{2}$, i.e., any set of vertices that intersects every hyperedge has size at least $\frac{n}{2}$. We show that the number of 2-colorings of H with no monochromatic hyperedge is at most $6^{\frac{n}{4}}$ and give a randomized algorithm that in expected $\text{poly}(n) \cdot 6^{\frac{n}{4}}$ time enumerates them. This bound is tight, by considering the disjoint union of $\frac{n}{4}$ cliques of size 4 (this hypergraph corresponds to the formula $\text{Maj}_{n,3}$).

Combinatorial interpretation

Our result can also be interpreted in a purely combinatorial manner. For instance, let $S(n, t, k)$ be the maximum number of weight t satisfying assignments of a k -CNF that has no solutions of weight less than t . By construction, we can show $S(n, n/2, 3) \geq 6^{n/4}$. It was conjectured in [5] that $S(n, n/2, 3) \leq 6^{n/4}$. We here prove $S(n, n/2, 3) \leq 6^{n/4}$ under an additional assumption that the 3-CNF formula F is negation closed: If α satisfies F , then negation of α also satisfies F . As pointed out in [5, 4], proving strong upper bounds for $S(n, n/2, k)$ for $k > 3$ is a major open problem and doing so for large k would lead to breakthrough circuit lower bounds for depth 3 circuits. The current best bounds were first obtained by [7] who showed $S(n, n/2, k) \leq 2^{n-O(n/k)}$. We also note that to refute SSETH, one not only needs to show strong upper bounds for $S(n, n/2, k)$ but also needs an enumeration algorithm.

1.2 Proof Strategy

As mentioned earlier, our enumeration algorithm is similar to [5] where we select an unsatisfied monotone clause C , randomly order the variables in C and for $1 \leq i \leq 3$, set the first $i - 1$ variables to 0, set variable i to 1 and recurse. This gives rise to a recursion tree (henceforth called transversal tree) where each leaf may correspond to a transversal (it could be a leaf corresponds to a falsified formula). We bound the expected number of leaves that our algorithm visits by carefully ensuring we do not double count leaves that correspond to the same transversal. This is also what [5] did. The key difference here is that in our algorithm, we very carefully choose which clause to develop next: we divide the transversal tree into stages and for each stage, carefully argue certain kinds of clauses must exist and carefully pick those clauses to develop the transversal tree. This is also where we use the not-all-equals assumption to show certain kinds of favorable clauses must exist. With careful accounting, we are able to obtain the tight bound.

2 Transversal trees and the TreeSearch algorithm

In this section we review the TREESEARCH algorithm from [5] which enumerates the solutions of a k -CNF solutions of minimum Hamming weight. We also make some additional observations that allow us to get a tight analysis of the algorithm for NAE-3-SAT.

The TREESEARCH algorithm is for k -SAT and we want algorithms for NAE- k -SAT. Define the *negation-clause* for clause C to be the clause C' whose literals are the negations of the literals of C . The *negation-closure* of a formula F is the formula \hat{F} obtained from F by adding the negation-clause of every clause of F (if it is not already in F). We say that F is negation-closed if $\hat{F} = F$. It is easy to see that the set of NAE solutions for a formula F is exactly the same as the set of SAT-solutions for its negation-closure \hat{F} . So any algorithm that enumerates the minimum-weight satisfying assignments for k -CNF formulas can be used to find the minimum weight NAE-assignments of a k -CNF formula F by applying the algorithm to \hat{F} .

2.1 Transversals and Transversal trees

2.1.1 Important definitions

► **Definition 2** (Transversals). *A set $S \subseteq X$ is a transversal of F if the assignment that sets the variables in S to 1 and the variables in $X \setminus S$ to 0 is a satisfying assignment of F .*

We say S is a *minimal transversal* of F if no subset of S is a transversal. We say that S is a *minimum-size transversal* of F if it has the smallest size over all transversals.

The definition of transversal of a formula can be seen as a natural generalization of the standard notion of *transversal of a hypergraph*, which is a set that has nonempty intersection with every edge. Indeed, if F is a monotone formula (where every clause consists of positive literals) then a transversal of F is exactly a transversal of the hypergraph H consisting of the set of clauses of F .

► **Definition 3** (Transversal number). *For a satisfiable k -CNF F , the transversal number $\tau(F)$ is the cardinality of the minimum-size transversal of F . The set of all minimum-size transversals of F is denoted by $\Gamma(F)$ and the cardinality of $\Gamma(F)$ is denoted by $\#\Gamma(F)$.*

We focus on two related problems: (1) Give an algorithm that enumerates $\Gamma(F)$ and analyze its complexity, and (2) Determine the maximum cardinality of $\Gamma(F)$ among all k -CNF (or NAE- k -CNF) with $\tau(F) = n/2$. Our main technical tool will be *transversal trees*, introduced in [5], and defined below. The definition applies to k -CNF with $\tau(F) = t$ for any k, t .

We need some preliminary definitions. We associate a subset Y of variables to the partial assignment that sets all variables of Y to 1. A clause C is said to be *satisfied by Y* if C contains a positive literal corresponding to a variable from Y .

► **Definition 4** (Simplification of clause / formula). *For a clause C that is not satisfied by Y , the simplification of C by Y , C/Y is the clause obtained by removing occurrences of negations of variables of Y . Similarly, the simplification of F by Y , denoted F/Y , is the formula on $X - Y$ obtained by deleting clauses satisfied by Y , and replacing each remaining clause C by C/Y .*

Note that the underlying set of variables of both F and F/Y is the same - X . Furthermore, the set of satisfying assignments of F with variables in Y set to 1 is in 1-1 correspondence with the set of satisfying assignments of F/Y . We say that F is *falsified by Y* if F/Y contains an empty clause.

We will be considering rooted trees with root r where each edge e is assigned a label $Q(e) \in X$.

► **Definition 5.** *For a node u and descendant node v we have the following definitions:*

- $P(u, v)$ denotes the path from u to v .
- The shoot from u to v , $S(u, v)$, consists of the edges of $P(u, v)$ together with all child edges of nodes on the path other than v .
- $Q(u, v) = \{Q(e) : e \in P(u, v)\}$. We write $Q(v)$ for $Q(r, v)$.
- A clause C is *live at v* provided that C contains no variable in $Q(v)$ (but it may contain the negation of variables in $Q(v)$). For such a clause we write C/v for $C/Q(v)$ and for a formula F , we write F/v for $F/Q(v)$.

2.1.2 Constructing the tree

We now define a process for growing a tree with node and edge labels $Q(e)$ starting from a root r . The leaves of the current tree are referred to as *frontier nodes*. Each frontier node v is examined and is either designated as a leaf (of the final tree) or a non-leaf as follows:

- If v is a node at depth at most $t - 1$ such that F/v contains no empty clause, then v is a non-leaf. It is labeled by a clause C for which C/v is a positive clause³. Node v is expanded to have $|C/v|$ children with the edges labeled by distinct variables in C/v . We denote the label of an edge e by $Q(e)$.
- If F/v contains an empty clause then v is a leaf of the final tree called a *falsified leaf*. For the parent u of such a leaf v , F/u has no empty clause. Since F/v is obtained from F/u by setting $Q(uv)$ to 1, the single literal clause $\neg Q(uv)$ is in F/u . We refer uv as a *falsifying edge*.
- If v is a node at depth t then v is a leaf. If it is not a falsified leaf then v is a *viable leaf*.

A tree constructed according to the above process is a *transversal tree*. The following easy fact (noted in [5]) justifies the name:

³ F/v must contain a positive clause, otherwise $Q(v)$ is a transversal of size less than t , contradicting $\tau(F) = t$.

► **Proposition 6.** *Let F be a formula with $\tau(F) = t$ and let T be a transversal tree for F . For every minimum-size transversal S of F , there is a depth t leaf v of T such that $S = Q(v)$.*

The depth- t leaves can be divided into three types: falsified leaves, viable leaves ℓ for which $Q(\ell)$ is a transversal, and viable leaves ℓ for which $Q(\ell)$ is not a transversal. For a leaf of the third type, $Q(\ell)$ might be a subset of one or more transversals of size larger than t .

► **Remark 7.** Notice that if a leaf u appears before depth t , then u must be a falsified leaf. Unless arising from “effective width 2 clauses”, for our analysis sake, we will continue expanding the tree below u . Since u is already falsified, we assume, without loss of generality, that all possible clauses of width at most 3 are live at u (including the empty clause). We then choose a monotone clause and recursively develop the tree underneath u until we reach depth t . Similarly, at all nodes v underneath u , we pretend this is the case, that all clauses are live at v . We do this since our algorithm will dictate for various nodes which kind of clause must be chosen and developed next. We will argue that certain kinds of clauses exist for such nodes. Hence, to simplify analysis, we will assume all clauses needed are already present.

Note that it is possible that many leaves correspond to the same transversal.

2.2 The TreeSearch algorithm for enumerating minimum-size transversals

From Proposition 6, we can enumerate all minimum-sized transversals of F by constructing a transversal tree for F , traversing the tree, and for each leaf ℓ computing $Q(\ell)$ and testing whether $Q(\ell)$ is a transversal. To make this procedure fully algorithmic we need to specify two things.

The first is a *clause selection strategy* which determines, for each node v , the clause of F that labels v . In general, the clause selection strategy may be randomized, but in this paper we consider only deterministic strategies.

The second thing to be specified is the order of traversal of the tree, which is specified a family $\pi = \{\pi(v) : \text{internal nodes } v \text{ of } T\}$ where $\pi(v)$ is a left-to-right ordering of the variables of C/v and thus of the child edges of v . We use the ordering to determine a depth first search traversal of the tree which upon arrival at each leaf ℓ , outputs $Q(\ell)$ if $Q(\ell)$ is a transversal. We will say that u is to the left of v if it is visited before v in the traversal. The running time of the algorithm is dominated by the size of the tree, which may be as large as $k^{\tau(F)}$.

To speed up the search, [5] described a simple criterion on edges such that in the pruned tree obtained by removing all edges meeting this criterion and the subtrees below, every minimum-size transversal corresponds to a unique leaf,

To formulate this criterion note that if u is an ancestor of v , the edges of shoot $S(u, v)$ are situated in one of three ways with respect to the path $P(u, v)$: to the left of the tree path, to the right of the tree path, or along the tree path.

► **Definition 8 (Superfluous edge).** *An edge (uv) is superfluous if there is an ancestor w of v that has a child edge wz to the left of $P(r, v)$ such that $Q(wz) = Q(uv)$.*

We define the algorithm TREESEARCH on the ordered tree (T, π) to be the depth first search procedure with the following modification: before traversing any edge check whether it is superfluous; if it is then skip that edge (thereby pruning the edge and the subtree below it). This indeed corresponds to setting the corresponding variable to 0.

► **Proposition 9** (Implicit in [5]). *For any transversal tree T of F and any ordering of T , the tree obtained by removing all superfluous edges and the subtrees below them satisfies: every minimum-size transversal S of F has a unique leaf ℓ of the pruned tree for which $Q(\ell) = S$, and this leaf is the leftmost leaf of the original tree for which $Q(\ell) = S$. Therefore TREESEARCH outputs each minimum-size transversals exactly once.*

The time complexity of TREESEARCH is bounded (up to $\text{poly}(n)$ factors) by the number of leaf nodes in the pruned tree. The number of leaves depends on the clause selection strategy and the order π . Our goal is to choose the strategy and the order so that we can prove a good upper bound on the number of leaves.

In [5], the algorithm for 3-SAT was analyzed under a specific clause selection rule by considering a randomly chosen ordering π , in which the children of each node in the transversal tree are ordered independently and uniformly at random. This enabled them to get an upper bound that is non-trivial, though probably not optimal.

In this paper, we use a similar approach to analyze NAE-3-SAT. As in [5] we choose π at random. We combine this with a carefully chosen clause selection strategy, described in Section 3, which leads to an optimal upper bound for enumeration of minimum-sized transversals.

2.3 Pruning under random edge ordering

Let $F = (X, \mathcal{C})$ be a k -CNF. Let $t = \tau(F)$ be the transversal number of F . Let T be a transversal tree for F . As indicated above we consider a random ordering π in which the child edges of each node are randomly ordered from left-right independent of other nodes. Under this distribution, whether or not a particular edge is superfluous is a random event.

► **Definition 10** (Survival and survival probability).

- *An edge uv is said to survive provided that v is not a falsified leaf v and uv is not superfluous. We say that path $P(u, v)$ survives if every edge of the path survives, and node v survives if path $P(r, v)$ survives.*
- *The conditional survival probability of an edge uv , $\sigma(uv)$ is defined to be $\mathbf{P}[v \text{ survives} \mid u \text{ survives}]$. More generally the conditional survival probability $\sigma(u, v)$ of path $P(u, v)$ is defined to be $\mathbf{P}[P(u, v) \text{ survives} \mid u \text{ survives}]$. It follows that $\sigma(u, v) = \prod_{e \in P(u, v)} \sigma(e)$.*
- *For a node u , let $L(u)$ be the number of leaves of the subtree $T(u)$ that survive, and define the survival value of u , $\psi(u)$, to be the conditional expectation $\mathbb{E}[L(u) \mid u \text{ survives}]$. It follows from linearity of expectation that $\psi(u) = \sum_{v \text{ is leaf of } T(u)} \sigma(u, v)$.*

► **Proposition 11.** *Let F be a 3-CNF with n variables and $\tau(F) = n/2$. Fix a clause selection rule for building transversal trees. Then the expected running time of TREESEARCH on a randomly selected order is $\psi(r)\text{poly}(n)$ and the number of transversals of F satisfies $\#\Gamma(F) \leq \psi(r)$.*

Proof. For the first statement, the running time of TREESEARCH for a given π is at most $L(r)\text{poly}(n)$ and so the expected running time on a random order is at most $\psi(r)\text{poly}(n)$. For the second statement, for any order π , Proposition 9 implies that $\#\Gamma(F) \leq L(r)$, and so $\#\Gamma(F) \leq \psi(r)$. ◀

In the sections that follow, we will describe a clause specification strategy and analyze $\psi(r)$ for the case of negation-closed 3-CNF formula (which as noted earlier, corresponds to the case of NAE-3-SAT). For the analysis we will need some additional observations.

First, we determine a condition under which we can exactly determine $\sigma(uv)$ for an edge uv . It follows from the definition of survival probability if uv is a falsifying edge then $\sigma(uv) = 0$. So we consider the case that uv is not falsifying. None of the edges on the path $P(r, u)$ are falsified (since the child node of a falsifying edge is always a leaf).

If uv is not falsifying then it survives if and only if it is not superfluous. The condition for being superfluous depends on the label of uv appearing as the label of a child edge of an ancestor of u . To account for this we use a system of marking of edges and vertices:

► **Definition 12** (Marking of edges and vertices).

- The marking set $M(e)$ of a tree edge $e = (u, v)$ is the set of nodes $w \neq u$ in the path $P(r, u)$ which have a child edge e' such that $Q(e) = Q(e')$. For $w \in M(e)$ we say that w marks the edge e and w marks the label $Q(e)$.
- An edge e is marked provided that $M(e) \neq \emptyset$.
- A node u is i -marked (for $i \in \{0, 1, 2, 3\}$) if exactly i child edges of u are marked.

Marked edges are edges that have a non-zero probability of being superfluous. The analysis in [5] uses this to obtain upper bounds on the survival probability of nodes. We now show that under favorable conditions we can calculate the survival probability of nodes exactly.

The event that edge uv survives is exactly the event that for every node $w \in M(uv)$, the child edge of w with the same label as uv is to the right of the path edge that comes from w , which happens with probability $1/2$. Thus whether uv survives is determined by the orderings $\pi(w)$ of the child edges of w for all $w \in M(uv)$, and $\mathbf{P}[uv \text{ survives}] = 2^{-|M(e)|}$.

The above computation gives the *unconditioned* probability that uv survives, but $\sigma(uv) = \mathbf{P}[uv \text{ survives} \mid u \text{ survives}]$ is a conditional probability. We now identify a condition under which the conditioning event is independent of the event that uv survives. Say that an edge uv is *disjointly marked* if its marking set $M(uv)$ is disjoint from $M(e)$ for every edge $e \in P(r, u)$. For a disjointly marked edge, the event that uv survives is independent of the event that u survives, i.e., that all edges on $P(r, u)$ survive, because the latter event is determined by the order $\pi(w)$ for nodes that mark some edge of $P(r, u)$ and since uv is disjointly marked, this set of nodes is disjoint from $M(uv)$. Since the order of child edges of nodes are chosen independently, the event that uv survives is independent of the event that all edges on $P(r, u)$ survive. From this we conclude:

► **Proposition 13.** *Let T be a transversal tree for the k -CNF F .*

1. *Let uv be an edge that is not a falsifying edge and is disjointly marked. Then $\sigma(uv) = 2^{-|M(e)|}$.*
2. *If v is a leaf that is not falsifying and each edge on $P(r, v)$ is disjointly marked then $\sigma(v) = 2^{-\sum_{e \in P(r, v)} |M(e)|}$.*

For the case of negation-closed 3-CNF (which corresponds to NAE 3-SAT) we note the following:

► **Proposition 14.** *If F is a negation-closed 3-CNF formula then in any transversal tree T , every edge that is not a falsifying edge is disjointly marked. Therefore Proposition 13 applies to every edge and every leaf of T .*

Proof. Let uv be an edge with ancestor edge wz and y is a node that marks both uv and wz . We claim that uv is a falsifying edge. y is necessarily an ancestor of w . Let y' be the child of y on the path to w . Then the clause labeling y is $Q(yy') \vee Q(wz) \vee Q(uv)$. Since F is negation closed, F also contains the clause $(\neg Q(yy')) \vee (\neg Q(wz)) \vee (\neg Q(uv))$. This clause is falsified at the node v since $Q(yy'), Q(wz), Q(uv) \in Q(v)$, and therefore v is a falsifying leaf. ◀

We need a few additional definitions.

► **Definition 15** (Mass of a node). *For a non-leaf node u in a transversal tree, the mass of u is the conditional expectation of the number of surviving children of u given that u survives. By linearity of expectation this is the same as $\sum_{v: \text{child node of } u} \sigma(u, v)$. We also refer to this as the mass of the clause at u .*

► **Definition 16** (Defect). *For node u , with e child edges, let defect of u to be $3 - e$. We similarly define defect of path / shoot as the sum of the defects of all nodes on the path / shoot.*

► **Definition 17** (Weight). *Let $S(u, v)$ be a shoot in T . The weight of $S(u, v)$ is defined as $W(u, v) \stackrel{\text{def}}{=} |\{e \in S : M(e) \neq \emptyset\}| + \text{defect of } S(u, v)$, i.e., the number of marked edges along $S(u, v)$ plus the defect of $S(u, v)$.*

The following fact provides a lower bound on the weight of each root to leaf shoot in T .

► **Fact 18.** Every root to leaf shoot $S(r, u)$ in T has a weight of at least $3t - n$.

Proof. Let f be the defect of $S(r, u)$. Since the depth of T is t , a root to leaf shoot has $3t - f$ edges. Partition the edges of the shoot into classes according to the label of the edge, and note that all edges in a class are at different depths. For each class, the edge of minimum depth is unmarked and all other edges in the class are marked, so the number of unmarked edges is at most n and the number of marked edges is at least $3t - f - n$. Hence, the weight of $S(r, u)$ is at least $(3t - f - n) + f = 3t - n$. ◀

3 Clause Selection Criterion

We here provide a clause selection criterion for transversal trees for 3-CNFs that will provide us with a tight analysis for the number of NAE-solutions and for the complexity of enumerating them. The clause selection is divided into three stages. Each node will be associated with a stage and a corresponding clause will be selected from that stage. The three stages are:

1. *Disjoint stage*
2. *Controlled stage*
3. *Arbitrary stage*

We begin the disjoint stage by selecting a pseudomaximum collection \mathcal{C}^0 of disjoint monotone clauses of width 3 from F (See Remark 20 for what we mean by pseudomaximum collection; on first reading, the readers should pretend as if we selected maximum such collection). Let $t_0 = |\mathcal{C}^0|$. Our further clause selection criterion depends on the value of t_0 :

- If $t_0 \geq \frac{n}{4}$, then we skip the controlled stage and directly enter the arbitrary stage where we select arbitrary monotone clauses for the rest of the TREESEARCH process.
- If $t_0 \leq \frac{n}{4}$, then we enter the controlled stage where we carefully control which clauses to select, argue about existence of certain clauses and our analysis leverages this control. This controlled stage also helps us get a handle on the kinds of clauses we can encounter in the arbitrary stage.

We here describe the disjoint stage further. The arbitrary stage needs no further explanation. We analyze the $t_0 \geq \frac{n}{4}$ case in Section 4. We will explain the controlled stage further in Section 5 and will analyze the $t_0 \leq \frac{n}{4}$ case depending upon it in Section 6.

3.1 The disjoint stage

Let the clauses in \mathcal{C}^0 be arbitrarily ordered as $C_0^0, C_1^0, \dots, C_{t_0-1}^0$. We will develop the transversal tree so that for all $0 \leq i \leq t_0 - 1$, all nodes at level i will be expanded using clause C_i^0 . We can do this because the disjointness of the clauses ensures that C_i^0 is a live clause at every node at level i . We record the useful observation that any clause used to develop a node appearing at level $\ell \geq t_0$ will have at least one marking:

► **Fact 19.** Let u be a node at level $\ell \geq t_0$ and let C be a width 3 clause used to expand at u . Then, u will not be 0-marked, i.e., u will be j -marked for $1 \leq j \leq 3$.

Proof. For u to be 0-marked, C must be a width 3 monotone clause and C must be disjoint from all clauses from \mathcal{C}^0 . However, that contradicts the fact that \mathcal{C}^0 is a maximal collection of disjoint width 3 monotone clauses. ◀

► **Remark 20** (Algorithmic aspect of maximum matching - pseudomaximum matching). Finding a collection \mathcal{C}^0 of maximum size is NP-hard, and may require large exponential overhead. To mitigate this, we implement an auxiliary data structure that maintains a pseudomaximum collection of disjoint clauses. Initially, the data structure will greedily pick a maximal such collection of clauses. Later in the algorithm, we might encounter scenarios where we can find a larger collection of disjoint clauses. In these cases, we will “reset” the algorithm with that larger collection of disjoint clauses. Such a reset can take place at most n times and hence, the overhead is $\text{poly}(n)$. In our analysis, we will make claims that take this for granted and allow them to use the fact that \mathcal{C}^0 has maximum size; for any claim where we use this, if the claim does not hold because \mathcal{C}^0 is not necessarily of maximum size, the proof of the claim actually will supply us with clauses that will help us form a larger disjoint collection of clauses and we will “reset” our algorithm.

4 Bounding ψ when $t_0 \geq n/4$

In this section, we will show that if in a transversal tree, the number of maximally disjoint clauses chosen in the disjoint stage is at least $n/4$, then $\psi(r) \leq 6^{n/4}$ where r is the root of the tree. Formally:

► **Theorem 21.** Let T be a transversal tree developed using the clause selection criterion as laid out in Section 3 with $t_0(T) \geq \frac{n}{4}$. Then, $\psi(r) \leq 6^{\frac{n}{4}}$ where r is the root of T .

We will use the following lemma to prove the theorem:

► **Lemma 22.** Let $0 \leq d \leq \frac{n}{2} - t_0, w \geq 0$ and let u be a node at level $\frac{n}{2} - d$ such that every u to leaf shoot in $T(u)$ has weight at least w . Then, $\psi(u) \leq F(w, d)$ where $F(w, d)$ is defined as follows:

$$F(w, d) = \begin{cases} \left(\frac{5}{2}\right)^{2d-w} 2^{w-d} & \text{if } w \leq 2d \\ 2^{3d-w} \left(\frac{3}{2}\right)^{w-2d} & \text{if } 2d \leq w. \end{cases}$$

Assuming this lemma, our main theorem easily follows:

Proof of Theorem 21. Let $t_0 = \frac{n}{4} + \Delta$ for some $\Delta \geq 0$. Let u be an arbitrary node at depth t_0 . Let $T(u)$ be the sub-tree rooted at u . The tree $T(u)$ has remaining depth $\frac{n}{4} - \Delta$, and minimum weight of every root to leaf shoot is at least $\frac{n}{2}$ since all edges in the disjoint

stage are unmarked. Thus, $T(u)$ satisfies the requirements of Lemma 22, and we infer that $\psi(u) \leq F(n/2, n/4 - \Delta)$. Summing over all $3^{n/4+\Delta}$ nodes at the end of the disjoint stage, we infer that

$$\begin{aligned} \psi(r) &\leq 3^{n/4+\Delta} F(n/2, n/4 - \Delta) \\ &= 3^{n/4+\Delta} 2^{n/4-3\Delta} \left(\frac{3}{2}\right)^{2\Delta} \\ &= 6^{n/4} \left(\frac{27}{32}\right)^\Delta \\ &\leq 6^{n/4} \end{aligned}$$

where the last inequality follows because $\Delta \geq 0$. ◀

5 Clause Selection Criterion: the Controlled Stage

Recall that this stage appears after the disjoint stage and applies to nodes u at level t_0 or larger provided $t_0 \leq \frac{n}{4}$. To analyze this stage we will need to construct the transversal tree more carefully, using an involved clause selection criterion which is described in this section.

We use the structural properties of the tree to obtain our lower bound in Section 6.

In the controlled stage, we will carefully select clauses in a way that imposes useful restrictions on the structure of the tree. After that, we enter the arbitrary stage during which we expand the tree using monotone clauses in an arbitrary order. We will show that the structure of the clauses in the controlled stage imposes useful restrictions on the structure of the clauses that arises in the arbitrary stage which will help us bound the survival probability. Nodes with same number of markings will appear consecutively in the controlled stage.

We will need the following definition:

► **Definition 23.** *A node u with clause C of width w has effective width w' if it has $w - w'$ falsifying edges (as defined in Section 2.1.2) arising out of it.*

Recall from Section 2.1.2 that a falsifying edge leads to a falsifying leaf, i.e. a node v for which F/v contains an empty clause. Therefore the effective width of a node corresponds to the number of children whose subtrees need to be explored.

Our controlled stage is divided into two substages:

κ_1 . 1-marked nodes.

κ_2 . 2-marked nodes corresponding to clauses of effective width 2.

► **Remark 24.** In stage κ_2 , an effective width 2 (monotone) clause C corresponds to a width 3 (monotone) clause where one of the variables, say x , in C will cause a falsifying edge. We will show that there exists a monotone clause C' s.t. $x \in C'$ and remaining two literals in C' are set to 1. Since F is negation-closed, the negation of C' will have both its literals set to 0, simplifying it to the clause $\neg x$. So, any node expanded using C will have the edge labelled with x as a falsifying edge.

We first introduce useful notation regarding the disjoint stage. We then provide a detailed construction of each of the substages in the controlled stage followed by the impact of the controlled stage on the arbitrary stage.

5.1 Additional notation for the disjoint stage

As described in Section 3.1, we select a pseudomaximum size disjoint collection of clauses $C_0^0, C_1^0, \dots, C_{t_0-1}^0$ in this stage. Our analysis in Section 6 will focus on obtaining an upper bound on $\psi(u_0)$ where u_0 is an arbitrary node at level t_0 . We will fix such a node u_0 for the rest of the section. Write $C_i^0 = \{p_i, x_i, x'_i\}$ where the variable p_i is the label of the edge along the path $P(r, u_0)$. Let $V_0 = \{0, 1, \dots, t_0 - 1\}$. For $i \in [t_0]$, let $X_i = \{x_i, x'_i\}$ and let $X = \cup_{i \in [t_0]} X_i$. We will also need the following useful fact:

► **Fact 25.** Let u be a node at level $\ell \geq t_0$ and let C be a width 3 monotone clause used to expand at u . Then, there must exist $i \in V_0$ such that $X_i \cap C \neq \emptyset$.

Proof. Indeed, assume such C existed. Then, C must be disjoint from all clauses in \mathcal{C}^0 . However, this contradicts the fact \mathcal{C}^0 is a pseudomaximum collection of clauses. ◀

5.2 Stage κ_1 : 1-marked nodes

Let F_1 be the set of width 3 monotone clauses that are live at u_0 and have exactly one marked variable at u_0 . We select a pseudomaximum size disjoint collection \mathcal{C}^1 of clauses from F_1 and expand u_0 to a sub-tree of uniform depth $t_1 = |\mathcal{C}^1|$.

► **Remark 26.** Similar to Remark 20, in the algorithmic implementation we will maintain an auxiliary data structure that will maintain the pseudomaximum collection \mathcal{C}^1 . Initially, the collection will be a maximal set of disjoint clauses from F_1 and whenever we come across a claim that uses maximum size property of \mathcal{C}^1 and is violated, we will reset \mathcal{C}^1 to the disjoint clauses supplied by the proof and “reset” this phase of the algorithm. Again, since a reset can happen at most n times per u_0 , this will increase the total runtime of the algorithm by factor of n .

Clauses in F_1 satisfy the following property:

► **Fact 27.** If $C = \{x_i, a, b\}$ and $C' = \{x'_i, c, d\}$ are in F_1 where $i \in V_0$, then $\{a, b\} \cap \{c, d\} \neq \emptyset$.

Proof. If C and C' are disjoint, we can replace the clause C_i^0 in \mathcal{C}^0 with C and C' to obtain a larger size collection violating the pseudomaximum size condition on \mathcal{C}^0 . ◀

Let $V_1 = \{i \mid \text{a variable from } X_i \text{ appears in a clause from } \mathcal{C}^1\}$. Note that $|V_1| = |\mathcal{C}^1| = t_1$. By Fact 25, for every $C \in F_1$, there is a unique $i \in V_0$ such that C contains exactly one of x_i or x'_i with a single marking, and the remaining two variables in C do not appear in any clause from \mathcal{C}^0 . Furthermore, by Fact 27, for each $i \in V_0$, at most one variable from X_i can appear in some clause of \mathcal{C}^1 ; that is, if x_i (x'_i) appears in some clause of \mathcal{C}^1 , then x'_i (x_i) does not appear in any other clause of \mathcal{C}^1 . These two indeed imply that $|V_1| = |\mathcal{C}^1| = t_1$.

We index the clauses in \mathcal{C}^1 using V_1 . So, for $i \in V_1$, we write clause C_i^1 as $\{\tilde{x}_i, y_i, y'_i\}$ where $\tilde{x}_i \in X_i$. For $i \in V_1$, let $Y_i = \{y_i, y'_i\}$. Let $V_B = V_0 - V_1$ and $m_B = |V_B| = t_0 - t_1$.

Before we describe stage κ_2 , we make some careful observations to prepare for it.

5.3 Preparation for Stage κ_2

For a node u at the end of the stage κ_1 , define $F_2(u)$ to be the set of monotone width 3 clauses C that are live at u and have exactly two singly marked variables (so that C has mass 2).

► **Lemma 28.** *Let u be an arbitrary node at the end of stage κ_1 . Let $C \in F_2(u)$ be arbitrary. Then, C must be of one of the following forms:*

1. $C = \{\tilde{x}_i, \tilde{y}_i, z\}$ for some $i \in V_1, \tilde{x}_i \in X_i \setminus C_i^0, \tilde{y}_i \in Y_i$ and where z does not appear in the shoot $S(r, u)$.
2. $C = \{\tilde{x}_i, \tilde{x}_j, z\}$ for some $i \in V_1, j \in V_B, \tilde{x}_i \in X_i \setminus C_i^0, \tilde{x}_j \in X_j$ and z does not appear in the shoot $S(r, u)$.
3. $\{\tilde{x}_i, \tilde{x}_j, z\}$ where $i, j \in V_B, \tilde{x}_i \in X_i, \tilde{x}_j \in X_j$ and z does not appear in the shoot $S(r, u)$.
4. $\{\tilde{x}_i, \tilde{y}_j, z\}$ where $i \in V_B, j \in V_1, \tilde{x}_i \in X_i, \tilde{y}_j \in Y_j$, and z does not appear in the shoot $S(r, u)$.

Proof. Since C has mass 2, there must be $z \in C$ that does not appear in the shoot $S(r, u)$. By Fact 25, there exists $i \in V_0$ such that $X_i \cap C \neq \emptyset$. We take cases on whether $i \in V_1$ or not and whether there exist two variables in C from X .

Case 1. $i \in V_1$. Since $C \in F_2(u)$ and \tilde{x}_i is singly marked, $\tilde{x}_i \notin C_i^1 \cap X_i$.

Assume that the second marked variable in C is from X . By Fact 27, the second marked variable must be from X_j where $j \in V_0$ and $j \neq i$. We claim that $j \in V_B$. Assume for contradiction that $j \in V_1$. Since \tilde{x}_j is singly marked, $\tilde{x}_j \notin C_j^1$. Then C_i^1, C_j^1 and C are disjoint clauses that we can use to replace clauses C_i^0, C_j^0 from \mathcal{C}^0 and obtain a larger set of disjoint clauses, contradicting the fact that \mathcal{C}^0 is a pseudomaximum collection of clauses. Hence, the claim follows.

If the second marked variable in C is not from X , then it must be from Y . Let $\tilde{y}_j \in C$ where $j \in V_1, \tilde{y}_j \in Y_j$. We claim that $i = j$ in this case. Indeed, if not then the clauses C_i^1 and C will be disjoint clauses containing distinct variables from X_i , violating Fact 27. Hence, the claim follows.

Case 2. $i \notin V_1$. As $V_B = V_0 \setminus V_1$, we infer that $i \in V_B$. In this case, if the second marked variable in C is from X , it must be from X_j where $j \in V_B$ (if $j \in V_1$, then we are in the previous case) and the claim follows. Otherwise, the second marked variable in C is from Y and the claim follows as well. ◀

Let u^* denote the unique node at the end of stage κ_1 where the path $P(u_0, u^*)$ consists of only marked edges, each of which is labeled by a variable from X_i for $i \in V_1$. We note a useful relationship between the live clauses at an arbitrary node u at the end of κ_1 and the live clauses at u^* :

► **Fact 29.** For every node u at the end of stage κ_1 , $F_2(u) \subseteq F_2(u^*)$.

Proof. Let $C \in F_2(u)$ be arbitrary. If C is live at u^* , then C must have exactly 2 markings since $S(r, u) = S(r, u^*)$ and the markings only depend on the shoot. Hence, we show that all such C are live at u^* . Since C is live at u , it is also live at u_0 . Hence, if C is not live at u^* , then it must be that C contains one of the variables from $P(u_0, u^*)$. Equivalently, C must have to contain \tilde{x}_i where $i \in V_1$ and $\tilde{x}_i \in C_i^0$. This cannot happen since by Lemma 28, it must be that $\tilde{x}_i \in X_i \setminus C_i^0$. ◀

For simplicity we write $F_2 := F_2(u^*)$. Let $F_{2R} = \{C \in F_2 : \tilde{x}_i \in C \text{ where } i \in V_1\}$. Let $F_{2B} = F_2 \setminus F_{2R}$.

► **Fact 30.** Any maximally disjoint set of clauses from F_{2B} has size at most m_B .

42:14 Local Enumeration: The Not-All-Equal Case

Proof. By choice of F_{2B} , it only contains clauses of type 3 or type 4 as laid out in Lemma 28. Assume that there exists a collection S of more than m_B disjoint clauses from F_{2B} . Let $T = \{C_i^0 : i \in V_1\}$. We see that clauses in S are disjoint from T , and S and T have no clauses in common. However then, $S \cup T$ is a disjoint collection of clauses of size

$$|S| + |T| \geq (m_B + 1) + |V_1| = |V_0| + 1 = t_0 + 1$$

violating the fact that \mathcal{C}^0 is a pseudomaximum collection of clauses. \blacktriangleleft

► **Remark 31.** We use this fact later in Lemma 37. So even though this fact does not algorithmically provide us with clauses to replace \mathcal{C}^0 with, for algorithm's sake, we only care about the maximal collection we encounter from Lemma 37 and there indeed, we can constructively find such a collection if pseudomaximum property is violated.

► **Fact 32.** Let $C \in F_{2R}$ be arbitrary. Let $i \in V_1$ be such that $\tilde{x}_i \in C$. Let u be arbitrary node at the end of stage κ_1 where a variable from X_i appears along the path $P(u_0, u)$. Then C is live at u .

Proof. By Lemma 28, C can only take one of two forms: If C is of the form $(\tilde{x}_i, \tilde{x}_j, z)$ where $j \in V_B$ and z does not appear along the shoot $S(u_0, u)$, then this follows. Otherwise, C must be of the form $(\tilde{x}_i, \tilde{y}_i, z)$ where z is not along the shoot $S(u_0, u)$ and $\tilde{y}_i \in Y_i$. By assumption, a variable from X_i is in the path $P(u_0, u)$. This can only happen at the node corresponding to the clause C_i^1 . As \tilde{y}_i appears exactly once along the shoot $S(u_0, u)$ - at the node corresponding to the clause C_i^1 - we infer that \tilde{y}_i is not along the path $P(u_0, u)$. Hence, the clause C is still live at u . \blacktriangleleft

Let $V_R = \{i \in V_1 \mid \exists C \in F_{2R} \text{ such that } X_i \cap C \neq \emptyset\}$. Let $m_R = |V_R|$. Let $m_I = |V_1 \setminus V_R|$. We have $m_B + m_R + m_I = t_0 = \frac{n}{4} - \Delta$. Fix a pseudomaximum size collection \mathcal{C}'_R of disjoint clauses from F_{2R} . Let $V'_R = \{i \in V_R \mid x'_i \text{ appears in a clause in } \mathcal{C}'_R\}$ and $m'_R = |V'_R| = |\mathcal{C}'_R|$. Observe that $m'_R \leq m_R$.

► **Remark 33.** Similar to Remark 20 and Remark 26, in the algorithmic implementation, we will maintain an auxiliary data structure that will maintain \mathcal{C}'_R to be a pseudomaximum set of disjoint clauses from F_{2R} and whenever we come across a claim that uses this property but is violated, we will reset \mathcal{C}'_R to be the disjoint clauses supplied by the proof and will reset stage κ_1 . Such reset can happen at most n times per u_0 and hence, the runtime of the algorithm can be increased by at most n . We will allow remaining claims in this section to use that \mathcal{C}'_R is a maximum sized collection.

5.4 Stage κ_2 : 2-marked nodes with effective width 2 clauses

Consider a node u at the end of the stage κ_1 . Let $V^{\text{marked}}(u)$ denote the set of marked variables along the root to u path $P(r, u)$. Let $\mathcal{C}'_R(u) = \{C \in \mathcal{C}'_R \mid \exists i \in V^{\text{marked}}(u) \text{ such that } C \cap X_i \neq \emptyset\}$. Let $\ell(u) = |\mathcal{C}'_R(u)| = |V^{\text{marked}}(u) \cap V'_R|$.

In stage κ_2 , we expand using the clauses in $\mathcal{C}'_R(u)$. By Fact 32, these clauses are live at u and disjoint from each other, the expansion will be carried out for exactly $\ell(u)$ levels where each clause in $\mathcal{C}'_R(u)$ will be used to expand one level. We record this fact here:

► **Fact 34.** Let u be a node at the end of stage κ_1 . Then, stage κ_2 underneath u has length $\ell(u) = |\mathcal{C}'_R(u)| = |V^{\text{marked}}(u) \cap V'_R|$

After this expansion, we finish the controlled stage and enter the arbitrary stage. We record the following useful property of nodes in this stage:

► **Lemma 35.** *All nodes in stage κ_2 will have effective width 2 and mass at most $\frac{3}{2}$.*

Proof. Let v be arbitrary node in stage κ_2 developed using clause C where $C \in \mathcal{C}'_R(u)$. Let $i \in V^{\text{marked}}(u)$ be such that $\tilde{x}_i \in C$. As $C_i^0 = (p_i \vee x_i \vee x'_i)$ is a clause in \mathcal{C}^0 , and F is negation-closed, the negation-clause of C_i^0 is also in F . At v , the negation-clause of C_i^0 simplifies to the unit clause $\neg\tilde{x}_i$. So, at v , the edge with label \tilde{x}_i will be a falsifying edge, making its effective width equal to 2. Moreover, since v is 2-marked, some other edge from C is also marked. This implies the mass of v will indeed be at most $\frac{3}{2}$ as desired. ◀

5.5 Arbitrary Stage

Let u be arbitrary node at the end of the controlled stage. In the controlled stage, we expand using any monotone clause that is available and put no restrictions. However, we will still be able to argue regarding the kinds of clauses that one could encounter in this stage.

We begin by showing that every 1-marked vertex in this stage will have mass at most $\frac{9}{4}$:

► **Lemma 36.** *Every 1-marked node in arbitrary stage has mass at most $\frac{9}{4}$.*

Proof. Indeed, if a 1-marked vertex v with clause C in arbitrary stage has a larger mass, then it must have mass $\frac{5}{2}$. In that case, there exists a unique $i \in V_0$ such that C contains exactly one element from X_i . Since mass of v is $\frac{5}{2}$ and $i \notin V_1$, the remaining two variables do not appear in the shoot $S(r, u)$. However, this implies C is disjoint from \mathcal{C}^1 , contradicting the fact that \mathcal{C}^1 is a pseudomaximum disjoint set of clauses with single marking. ◀

► **Lemma 37.** *For any root to leaf shoot S in $T(u)$, the number of nodes with 2 marked edges and mass 2 is at most $m'_R + m_B - \ell(u)$.*

Proof. Call such clauses as *heavy clauses*. By Fact 29, heavy clauses along any shoot during arbitrary stage are all in F_2 . Observe that these heavy clauses must be disjoint from $\mathcal{C}'_R(u)$. We first claim that there can be at most $m'_R - \ell(u)$ heavy clauses from F_{2R} during arbitrary stage along S . Indeed, if there were more, then these heavy clauses combined with $\mathcal{C}'_R(u)$ would form a collection of disjoint clauses of size at least $(m'_R - \ell(u) + 1) + \ell(u) = m'_R + 1$, violating the fact that \mathcal{C}'_R is a pseudomaximum collection of disjoint clauses.

The only other heavy clauses that can occur in arbitrary stage are from the set F_{2B} . By Fact 30, there are at most m_B disjoint clauses in F_{2B} . Since heavy clauses along a shoot must be disjoint from each other, there can be at most $m'_R + m_B - \ell(u)$ heavy clauses in S . ◀

6 Bounding ψ when $t_0 \leq \frac{n}{4}$

In this section, we show that if in a transversal tree T , the number of maximally disjoint clauses chosen in the disjoint stage is at most $n/4$, then $\psi(r) \leq 6^{n/4}$ where r is the root of T . Formally, our main theorem is:

► **Theorem 38.** *Let T be a transversal tree developed using the clause selection criteria as laid out in Section 3 and Section 5 with $t_0 \leq \frac{n}{4}$. Then, $\psi(r) \leq 6^{n/4}$ where r is the root of T .*

To bound $\psi(r)$, we will carefully count and sum up the survival values of all nodes that appear at the end of the disjoint stage. To facilitate that, we need a handle on the survival value of a subtree that is developed in arbitrary stage. We introduce the following quantity to help us with that:

42:16 Local Enumeration: The Not-All-Equal Case

Let $M(w, d, h) = \max_u \psi(u)$ where u is a node at depth $\frac{n}{2} - d$ in arbitrary stage, every root to leaf shoot in $T(u)$ has weight at least w , and for every root to leaf shoot, the number of 2-marked nodes with mass 2 is bounded by h .

As u is a node at level $\frac{n}{2} - d$, the depth of $T(u)$ is d . Moreover, as u is in arbitrary stage, every node in T has at least one marked edge coming out of it, and by Lemma 36, every 1-marked node has that marked edge with survival probability at most $1/4$.

We will define the following useful function:

$$F(w, d, h) = \begin{cases} \left(\frac{9}{4}\right)^d & \text{if } w \leq d \\ \left(\frac{9}{4}\right)^{2d-w} 2^{w-d} & \text{if } d \leq w \leq d+h \\ \left(\frac{9}{4}\right)^{2d-w} 2^h \left(\frac{27}{8}\right)^{(w-d-h)/2} = 2^h \left(\frac{27}{8}\right)^{(3d-w-h)/2} \left(\frac{3}{2}\right)^{w-2d} & \text{if } d+h \leq w \leq 3d-h \\ 2^{3d-w} \left(\frac{3}{2}\right)^{w-2d} & \text{if } 3d-h \leq w \end{cases}$$

We will use the following bound on $M(w, d, h)$ that we prove in the appendix of the full version.

► **Lemma 39.** For all w, d, h : $M(w, d, h) \leq F(w, d, h)$.

With this, we are ready to prove Theorem 38:

Proof of Theorem 38. Let $u_0 \in T$ be an arbitrary node at depth t_0 . Then, we can associate quantities $m_B(u_0), m'_R(u_0), t_1(u_0)$ with the subtree $T(u_0)$. We bound $\psi(u_0)$ in terms of these quantities and function F from Lemma 39. We will sum over $\psi(u)$ where u is a node at the end of controlled stage and then use F to bound $\psi(u)$ as u will be at the beginning of arbitrary stage. We will also need to precisely compute $\sigma(u_0, u)$, for such u and for that, we keep track of how many marked edges (say i) from stage κ_1 corresponding to $m'_R(u_0)$ are on the path from u_0 to u . This i will also be the length of stage κ_2 , which will further help us in bounding $\sigma(u_0, u)$.

To do that, we first introduce the following quantities that we will use in the upper bound: Let $w(u_0, i) = \frac{n}{2} - 2i - t_1(u_0)$, $d(u_0, i) = \frac{n}{2} - t_0 - t_1(u_0) - i$, $h(u_0, i) = m'_R(u_0) + m_B(u_0) - i$. Using these quantities, we will upper bound $\psi(u_0)$ using the following expression: Define

$$N(u_0) = \left(\frac{5}{2}\right)^{t_1(u_0)} \left(\frac{4}{5}\right)^{m'_R(u_0)} \sum_{i=0}^{m'_R(u_0)} \binom{m'_R(u_0)}{i} \left(\frac{3}{8}\right)^i \cdot F(w(u_0, i), d(u_0, i), h(u_0, i))$$

Using this function, we will obtain the following as our main lemma:

► **Lemma 40.** $\psi(u_0) \leq N(u_0)$.

We assume this bound holds and continue our proof of Theorem 38. We will prove the lemma in the appendix of the full version of the paper.

As $N(u_0)$ depends on F , we want to figure out what case for the function of F applies. Recall that this depends on the relationship between $w(u_0, i), d(u_0, i), h(u_0, i)$. To help with this, we introduce the following function:

$$I(u_0) = 3t_0 + 2t_1(u_0) + m'_R(u_0) + m_B(u_0)$$

We show that the values of I and n govern which of the 4 functions will F equal. This is surprising since I is a function of u_0 and not a function of i . We first show that only 2 function choices of F can arise. We do this by showing:

▷ **Claim 41.** $d(u_0, i) + h(u_0, i) \leq w(u_0, i)$.

Proof of Claim 41. Indeed we compute:

$$\begin{aligned} d(u_0, i) + h(u_0, i) &= \frac{n}{2} - t_0 - t_1(u_0) + m'_R(u_0) + m_B(u_0) - 2i \\ &= w(u_0, i) + (m'_R(u_0) + m_B(u_0) - t_0) \\ &\leq w(u_0, i) \end{aligned}$$

where the last inequality follows because $m'_R(u_0) + m_B(u_0) \leq t_1(u_0) + m_B(u_0) = t_0$. \triangleleft

We next show that the value of I decides the choice function for F :

\triangleright **Claim 42.** $w(u_0, i) \leq 3d(u_0, i) - h(u_0, i)$ if and only if $I(u_0) \leq n$.

Proof of Claim 42. We compute the following:

$$\begin{aligned} 3d(u_0, i) - h(u_0, i) &= \frac{3n}{2} - 3t_0 - 3t_1(u_0) - m'_R(u_0) - m_B(u_0) - 2i \\ &= w(u_0, i) + n - 3t_0 - 2t_1(u_0) - m'_R(u_0) - m_B(u_0) \end{aligned}$$

and hence,

$$w(u_0, i) \leq 3d(u_0, i) - h(u_0, i) \iff 3t_0 + 2t_1(u_0) + m'_R(u_0) + m_B(u_0) = I(u_0) \leq n \quad \triangleleft$$

With this, we bound $\psi(r)$ as follows:

$$\begin{aligned} \psi(r) &= \sum_{u_0:u_0 \text{ at depth } t_0} \psi(u_0) \\ &\leq \sum_{u_0:u_0 \text{ at depth } t_0} N(u_0) \\ &= \sum_{u_0:u_0 \text{ at depth } t_0, I(u_0) \leq n} N(u_0) + \sum_{u_0:u_0 \text{ at depth } t_0, I(u_0) > n} N(u_0) \\ &\leq 3^{t_0} \max \left(\max_{u_0: I(u_0) \leq n} N(u_0), \max_{u_0: I(u_0) > n} N(u_0) \right) \\ &= \max \left(\max_{u_0: I(u_0) \leq n} 3^{t_0} \cdot N(u_0), \max_{u_0: I(u_0) \geq n} 3^{t_0} \cdot N(u_0) \right) \end{aligned}$$

We show that the inner quantities are maximized when $I(u_0) = n$ by the following two claims. These claims just rely on the inequalities listed in the claim, proving that some expression is bounded. We see these claims as solving an optimization problem and do not rely on any properties of the transversal tree.

\triangleright **Claim 43.** Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0$, $m'_R(u_0) \leq t_1(u_0)$, $I(u_0) \leq n$. Then, $3^{t_0} \cdot N(u_0)$ is maximised when $I(u_0) = n$.

\triangleright **Claim 44.** Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0$, $m'_R(u_0) \leq t_1(u_0)$, $I(u_0) \geq n$. Then, $3^{t_0} \cdot N(u_0)$ is maximised when $I(u_0) = n$.

Lastly, we show that when $I(u_0) = n$, then the inner quantity is bounded by $6^{n/4}$:

\triangleright **Claim 45.** Let u_0 be such that $m_B(u_0) + t_1(u_0) \leq t_0$, $m'_R(u_0) \leq t_1(u_0)$, $I(u_0) = n$. Then, $3^{t_0} \cdot N(u_0) \leq 6^{n/4}$.

These 3 claims together indeed show that $\psi(r) \leq 6^{n/4}$ as desired. We defer the proofs of all these claims to the appendix of the full version of the paper. \blacktriangleleft

7 Conclusion


We gave an optimal algorithm for the Not-All-Equal variant of $\text{ENUM}(k, \frac{n}{2})$ for $k = 3$. Extending the analysis of our algorithm to large k would break SETH. However, extending this to even $k = 4$ poses a great challenge.

References

- 1 Kazuyuki Amano. Depth-three circuits for inner product and majority functions. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPICs*, pages 7:1–7:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ISAAC.2023.7.
- 2 Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon M. Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2 - 2/(k + 1))^n$ algorithm for k -SAT based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002. doi:10.1016/S0304-3975(01)00174-8.
- 3 Peter Frankl, Svyatoslav Gryaznov, and Navid Talebanfard. A variant of the VC-dimension with applications to depth-3 circuits. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPICs*, pages 72:1–72:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.72.
- 4 Mohit Gurumukhani, Marvin Künnemann, and Ramamohan Paturi. On extremal properties of k -cnf: Capturing threshold functions. *CoRR*, abs/2412.20493, 2024. arXiv:2412.20493.
- 5 Mohit Gurumukhani, Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Navid Talebanfard. Local enumeration and majority lower bounds. In Rahul Santhanam, editor, *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPICs*, pages 17:1–17:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.CCC.2024.17.
- 6 Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k -SAT algorithms using biased-PPSZ. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 578–589. ACM, 2019. doi:10.1145/3313276.3316359.
- 7 Johan Håstad, Stasys Jukna, and Pavel Pudlák. Top-down lower bounds for depth-three circuits. *Comput. Complex.*, 5(2):99–112, 1995. doi:10.1007/BF01268140.
- 8 Timon Hertli. Breaking the PPSZ barrier for unique 3-SAT. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 600–611. Springer, 2014. doi:10.1007/978-3-662-43948-7_50.
- 9 Victor Lecomte, Prasanna Ramakrishnan, and Li-Yang Tan. The composition complexity of majority. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPICs*, pages 19:1–19:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CCC.2022.19.
- 10 Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.*, 10(3):287–295, 1985. doi:10.1016/0166-218X(85)90050-2.
- 11 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k -SAT. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.
- 12 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chic. J. Theor. Comput. Sci.*, 1999, 1999. URL: <http://cjtcs.cs.uchicago.edu/articles/1999/11/contents.html>.

- 13 Dominik Scheder. PPSZ is better than you think. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 205–216. IEEE, 2021. doi:10.1109/FOCS52979.2021.00028.
- 14 Dominik Scheder and Navid Talebanfard. Super strong ETH is true for PPSZ with small resolution width. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 3:1–3:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.3.
- 15 Uwe Schöning. A probabilistic algorithm for k -SAT based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002. doi:10.1007/s00453-001-0094-7.
- 16 Nikhil Vyas and R. Ryan Williams. On super strong ETH. *J. Artif. Intell. Res.*, 70:473–495, 2021. doi:10.1613/JAIR.1.11859.

Approximating Densest Subgraph in Geometric Intersection Graphs

Sariel Har-Peled ✉ 

Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Saladi Rahul ✉ 

Indian Institute of Science (IISc), Bangalore, India

Abstract

For an undirected graph $G = (V, E)$, with n vertices and m edges, the *densest subgraph* problem, is to compute a subset $S \subseteq V$ which maximizes the ratio $|E_S|/|S|$, where $E_S \subseteq E$ is the set of all edges of G with endpoints in S . The densest subgraph problem is a well studied problem in computer science. Existing exact and approximation algorithms for computing the densest subgraph require $\Omega(m)$ time. We present near-linear time (in n) approximation algorithms for the densest subgraph problem on *implicit* geometric intersection graphs, where the vertices are explicitly given but not the edges. As a concrete example, we consider n disks in the plane with arbitrary radii and present two different approximation algorithms.

As a by-product, we show a reduction from (shallow) range-reporting to approximate counting/sampling which seems to be new and is useful for other problems such as independent query sampling.

2012 ACM Subject Classification Theory of computation \rightarrow Computational geometry

Keywords and phrases Geometric intersection graphs, Densest subgraph, Range searching, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.43

Related Version *Full Version*: <https://www.arxiv.org/abs/2405.18337>

Funding Research of Saladi Rahul is funded by Walmart Center for Tech Excellence.

1 Introduction

Given an undirected graph $G = (V, E)$, the *density* of a set $S \subseteq V(G)$ is $|E_S|/|S|$, where each edge in $E_S \subseteq E(G)$ has both its vertices in S . In the *densest subgraph problem*, the goal is to find a subset of $V(G)$ with the maximum density. Computing the densest subgraph is a primitive operation in large-scale graph processing, and has found applications in mining closely-knit communities [12], link-spam detection [18], and reachability and distance queries [14]. See [7] for a detailed discussion on the applications of densest subgraph, and [22] for a survey on the recent developments on densest subgraph.

Exact algorithms for densest subgraph. Unlike the (related) problem of computing the largest clique (which is NP-HARD), the densest subgraph can be computed (exactly) in polynomial time. Goldberg [19] show how to reduce the problem to $O(\log n)$ instances of $s-t$ min-cut problem. Gallo et al. [17] improved the running time slightly by using parametric max flow computation. Charikar [10] presented an LP based solution to solve the problem, for which Khuller and Saha [21] gave a simpler rounding scheme.

Approximation algorithms for densest subgraph. The exact algorithms described above require solving either an LP or an $s-t$ min-cut instance, both of which are relatively expensive to compute. To obtain a faster algorithm, Charikar [10] analyzed a 2-approximation algorithm



© Sariel Har-Peled and Saladi Rahul;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 43; pp. 43:1–43:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

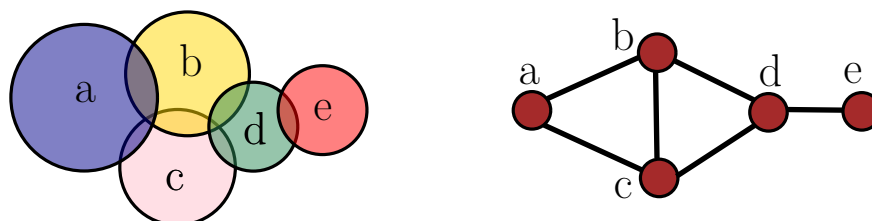


which repeatedly removes the vertex with the smallest degree and calculates the density of the remaining graph. This algorithm runs in linear time. After that, Bahmani et al. [6] used the primal-dual framework to give a $(1 + \varepsilon)$ -approximation algorithm which runs in $O((m/\varepsilon^2) \log n)$ time. The problem was also studied in the streaming model the focus is on approximation using bounded space. McGregor et al. [24] and Esfandiari et al. [16] presented a $(1 + \varepsilon)$ -approximation streaming algorithm using roughly $\tilde{O}(|V|)$ space. There has been recent interest in designing *dynamic* algorithms for approximate densest subgraph, where an edge gets inserted or deleted in each time step [8, 26, 15].

Geometric intersection graphs. The geometric intersection graph of a set \mathcal{D} of n objects is a graph $G = (V, E)$, where each object in \mathcal{D} corresponds to a unique vertex in V , and an edge exists between two vertices if and only if the corresponding objects intersect. Further, in an *implicit geometric intersection graph*, the input is only the set of objects \mathcal{D} and *not* the edge set E , whose size could potentially be quadratic in terms of $|V|$.

Unlike general graphs, the geometric intersection graphs typically have more structure, and as such computing faster approximation algorithms [4, 11], or obtaining better approximation algorithms on implicit geometric intersection graphs has been an active field of research.

One problem closely related to the densest subgraph problem is that of finding the maximum clique. Unlike the densest subgraph problem, the maximum clique problem is NP-hard for various geometric intersection graphs as well (for e.g., segment intersection graphs [9]). However, for the case of unit-disk graphs, an elegant polynomial time solution by Clark et al. [13] is known.



■ **Figure 1.1** Five disks and their corresponding graph. The densest subgraph is $\{a, b, c, d\}$ with density $5/4$.

Motivation. Densest subgraph computation on geometric intersection graphs can help detect regions which have strong cellular network coverage, or have many polluting factories. The region covered (resp., polluted) by cellular towers (resp., or factories) can be represented by disks of varying radii.

1.1 Our results

Here, we study the densest subgraph problem on (implicit) geometric intersection graphs, and present near-linear time (in terms of $|V|$) approximation algorithms.

From reporting to approximate counting/sampling. We show a reduction from (shallow) range-reporting to approximate counting/sampling: given a set of objects \mathcal{D} , and a reporting data-structure for \mathcal{D} , we show a reduction to building a data-structure, such that given a query object q , it returns approximately-at-uniform an object from $q \cap \mathcal{D} = \{x \in \mathcal{D} \mid x \cap q \neq \emptyset\}$, and also returns an $(1 \pm \varepsilon)$ -approximation for the size of this set. Previous work on closely

Approximation	Running time	Ref
$2 + \varepsilon$	$O\left(\frac{n \log n}{\varepsilon^4}\right)$	Theorem 19
$1 + \varepsilon$	$O\left(\frac{n \log^2 n}{\varepsilon^2} \left(\frac{1}{\varepsilon^2} + \log \log n\right)\right)$	Theorem 29

■ **Figure 1.2** Our results.

related problem includes the work by Afshani and Chan [1] (that uses shallow counting queries), the work by Afshani et al. [2], and the work by Afshani and Philips [3]. For our application, this data-structure enables us to sample $(1 \pm \varepsilon)$ -uniformly a disk from the set of disks intersecting a given disk. See Section 3 and Theorem 12 for details.

The reduction seems to be new, and should be useful for other problems. For example, in databases, motivated by summarizing large query output size, and to incorporate fairness and diversity in the output results, *independent query sampling (IQS)* has received quite a bit of attention. In IQS the goal is to report a uniform sample of the query output (i.e., of $\mathbf{q} \cap \mathcal{D}$). Crucially, the query output should be *independent* of all the previous queries posed (for example, if the same query is posed again, then the sample reported should be independent of the previous sample). See the survey by Tao [27] on IQS (and the references within). Our reduction addresses two of the future directions proposed in the survey: (a) designing a *generic* IQS data structure which works for a broad class of geometric queries, and (b) *approximate IQS* where the probability of reporting an object is almost-uniform, which is usually good enough in practice.

The application. For the sake of concreteness, we consider the case of n disks (with arbitrary radii) lying in the plane and present two different approximation algorithms. See Figure 1.1. However, our algorithms would work for other shapes with minor modifications.

A $(2 + \varepsilon)$ -approximation. Our first approximation algorithm uses the greedy strategy of removing disks of low-degree from the intersection graph. By batching the queries, and using the above data-structure, we get a $(2 + \varepsilon)$ -approximation for the densest subset of disks in time $O_\varepsilon(n \log n)$, where O_ε hides constants polynomial in $1/\varepsilon$. Getting this running time requires some additional ideas such as establishing densest subgraph's connection with deepest point in the arrangement of disks (defined later). The running time is optimal in terms of n in the comparison-based model, see Remark 20.

A $(1 + \varepsilon)$ -approximation. A more promising approach is to randomly sample edges from the intersection graph, and then apply known approximation algorithms. This requires some additional work since unlike previous work, we can only sample approximately in uniform. See Section 5 for details. The running time of the new algorithm is $O_\varepsilon(n \log^2 n \log \log n)$ which is slower than the first $(2 + \varepsilon)$ -approximation algorithm. The results are summarized in Figure 1.2.

2 Preliminaries

2.1 Definitions

In the following, \mathcal{D} denotes a (given) set of n objects (i.e., disks). For $S \subseteq \mathcal{D}$ and $u \in \mathcal{D}$, we use the shorthands $S + u = S \cup \{u\}$ and $S - u = S \setminus \{u\}$. For $\varepsilon \in (0, 1)$ and a real number $\alpha > 0$, let $(1 \pm \varepsilon)\alpha$ denote the interval $((1 - \varepsilon)\alpha, (1 + \varepsilon)\alpha)$. Throughout, a statement holds *with high probability*, if it holds with probability at least $1 - n^{-c}$, where c is a sufficiently large constant.

43:4 Approximating Densest Subgraph in Geometric Intersection Graphs

► **Observation 1.**

- (I) For any ε , we have $\frac{1}{1+\varepsilon} \geq 1 - \varepsilon$.
- (II) For $\varepsilon \in (0, 1/2)$, we have $\frac{1}{1\pm\varepsilon} = (1/(1+\varepsilon), 1/(1-\varepsilon)) \subseteq 1 \pm 2\varepsilon$ since $1 - \varepsilon \leq \frac{1}{1+\varepsilon} \leq \frac{1}{1-\varepsilon} \leq 1 + 2\varepsilon$.
- (III) For $\varepsilon \in (0, 1/3)$, we have $(1 \pm \varepsilon)^2 \subseteq (1 \pm 3\varepsilon)$.
- (IV) For $\varepsilon \in (0, 1)$ and constants c, c_1 and c_2 , such that $c \geq c_1 c_2$, we have $(1 \pm c_1 \varepsilon / c)(1 \pm c_2 \varepsilon / c) \subseteq 1 \pm \frac{c_1 + c_2 + 1}{c} \varepsilon^1$.

► **Definition 2.** Given a set of objects \mathcal{D} (say in \mathbb{R}^d), their **intersection graph** has an edge between two objects if and only if they intersect. Formally,

$$G_{\cap \mathcal{D}} = (\mathcal{D}, \{uv \mid u, v \in \mathcal{D}, \text{ and } u \cap v \neq \emptyset\}).$$

► **Definition 3.** For a graph G , and a subset $S \subseteq V(G)$, let

$$E_S = E_S(G) = \{uv \in E(G) \mid u, v \in S\}.$$

The **induced subgraph** of G over S is $G_S = (S, E_S)$, and let $m(S) = |E_S|$ denote the number of edges in this subgraph.

► **Definition 4.** For a set $S \subseteq V(G)$, its **density** in G is $\nabla(S) = \nabla_G(S) = m(G_S) / |S|$, where $m(G_S)$ is the number of edges in G_S . Similarly, for a set of objects \mathcal{D} , and a subset $S \subseteq \mathcal{D}$, the **density** of S is $\nabla(S) = m(G_{\cap S}) / |S|$.

► **Definition 5.** For a graph G , its **max density** is the quantity $d(G) = \max_{S \subseteq V(G)} \nabla(S)$, and analogously, for a set of objects \mathcal{D} , its **max density** is $d(\mathcal{D}) = \max_{S \subseteq \mathcal{D}} \nabla(S)$,

The problem at hand is to compute (or approximate) the maximum density of a set of objects \mathcal{D} . If a subset S realizes this quantity, then it is the **densest subset** of \mathcal{D} (i.e., $(G_{\cap \mathcal{D}})_S$ is the densest subgraph of $G_{\cap \mathcal{D}}$). One can make the densest subset unique, if there are several candidates, by asking for the lexicographic minimal set realizing the maximum density. For simplicity of exposition we treat the densest subset as being unique.

► **Lemma 6.** Let $\mathcal{O} \subseteq \mathcal{D}$ be the densest subset, and let $\nabla = \nabla(\mathcal{O})$. For $u \in \mathcal{O}$, let $d_{\mathcal{O}}(u) = |u \cap (\mathcal{O} - u)|$, where $u \cap (\mathcal{O} - u) = \{x \in \mathcal{O} - u \mid x \cap u \neq \emptyset\}$. Then, for all $u \in \mathcal{O}$, we have $d_{\mathcal{O}}(u) \geq \nabla$,

Proof. Observe that

$$\nabla = \frac{|E_{\mathcal{O}}|}{|\mathcal{O}|} = \frac{|E_{\mathcal{O}-u}| + d_{\mathcal{O}}(u)}{|\mathcal{O}| - 1 + 1}.$$

As such, if $d_{\mathcal{O}}(u) < \nabla$, then

$$\frac{d_{\mathcal{O}}(u)}{1} < \nabla = \frac{d_{\mathcal{O}}(u) + |E_{\mathcal{O}-u}|}{1 + |\mathcal{O}| - 1} < \frac{|E_{\mathcal{O}-u}|}{|\mathcal{O}| - 1}.$$

But this implies that $\mathcal{O} - u$ is denser than \mathcal{O} , which is a contradiction. ◀

¹ Indeed, $(1 + c_1 \varepsilon / c)(1 + c_2 \varepsilon / c) \leq 1 + (c_1 / c + c_2 / c + c_1 c_2 / c^2) \varepsilon \leq 1 + (c_1 + c_2 + 1) \varepsilon / c$.

2.2 Reporting all intersecting pairs of disks

The algorithm of Section 5 requires an efficient algorithm to report all the intersecting pairs of disks.

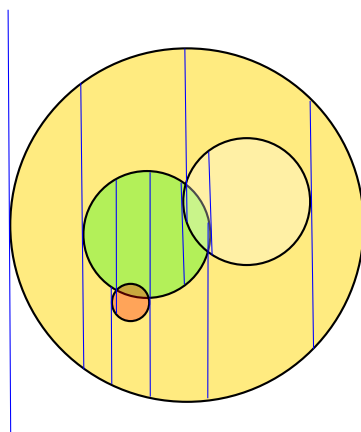
► **Lemma 7.** *Given a set \mathcal{D} of n disks, all the intersecting pairs of disks of \mathcal{D} can be computed in $O(n \log n + k)$ expected time, where k is the number of intersecting pairs.*

Proof. We break the boundary of each disk at its two x -extreme points, resulting in a set of $2n$ x -monotone curves. Computing the vertical decomposition of the arrangement of these disks (curves) $\mathcal{A}(\mathcal{D})$ can be done in $O(n \log n + k)$ expected time [20]. See Figure 2.1 for an example. This gives us readily all the pairs that their boundaries intersect.

As for the intersections that rise out of containment, perform a traversal of the dual graph of the vertical decomposition (i.e., each vertical trapezoid is a vertex, and two trapezoids are adjacent if they share a boundary edge). The dual graph is planar with $O(n + k)$ vertices and edges, and as such the graph can be traversed in $O(n + k)$ time. During the traversal, by appropriate bookkeeping, it is straightforward to maintain the list of disks containing the current trapezoid, in $O(1)$ per edge traversed, as any edge traversed changes this set by at most one.

For a disk d , let p_d be the rightmost point of d . For each disk d , pick any trapezoid Δ such that $p_d \in \Delta$ and either the top boundary of d is the ceiling of Δ or the bottom boundary of d is the floor of Δ . Assign $\Delta_d \leftarrow \Delta$ as the *representative* trapezoid of d .

During the traversal of the dual graph, consider the case where we arrive at a representative trapezoid of a disk d . Let $L(d)$ be the list of disks containing Δ_d . Then scan $L(d)$ to report all the containing pairs of d . Each disk in $L(d)$ either intersects the boundary of d , or contain it. Therefore, the total time spent at the representative trapezoids is $\sum_{d \in \mathcal{D}} |L(d)| = O(k)$. ◀



■ **Figure 2.1** Vertical decomposition of four disks.

3 From reporting to approximate sampling/counting

In this section, given a set of objects \mathcal{D} , and a reporting data-structure for \mathcal{D} , we show a reduction to building a data-structure, such that given a query object q , it returns an object from $q \cap \mathcal{D} = \{x \in \mathcal{D} \mid x \cap q \neq \emptyset\}$, with almost uniform probability, and also returns an $(1 \pm \varepsilon)$ -approximation for the size of this set.

3.1 The data-structure

The given reporting data-structure. Let \mathcal{D} be a set of n objects, and assume for any subset $X \subseteq \mathcal{D}$ of size m , one can construct, in $C(m)$ time, a data-structure that given a query object \mathbf{d} , returns, in $O(Q(m) + k)$ time, all the objects in X that intersects \mathbf{d} , where $k = |\mathbf{d} \cap X|$. Furthermore, we assume that if a parameter k' is specified by the query, then the data-structure stops after $O(Q(m) + k')$ time, if $k > k'$, and indicate that this is the case.

► **Example 8.** If \mathcal{D} is a set of disks, and the query \mathbf{d} is a disk, then this becomes a query reporting of the k -nearest neighbors in an additive weighted Voronoi diagram. Liu [23] showed how to build such a reporting data-structure, in $O(n \log n)$ expected preprocessing time, and query time $O(\log n + k)$.

Data-structure construction. We build a random binary tree over the objects of \mathcal{D} , by assigning each object of \mathcal{D} with equal probability either a 0 or a 1 label. This partitions \mathcal{D} into two sets \mathcal{D}_0 (label 0) and \mathcal{D}_1 (label 1). Recursively build random trees T_0 and T_1 for \mathcal{D}_0 and \mathcal{D}_1 , respectively, with the output tree having T_0 and T_1 as the two children of the root. The constructions bottoms out when the set of objects is of size one. Let \mathcal{T} be the resulting tree that has exactly n leaves. For every node u of \mathcal{T} , we construct the reporting data-structure for the set of objects $\mathcal{D}(u)$ – that is, the set of objects stored in the subtree of u .

Finally, create an array L_i for each level i of the tree \mathcal{T} , containing pointers to all the nodes in this level of the tree.

Answering a query. Given a query object \mathbf{q} , and a parameter $\varepsilon \in (0, 1/2)$, the algorithm starts from an arbitrary leaf v of \mathcal{T} . The leaf v has a unique root to v path, denoted by $\pi = u_0 u_1 \dots u_t$, where $u_0 = \text{root}(\mathcal{T})$ and $u_t = v$. The algorithm performs a binary search on π , using the reporting data-structure associated with each node, to find the maximal i , such that $|\mathbf{q} \cap \mathcal{D}(u_i)| > \psi = c \log n$, where c is a sufficiently large constant. Here, we use the property that one can abort the reporting query if the number of reported objects exceeds ψ . This implies that each query takes $Q(n) + O(\log n)$ time (with no dependency on ε). Next, the algorithm computes the maximal j such that $|\mathbf{q} \cap \mathcal{D}(u_j)| > \psi_\varepsilon = c\varepsilon^{-2} \log n$ (or set $j = 0$ if such a j does not exist). This is done by going up the path π from u_i , trying $u_{i-1}, u_{i-2}, \dots, u_j$ using the reporting data-structure till the condition is fulfilled². Next, the algorithm chooses a vertex $u \in L_j$ uniformly at random. It computes the set $S = \mathbf{q} \cap \mathcal{D}(u)$ using the reporting data-structure. The algorithm then returns a random object from S uniformly at random, and the number $2^j |S|$. The first is a random element chosen from $\mathbf{q} \cap \mathcal{D}$, and the second quantity returned is an estimate for $|\mathbf{q} \cap \mathcal{D}|$.

3.2 Analysis

3.2.1 Correctness

► **Lemma 9.** *Let $\varepsilon \in (0, 1/2)$, $\psi_\varepsilon = c\varepsilon^{-2} \log n$, and \mathbf{q} be a query object. Let $M \geq 1$ be the integer such that $\psi_\varepsilon/16 \leq |\mathcal{D} \cap \mathbf{q}|/2^M < \psi_\varepsilon/8$. Then, for all nodes v at distance $i \leq M$ from the root of \mathcal{T} , we have $\mathbb{P}\left[|\mathcal{D}(v) \cap \mathbf{q}| \notin (1 \pm \varepsilon/2) \frac{|\mathcal{D} \cap \mathbf{q}|}{2^i}\right] \leq \frac{1}{n^{\Omega(c)}}$.*

² One can also “jump” to level $i + \log_2(1/\varepsilon^2) - 2$, and do a local search there for j , but this “improvement” does not effect the performance.

Proof. Consider a node v at a distance i from the root, and let $Y_v = |\mathcal{D}(v) \sqcap \mathbf{q}|$. Clearly, $\mu_v = \mathbb{E}[Y_v] = |\mathcal{D} \sqcap \mathbf{q}| / 2^i$. Since $i \leq M$, we have $\mu_v \geq \psi_\varepsilon / 16$. By Chernoff's inequality, we have

$$\mathbb{P}[Y_v \notin (1 \pm \varepsilon/2)\mu_v] \leq 2 \exp(-\varepsilon^2 \mu_v / 3) \leq 2 \exp(-\varepsilon^2 \psi_\varepsilon / 48) \leq 2 \exp(-(c/48) \log n) \leq \frac{2}{n^{c/48}}.$$

The number of nodes in \mathcal{T} is $O(n)$, and hence, by the union bound, for all nodes v at distance $i \leq M$ from the root, we have $\mathbb{P}[Y_v \notin (1 \pm \varepsilon/2)\mu_v] \leq 1/n^{\Omega(c)}$. ◀

▶ **Observation 10.** *Lemma 9 implies that for all nodes v at distance $i > M$ from the root of \mathcal{T} , we have $\mathbb{P}[|\mathcal{D}(v) \sqcap \mathbf{q}| > \psi_\varepsilon] \leq 1/n^{\Omega(c)}$. Indeed, Lemma 9 implies this for all nodes at distance M from the root, and the sizes of these sets are monotonically decreasing along any path down the tree.*

▶ **Lemma 11.** *Assume that the number of distinct sets $\mathbf{q}' \sqcap \mathcal{D}$, over all possible query objects \mathbf{q}' , is bounded by a polynomial $O(n^d)$, where d is some constant. Then, for a query \mathbf{q} , the probability that the algorithm returns a specific object $\mathbf{o} \in \mathcal{D} \sqcap \mathbf{q}$, is in $(1 \pm \varepsilon)/\beta$, where $\beta = |\mathcal{D} \sqcap \mathbf{q}|$. Similarly, the estimate the algorithm outputs for β is in $(1 \pm \varepsilon)\beta$. The answer is correct for all queries, with probability $\geq 1 - 1/n^{\Omega(c)}$, for a sufficiently large constant c .*

Proof. It is easy to verify the algorithm works correctly if $|\mathbf{q} \sqcap \mathcal{D}(u_j)| < \psi_\varepsilon$. Otherwise, for the node u_j computed by the algorithm, we have $|\mathbf{q} \sqcap \mathcal{D}(u_j)| > \psi_\varepsilon = c\varepsilon^{-2} \log n$. By Observation 10, with high probability, we have that $j \leq M$. By Lemma 9, it implies that for any node $u \in L_j$, we have $2^j |\mathcal{D}(u) \sqcap \mathbf{q}| \in (1 \pm \varepsilon/2) |\mathcal{D} \sqcap \mathbf{q}| = (1 \pm \varepsilon/2)\beta$, which implies that the estimate for the size of β is correct, as $u \in L_j$. This readily implies that the probability of returning a specific object $\mathbf{o} \in \mathcal{D} \sqcap \mathbf{q}$ is in $(1 \pm \varepsilon)/\beta$, since

$$\frac{1 - \varepsilon}{\beta} \leq \frac{1}{(1 + \varepsilon/2) |\mathcal{D} \sqcap \mathbf{q}|} \leq \frac{1}{|L_j| \cdot |\mathcal{D}(u) \sqcap \mathbf{q}|} \leq \frac{1}{(1 - \varepsilon/2) |\mathcal{D} \sqcap \mathbf{q}|} \leq \frac{1 + \varepsilon}{\beta}.$$

As for the probabilities, there are n nodes in \mathcal{T} , and $O(n^d)$ different queries, and thus the probability of failure is at most $n^{d+1}/n^{\Omega(c)} < 1/n^{\Omega(c)}$, by Lemma 9. ◀

3.2.2 Running times

Query time. The depth of \mathcal{T} is $h = O(\log n)$ with high probability (follows readily from Chernoff's inequality). Thus, the first stage (of computing the maximal i) requires $O(\log \log n)$ queries on the reporting data-structure, where each query takes $O(Q(n) + \log n)$ time. The second stage (of finding maximal j) takes

$$\tau = \mathbb{E} \left[\sum_{t=j}^i O(Q(n) + |\mathcal{D}(u_t) \sqcap \mathbf{q}|) \right].$$

Thus, we have

- (A) If $Q(n) = O(\log n)$, then $\tau = O(\psi_\varepsilon)$, as the cardinality of $\mathcal{D}(u_t) \sqcap \mathbf{q}$ decreases by a factor of two (in expectation) as one move downward along a path in the tree. Thus τ is a geometric summation dominated by the largest term.
- (B) If $Q(n) = \Omega(\log n)$, we have (in expectation) that $|i - j| \leq O(\log(1/\varepsilon))$, and thus $\tau = O(Q(n) \log(1/\varepsilon) + \psi_\varepsilon)$ time.
- (C) If $Q(n) = O(n^\lambda)$, for $0 < \lambda \leq 1$, then the query time is dominated by the query time for the top node (i.e., u_j) in this path, and $\tau = O(Q(n))$, as can easily be verified.

Construction time. The running time bounds of the form $O(C(n))$ are *well-behaved*, if for any non-negative integers n_1, n_2, \dots , such that $\sum_{i=1} n_i = n$, implies that $\sum_{i=1} C(n_i) = O(C(n))$. Under this assumption on the construction time, we have that the total construction time is $O(C(n) \log n)$.

3.2.3 Summary

► **Theorem 12.** Let \mathcal{D} be a set of n objects, and assume we are given a well-behaved range-reporting data-structure that can be constructed in $C(m)$ time, for m objects, and answers a reporting query \mathbf{q} in $O(Q(m) + |\mathbf{q} \cap \mathcal{D}|)$ time. Then, one can construct a data-structure, in $O(C(n) \log n)$ time, such that given a query object \mathbf{q} , it reports an $(1 \pm \varepsilon)$ -estimate for $\beta = |\mathbf{q} \cap \mathcal{D}|$, and also returns an object from $\mathbf{q} \cap \mathcal{D}$, where each object is reported with probability $(1 \pm \varepsilon)/\beta$. The data-structure answers all such queries correctly with probability $\geq 1 - 1/n^{\Omega(1)}$. The expected query time is:

- (i) $O((\varepsilon^{-2} + \log \log n) \log n)$ if $Q(m) = O(\log m)$.
- (ii) $O(Q(n))$ if $Q(m) = O(m^\lambda)$, for some constant $\lambda > 0$.
- (iii) $O(\varepsilon^{-2} \log n + Q(n) \log \frac{\log n}{\varepsilon})$ otherwise.

Plugging in the data-structure of Liu [23] for disks, with $C(m) = O(m \log m)$, and $Q(m) = O(\log m)$, in the above theorem, implies the following.

► **Corollary 13.** Let \mathcal{D} be a set of n disks in the plane. One can construct in $O(n \log^2 n)$ time a data-structure, such that given a query disk \mathbf{q} and a parameter $\varepsilon \in (0, 1/2)$, it outputs an $(1 \pm \varepsilon)$ -estimate for $\beta = |\mathbf{q} \cap \mathcal{D}|$, and also returns a disk in $\mathbf{q} \cap \mathcal{D}$ with a probability that is $(1 \pm \varepsilon)$ -uniform. The expected query time is $O((\varepsilon^{-2} + \log \log n) \log n)$, and the result returned is correct with high probability for all possible queries.

Approximate depth queries for a fixed threshold. If we are interested only in a single threshold, the above data-structure is an overkill, as one can directly construct a single sample and perform the query directly on this sample. This implies the following.

► **Corollary 14.** Let \mathcal{D} be a set of n disks in the plane. One can construct, in $O(n \log n)$ time, a data-structure, such that given a query disk \mathbf{q} , and a parameter $\varepsilon \in (0, 1/2)$, it reports whether $|\mathbf{q} \cap \mathcal{D}| \geq \beta$ or $|\mathbf{q} \cap \mathcal{D}| < \beta$. The data structure is allowed to make a mistake when $|\mathbf{q} \cap \mathcal{D}| \in [(1 - \varepsilon)\beta, (1 + \varepsilon)\beta]$. The data-structure answers the query correctly with probability $\geq 1 - 1/n^{\Omega(1)}$, and the query time is $O(\varepsilon^{-2} \log n)$.

4 A $(2 + \varepsilon)$ -approximation for densest subset disks

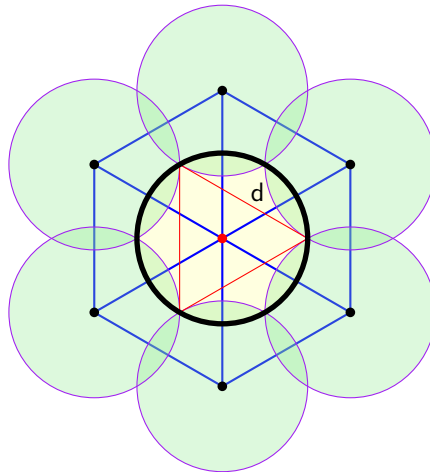
In this section, we design a $(2 + \varepsilon)$ -approximation algorithm to compute the densest subset of disks in $O(\varepsilon^{-4} n \log n)$ time.

4.1 Constant approximation via depth

The *depth* of a point in the arrangement of disks is the number of disks containing the point. The *deepest* point in the plane is the point with the maximum depth. The deepest point in the arrangement of disks and the densest subgraph in the geometric intersection graph of disks are the same (up to a constant factor).

► **Lemma 15.** *Let h be the depth of the deepest point in \mathcal{D} , where \mathcal{D} is a set of disks in the plane, and let d be maximum density of \mathcal{D} . Then $(h - 1)/2 \leq d \leq 7 \cdot h$, where c is a sufficiently large constant. In particular, in $O(n \log n)$ time, one can compute a quantity t , such that $(t - 1)/2 \leq d \leq 8t$.*

Proof. Lemma 6 readily implies that $d \geq (h - 1)/2$, as this is the density provided by the disks containing the deepest point.



■ **Figure 4.1** Defending a disk by a guard set made of 7 points.

As for the other direction, consider the densest subset $\mathcal{O} \subseteq \mathcal{D}$, and let d be the smallest disk in S . By Lemma 6, the set $T = d \cap (\mathcal{O} - d)$ has size least d . Let c be the center of d and inscribe an equilateral triangle in d . Consider a tiling of 6 such triangles that are interior disjoint, all sharing a vertex at c . Let P be the set of 7 vertices used by these triangles (including c), see Figure 4.1. One can verify³ that any disk d' at least as large as d that intersect d , must contain at least one point of P .

Thus, each disk in T is stabbed by at least one point in P , which readily implies that there is a point in P that has depth $|T|/|P| \geq d/7$.

As for the last part, a 1.1-approximation of the deepest point in the arrangement of the disks can be computed in $O(n \log n)$ time [5]. The returned approximate deepest point provides the desired approximation. ◀

4.2 The algorithm

The input is a set \mathcal{D} of n disks in the plane. Let $\vartheta = \varepsilon/15$. The algorithm starts by computing a number ξ , such that $d \in [(\xi - 1)/2, 8\xi]$, using the algorithm of Lemma 15. The basic idea is to try a sequence of exponentially decaying values to the optimal density d . To this end, in the i th round, the algorithm tries the degree threshold $\beta = 8\xi(1 - \vartheta)^i$, for $i = 1, \dots, \log_{1+\vartheta} 16 = O(1/\varepsilon)$.

In the beginning of such a round, let $\mathcal{L} \leftarrow \mathcal{D}$. During a round, the algorithm repeatedly removes “low-degree” objects, by repeatedly doing the following:

³ Indeed, if the center c' of d' is in the union of the seven copies of d centered at the points of P , see Figure 4.1, then the disk d' must contain one of the points of P . Otherwise, it can be verified that d and d' do not intersect.

43:10 Approximating Densest Subgraph in Geometric Intersection Graphs

- (I) Construct the data-structure of Corollary 14 on the objects of \mathcal{L} .
- (II) Let $\mathcal{L}_< \subseteq \mathcal{L}$ be the objects whose degree in \mathcal{L} is smaller than $(1 + \vartheta)\beta$ according to this data-structure (i.e., query all the objects of \mathcal{L} for their degree). Let $\mathcal{L}_\geq = \mathcal{L} \setminus \mathcal{L}_<$.
- (III) If \mathcal{L}_\geq is empty, then this round failed, and the algorithm continues to the next round.
- (IV) If $|\mathcal{L}_<| < \vartheta |\mathcal{L}|$, then the algorithm returns \mathcal{L} as the desired approximate densest subset.
- (V) Otherwise, let $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L}_<$. The algorithm continues to the next iteration (still inside the same round).

4.3 Analysis

► **Lemma 16.** *When the algorithm terminates, we have $\beta \geq (1 - \vartheta)^3 d$, with high probability, where d is the optimal density.*

Proof. Consider the iteration when $\beta \in [(1 - \vartheta)^3 d, (1 - \vartheta)^2 d]$. By definition of $\mathcal{L}_<$, all the objects in it have a degree at most $(1 + \vartheta)\beta \leq (1 + \vartheta)(1 - \vartheta)^2 d \leq (1 - \vartheta)d$. By Lemma 6, all the objects in the optimal solution have degree $\geq d$ (when restricted to the optimal solution). Therefore, none of the objects in the optimal solution are in $\mathcal{L}_<$ and hence, the set \mathcal{L}_\geq is not empty (and contains the optimal solution). Inside this round, the loop is performed at most $O(\vartheta^{-1} \log n)$ times, as every iteration of the loop shrinks \mathcal{L} by a factor of $1 - \vartheta$. This implies that the algorithm must stop in this round. ◀

► **Lemma 17.** *The above algorithm returns a $(2 + \varepsilon)$ -approximation of the densest subset.*

Proof. Consider the set \mathcal{L} , the value of β when the algorithm terminated, and let $\nu = |\mathcal{L}|$. By Lemma 16, $\beta \geq (1 - \vartheta)^3 d$, and by the algorithm stopping condition, we have $|\mathcal{L}_<| < \vartheta |\mathcal{L}|$. In addition, all the objects in \mathcal{L}_\geq have degree at least $(1 - \vartheta)\beta$. Thus, the number of induced edges on \mathcal{L} is

$$m(\mathcal{L}) \geq \frac{(1 - \vartheta)\beta |\mathcal{L}_\geq|}{2} > \frac{(1 - \vartheta)^2 \beta |\mathcal{L}|}{2} \geq (1 - \vartheta)^5 \frac{d}{2} |\mathcal{L}| \geq (1 - 5\vartheta) \frac{d}{2} |\mathcal{L}|.$$

Thus $\nabla(\mathcal{L}) = \frac{m(\mathcal{L})}{|\mathcal{L}|} \geq (1 - 5\vartheta)d/2 = (1 - \varepsilon/3)d/2$, as $\vartheta = \varepsilon/15$. Observe that $2/(1 - \varepsilon/3) \leq 2(1 + \varepsilon/2) \leq 2 + \varepsilon$. ◀

► **Lemma 18.** *The expected running time of the above algorithm is $O(n\varepsilon^{-4} \log n)$.*

Proof. The expected time spent, in an iteration, in step I and II is $O(\vartheta^{-2} n \log n)$, where $n = |\mathcal{L}|$, and this dominates the running time of this iteration. Let n_i be the size of \mathcal{L} in the i th iteration (inside the same round), and observe that $n_{i+1} \leq (1 - \vartheta)n_i$. Assume t iterations are performed inside this round. As $\sum_{i=1}^t n_i = O(n/\vartheta)$, the expected running time for the round is $\sum_{i=1}^t O(\vartheta^{-2} n_i \log n_i) = O(\vartheta^{-3} n \log n)$. The value of β can change $O(1/\varepsilon)$ times during the algorithm and hence, the overall expected running time of the algorithm is $O(n\vartheta^{-4} \log^2 n)$, which implies the result since $\vartheta = \Theta(\varepsilon)$. ◀

Summarizing, we get the following.

► **Theorem 19.** *Let \mathcal{D} be a set of n disks in the plane, and let $\varepsilon \in (0, 1/2)$ be a parameter. The overall algorithm computes, in $O(\varepsilon^{-4} n \log n)$ expected time, a $(2 + \varepsilon)$ -approximation to the densest subgraph of $G_{\cap \mathcal{D}}$, and the result is correct with high probability.*

► **Remark 20** (A lower bound). Consider the element uniqueness problem – of deciding if n real numbers are all distinct. It is known that in the comparison model this requires $\Omega(n \log n)$ time. Treating each number as a disk of radius zero, readily implies there is a subgraph of density ≥ 1 (and in particular, non-zero), if and only if there are two identical numbers. Thus, any approximation algorithm for the maximum density problem requires $\Omega(n \log n)$ time (in the comparison model).

5 An $(1 + \varepsilon)$ -approximation for densest subset disks

Here, we present a $(1 + \varepsilon)$ -approximation algorithm for the densest subset of disks, which is based on the following intuitive idea – if the intersection graph is sparse, then the problem is readily solvable. If not, then one can sample a sparse subgraph, and use an approximation algorithm on the sampled graph.

5.1 Densest subgraph estimation via sampling

Let $G = (V, E)$ be a graph with n vertices and m edges, with maximum subgraph density d . Let $\vartheta \in (0, 1/6)$ be a parameter, and assume that $m > c'n\vartheta^{-2} \log n$, where c' is some sufficiently large constant, which in particular implies that

$$d = d(G) \geq \frac{m}{n} \geq \frac{c'}{\vartheta^2} \log n.$$

Assume we have an estimate $\bar{m} \in (1 \pm \vartheta)m$ of m . For a constant c to be specified shortly, with $c < c'$, let

$$\psi = c \frac{n}{\bar{m}} \vartheta^{-2} \log n \leq \frac{c}{c'(1 - \vartheta)} \leq \frac{6c}{5c'} < 1.$$

Let $F = \{e_1, \dots, e_r\}$ be a random sample of $r = \lceil \psi \bar{m} \rceil$ edges from G . Specifically, in the i th iteration, an edge e_i is picked from the graph, where the probability of picking any edge is in $(1 \pm \vartheta)/m$. Let $H = (V, F)$, and observe that H is a sparse graph with n vertices and $r = O(\vartheta^{-2} n \log n)$ edges. The claim is that the densest subset $D \subseteq V$ in H , or even approximate densest subset, is close to being the densest subset in G . The proof of this follows from previous work [24], but requires some modifications, since we only have an estimate to the number of edges m , and we are also interested in approximating the densest subgraph on the resulting graph. We include the details here so that the presentation is self contained. The result we get is summarized in Lemma 24, if the reader is uninterested in the (somewhat tedious) analysis of this algorithm.

5.1.1 Analysis

► **Lemma 21.** *Let $d = d(G)$, and let $U \subseteq V$, be an arbitrary set of k vertices. If $\nabla_G(U) \leq d/60$, then $\mathbb{P}[\nabla_H(U) \geq \psi d/5] \leq n^{-100k}$.*

Proof. We have $d \geq m/n$, where $n = |V(G)|$ and $m = |E(G)|$, and thus

$$\psi = c \frac{n}{\bar{m} \vartheta^2} \log n \geq c \frac{n}{(1 + \vartheta)m \vartheta^2} \log n \geq \frac{1}{d} \cdot \frac{c}{(1 + \vartheta)\vartheta^2} \log n$$

Let $X_i = 1$ if the edge sampled in the i th round belongs to H_U and zero, otherwise. Let $X = \sum_i X_i$ be the number of edges in H_U . Then

$$\mathbb{P}[X_i = 1] \in (1 \pm \vartheta) \frac{|E(G_U)|}{m} = (1 \pm \vartheta) \frac{k \nabla_G(U)}{m}.$$

43:12 Approximating Densest Subgraph in Geometric Intersection Graphs

By linearity of expectations, and as $\bar{m} \in (1 \pm \vartheta)m$, we have

$$\mathbb{E}[X] \in (1 \pm \vartheta)\psi\bar{m}\frac{k\nabla_{\mathbf{G}}(U)}{m} \subseteq (1 \pm \vartheta)^2\psi k\nabla_{\mathbf{G}}(U). \quad (5.1)$$

By assumption $\nabla_{\mathbf{G}}(U) \leq d/60$, implying that $\mathbb{E}[X] \leq (1 + 3\vartheta)\psi kd/60 \leq \psi kd/30$, if $\vartheta \in (0, 1/3)$, by Observation 1. Observe that $(1 + (2e - 1))\mathbb{E}[X] \leq \frac{\psi kd}{5}$. By Chernoff's inequality, Lemma 30, we have

$$\mathbb{P}\left[\nabla_H(U) \geq \frac{\psi d}{5}\right] = \mathbb{P}\left[X \geq \frac{\psi kd}{5}\right] \leq 2^{-\psi kd/5} \leq \frac{1}{n^{100k}},$$

by picking c to be sufficiently large. \blacktriangleleft

► **Lemma 22.** *Let $d = d(\mathbf{G})$, and let $U \subseteq V$, be an arbitrary set of k vertices. If $\nabla_{\mathbf{G}}(U) \geq d/60$, then $\mathbb{P}[\nabla_H(U) \in (1 \pm \vartheta)^3\psi\nabla_{\mathbf{G}}(U)] \geq 1 - n^{-100k}$.*

Proof. Following the argument of Lemma 21 and as $\nabla_{\mathbf{G}}(U) \geq d/60$, we have that $\mathbb{E}[X] = \Omega(k \cdot \psi\nabla_{\mathbf{G}}(U)) = \Omega(k \cdot \psi d) = \Omega(k \cdot c\vartheta^{-2} \log n)$. Chernoff's inequality, Theorem 31, then implies that $X \in (1 \pm \vartheta)\mathbb{E}[X]$ with probability at least $1 - 2\exp(-\vartheta^2 \mathbb{E}[X]/4) \geq 1 - 1/n^{100k}$, for n sufficiently large. The claim now readily follows from Eq. (5.1). \blacktriangleleft

► **Lemma 23.** *Let $\alpha \in (0, 1/6)$ be a parameter. For all sets $U \subseteq V$, such that $\nabla_H(U) \geq (1 - \alpha)d(H)$, we have that $\nabla_{\mathbf{G}}(U) \geq (1 - 6\vartheta)(1 - \alpha)d$, and this holds with high probability.*

Proof. Let X be the densest subset in \mathbf{G} . By Lemma 22, we have that

$$\nabla_H(X) \in (1 \pm \vartheta)^3\psi d(\mathbf{G}) \implies d(H) \geq (1 - \vartheta)^3\psi d \geq \frac{\psi d}{2}.$$

By Lemma 21, we have that for all the sets $T \subseteq V$, with $\nabla_{\mathbf{G}}(T) \leq d/60$, we have $\nabla_H(T) < \psi d/5 < d(H)/2$, and this happens with probability $\sum_{k=2}^n \sum_{T \subseteq V: |T|=k} 1/n^{100k} \leq \sum_{k=2}^n \binom{n}{k}/n^{100k} \leq 1/n^{99}$.

Thus, all the sets $U \subseteq V$ under consideration have $\nabla_{\mathbf{G}}(U) > d/60$. By Lemma 22, for all such sets, with probability $1 - n^{-100k} \geq 1 - 1/n^{99}$, we have $\nabla_H(U) \in (1 \pm \vartheta)^3\psi\nabla_{\mathbf{G}}(U)$, which implies $\nabla_{\mathbf{G}}(U) \in \frac{1}{(1 \pm \vartheta)^3\psi}\nabla_H(U)$. Thus, we have

$$\nabla_{\mathbf{G}}(U) \geq \frac{1}{(1 + \vartheta)^3\psi}\nabla_H(U) \geq \frac{(1 - \alpha)d(H)}{(1 + \vartheta)^3\psi} \geq \frac{(1 - \alpha)(1 - \vartheta)^3\psi d}{(1 + \vartheta)^3\psi} \geq (1 - \alpha)(1 - 6\vartheta)d,$$

since $1/(1 + \vartheta) \geq 1 - \vartheta$, and $(1 - \vartheta)^6 \geq 1 - 6\vartheta$. \blacktriangleleft

5.1.2 Summary

► **Lemma 24.** *Let $\varepsilon \in (0, 1)$ be a parameter, and let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a graph with n vertices and m edges, with $m = \Omega(\varepsilon^{-2}n \log n)$. Furthermore, let \bar{m} be an estimate to m , such that $\bar{m} \in (1 \pm \vartheta)m$, where $\vartheta = \varepsilon/10$. Let $\psi = c(n/\bar{m})\vartheta^{-2} \log n$, and let F be a random sample of $\psi\bar{m} = O(\varepsilon^{-2}n \log n)$ edges, with repetition, where the probability of any specific edge to be picked is $(1 \pm \vartheta)/m$, and c is a sufficiently large constant. Let $H = (\mathbf{V}, F)$ be the resulting graph, and let $X \subseteq V$ be subset of H with $\nabla_H(X) \geq (1 - \varepsilon/6)d(H)$. Then, $\nabla(X) \geq (1 - \varepsilon)d(\mathbf{G})$.*

Proof. This follows readily from the above, by setting $\alpha = \varepsilon/6$, and using Lemma 23. \blacktriangleleft

5.2 Random sampler

To implement the above algorithm, we need an efficient algorithm for sampling edges from the intersection graph of disks, which we describe next.

5.2.1 The algorithm

The algorithm consists of the following steps:

- (I) Build the data-structure of Corollary 13 on the disks of \mathcal{D} with error parameter ε/c , where c is a sufficiently large constant. Also, build the range-reporting data-structure of Liu [23] on the disks of \mathcal{D} .
- (II) For each object $\mathfrak{o} \in \mathcal{D}$, query the data-structure of Corollary 13 with \mathfrak{o} . Let the estimate returned be d' . If $d' < c'/\varepsilon$ (for a constant $c' \gg c$), then report $\mathfrak{o} \cap \mathcal{D}$ by querying the range-reporting data-structure with \mathfrak{o} , and set $d_{\mathfrak{o}} \leftarrow |\mathfrak{o} \cap \mathcal{D}| - 1$. Otherwise, set $d_{\mathfrak{o}} \leftarrow d'$.
- (III) We perform $|F|$ iterations and in each iteration, sample a random edge from $G_{\cap \mathcal{D}}$. In a given iteration, sample a disk $\mathfrak{o} \in \mathcal{D}$, where \mathfrak{o} has a probability of $\frac{d_{\mathfrak{o}}}{\sum_{\mathfrak{o} \in \mathcal{D}} d_{\mathfrak{o}}}$ being sampled. If $d_{\mathfrak{o}} < c'/\varepsilon$, then uniformly-at-random report a disk from $\mathfrak{o} \cap (\mathcal{D} - \mathfrak{o})$. Otherwise, query the data-structure of Corollary 13 with \mathfrak{o} which returns a disk in $\mathfrak{o} \cap \mathcal{D}$ (keep querying till a disk other than \mathfrak{o} is returned).

5.2.2 Analysis

► **Lemma 25.** *For each object $\mathfrak{o} \in \mathcal{D}$, we have $d_{\mathfrak{o}} \in (1 \pm \varepsilon/c'')|\mathfrak{o} \cap (\mathcal{D} - \mathfrak{o})|$, where $c \gg c''$ with high probability.*

Proof. Fix an object \mathfrak{o} . When $d' < c'/\varepsilon$, then the statement holds trivially. Let $d = |\mathfrak{o} \cap (\mathcal{D} - \mathfrak{o})|$. Now we consider the case $d' \geq c'/\varepsilon$. We know that $d' \in (1 \pm \varepsilon/c)(d+1)$. Firstly, $d' \leq (1 + \varepsilon/c)(d+1) \leq (1 + \varepsilon/c'')d$ hold if

$$d \geq \frac{1}{\varepsilon} \cdot \frac{1 + \varepsilon/c}{1/c'' - 1/c} \geq 1/2\varepsilon.$$

Observe that $d \geq 1/2\varepsilon$, since $c'/\varepsilon \leq d' \leq (1 + \varepsilon/c)(d+1)$ implies that $d \geq \frac{c'}{\varepsilon(1 + \varepsilon/c)} - 1 \geq 1/2\varepsilon$. Finally, $d' \geq (1 - \varepsilon/c)(d+1) \geq (1 - \varepsilon/c'')d$ holds trivially. Therefore,

$$d_{\mathfrak{o}} = d' \in (1 \pm \varepsilon/c'')d. \quad \blacktriangleleft$$

► **Lemma 26.** *In each iteration, the probability of sampling any edge in $G_{\cap \mathcal{D}}$ is $(1 \pm \varepsilon)/m$.*

Proof. An edge (u, v) in $G_{\cap \mathcal{D}}$ can get sampled in step (III) in two ways. In the first way, the disk corresponding to u gets sampled and then v gets reported as the random neighbor of u , and vice-versa for the second way.

Let $d = |\mathfrak{o} \cap (\mathcal{D} - \mathfrak{o})|$, where \mathfrak{o} is the disk corresponding to u . Consider the case, where $d_{\mathfrak{o}} \geq c'/\varepsilon$. As such, the first way of sampling the edge (u, v) has probability lower-bounded by:

$$\frac{d_{\mathfrak{o}}}{\sum d_{\mathfrak{o}}} \cdot \underbrace{\frac{1 \pm \varepsilon/c}{d}}_{\text{almost-uniformity}} \subseteq \underbrace{\frac{d_{\mathfrak{o}}}{(1 \pm \varepsilon/c'')2m}}_{\text{Lemma 25}} \cdot \frac{1 \pm \varepsilon/c}{d} \subseteq \frac{(1 \pm \varepsilon/c'')d}{(1 \pm \varepsilon/c'')2m} \cdot \frac{1 \pm \varepsilon/c}{d} \subseteq \frac{1 \pm \varepsilon}{2m}.$$

Therefore, for the case $d_{\mathfrak{o}} \geq c'/\varepsilon$, the probability of sampling the edge (u, v) is $(1 \pm \varepsilon)/m$. Similarly, the statement holds for the case $d_{\mathfrak{o}} < c'/\varepsilon$. \blacktriangleleft

► **Lemma 27.** *The expected running time of the algorithm is $O(\varepsilon^{-4}n \log^2 n + \varepsilon^{-2}n \log^2 n \cdot \log \log n)$.*

Proof. The first step of the algorithm takes $O(n \log^2 n)$ expected time. The second step of the algorithm takes $O(n(\varepsilon^{-2} + \log \log n) \log n)$ expected time. In the third step of the algorithm, each iteration requires querying the data-structure of Corollary 13 $O(1)$ times in expectation. Therefore, the third step takes $O(|F|) \cdot O((\varepsilon^{-2} + \log \log n) \log n) = O(\varepsilon^{-4}n \log^2 n + \varepsilon^{-2}n \log^2 n \cdot \log \log n)$ expected time. ◀

The above implies the following result.

► **Lemma 28.** *Let \mathcal{D} be a collection of n disks in the plane, and let $G_{\cap \mathcal{D}}$ be the corresponding geometric intersection graph with m edges. Let F be a random sample of $O(\varepsilon^{-2}n \log n)$ edges from $G_{\cap \mathcal{D}}$, with repetition, where the probability of any specific edge to be picked is $(1 \pm \varepsilon)/m$. The edges are all chosen independently into F . Then, the algorithm described above, for computing F , runs in $O(\varepsilon^{-4}n \log^2 n + \varepsilon^{-2}n \log^2 n \cdot \log \log n)$ expected time.*

5.3 The result

► **Theorem 29.** *Let \mathcal{D} be a collection of n disks in the plane. A $(1 + \varepsilon)$ -approximation to the densest subgraph of \mathcal{D} can be computed in $O(\varepsilon^{-4}n \log^2 n + \varepsilon^{-2}n \log^2 n \cdot \log \log n)$ expected time. The correctness of the algorithm holds with high probability.*

Proof. The case of intersection graph having $O(\varepsilon^{-2}n \log n)$ edges can be handled directly by computing the whole intersection graph in $O(\varepsilon^{-2}n \log n)$ expected time (using Lemma 7).

To handle the other case, we use Lemma 28 to generate a graph $H = (V, F)$ in $O(\varepsilon^{-4}n \log^2 n + \varepsilon^{-2}n \log^2 n \cdot \log \log n)$ expected time. Since the intersection graph has $\Omega(\varepsilon^{-2}n \log n)$ edges, using Lemma 24, it suffices to compute a $(1 - \varepsilon/6)$ approximate densest subgraph of H , which can be computed by the algorithm of [6] in $O(\varepsilon^{-4}n \log^2 n)$ expected time. ◀

6 Extension to other geometric intersection graphs

Although the focus of the paper has been on disks, the techniques described in this paper can be extended to other geometric intersection graphs as well. For any geometric intersection graph with a well-behaved range-reporting data-structure that can be constructed in $C(n)$ time and query time $O(Q(n) + |\mathbf{q} \cap \mathcal{D}|)$, the technique in Section 5 gives a $(1 + \varepsilon)$ -approximation algorithm with a running time of $\tilde{O}(C(n) + nQ(n))$ (where the \tilde{O} notation hides polylogarithmic factors in n). Therefore, if $C(n) = O(n^{2-\lambda_1})$ and $Q(n) = O(n^{1-\lambda_2})$, it leads to a $(1 + \varepsilon)$ -approximation algorithm with running time $O(n^{2-\lambda_3})$, where $\lambda_1, \lambda_2, \lambda_3 \in (0, 1)$. For example, data structures with such bounds exist for axis-aligned boxes in d -dimension (with $n \log^{O(d)} n$ running time) and spheres in d -dimensions (via reduction to halfspace range reporting). This improves upon the running time of $\Omega(m) = \Omega(n^2)$ obtained for general graphs where the edges are given explicitly.

The technique in Section 4 for disks performs $O(1/\varepsilon)$ rounds by establishing a connection between the deepest point in the arrangement of disks with the maximum density. This connection extends to spheres in d -dimensions but does not hold, for example, for axis-aligned boxes in d -dimensions (in 2-d the depth can be at most two but the maximum density can be $\Omega(n)$). In any case, the technique in Section 4 will perform at most $O(\varepsilon^{-1} \log n)$ rounds if we start guessing the degree threshold from n . Then the running time of the algorithm will be $\tilde{O}(C(n) + nQ(n))$, where $C(n)$ and $Q(n)$ are the construction time and the query

time, respectively, of the approximate depth data-structure for a fixed threshold (analogous version of Corollary 14). Once again this leads to $n \log^{O(d)} n$ running time for axis-aligned boxes in d -dimensions and truly sub-quadratic running time for spheres in d -dimensions.

7 Conclusions

We presented two near-linear time approximation algorithms to compute the densest subgraph on (implicit) geometric intersection graph of disks. We conclude with a few open problems. Are there implicit geometric intersection graphs, such as unit-disk graphs or say, interval graphs, for which the *exact* densest subgraph can be computed in *sub-quadratic time* (in terms of n)? Finally, maintaining the approximate densest subgraph in sub-linear time (again in terms of n) under insertions and deletions of objects, looks to be a challenging problem (in prior work on general graphs an edge gets deleted or inserted, but in an intersection graph a vertex gets deleted or inserted).

References

- 1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In Claire Mathieu, editor, *Proc. 20th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 180–186. SIAM, 2009. doi:10.1137/1.9781611973068.21.
- 2 Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Comput. Geom.*, 43(8):700–712, 2010. doi:10.1016/J.COMGEO.2010.04.003.
- 3 Peyman Afshani and Jeff M. Phillips. Independent range sampling, revisited again. In *Proceedings of Symposium on Computational Geometry (SoCG)*, volume 129, pages 4:1–4:13, 2019. doi:10.4230/LIPICs.SOCG.2019.4.
- 4 Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of Symposium on Computational Geometry (SoCG)*, page 271, 2014. doi:10.1145/2582112.2582152.
- 5 Boris Aronov and Sarel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008. doi:10.1137/060669474.
- 6 Bahman Bahmani, Ashish Goel, and Kamesh Munagala. Efficient primal-dual graph algorithms for mapreduce. In *Algorithms and Models for the Web Graph: 11th International Workshop*, pages 59–78, 2014. doi:10.1007/978-3-319-13123-8_6.
- 7 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *Proceedings of the VLDB Endowment*, 5(5):454–465, 2012. doi:10.14778/2140436.2140442.
- 8 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 173–182, 2015. doi:10.1145/2746539.2746592.
- 9 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & Computational Geometry*, 50(3):771–783, 2013. doi:10.1007/S00454-013-9538-5.
- 10 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 84–95, 2000. doi:10.1007/3-540-44436-X_10.
- 11 Chandra Chekuri, Sarel Har-Peled, and Kent Quanrud. Fast LP-based approximations for geometric packing and covering problems. In *Proc. 31st ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 1019–1038, 2020. doi:10.1137/1.9781611975994.62.

- 12 Jie Chen and Yousef Saad. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(7):1216–1230, 2012. doi:10.1109/TKDE.2010.271.
- 13 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990. doi:10.1016/0012-365X(90)90358-0.
- 14 Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal of Computing*, 32(5):1338–1355, 2003. doi:10.1137/S0097539702403098.
- 15 Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient densest subgraph computation in evolving graphs. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 300–310, 2015. doi:10.1145/2736277.2741638.
- 16 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Brief announcement: Applications of uniform sampling: Densest subgraph and beyond. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 397–399, 2016. doi:10.1145/2935764.2935813.
- 17 Giorgio Gallo, Michael D. Grigoriadis, and Robert Endre Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal of Computing*, 18(1):30–55, 1989. doi:10.1137/0218003.
- 18 David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of Very Large Data Bases (VLDB)*, pages 721–732, 2005. URL: <http://www.vldb.org/archives/website/2005/program/paper/thu/p721-gibson.pdf>.
- 19 A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1984.
- 20 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. doi:10.1090/surv/173.
- 21 Samir Khuller and Barna Saha. On finding dense subgraphs. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 597–608, 2009. doi:10.1007/978-3-642-02927-1_50.
- 22 Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzino, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *CoRR*, abs/2303.14467, 2023. doi:10.48550/arXiv.2303.14467.
- 23 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM J. Comput.*, 51(3):723–765, 2022. doi:10.1137/20M1388371.
- 24 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In *Proceedings of Mathematical Foundations of Computer Science (MFCS)*, pages 472–482, 2015. doi:10.1007/978-3-662-48054-0_39.
- 25 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995. doi:10.1017/CB09780511814075.
- 26 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 181–193, 2020. doi:10.1145/3357713.3384327.
- 27 Yufei Tao. Algorithmic techniques for independent query sampling. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 129–138, 2022. doi:10.1145/3517804.3526068.

A Chernoff’s inequality


The following are standard forms of Chernoff’s inequality, see [25].

► **Lemma 30.** *Let X_1, \dots, X_n be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \dots, n$. Let $X = \sum_{i=1}^n X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta > 2e - 1$, we have $\mathbb{P}[X > (1 + \delta)\mu] < 2^{-\mu(1+\delta)}$.*

► **Theorem 31.** *Let $\varepsilon \in (0, 1)$ be a parameter. Let $X_1, \dots, X_n \in \{0, 1\}$ be n independent random variables, let $X = \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X]$. We have that*

$$\max\left(\mathbb{P}[X < (1 - \varepsilon)\mu], \mathbb{P}[X > (1 + \varepsilon)\mu]\right) \leq \exp(-\varepsilon^2\mu/3)$$

Independence and Domination on Bounded-Treewidth Graphs: Integer, Rational, and Irrational Distances

Tim A. Hartmann  

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Dániel Marx  

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

The distance- d variants of INDEPENDENT SET and DOMINATING SET problems have been extensively studied from different algorithmic viewpoints. In particular, the complexity of these problems are well understood on bounded-treewidth graphs [Katsikarelis, Lampis, and Paschos, *Discret. Appl. Math* 2022][Borradaile and Le, IPEC 2016]: given a tree decomposition of width t , the two problems can be solved in time $d^t \cdot n^{O(1)}$ and $(2d+1)^t \cdot n^{O(1)}$, respectively. Furthermore, assuming the Strong Exponential-Time Hypothesis (SETH), the base constants are best possible in these running times: they cannot be improved to $d-\varepsilon$ and $2d+1-\varepsilon$, respectively, for any $\varepsilon > 0$. We investigate continuous versions of these problems in a setting introduced by Megiddo and Tamir [SICOMP 1983], where every edge is modeled by a unit-length interval of points. In the δ -DISPERSION problem, the task is to find a maximum number of points (possibly inside edges) that are pairwise at distance at least δ from each other. Similarly, in the δ -COVERING problem, the task is to find a minimum number of points (possibly inside edges) such that every point of the graph (including those inside edges) is at distance at most δ from the selected point set. We provide a comprehensive understanding of these two problems on bounded-treewidth graphs.

1. Let $\delta = a/b$ with a and b being coprime. If $a \leq 2$, then δ -DISPERSION is polynomial-time solvable. For $a \geq 3$, given a tree decomposition of width t , the problem can be solved in time $(2a)^t \cdot n^{O(1)}$, and, assuming SETH, there is no $(2a - \varepsilon)^t \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$.
2. Let $\delta = a/b$ with a and b being coprime. If $a = 1$, then δ -COVERING is polynomial-time solvable. For $a \geq 2$, given a tree decomposition of width t , the problem can be solved in time $((2 + 2(b \bmod 2))a)^t \cdot n^{O(1)}$, and, assuming SETH, there is no $((2 + 2(b \bmod 2))a - \varepsilon)^t \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$.
3. For every fixed irrational number $\delta > 0$ satisfying some mild computability condition, both δ -DISPERSION and δ -COVERING can be solved in time $n^{O(t)}$ on graphs of treewidth t . We show a very explicitly defined irrational number $\delta = (4 \sum_{j=1}^{\infty} 2^{-2^j})^{-1} \approx 0.790085$ such that δ -DISPERSION and $\delta/2$ -COVERING are W[1]-hard parameterized by the treewidth t of the input graph, and, assuming ETH, cannot be solved in time $f(t) \cdot n^{o(t)}$.

As a key step in obtaining these results, we extend earlier results on distance- d versions of INDEPENDENT SET and DOMINATING SET: We determine the exact complexity of these problems in the special case when the input graph arises from some graph G' by subdividing every edge exactly b times.

2012 ACM Subject Classification Theory of computation \rightarrow Discrete optimization; Mathematics of computing \rightarrow Graph algorithms; Theory of computation \rightarrow Problems, reductions and completeness; Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Independence, Domination, Irrationals, Treewidth, SETH

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.44



© Tim A. Hartmann and Dániel Marx;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 44; pp. 44:1–44:19



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

An independent set of a graph G is a subset of vertices that have pairwise distance at least 2. A well-known generalization to higher distance is the notion of a -independent set for some integer a , which is a subset of vertices that have pairwise distance at least a . Receiving extensive attention in the literature, e.g. [12, 29, 13, 4, 32, 21], the problem seems reasonably well understood. The dual notion of $distance-d$ dominating set, which is a set D of vertices such that every vertex of the graph is at distance at most d from S , was also similarly well studied. In this paper, we present an extensive study of both problems, focusing on their complexity on subdivided and bounded-treewidth graphs. Furthermore, we explore the generalization of these problems to noninteger (even irrational!) distances in an appropriate continuous model [8, 33] that received renewed attention lately [16, 17, 18, 14].

Independent Set and Dispersion

Integer distances. Finding a maximum a -independent set is NP-hard for $a = 2$ (as it is the same as the classic INDEPENDENT SET problem) and it is not difficult to show that it remains NP-hard for any fixed $a \geq 2$. In contrast, there are polynomial time algorithms for $a \in \{2, 4\}$ when the input is a 2-subdivided graph, that is a graph G that resulted from a graph G' by replacing every edge by a path of length two.

► **Theorem 1.1** (Grigoriev et al. [14]). *For $a \in \{2, 4\}$, a maximum a -independent set on a 2-subdivided graph can be found in linear time.¹*

Due to an important connection to the dispersion problem (see later in this section), we are particularly interested in the complexity of finding a maximum a -independent set on subdivided graphs, where every edge is replaced by a path of length b . Formally, let $\alpha_a(G)$ be the maximum cardinality of an a -independent set of a graph G . For a graph class \mathcal{G} , let a -INDEPENDENT SET(\mathcal{G}) be the corresponding decision problem, which is, given a graph $G \in \mathcal{G}$ and integer k , deciding whether $\alpha_a(G) \geq k$. Let \mathcal{G}_b be class of graphs G that are b -subdivisions, meaning G results from a graph G' by replacing every edge by a path of length b .

As a first contribution, for every fixed integer a, b , we settle the NP-hardness and the parameterized complexity of finding a maximum a -independent set when parameterized by the solution size (as color-coded in Figure 1). If the ratio $\frac{a}{b}$ is smaller than 2, then the problem is FPT, and otherwise it is W[1]-hard unless it is a polynomial time solvable case.

► **Theorem 1.2** (Section 4). *a -INDEPENDENT SET(\mathcal{G}_b) is*

- *polynomial time solvable if $b = ca$ or if $b = c\frac{a}{2}$ and $\frac{a}{2}$ is even for some integer c ; and NP-hard for all other integers a, b ; and is*
- *fixed-parameter tractable for the solution size as parameter if $\frac{a}{b} < 2$ or if $\frac{a}{b} = 2$, b even; and W[1]-hard for all other integers a, b .*

Next, we consider the problem parameterized by treewidth. Intuitively, the a -INDEPENDENT SET problem is harder for larger a . Indeed, for all $a \geq 2$, there is a matching upper and lower bound with a in the base of an exponential run time for graphs of bounded treewidth, assuming the Strong Exponential Time Hypothesis (SETH).

¹ The original statement is about a continuous dispersion problem, but can be put as above using a connection of these two problem which we mention later.

$a \backslash b$	1	2	3	4	5	6
1	$ V $	$ V + E $	$ V + 2 E $	$ V + 3 E $	$ V + 4 E $	$ V + 5 E $
2	IS	1-DISP	IS+ $ E $	1/2-DISP	IS+2 $ E $	1/3-DISP
3	3-IS		$ V $	3-IS+ $ E $		$ V + E $
4	4-IS	2-DISP		1-DISP	4-IS+ $ E $	2/3-DISP
5	5-IS				$ V $	5-IS+ $ E $
6	6-IS	3-DISP	IS	3/2-DISP		1-DISP

■ **Figure 1** Some problems, such as **INDEPENDENT SET** and **DISPERSION**, (or solution sizes) corresponding to a -**INDEPENDENT SET** on a graph G_b for small values of a, b and $V = V(G)$ and $E = E(G)$. A light green cell indicates a polynomial time solvable case, an orange NP-hardness & FPT, and a dark red NP-hardness & W[1]-hardness.

► **Theorem 1.3** (Katsikarelis et al. [20]). *For $a \geq 2$, given a tree decomposition of width t of an n vertex input graph, a maximum a -independent set can be found in time $a^t \cdot n^{O(1)}$. Assuming **SETH**, there is no $(a - \varepsilon)^{\text{tw}(G)} \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$, even for graphs without a cycle of length $< a$.²*

We refine Theorem 1.3 by restricting the a -**INDEPENDENT SET** problem to b -subdivided graphs and determining the optimal base of the exponent for all integers a and b . We expect that larger a makes the problem harder (as in Theorem 1.3) and larger b makes the problem easier (as the graphs become more restricted), but it turns out that the optimal base depends on a and b in a very subtle way. Let $\text{gcd}(a, b)$ denote the greatest common divisor of integers a and b .

► **Theorem 1.4** (Section 5). *Let a', b' integers with $\text{gcd}(a', b') = c$, $ca = a'$ and $cb = b'$. Assume **SETH**, an $\varepsilon > 0$ and that a tree decomposition of width t is part of the input.*

- *If $\text{gcd}(a', b')$ is odd: If $a = 1$, a' -**INDEPENDENT SET**($\mathcal{G}_{b'}$) is in **P**, else a' -**INDEPENDENT SET**($\mathcal{G}_{b'}$) can be solved in time $a^t \cdot n^{O(1)}$ but not in $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.*
- *If $\text{gcd}(a', b')$ is even: If $a \in \{1, 2\}$, a' -**INDEPENDENT SET**($\mathcal{G}_{b'}$) is in **P**, else a' -**INDEPENDENT SET**($\mathcal{G}_{b'}$) can be solved in time $(2a)^t \cdot n^{O(1)}$ but not in $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.*

The proof heavily uses hidden symmetries of a -independent sets on b -subdivided graphs for different values of a and b . Such symmetries were explored first for a continuous version of a -independent set, in a series of work [14, 17, 15]. We show that these results hold in similar form for a -independent sets as well.

Rational distances. As the distance between any two vertices in a graph is an integer, it makes no sense to consider a -independent sets for noninteger a . However, noninteger distances can be highly relevant if we consider the complexity of the said continuous version of a -independent. The continuous version, introduced by Dearing and Francis [8], is known as δ -dispersion for a positive real distance δ . In this setting, instead of requiring a selection of vertices of a graph G , we allow the selection of *points* that may be on a vertex or somewhere on the continuum of an edge. We fix the length of the edges to 1, which defines a distance relation of the points in the graph G . A δ -dispersed set then is a subset of *points* S where

² The restriction to graphs without short cycles is not explicitly given, but easily observed. We will rely on this restriction later.

every distinct points $p, q \in S$ have distance at least δ ; as studied for example in [35, 14, 17]. The problem δ -DISPERSION is the decision version asking for a δ -dispersed set of size at least k , for some budget k given in the input. It turns out that the notion of $\frac{a}{b}$ -dispersed sets is similar to a -independent sets on b -subdivided graphs. Indeed, a crucial connection between the two types of sets is that $\frac{a}{b}$ -dispersed sets are in one-to-one correspondence to $2a$ -independent sets on the $2b$ -subdivided graph, as follows from a discretization argument by Grigoriev et al. [14]. Particularly, the polynomial time solvable case of a -independent set, as stated in Theorem 1.1, follow from this discretization argument and a characterization of the polynomial time solvable cases of δ -dispersion. Finding a maximum δ -dispersed set is polynomial time solvable if δ is a twice a unit-fraction (including 1 and 2), and all other cases are NP-hard [14]. Further, δ -dispersion when parameterized by the solution size is FPT when $\delta \leq 2$ and otherwise W[1]-hard, as shown by Hartmann et al. [17].

With such connections and Theorem 1.4 at our hands, we can turn the results on a -independent set on b -subdivided graphs into tight results for δ -DISPERSION on bounded treewidth graphs for every fixed *rational* δ .

► **Theorem 1.5** (Section 5). *Let coprime a, b define $\delta = \frac{a}{b}$. If $a \leq 2$, then δ -DISPERSION is in P. For $a \geq 3$, given a tree decomposition of width t , the problem can be solved in time $(2a)^t \cdot n^{O(1)}$, and, assuming SETH, there is no $(2a - \varepsilon)^t \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$.*

Irrational distances. By Theorem 1.5, for a fixed *rational* $\delta = \frac{a}{b}$, finding a maximum size $\frac{a}{b}$ -dispersed set is fixed-parameter tractable in the treewidth of the input graph. This is not necessarily the case for *irrational* δ . Deciding δ -DISPERSION can be as hard as outputting the digits of δ , which for some δ is not even computable. Consider, for example, a path of length ℓ . Then there is a dispersed set of size $k + 1$ if and only if $\frac{\ell}{k} \leq \delta$. Hence it is reasonable to consider the question of efficient algorithms only if δ is *efficiently comparable* to rationals, meaning that there is an algorithm that, given $\frac{x}{y}$, decides whether $\frac{x}{y} \leq \delta$ in time polynomial in $\log x + \log y$.

For every fixed efficiently comparable δ , it is possible to find a maximum δ -dispersed set in an n -vertex graph in time $n^{O(\text{tw}(G))}$, i.e., there is an XP algorithm parameterized by treewidth. This follows from a rounding procedure by Hartmann et al. [17], by which for an n -vertex graph the dispersion number of δ equals to the dispersion number of the smallest rational $\frac{x}{y}$ where $\delta \leq \frac{x}{y}$ with $x \leq 2n$. Using that δ is efficiently comparable, we can find this rational in polynomial time since $x \leq 2n$ and y is in the order of n for a fixed δ . Then it remains to apply the algorithm of Theorem 1.5 to find a maximum $\frac{x}{y}$ -dispersed set.

In contrast, the above algorithm cannot be improved to a fixed-parameter tractable under standard complexity assumptions. As we show, there is a very explicitly defined and efficiently comparable irrational $\delta = (4 \sum_{j=1}^{\infty} 2^{-2^j})^{-1} \approx 0.790085$, for which computing the δ -dispersion number is W[1]-hard parameterized by the treewidth (in fact even for pathwidth), and an according lower bound holds under the Exponential Time Hypothesis (ETH).

► **Theorem 1.6** (Section 6). *There is an efficiently comparable irrational δ for which δ -DISPERSION is W[1]-hard in the pathwidth $\text{pw}(G)$ of the n -vertex input graph G and, assuming ETH, cannot be solved in time $f(\text{pw}(G)) \cdot n^{o(\text{pw}(G))}$ for any computable function f .*

Domination Problems

In addition to distance a -independent set, we perform a similar study of the dual domination problems. As we show, the results for a -independence hold quite similarly for according domination problems. We use a definition that unifies several concepts such as that of a dominating set and a vertex cover.

A *distance- d dominating set* D is a subset of vertices such that every other vertex is at distance at most d to a vertex in D . The literature contains several more distance domination-like problems, which are often quite well understood on bounded treewidth graphs. A well-studied example is *mixed dominating set*, for example [38, 19, 25, 37, 10] and under the name *total covering* [1, 11, 28, 2, 31], which is (even though not directly phrased as such) equivalent to a distance-2 dominating set of the 2-subdivision of a graph G . Similarly, a *vertex-edge dominating set* is a subset of vertices D such that every edge has one of its end vertices in distance at most 1 to a vertex in D , as studied in [23, 40, 39]. More generally, a *distance- d vertex cover* (not to be confused with a *d -path vertex cover*) is a subset of vertices D such that every edge has one of its end vertices in distance at most d to a vertex in D , as studied in [3, 7].

We unify all above concepts by the notion of an *a -walk dominating set* for an integer a , which is a subset of vertices D such that for every edge $e \in E(G)$, there are (possibly identical) vertices $w_1, w_2 \in D$ and a w_1, w_2 -walk of length at most a containing edge e .

► **Observation 1.7.** *For a graph G without isolated vertices, the following notions coincide:*

- *a vertex cover and a 2-walk dominating set,*
- *a dominating set and a 3-walk dominating set,*
- *a vertex-edge dominating set and a 4-walk dominating set,*
- *a distance- d dominating set and a $(2d + 1)$ -walk dominating set, for every $d \geq 1$,*
- *a mixed dominating set in G and a 5-walk dominating set in the 2-subdivision G_2 , and*
- *a distance- d vertex cover and a $(2d + 2)$ -walk dominating set, for every $d \geq 0$.*

Integer distances. Finding a minimum a -walk dominating set is NP-hard for $a = 2$ (i.e., finding minimum vertex cover) and it is not difficult to show that it remains NP-hard for any fixed $a \geq 2$. In some cases, the hardness also extends to when we restrict the input to 2-subdivided graphs: Finding a minimum *mixed dominating set*, i.e., a 5-walk dominating set of 2-subdivided graphs, is NP-hard, as shown by Majumdar [26]. In contrast, there are polynomial time algorithms for $a \in \{2, 4\}$ when the input is a 2-subdivided graph.

► **Theorem 1.8** (Hartmann et al. [18]). *For $a \in \{2, 4\}$, a minimum a -walk dominating set on a 2-subdivided graph can be found in linear time.³*

These examples give a glimpse into the complexity of finding an a -walk dominating set of a b -subdivided graphs for integers a, b . This work settles, for every fixed integer a, b , whether finding a minimum a -walk dominating set of a b -subdivided graph is polynomial time solvable or NP-hard, and additionally settles the parameterized complexity for the solution size as parameter (as color-coded in Figure 2). Formally, let $\bar{\gamma}_a(G)$ be the minimum cardinality of an a -walk dominating set of a graph G . For a graph class \mathcal{G} , let a -WALK DOMINATING SET(\mathcal{G}) be the according decision problem, which is, given a graph $G \in \mathcal{G}$ and an integer k , deciding whether $\bar{\gamma}_a(G) \geq k$.

► **Theorem 1.9** (\star). *a -WALK DOMINATING SET(\mathcal{G}_b) is*

- *polynomial time solvable if $b = ca$ or if $b = c\frac{a}{2}$ and $\frac{a}{2}$ is even for some integer c ; and is NP-hard for all other integers a, b ; and is*
- *fixed-parameter tractable for the parameter solution size if $\frac{a}{b} < 3$; and $W[2]$ -hard for all other integers a, b .*

³ The original statement is about a continuous covering problem, but can be phrased as here by using a discretization argument given in the same work.

$a \backslash b$	1	2	3	4	5	6
1	$ V $	$ V + E $	$ V + 2 E $	$ V + 3 E $	$ V + 4 E $	$ V + 5 E $
2	VC	1/2-COVER	VC+ E	1/4-COVER	VC+2 E	1/6-COVER
3	DS	DS(G_2)	$ V $	DS+ E	DS(G_2) + E	$ V + E $
4	VED	1-COVER		1/2-COVER	VED+ E	1/3-COVER
5	2-DS	MDS		MDS	$ V $	2-DS+ E
6		3/2-COVER	VC	DS(G_2)		1/2-COVER
7	3-DS					
8		2-COVER		1-COVER		2/3-COVER
9	4-DS		DS			DS(G_2)

■ **Figure 2** Some problems, such as VERTEXCOVER, (MIXED)DOMINATINGSET, VERTEXEDGE DOMINATION, (or solution sizes) corresponding to a -WALK DOMINATING SET of a graph G_b for small values of a, b , where $V = V(G)$ and $E = E(G)$. A light green cell indicates a polynomial time solvable case, an orange NP-hardness and FPT, and a dark red NP-hardness & W[2]-hardness.

We note that a -WALK DOMINATING SET(\mathcal{G}_b) is polynomial time solvable for the same set of integers a, b where a -INDEPENDENT SET(\mathcal{G}_b) is polynomial time solvable. In contrast, the threshold which separates the fixed-parameter tractable cases from the W[1]-hard/W[2]-hard cases is shifted, which should be expected as VERTEX COVER and DOMINATING SET are to be separated by this threshold.

Further, we study the problem parameterized by treewidth. Again, intuitively, the a -walk dominating set problem is harder for larger a . Indeed, for many cases the notion of an a -walk dominating set corresponds to a known problem (as in Observation 1.7) where the literature knows a matching upper and lower bound with a in the base of an exponential run time for graphs of bounded treewidth, assuming SETH. This is also the case for $a = 5$ on 2-subdivided graphs, as this corresponds to a mixed dominating set.

▶ **Theorem 1.10** ([30, 36, 5, 24, 10]). *For $a \in \{2\} \cup \{3, 5, 7, \dots\}$, given a tree decomposition of width t of an n vertex input graph, a minimum a -walk dominating set can be found in time $a^t \cdot n^{O(1)}$, and, assuming SETH, there is no $(a - \varepsilon)^t \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$. Moreover, for $a = 5$ this even holds when the input graph is restricted to 2-subdivided graphs.*

We refine Theorem 1.10 by including also even distances $a \geq 4$ and by considering the restriction of a -WALK DOMINATING SET to b -subdivided graphs, beyond the case $a = 5$ and $b = 2$. We determine the optimal base of the exponent for all integers a and b . As it turns out, the optimal base depends on a and b in a very subtle way.

▶ **Theorem 1.11** (\star). *Let a', b' integers with $\gcd(a', b') = c$ and $ca = a'$ and $cb = b'$. Assume SETH, $\varepsilon > 0$, and that a tree decomposition of width t is part of the input.*

- *If $\gcd(a', b')$ is odd: If $a = 1$, a' -WALK DOMINATING SET($\mathcal{G}_{b'}$) is in P, else a' -WALK DOMINATING SET($\mathcal{G}_{b'}$) can be solved in time $a^t \cdot n^{O(1)}$ but not in $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.*
- *If $\gcd(a', b')$ is even: If $a \in \{1, 2\}$, a' -WALK DOMINATING SET($\mathcal{G}_{b'}$) is in P, else a' -WALK DOMINATING SET($\mathcal{G}_{b'}$) can be solved in time $(2a)^t \cdot n^{O(1)}$ but not $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.*

The proof of Theorem 1.11 heavily uses hidden symmetries of a -walk dominating sets on b -subdivided graphs, which are of similar nature as for independent sets. Such symmetries were explored first for a continuous version of a -walk dominating set [18]. We show that these results hold in similar form for a -walk dominating set as well.

Regarding distance- d domination, our results so far imply the following.

- **Corollary 1.12.** *Finding a minimum distance- d dominating set in b -subdivided graphs*
- *is polynomial time solvable if b is a multiple of $2d + 1$, otherwise is NP-hard;*
 - *if b is not a multiple of $2d+1$, with $\gcd(2d+1, b) = c$ can be solved in time $((2d+1)/c)^t \cdot n^{O(1)}$ if a tree decomposition of width t is part of the input, and, assuming SETH, cannot be solved in time $((2d + 1)/c - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$; and*
 - *fixed-parameter tractable for the parameter solution size if $\frac{2d+1}{b} < 3$; and W[2]-hard for all other values of d, b .*

Rational distances. The continuous version of a -walk dominating set, as introduced by Shier [33], is known as δ -covering for a positive real distance δ . Similarly to δ -dispersion, we allow the selection of *points* that may be on a vertex or somewhere on the continuum of an edge. We fix the length of the edges to 1, which defines a distance relation of the points in the graph G . A δ -cover is a set of points S that covers every point p in the graph, that is there is a point $q \in S$ such that p, q have distance at most δ ; as studied for example in [27, 34] and receiving renewed attention lately [16, 18]. The problem δ -COVERING is the decision version asking for a δ -cover of size at most k , for some budget k given in the input. The notion of a $\frac{a}{b}$ -cover is quite similar to an a -walk dominating set of a b -subdivided graph; though the connection is more subtle compared to the independence problems. Based on a discretization argument by Hartmann et al. [18] we show that $\frac{a}{b}$ -covers are in one-to-one correspondence to $2a$ -walk dominating sets on b -subdivided graphs, if b is even; while if b is odd, $\frac{a}{b}$ -covers are in one-to-one correspondence to $4a$ -walk dominating sets on $2b$ -subdivided graphs. Particularly, we obtain the polynomial time solvable cases of δ -covering based on this connection. Finding a minimum δ -cover is polynomial time solvable if δ is a unit-fraction and otherwise NP-hard [18]. By the same work, δ -COVERING parameterized by the solution size is FPT in case $\delta < \frac{3}{2}$ an otherwise W[2]-hard. (We observe a similar dichotomy for a -independent sets on b -subdivided graphs, as stated in Theorem 1.9.)

With such connections and Theorem 1.11 at our hands, we can turn the results on a -independent set on b -subdivided graphs into tight results for δ -COVERING on bounded treewidth graphs for every fixed *rational* δ .

► **Theorem 1.13** (*). *Let a', b' integers with $\gcd(a', b') = c$ and $ca = a'$ and $cb = b'$. Assume SETH, $\varepsilon > 0$, and that a tree decomposition of width t is part of the input.*

- *$\frac{a'}{b'}$ -COVERING for $a = 1$ is in P; if $a \geq 2$ and b is odd, can be solved in time $(4a)^t \cdot n^{O(1)}$ but not in $(4a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$; if $a' \geq 2$ and b is even, can be solved in time $(2a)^t \cdot n^{O(1)}$ but not in $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.*

Irrational distances. By Theorem 1.13, for every fixed *rational* $\delta = \frac{a}{b}$, finding a minimum $\frac{a}{b}$ -cover is fixed-parameter tractable parameterized by the treewidth of the input graph. As is the case for δ -covering, this is not necessarily true for *irrational* δ . Deciding δ -COVERING can be as hard as outputting the digits of δ , which for some δ is not even computable. For a path of length ℓ , there is a covering set of size k if and only if $\delta \geq \frac{\ell}{2k}$. Hence it is reasonable to consider only δ which are efficiently comparable.

For every fixed *efficiently comparable* δ , it is possible to find a minimum δ -cover in time $n^{O(\text{tw}(G))}$, i.e., there is an XP algorithm parameterized by treewidth. This follows from a rounding procedure by Tamir [34], by which for an n -vertex graph the covering number of δ equals to the covering number of the largest rational $\frac{x}{y} \leq \delta$ with $x \leq 2n$. Using that δ is

efficiently comparable, we can find this rational in polynomial time since $x < 2n$ and y is in the order of n for a fixed δ . Then it remains to apply the algorithm of Theorem 1.13 to find a minimum $\frac{x}{y}$ -covering set.

In contrast, the above algorithm cannot be improved to a fixed-parameter tractable under standard complexity assumptions. As we show, there is a very explicitly defined and efficiently comparable irrational $\delta' = (2 \sum_{j=1}^{\infty} 2^{-2^j})^{-1} \approx 0.395043$, for which computing the δ' -covering number is $W[1]$ -hard parameterized by the treewidth (in fact even for pathwidth), and an according lower bound under ETH.

► **Theorem 1.14** (★). *There is an efficiently comparable irrational δ' such that δ' -COVERING is $W[1]$ -hard in the pathwidth $\text{pw}(G)$ of the n -vertex input graph G and, assuming ETH, cannot be solved in time $f(\text{pw}(G)) \cdot n^{o(\text{pw}(G))}$ for any computable function f .*

2 Preliminaries

All our graphs are simple and undirected. Usually we assume that our graphs as input do not contain isolated vertices, as they can easily be preprocessed for the studied problems.

b -Subdivision. For a graph G and an integer b , the b -subdivision of G , denoted as G_b , results from G by replacing every edge $\{u, v\} \in E(G)$ by a u, v -path of length b . For example, $G = G_1$. For an edge $\{u, v\} \in E(G)$ and $\beta \in \{0, \dots, b\}$, let $v(u, v, \beta)$ be the unique vertex on the unique shortest u, v -path in G_b with distance β to u and distance $b - \beta$ to v . Let \mathcal{G}_b be the class of every graph H that is the b -subdivision G_b of a graph G .

Point space. For a graph G , we assume that its edges have unit length. Let $p(u, v, \lambda)$, for an edge $\{u, v\} \in E(G)$ and a real $\lambda \in [0, 1]$, denote the *point* on the edge $\{u, v\}$ with distance λ to u and distance $1 - \lambda$ to v . Hence $p(u, v, \lambda)$ coincides with $p(u, v, 1 - \lambda)$, and the point $p(u, v, 0)$ coincides with the vertex u . By $P(G)$ we denote the set of points of a graph G . Let $d(p, q)$, for two points $p, q \in P(G)$, denote the distance of p, q of the underlying metric space on $P(G)$.

Graph parameters. We use the following well known relation of graph parameters. By $\nu(G)$ we denote the maximum size of a matching in a graph G . A tree decomposition (T, β) of G consists of a tree T and a mapping β from the vertices of T (referred to as *nodes*) to subsets of $V(G)$ (referred to as *bags*), such that (1) $\bigcup_{n \in V(T)} \beta(n) = V(G)$, (2) $\{u, v\} \in E(G)$ implies a node $n \in V(T)$ with $\{u, v\} \subseteq \beta(n)$, and (3) for nodes $n_1, n_2, n_3 \in V(T)$ where n_2 lies on the path from n_1 to n_3 , we have $\beta(n_1) \cap \beta(n_3) \subseteq \beta(n_2)$. The width of a tree decomposition is the maximum of $|\beta(n)| - 1$ for all $n \in V(T)$. A path decomposition of G is a tree decomposition (P, β) where P is a path. We let $(\beta(n_1), \dots, \beta(n_t))$ denote the path decomposition using a path (n_1, \dots, n_t) . The *treewidth* $\text{tw}(G)$ of a graph is the minimum width of a tree decomposition, and likewise the *pathwidth* $\text{pw}(G)$ is the minimum width of a path decomposition.

It is well known that $2\nu(G) \geq \text{pw}(G) \geq \text{tw}(G)$. Hence a parameterized algorithm with the treewidth as parameter is more general result than using the pathwidth (assuming a respective decomposition is given in the input). On the other hand, a lower bound for the parameter pathwidth also holds for the parameter treewidth.

Efficiently comparable real. A real δ is *efficiently comparable* if there is an algorithm that, given $\frac{y}{x}$, decides whether $\frac{x}{y} \leq \delta$ in time polynomial in $\log x + \log y$.

3 Independent and Dispersed Sets

This section explores the close relationship between a -independent sets and δ -dispersed sets. Our goal is to establish the following two transformations of δ -dispersed sets also for independent sets. The first relates the dispersion number to the subdividing the graph.

► **Lemma 3.1** ([14]). *For every real $\delta > 0$ and integer $c \geq 1$, $\delta\text{-disp}(G) = c\delta\text{-disp}(G_c)$.*

The second relates the dispersion number of the same graph but of different distances. For certain distances the solution size differs by exactly one point for every edge.

► **Lemma 3.2** ([17, 15]). *$\delta\text{-disp}(G) + |E(G)| = \frac{\delta}{1+\delta}\text{-disp}(G)$ for every real $\delta > 0$ and graph G without cycles of length $< \delta$.*

In a key result later, we will apply Lemma 3.2 multiple time, stated as follows. (The proof thereof and of other statements marked with (\star) can be found in the full version of this paper.)

► **Corollary 3.3** (\star) . *$\delta\text{-disp}(G) = \frac{\delta}{1+b\delta}\text{-disp}(G) - b|E(G)|$ for an integer $b \geq 1$, real $\delta > 0$ and graph G without cycles of length $< \delta$.*

To obtain Lemma 3.1 and Lemma 3.2 in terms of a -independent sets, we consider certain normalized dispersed sets. To that end, recall that a point $p(u, v, \lambda) \in P(G)$ is c -simple if λ is a multiple of c . Further, let $S \subseteq P(G)$ be c -simple if all its points are c -simple. The good news is that there is a direct correspondence of b -simple $\frac{a}{b}$ -dispersed sets in a graph G and a -independent sets in the b -subdivision G_b .

► **Observation 3.4.** *Let $k \in \mathbb{N}$. There is an a -independent set of size k in G_b , if and only if there is a b -simple $\frac{a}{b}$ -dispersed set of size k in G .*

Proof. The vertex $v(u, v, \beta)$ for an edge $\{u, v\} \in E(G)$ and $\beta \in \{0, \dots, b\}$ corresponds to the b -simple point $p(u, v, \frac{\beta}{b})$ and vice versa. The distance between vertices corresponds to the same distance of corresponding points multiplied by the factor $\frac{1}{b}$. ◀

Unfortunately, there may be no minimum $\frac{a}{b}$ -dispersed set that is b -simple. On the positive side, a minimum $\frac{a}{b}$ -dispersed set S can be modified to a $2b$ -simple dispersed set S^* of same size. Actually, one can observe that S^* is not b -simple only because of certain points in S .

► **Lemma 3.5** ([14]). *For an $\frac{a}{b}$ -dispersed set S , the following set $S^* = \{p^* \mid p \in S\}$ is $2b$ -simple and $\frac{a}{b}$ -dispersed. For each point $p(u, v, \lambda)$ with $\lambda \in (\frac{i}{b} - \frac{1}{2b}, \frac{i}{b} + \frac{1}{2b})$ for some integer $i \geq 0$, let $p^* := p(u, v, \frac{i}{b})$; for all other $p \in S$, let $p^* = p$.*

► **Corollary 3.6.** *$\frac{a}{b}\text{-disp}(G) = \alpha_{2a}(G_{2b})$ for every $a, b \in \mathbb{N}_+$ and graph G .*

This already implies a subdivision argument almost as for a -independent sets (Lemma 3.1).

► **Lemma 3.7.** *For every graph G , $\alpha_a(G_b) = \alpha_{ca}(G_{cb})$ if c is odd, or a, b are even.*

Proof. First consider that a and b are even. Then $\alpha_a(G_b) = \alpha_{2a'}(G_{2b'})$ for some integers a', b' . Then $\alpha_{2a'}(G_{2b'}) = \frac{a'}{b'}\text{-disp} = \frac{ca'}{cb'}\text{-disp} = \alpha_{2ca'}(G_{2cb'}) = \alpha_{ca}(G_{cb})$, because of Corollary 3.6.

Now consider that c is odd. Clearly, an a -independent set in G_b corresponds to a ca -independent set in G_{cb} . For the reverse direction, let $I \subseteq V(G_{cb})$ be a ca -independent set of G_{cb} . Consider the corresponding a -dispersed set $S \subseteq V(G_b)$ in G_b , which is bc -simple. We apply the construction of Lemma 3.5. As c is odd, there is no point $p \in S$ with edge position $\frac{1}{2}$. Hence the construction only produces points with edge position 0 or 1. That is, the constructed set S^* is 1-simple, and hence S^* corresponds to an a -independent set in G_b . ◀

Next, we obtain the second connection (Lemma 3.2) quite similarly for dispersed sets. That is, we relate a -independent sets in the subdivided graphs G_b and G_{a+b} . The basic idea is to translate the independent set to a dispersed set and apply Lemma 3.2. However, the construction of Lemma 3.2 does not preserve simplicity. Hence we adapt the construction slightly such that it always maps b -simple inputs to $(a+b)$ -simple outputs. As we show in the appendix, the correctness follows with almost the same proof as for Lemma 3.2.

► **Lemma 3.8** (\star). *Let G be a graph without a cycle of length $< \frac{a}{b}$. A b -simple $\frac{a}{b}$ -dispersed set S implies an $(a+b)$ -simple $\frac{a}{a+b}$ -dispersed set of size $|S| + |E(G)|$. Further, an $(a+b)$ -simple $\frac{a}{a+b}$ -dispersed set S' implies a b -simple $\frac{a}{b}$ -dispersed set of size $|S'| - |E(G)|$.*

Equipped with Lemma 3.8, we obtain an result analogous to Lemma 3.2.

► **Theorem 3.9.** $\alpha_a(G_b) + |E(G)| = \alpha_a(G_{a+b})$ if graph G_b contains no cycle of length $< a$.

Proof. Let I be an a -independent set I in G_b . Then I corresponds to a b -simple $\frac{a}{b}$ -dispersed set S in G , by Observation 3.4. Lemma 3.8 maps S to an $(a+b)$ -simple $\frac{a}{a+b}$ -dispersed set S' in G of size $|I| + |E(G)|$. This construction is applicable since G_b contains no cycle of length $< a$, and hence G contains no cycle of length $< \frac{a}{b}$. Then the $(a+b)$ -simple set S' corresponds to an a -independent set in G_{a+b} of size $|I| + |E(G)|$, by Observation 3.4. That means $\alpha_a(G_b) + |E(G)| \leq \alpha_a(G_{a+b})$.

Vice versa, let I' be an a -independent set in G_{a+b} . Then I' corresponds to an $(a+b)$ -simple $\frac{a}{a+b}$ -dispersed set S' in G of size $|I'|$, by Observation 3.4. Lemma 3.8 maps S' to an a -simple $\frac{a}{b}$ -dispersed set S in G of size $|I'| - |E(G)|$. Then the b -simple set S corresponds to an a -independent set in G_b of size $|I'| - |E(G)|$, by Observation 3.4. That means $\alpha_a(G_b) + |E(G)| \geq \alpha_a(G_{a+b})$. ◀

Now with these two relation of independent sets established (Corollary 3.6 and Theorem 3.9), we easily obtain the the integers a, b for which finding a maximum a -independent set on b -subdivided graphs is polynomial time solvable. A simple case is, that a is odd and b is a multiple of a . By Lemma 3.7, this is equivalent to finding a maximum 1-independent set on a $\frac{a}{b}$ -subdivided graph, which trivially consist of all vertices. In the case that a is even and b is a multiple of a , and that $\frac{a}{2}$ is even and b is a multiple of $\frac{a}{2}$, Lemma 3.7 and Theorem 3.9 allow to reduce the problem to a polynomial time solvable case from Theorem 1.2. Theorem 3.9 is applicable as in above cases $\frac{a}{b} \leq 2$.

► **Theorem 3.10.** a -INDEPENDENT SET on b -subdivided graph is polynomial time solvable if b is a multiple of a , or $\frac{a}{2}$ is even and b is an odd multiple of $\frac{a}{2}$.

4 Independent Set with Parameter Solution Size

This section settles the parameterized complexity of finding a maximum a -independent set on b -subdivided graphs with the solution size as parameter, for every integer a, b . These results are also color-coded in Figure 1 and summarized as follows.

- **Theorem 4.1** (Theorem 1.2 restated). a -INDEPENDENT SET(G_b) is
- polynomial time solvable if $b = ca$ or if $b = c\frac{a}{2}$ and $\frac{a}{2}$ is even for some integer c ; and NP-hard for all other integers a, b ; and is
 - fixed-parameter tractable for the solution size as parameter if $\frac{a}{b} < 2$ or if $\frac{a}{b} = 2$, b even; and W[1]-hard for all other integers a, b .

The polynomial time solvable cases are already settled by Theorem 3.10. As is well-known, 2-INDEPENDENT SET (on general graphs, hence 1-subdivided graphs) is W[1]-hard [9]. This also puts all cases where $\frac{a}{b} = 2$ and b odd to W[1]-hard, by applying Lemma 3.7; while all over cases with $\frac{a}{b} = 2$ are polynomial time solvable. The fixed-parameter tractable cases rely on bounding the solution size below by the size of a maximum matching in a graph G , denoted as $\nu(G)$; similarly as for the covering problem [17].

► **Lemma 4.2.** $\nu(G) \leq \alpha_a(G_b)$ for every graph G and integers a, b with $\frac{a}{b} < 2$.

Proof. If $\frac{a}{b} \leq 1$, then $V(G)$ is an a -independent set in G_b . Since $|V(G)| \geq \nu(G)$, the statement follows for $\frac{a}{b} \leq 1$. Otherwise, consider a maximum matching $M \subseteq E(G)$. Then the two vertices $v(u, v, \lfloor \frac{b}{2} \rfloor)$ and $v(u', v', \lfloor \frac{b}{2} \rfloor)$ for distinct matching edges $\{u, v\}, \{u', v'\} \in M$ have distance at least $b + 2\lfloor \frac{b}{2} \rfloor \geq b + (b - 1) = 2b - 1 \geq a$. The last inequality holds since otherwise $2b \leq a$ in contradiction to $\frac{a}{b} < 2$. In conclusion $\{v(u, v, \lfloor \frac{b}{2} \rfloor) \mid \{u, v\} \in M, u \prec v\}$ is an a -independent set of G_b , where \prec is an arbitrary ordering of $V(G)$. ◀

As the maximum size of a matching in a graph upper bounds the treewidth, an FPT-algorithm results from a win-win situation. Either the input asks for an independent set that is relatively large compared to $\nu(G)$ and hence also compared to the treewidth, in which case we can use Theorem 1.3, or the answer is trivially “yes”.

► **Lemma 4.3** (★). For every a, b with $\frac{a}{b} < 2$, a -INDEPENDENT SET(\mathcal{G}_b) is FPT for the parameter solution size.

It remains to show W[1]-hardness if $\frac{a}{b} > 2$, which follows from two parameter preserving reductions from INDEPENDENT SET showing W[1]-hardness, the first for $\frac{a}{b} \in (2, 3)$, the second for $\frac{a}{b} \geq 3$; similarly as for the covering problem [17].

► **Lemma 4.4** (★). For integers a, b where $\frac{a}{b} > 2$, a -INDEPENDENT SET(\mathcal{G}_b) is W[1]-hard.

5 Dispersion for Rational Distances

This section derives the upper and lower bounds under SETH for finding a minimum a -independent set on b -subdivided graphs for the parameter treewidth, for all integers a, b . All lower bounds follow from the mere lower bound for even distance $a \geq 6$.

► **Theorem 5.1** (★). Let $a \geq 6$ be even. Assuming SETH, a -INDEPENDENT SET has no $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$, even when the input is restricted to 2-subdivided graphs without a cycle of length $< a$.

In fact, we show this lower bound assuming the Primal Pathwidth SETH, recently introduced by Lampis [22]. We provide the details in the full version.

► **Theorem 5.2** (Theorem 1.4, Theorem 1.5 combined). Let integers a', b' define $\text{gcd}(a', b') = c$ and $ca = a'$ and $cb = b'$. Assume SETH, an $\varepsilon > 0$ and that a tree decomposition of width t is part of the input.

- If $\text{gcd}(a', b')$ is odd: If $a = 1$, a' -INDEPENDENT SET($\mathcal{G}_{b'}$) is in P, else a' -INDEPENDENT SET($\mathcal{G}_{b'}$) can be solved in time $a^t \cdot n^{O(1)}$ but not in $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.
- If $\text{gcd}(a', b')$ is even: If $a \in \{1, 2\}$, a' -INDEPENDENT SET($\mathcal{G}_{b'}$) is in P, else a' -INDEPENDENT SET($\mathcal{G}_{b'}$) can be solved in time $(2a)^t \cdot n^{O(1)}$ but not in $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.
- If $a' \in \{1, 2\}$, $\frac{a'}{b}$ -DISPERSION is in P; while if $a' \geq 3$ can be solved in $(2a)^t \cdot n^{O(1)}$ time but not in time $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$.

Proof. First, we consider that $c = \gcd(a', b')$ is odd. Then ca -INDEPENDENT SET(\mathcal{G}_{cb}) is equivalent to a -INDEPENDENT SET(\mathcal{G}_b) by Lemma 3.7. In case $a = 1$, then 1-INDEPENDENT SET(\mathcal{G}_b) has the trivial 1-independent set $|V(G)|$. Otherwise, a -INDEPENDENT SET(\mathcal{G}_b) can be solved in time $a^t \cdot n^{O(1)}$ using Theorem 1.3.

For the lower bound we use that $yb = 1 + xa$ for some integers x, y . Assume SETH. Then we know from Theorem 1.3 that a -INDEPENDENT SET(\mathcal{G}_1) has no $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$. Particularly, this lower bound relies on graphs without a cycle of length $< a$. Then Theorem 3.9 applied x times yields that a -INDEPENDENT SET(\mathcal{G}_{1+xa}), and equivalently a -INDEPENDENT SET(\mathcal{G}_{yb}), also has no $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$. Thus especially a -INDEPENDENT SET(\mathcal{G}_b) has no $(a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ time algorithm. This settles the cases for independent set with odd $\gcd(a', b')$.

Next, we consider that $c = \gcd(a', b')$ is even, hence that $c = 2\hat{c}$ for some integer \hat{c} . Then $(2\hat{c}a)$ -INDEPENDENT SET($\mathcal{G}_{2\hat{c}b}$) is equivalent to $2a$ -INDEPENDENT SET(\mathcal{G}_{2b}), by Lemma 3.7. Again, by Theorem 1.3, $2a$ -INDEPENDENT SET(\mathcal{G}_{2b}) can be solved in time $(2a)^t \cdot n^{O(1)}$.

If $a \in \{1, 2\}$, then a -DISPERSION is polynomial time solvable, by Theorem 1.1. Further, as $\frac{a}{b} \leq 2$, applying Lemma 3.2 yields that also $\frac{a'}{b'}$ -DISPERSION is polynomial time solvable (as also observed in [14]). By Corollary 3.6, $2a$ -INDEPENDENT SET(\mathcal{G}_{2b}) is equivalent to $\frac{a'}{b'}$ -DISPERSION, hence polynomial time solvable.

In case $a \geq 3$, and assuming SETH, Theorem 5.1 provides that $2a$ -INDEPENDENT SET(\mathcal{G}_2) has no $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$. Particularly, this lower bound does not rely on graphs with a cycle of length $< a$. Since a, b are co-prime, again Theorem 3.9 applies, and we obtain that $2a$ -INDEPENDENT SET(\mathcal{G}_{2b}) has no $(2a - \varepsilon)^{\text{pw}(G)} \cdot n^{O(1)}$ for any $\varepsilon > 0$. This settles the cases for independent set with even $\gcd(a', b')$.

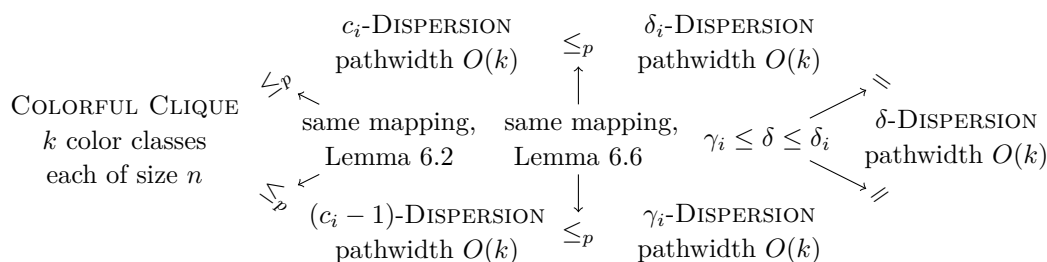
Finally, $\frac{a'}{b'}$ -DISPERSION is equivalent to $\frac{a}{b}$ -DISPERSION by definition. Then $\frac{a}{b}$ -DISPERSION is equivalent to $2a$ -INDEPENDENT SET(\mathcal{G}_{2b}) by Corollary 3.6. Since a, b are co-prime, $2a, 2b$ have greatest common divisor 2. By the discussion for an greatest common divisor which is even, we follow that $\frac{a}{b}$ -DISPERSION has an $(2a)^t \cdot n^{O(1)}$ time algorithm, and, assuming SETH, has no $(2a - \varepsilon)^t \cdot n^{O(1)}$ time algorithm for any $\varepsilon > 0$. ◀

6 Dispersion for Irrational Distance

This section derives the hardness result for computing a maximum δ -dispersed set for the *efficiently comparable* irrational $\delta = (\sum_{j \in [i+1]} 2^{2-2^j})^{-1} \approx 0.790085$.

► **Theorem 6.1** (Theorem 1.6 restated). *There is an efficiently comparable irrational δ for which δ -DISPERSION is $W[1]$ -hard in the pathwidth $\text{pw}(G)$ of the n -vertex input graph G and, assuming ETH, cannot be solved in time $f(\text{pw}(G)) \cdot n^{o(\text{pw}(G))}$ for any computable function f .*

Our proof is based on two main reductions. The first one is fairly standard: It is a reduction from a colorful clique problem to c_i -DISPERSION, where $c_i = 2^{2^i}$ is a sufficiently large integer (polynomially bounded in the number of vertices of the colorful clique problem). The reduction is robust in the sense that it is simultaneously a reduction to $(c_i - 1)$ -DISPERSION as well, i.e., the yes/no answer does not change if we reduce the radius by 1. The second reduction is the main nontrivial part of the proof: We reduce c_i -DISPERSION to δ_i -DISPERSION for some rational δ_i in a robust way. That is, the reduction can be interpreted also as a reduction from $(c_i - 1)$ -DISPERSION to γ_i -DISPERSION for some $\gamma_i < \delta_i$. Thus if the source instance has the same yes/no answer for radius c_i and $c_i - 1$, then the target problem has the same answer for any radius $\delta \in [\gamma_i, \delta_i]$. We manage to define γ_i, δ_i in such a way that there is an irrational δ that is in $[\gamma_i, \delta_i]$ for every i . Thus for every i , the problem can be reduced to δ -DISPERSION.



■ **Figure 3** Reductions used for the proof of Theorem 6.1. Lemma 6.2 is simultaneously a reduction to c_i -DISPERSION and to $(c_i - 1)$ -DISPERSION. ‘Agnostic’ to whether the distance is c_i or $c_i - 1$, Lemma 6.6 reduces to DISPERSION with distance γ_i respectively δ_i . These rationals γ_i and δ_i form an approximation of δ from below and above. Thus the combined reduction reduces to δ -DISPERSION.

Our main tools so far are subdividing (using Lemma 3.1), and translating (using Lemma 3.2). Translating only applies to instances $\langle \delta, G, k \rangle$ where the graph G contains no cycle of length $< \delta$. Hence, for convenience, let DISPERSION^* be the DISPERSION problem restricted to instances where the graph G contains no cycle of length $< \delta$.

The starting point is COLORFUL CLIQUE where, given a graph G and an integer k and a proper k -coloring of G , the task is to decide whether G contains a k -clique that contains exactly one vertex of each color. It is known [6] that COLORFUL CLIQUE is $\text{W}[1]$ -hard parameterized by the solution size k and, assuming ETH , has no $f(k) \cdot n^{o(k)}$ time algorithm for any computable function f .

The first reduction is based on a reduction from COLORFUL CLIQUE to the task of finding a maximum a -independent set with a as part of the input, as given by Katsikarelis et al. [20]. We output a graph with enough leeway such that the δ -dispersion number does not change for δ in the interval $[c, c - 1]$ for some integer c . Doing so, our construction constitutes a reduction from COLORFUL CLIQUE to c -DISPERSION and, at the same time, a reduction from COLORFUL CLIQUE to $(c - 1)$ -DISPERSION. Further, we make sure that the construction does not introduce any short cycles.

► **Lemma 6.2** (\star). *There is a polynomial time reduction that, given a COLORFUL CLIQUE -instance $\langle G, k \rangle$ with k color classes of size n , outputs a graph G' of pathwidth $O(k)$ and integer k' , such that: $\langle G, k \rangle$ is a yes-instance of COLORFUL CLIQUE if and only if $\langle G', k' \rangle$ is a yes-instance of $32n$ -DISPERSION * if and only if $\langle G', k' \rangle$ is a yes-instance of $(32n - 1)$ -DISPERSION * .*

Next, let us define δ in an abstract sense. Distance δ is approximated by values δ_i and γ_i from below and above with increasing precision. The idea is to define distance δ_i by a fraction $\frac{a_i}{b_i}$ that results from applying translation (Lemma 3.2) and subdivision (Lemma 3.1) to a (quite large) distance c_i . Later, our second reduction then reduces to δ_i -DISPERSION and to γ_i -DISPERSION by applying the according translation and subdivision.

► **Definition 6.3.** *Let $c_1, c_2, \dots \in \mathbb{N}_+$ be an increasing integer sequence. Then $a_0 = b_0 = 1$ and, for $i \geq 1$,*

$$a_i := a_{i-1}c_i, \quad b_i := b_{i-1}c_i + 1, \quad \delta_i := \frac{a_i}{b_i}, \quad \gamma_i := \frac{a_i - a_{i-1}}{b_i - b_{i-1}}.$$

This defines $\delta := \lim_{i \rightarrow \infty} \delta_i$.

The sequence is decreasing and bounded from below, hence the limit $\delta := \lim_{i \rightarrow \infty} \delta_i$ is well defined.

44:14 Independence and Domination on Bounded-Treewidth Graphs

► **Lemma 6.4** (★). For $i \geq 2$, and $\gamma_i, \delta, \delta_i$ as defined in Definition 6.3, we have $\gamma_{i-1} < \gamma_i < \delta < \delta_i < \delta_{i-1}$

We obtain nice computational properties if we use the double-exponential sequence for c_i .

► **Lemma 6.5.** Using sequence $c_i := 2^{2^i}$ for Definition 6.3, integers a_i, b_i are polynomial-time computable given c_i , and a_i is polynomial in c_i . Further, δ is efficiently comparable.

Proof. We observe that $a_i = \prod_{j=1}^i c_j = 2^{2^1} \cdot 2^{2^2} \cdots 2^{2^i} = 2^{2^{i+1}-2} = 2^{2^{i+1}}/4 = c_i^2/4$, hence that a_i is polynomial-time computable given c_i and is polynomial in c_i . Further, $b_i = (\dots((c_1 + 1)c_2 + 1)\dots)c_i + 1 = \sum_{j=1}^i c_j c_{j+1} \cdots c_i \leq i \cdot a_i = \log \log c_i \cdot c_i^2/4$. Hence b_i is polynomial-time computable given c_i , by at most $2i$ multiplications and additions of integers that are polynomial in c_i .

Let us determine δ . We let $\eta_i = \sum_{j=1}^i 2^{-2^j}$. Then

$$b_i = \sum_{j=1}^{i+1} \prod_{k=j}^i c_k = \prod_{k=1}^i c_k \sum_{j=1}^{i+1} \prod_{k=1}^{j-1} c_k^{-1} = a_i \sum_{j=1}^{i+1} 2^{-(2^j-2)} = 4a_i \eta_{i+1}.$$

This yields $\delta = \lim_{i \rightarrow \infty} \frac{a_i}{b_i} = \left(4 \sum_{j=1}^{\infty} 2^{-2^j}\right)^{-1} \approx 0.790085$.

We show that δ is efficiently comparable, that is there is an algorithm that, given a rational $\frac{x}{y}$, decides whether $\frac{x}{y} < \delta$ in time polynomial in $\log x + \log y$. Our algorithm first checks whether $\frac{1}{2} < \frac{x}{y} < 1$, and if not can conclude that $\frac{x}{y} < \delta$ or $\frac{x}{y} > \delta$. Instead of comparing $\frac{x}{y}$ with δ , we compare their inverses $\frac{y}{x}$ and δ^{-1} , and output the negated answer. In base 2, we obtain that $\delta^{-1} = 1.01000100000001\dots$, which is that the i -th digit 1 succeeds the $(i-1)$ -st digit 1 in 2^i steps. Hence the first j digits (after the dot) of δ^{-1} can be output in time polynomial in j . We may also output the first j digits (after the dot) of $\frac{y}{x}$ in time polynomial in j . If there is a position where the digits differ, we can conclude whichever is larger. It remains to show that there will be a difference in the first $j = O(\log x + \log y)$ digits of δ^{-1} and $\frac{y}{x}$, hence that comparing the first j digits suffices. Indeed, the digits of $\frac{y}{x}$ as a string cannot contain the substring $0^{\lceil \log x \rceil} 1$ after the dot. Otherwise $\frac{y}{x} + \frac{y}{x}$ contains the substring $0^{\lceil \log x \rceil - 1} 1$ after the dot, and by induction $\frac{y}{x} \cdot x = y$ contains the substring 1 after the dot, in contradiction that y is integer. In contrast, the first $O(\log x)$ digits of δ^{-1} do include the substring $0^{\lceil \log x \rceil} 1$. Thus it suffices to compare the first $\log x$ digits of δ^{-1} and $\frac{y}{x}$, which concludes the proof. ◀

The following lemma lies at the heart of our result: the definition of the sequences a_i, b_i, c_i allows us to reduce c_i -DISPERSION to δ_i -DISPERSION and, at the same time, $(c_i - 1)$ -DISPERSION to γ_i -DISPERSION with the same reduction.

► **Lemma 6.6.** Let sequence $(c_i)_{i \geq 1}$ be as in Definition 6.3. There is a polynomial-time reduction that, given integers c_i, k' and a graph G' , outputs a subdivision G'' of G' and integer k'' , such that: $\langle G', k' \rangle$ is a yes-instance of c_i -DISPERSION*, if and only if the output $\langle G'', k'' \rangle$ is a yes-instance of δ_i -DISPERSION*. Also, $\langle G', k' \rangle$ is a yes-instance of $(c_i - 1)$ -DISPERSION*, if and only if the output $\langle G'', k'' \rangle$ is a yes-instance of γ_i -DISPERSION*.

Proof. Let a_i, b_i and sequence $(c_i)_{i \geq 1}$ be defined as in Definition 6.3. Our algorithm begins by addressing some border cases. If $k' = 0$, we output a trivial simultaneous yes-instance of δ_i -DISPERSION* and γ_i -DISPERSION*. Else, if c_i exceeds $|V(G')|$ and $k' \geq 1$, we output a trivial simultaneous no-instance. Else, we output an a_{i-1} -subdivision of the input graph G' as G'' and as budget $k'' = k' + b_{i-1}|E(G')|$. Hence G' and G'' have the same pathwidth up to subdividing the edges. Any number of subdivisions of edges may increase the pathwidth only by a total of one. To compute g_i with c_i as part of the input, we use Lemma 6.5

to compute a_{i-1} and b_{i-1} in polynomial time. In particular, a_{i-1} is polynomial in c_i and hence polynomial in $|V(G')|$, such that we may output G'' , the a_{i-1} -subdivision of G' , in polynomial time.

We have $c_i\text{-disp}(G') = \frac{c_i}{1+c_i}\text{-disp}(G') - |E(G')|$ by Lemma 3.2 and as G' contains no cycle of length $< a$. Applying this translation not only once but b_{i-1} times, by Corollary 3.3, we obtain $c_i\text{-disp}(G') = \frac{c_i}{1+b_{i-1}c_i}\text{-disp}(G') - b_{i-1}|E(G')|$. Then by an a_{i-1} -subdivision of the input graph we have $\frac{c_i}{1+b_{i-1}c_i}\text{-disp}(G') = \frac{a_{i-1}c_i}{1+b_{i-1}c_i}\text{-disp}(G'_{a_{i-1}}) = \frac{a_i}{b_i}\text{-disp}(G'') = \delta_i\text{-disp}(G'')$ by Lemma 3.1. Thus the input $\langle G', k' \rangle$ is a yes-instance of $c_i\text{-DISPERSION}^*$, if and only if the output $\langle G'', k'' \rangle$ is a yes-instance of $\delta_i\text{-DISPERSION}^*$.

The analogous transformations yields that $(c_i - 1)\text{-disp}(G') = \frac{a_{i-1}(c_i-1)}{1+b_{i-1}(c_i-1)}\text{-disp}(G'') - b_{i-1}|E(G')|$. We observe that the numerator of the latter is $a_{i-1}(c_i - 1) = a_{i-1}c_i - a_{i-1} = a_i - a_{i-1}$, while the denominator is $1 + b_{i-1}(c_i - 1) = 1 + b_{i-1}c_i - b_{i-1} = b_i - b_{i-1}$. Hence this rational is equal to γ_i . Thus $\langle G', k' \rangle$ is a yes-instance of $(c_i - 1)\text{-DISPERSION}^*$, if and only if the output $\langle G'', k'' \rangle$ is a yes-instance of $\gamma_i\text{-DISPERSION}^*$. ◀

Proof of Theorem 6.1. Let δ be defined by integer sequence $c_i = 2^{2^i}$ for $i \geq 1$. Then δ is efficient comparable by Lemma 6.5. Consider a COLORFUL CLIQUE-instance $\langle G, k \rangle$ with color classes of size \hat{n} . Let i be such that $\hat{n} \leq c_i/32 = 2^{2^i-5} =: n$, hence $32n = c_i$. We note that $c_{j+1} = c_j^2$, for $j \geq 0$, and hence $n \leq \hat{n}^2$. Thus n and c_i are polynomial-time computable, and n, c_i are polynomial in \hat{n} . We extend the color-classes of $\langle G, k \rangle$ with $n - \hat{n}$ isolated vertices each, resulting in color classes of size n . Next, we apply the reduction of Lemma 6.2 on $\langle G, k \rangle$, now with c_i color classes, which outputs $\langle G', k' \rangle$. In turn, we apply the reduction of Lemma 6.6 on $\langle G', k' \rangle$ which outputs $\langle G'', k'' \rangle$, forming our final output.

We note that the reductions of Lemma 6.2 and Lemma 6.6 are polynomial-time computable. The former outputs a graph of pathwidth $O(k)$, the latter does not change the pathwidth up to a constant. Hence overall we output a graph G'' of pathwidth $O(k)$.

By Lemma 6.2 and Lemma 6.6, $\langle G, k \rangle$ is a yes-instance of COLORFUL CLIQUE, if and only if $\langle G'', k'' \rangle$ is a yes-instance of $\delta_i\text{-DISPERSION}$, if and only if $\langle G'', k'' \rangle$ is a yes-instance of $\gamma_i\text{-DISPERSION}$. Since $\gamma_i < \delta < \delta_i$, by Lemma 6.4, the dispersion numbers satisfy $\gamma_i\text{-disp}(G'') = \delta\text{-disp}(G'') = \delta_i\text{-disp}(G'')$. Thus the output $\langle G'', k'' \rangle$ is a yes-instance of $\delta\text{-DISPERSION}$, if and only if $\langle G, k \rangle$ is a yes-instance of COLORFUL CLIQUE.

Since COLORFUL CLIQUE is W[1]-hard parameterized by k , also $\delta\text{-DISPERSION}$ is W[1]-hard parameterized by the pathwidth of the input graph. For the lower bound under ETH, assume an $f(\text{pw}(G)) \cdot n^{o(\text{pw}(G))}$ time algorithm for $\delta\text{-DISPERSION}$ for a computable function f . Then using the above reduction on a COLORFUL CLIQUE-instance yields an $f(k) \cdot n^{o(k)}$ time algorithm for COLORFUL CLIQUE, in contradiction to ETH. ◀

7 Domination and Covering

Finally, we turn to the domination problems and covering as their continuous counterpart. This section outlines the connection of a -walk dominating set and δ -covers. We establish tools that relate a -walk dominating set on b -subdivided graphs for different values of a, b , similarly as we did for the independent set problem. These tools then allow to derive the complexity results for $a\text{-WALK DOMINATING SET}$ on b -subdivided graphs and $\delta\text{-COVERING}$ as stated in the introduction. The details thereof are deferred to the full version.

The notion of an a -walk dominating set can be defined in three different ways, (D1), (D2) and (D3), which are useful for different kind of proofs. Let G be a graph without isolated vertices. For an integer a , a subset $D \subseteq V(G)$ a -walk dominates some subset of edges $E' \subseteq E(G)$, defining $V' := V(G[E'])$, if:

- (D1) For every edge $e \in E'$, there are (possibly identical) vertices $w_1, w_2 \in D$ and a w_1, w_2 -walk in G of length at most a that contains e ; or
- (D2) Every vertex $u \in V(G[E'])$ has $d(u, D) \leq \frac{a}{2}$, and the set vertices $u \in V(G[E'])$ where $d(u, D) = \frac{a}{2}$ forms an independent set.
- (D3) $D \subseteq V(G)$ a -dominates $V' \cup E'$ in the 2-subdivision G_2 of G (when identifying an edge $\{u, v\}$ with the vertex with neighborhood $\{u, v\}$ in G_2). That is, for every vertex $u \in V' \cup E'$, there is a vertex $w \in D$ with $d_{G_2}(u, w) \leq a$.

An a -walk dominating set of G is a subset $D \subseteq V(G)$ that a -walk dominates $E(G)$.

► **Lemma 7.1** (*). *Conditions (D1), (D2), (D3) are equivalent.*

We have the following two transformation of δ -dispersed sets for different values of δ , as shown by Hartmann et al. [18].

► **Lemma 7.2** ([18]). *For every real $\delta > 0$ and integer $c \geq 1$, δ -cover(G) = $c\delta$ -cover(G_c).*

► **Lemma 7.3** ([18]). δ -cover(G) + $|E(G)| = \frac{\delta}{1+2\delta}$ -cover(G).

Aiming to translate these modifications to the realm of a -walk dominating set on b -subdivided graphs, we observe the following connection.

► **Observation 7.4** (*). *Let $k \in \mathbb{N}$. There is a b -simple $\frac{a}{2b}$ -covering set of size k of a graph G without isolated vertices, if and only if there is an a -walk dominating set of G_b of size k .*

A minimum $\frac{a}{b}$ -cover S can be assumed to be b -simple [18]. Actually, if b is even, we observation can be improved. For example, a $\frac{1}{2}$ -covering set implies a 2-simple $\frac{1}{2}$ -covering set of same size.

► **Lemma 7.5** (*). *Let S be an $\frac{a}{b}$ -cover of a graph G for integers $a, b \in \mathbb{N}$. Then there is an $\frac{a}{b}$ -cover S^* of G of size $|S^*| = |S|$ that is $2b$ -simple and, if b is a multiple of 2, is b -simple.*

Assuming that S contains no point at a position $\frac{2i-1}{2b}$ for $i \in \mathbb{N}$, then S^ is b -simple, and, if additionally b is a multiple of 2, S^* is $\frac{b}{2}$ -simple.*

With this connection at hand, we can state a refined connection of minimum $\frac{a}{b}$ -covers of a graph G and minimum a -walk dominating set on b -subdivided graphs.

► **Corollary 7.6.** $\frac{a}{b}$ -cover(G) = $\bar{\gamma}_{4a}(G_{2b}) = \bar{\gamma}_{4ca}(G_{2cb})$; $\frac{a}{2b}$ -cover(G) = $\bar{\gamma}_{2a}(G_{2b}) = \bar{\gamma}_{2ca}(G_{2cb})$, for any $a, b, c \in \mathbb{N}$ and graph G without isolated vertices.

Now we can put the earlier stated transformation of δ -dispersed sets in terms of a -walk dominating on b -subdivided graphs.

► **Theorem 7.7** (*). $\bar{\gamma}_a(G_b) = \bar{\gamma}_{ca}(G_{cb})$ when c is odd, or a, b are even.

► **Theorem 7.8** (*). $\bar{\gamma}_a(G_b) + |E(G)| = \bar{\gamma}_a(G_{a+b})$.

These two transformation lay the groundwork for show Theorem 1.11, Theorem 1.13 and Theorem 1.14. For details, we refer to the full version.

References

- 1 Yousef Alavi, M. Behzad, Linda M. Lesniak-Foster, and E. A. Nordhaus. Total matchings and total coverings of graphs. *Journal of Graph Theory*, 1(2):135–140, 1977. doi:10.1002/jgt.3190010209.
- 2 Yousef Alavi, Jiuqiang Liu, Jianfang Wang, and Zhongfu Zhang. On total covers of graphs. *Discret. Math.*, 100(1-3):229–233, 1992. doi:10.1016/0012-365X(92)90643-T.

- 3 José D. Alvarado, Simone Dantas, and Dieter Rautenbach. Distance k -domination, distance k -guarding, and distance k -vertex cover of maximal outerplanar graphs. *Discret. Appl. Math.*, 194:154–159, 2015. doi:10.1016/J.DAM.2015.05.010.
- 4 Gábor Bacsó, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Zsolt Tuza, and Erik Jan Van Leeuwen. Subexponential-time algorithms for maximum independent set in P_t -free and broom-free graphs. *Algorithmica*, 81:421–438, 2019.
- 5 Glencora Borradaile and Hung Le. Optimal dynamic program for r -domination problems over tree decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24–26, 2016, Aarhus, Denmark*, volume 63 of *LIPICs*, pages 8:1–8:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.IPEC.2016.8.
- 6 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.
- 7 Clément Dallard, Mirza Krbežlija, and Martin Milanic. Vertex cover at distance on H -free graphs. In Paola Flocchini and Lucia Moura, editors, *Combinatorial Algorithms – 32nd International Workshop, IWOCA 2021, Ottawa, ON, Canada, July 5–7, 2021, Proceedings*, volume 12757 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2021. doi:10.1007/978-3-030-79987-8_17.
- 8 Perino M. Dearing and Richard L. Francis. A minimax location problem on a network. *Transportation Science*, 8(4):333–343, 1974.
- 9 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for $W[1]$. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. doi:10.1016/0304-3975(94)00097-3.
- 10 Louis Dublois, Michael Lampis, and Vangelis T. Paschos. New algorithms for mixed dominating set. *Discret. Math. Theor. Comput. Sci.*, 23(1), 2021. doi:10.46298/DMTCS.6824.
- 11 Paul Erdős and Amram Meir. On total matching numbers and total covering numbers of complementary graphs. *Discret. Math.*, 19(3):229–233, 1977. doi:10.1016/0012-365X(77)90102-9.
- 12 Hiroshi Eto, Fengrui Guo, and Eiji Miyano. Distance- d independent set problems for bipartite and chordal graphs. *J. Comb. Optim.*, 27(1):88–99, 2014. doi:10.1007/s10878-012-9594-4.
- 13 Hiroshi Eto, Takehiro Ito, Zhilong Liu, and Eiji Miyano. Approximation algorithm for the distance-3 independent set problem on cubic graphs. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29–31, 2017, Proceedings*, volume 10167 of *Lecture Notes in Computer Science*, pages 228–240. Springer, 2017. doi:10.1007/978-3-319-53925-6_18.
- 14 Alexander Grigoriev, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. Dispersing obnoxious facilities on a graph. *Algorithmica*, 83(6):1734–1749, 2021. doi:10.1007/s00453-021-00800-3.
- 15 Tim A. Hartmann. *Facility location on graphs*. Dissertation, RWTH Aachen University, Aachen, 2022. doi:10.18154/RWTH-2023-01837.
- 16 Tim A. Hartmann and Tom Janßen. Approximating δ -covering (to appear). In *Approximation and Online Algorithms – 22nd International Workshop, WAOA 2024, Egham, United Kingdom, September 5–6, 2024, Proceedings*, Lecture Notes in Computer Science. Springer, 2024.
- 17 Tim A. Hartmann and Stefan Lendl. Dispersing obnoxious facilities on graphs by rounding distances. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22–26, 2022, Vienna, Austria*, volume 241 of *LIPICs*, pages 55:1–55:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.MFCS.2022.55.
- 18 Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. Continuous facility location on graphs. *Math. Program.*, 192(1):207–227, 2022. doi:10.1007/s10107-021-01646-x.

- 19 Pallavi Jain, Jayakrishnan Madathil, Fahad Panolan, and Abhishek Sahu. Mixed dominating set: A parameterized perspective. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science – 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 330–343. Springer, 2017. doi:10.1007/978-3-319-68705-6_25.
- 20 Ioannis Katsikarelis, Michael Lampis, and Vangelis T. Paschos. Structurally parameterized d-scattered set. *Discret. Appl. Math.*, 308:168–186, 2022. doi:10.1016/j.dam.2020.03.052.
- 21 Ioannis Katsikarelis, Michael Lampis, and Vangelis T. Paschos. Improved (in-)approximability bounds for d-scattered set. *J. Graph Algorithms Appl.*, 27(3):219–238, 2023. doi:10.7155/JGAA.00621.
- 22 Michael Lampis. The primal pathwidth SETH. *CoRR*, abs/2403.07239, 2024. doi:10.48550/arXiv.2403.07239.
- 23 Jason Lewis, Stephen T. Hedetniemi, Teresa W. Haynes, and Gerd H. Fricke. Vertex-edge domination. *Utilitas mathematica*, 81:193–213, 2010.
- 24 Daniel Lokshantov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.
- 25 Jayakrishnan Madathil, Fahad Panolan, Abhishek Sahu, and Saket Saurabh. On the complexity of mixed dominating set. In René van Bevern and Gregory Kucherov, editors, *Computer Science – Theory and Applications – 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 262–274. Springer, 2019. doi:10.1007/978-3-030-19955-5_23.
- 26 Aniket Majumdar. *Neighborhood hypergraphs: A framework for covering and packing parameters in graphs*. Dissertation, Clemson University, 1992.
- 27 Nimrod Megiddo and Arie Tamir. New results on the complexity of p -center problems. *SIAM J. Comput.*, 12(4):751–758, 1983. doi:10.1137/0212051.
- 28 Amram Meir. On total covering and matching of graphs. *J. Comb. Theory B*, 24(2):164–168, 1978. doi:10.1016/0095-8956(78)90017-5.
- 29 Pedro Montealegre and Ioan Todinca. On distance- d independent set and other problems in graphs with “few” minimal separators. In Pinar Heggernes, editor, *Graph-Theoretic Concepts in Computer Science – 42nd International Workshop, WG 2016, Istanbul, Turkey, June 22-24, 2016, Revised Selected Papers*, volume 9941 of *Lecture Notes in Computer Science*, pages 183–194, 2016. doi:10.1007/978-3-662-53536-3_16.
- 30 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.
- 31 Uri N. Peled and Feng Sun. Total matchings and total coverings of threshold graphs. *Discret. Appl. Math.*, 49(1-3):325–330, 1994. doi:10.1016/0166-218X(94)90216-X.
- 32 Michal Pilipczuk and Sebastian Siebertz. Kernelization and approximation of distance- r independent sets on nowhere dense graphs. *Eur. J. Comb.*, 94:103309, 2021. doi:10.1016/J.EJC.2021.103309.
- 33 Douglas R. Shier. A min-max theorem for p -center problems on a tree. *Transportation Science*, 11(3):243–252, 1977. URL: <http://www.jstor.org/stable/25767877>.
- 34 Arie Tamir. On the solution value of the continuous p -center location problem on a graph. *Math. Oper. Res.*, 12(2):340–349, 1987. doi:10.1287/moor.12.2.340.
- 35 Arie Tamir. Obnoxious facility location on graphs. *SIAM J. Discret. Math.*, 4(4):550–567, 1991. doi:10.1137/0404048.
- 36 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms – ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

- 37 Mingyu Xiao and Zimo Sheng. Improved parameterized algorithms and kernels for mixed domination. *Theor. Comput. Sci.*, 815:109–120, 2020. doi:10.1016/J.TCS.2020.02.014.
- 38 Yancai Zhao, Liying Kang, and Moo Young Sohn. The algorithmic complexity of mixed domination in graphs. *Theor. Comput. Sci.*, 412(22):2387–2392, 2011. doi:10.1016/J.TCS.2011.01.029.
- 39 Radosław Ziemann and Paweł Żyliński. Vertex-edge domination in cubic graphs. *Discret. Math.*, 343(11):112075, 2020. doi:10.1016/J.DISC.2020.112075.
- 40 Paweł Żyliński. Vertex-edge domination in graphs. *Aequationes mathematicae*, 93(4):735–742, 2019.

Forbidden Patterns in Mixed Linear Layouts

Deborah Haun  

Karlsruhe Institute of Technology, Germany

Laura Merker  

Karlsruhe Institute of Technology, Germany

Sergey Pupyrev  

Menlo Park, CA, USA

Abstract

An ordered graph is a graph with a total order over its vertices. A linear layout of an ordered graph is a partition of the edges into sets of either non-crossing edges, called stacks, or non-nesting edges, called queues. The stack (queue) number of an ordered graph is the minimum number of required stacks (queues). Mixed linear layouts combine these layouts by allowing each set of edges to form either a stack or a queue. The minimum number of stacks plus queues is called the mixed page number. It is well known that ordered graphs with small stack number are characterized, up to a function, by the absence of large twists (that is, pairwise crossing edges). Similarly, ordered graphs with small queue number are characterized by the absence of large rainbows (that is, pairwise nesting edges). However, no such characterization via forbidden patterns is known for mixed linear layouts.

We address this gap by introducing patterns similar to twists and rainbows, which we call thick patterns; such patterns allow a characterization, again up to a function, of mixed linear layouts of bounded-degree graphs. That is, we show that a family of ordered graphs with bounded maximum degree has bounded mixed page number if and only if the size of the largest thick pattern is bounded. In addition, we investigate an exact characterization of ordered graphs whose mixed page number equals a fixed integer k via a finite set of forbidden patterns. We show that for $k = 2$, there is no such characterization, which supports the nature of our first result.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases Ordered Graphs, linear Layout, mixed linear Layout, Stack Layout, Queue Layout

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.45

Related Version *Full Paper*: <https://doi.org/10.48550/arXiv.2412.12786> [41]

Funding *Deborah Haun*: supported by the Deutsche Forschungsgemeinschaft – 520723789.

1 Introduction

An *ordered graph* is a graph given with a fixed linear vertex order, \prec . A *linear layout* of an ordered graph is a partition of its edges such that each part satisfies certain requirements with respect to the order. In a *stack layout* each part, also called a *stack*, is required to be crossing-free with respect to \prec , that is, two edges in the same stack may not have alternating endpoints. A *queue layout* is a “dual” concept which forbids two edges to nest in the same part, called a *queue*; that is, if $u \prec x \prec y \prec v$, then edges uv and xy must be in different queues. The two concepts are generalized in *mixed linear layouts*, where each part (called a *page* then) may either be a stack or a queue. A linear layout using s stacks and/or q queues is called pure s -stack, pure q -queue, and mixed s -stack q -queue, respectively. In all three cases, the objective is to minimize the number of parts. The *stack number* $sn(G)$ (*queue number* $qn(G)$), *mixed page number* $mn(G$)) of an ordered graph G is the smallest k such that there is a stack (queue, mixed) layout with at most k stacks (queues, pages).



© Deborah Haun, Laura Merker, and Sergey Pupyrev;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 45; pp. 45:1–45:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Stack layouts and queue layouts are well understood and a rich collection of tools has been developed. Most notably, the product structure theory [22], layered path decompositions [7, 26], track layouts [23], and different kinds of H -partitions that were used successfully both for queue layouts [8, 22, 35, 45] and stack layouts [48]. A fundamental technique in this context is a characterization of stack and queue layouts via forbidden ordered patterns. Formally, a *pattern* is an ordered graph with at least one edge. The *size* of a pattern is the number of edges. An ordered graph (G, \prec_1) *contains* a pattern (H, \prec_2) if H is a subgraph of G and \prec_2 is a suborder of \prec_1 ; otherwise, the graph *avoids* the pattern. A k -*twist* denotes a set of k pairwise crossing edges with respect to some vertex order, and a k -*rainbow* is a set of k pairwise nesting edges, where symbol k can be omitted if not needed. We define a *graph parameter* to be a function assigning a non-negative integer to every graph. A parameter p is *bounded* for a family of graphs if there is a constant c such that $p(G) \leq c$ for every graph G of the family. Now, the characterization of stack and queue layouts can be formulated as follows.

► **Theorem 1** ([17, 40]). *A family of ordered graphs has bounded stack number if and only if the size of the largest twist is bounded.*

► **Theorem 2** ([44]). *A family of ordered graphs has bounded queue number if and only if the size of the largest rainbow is bounded.*

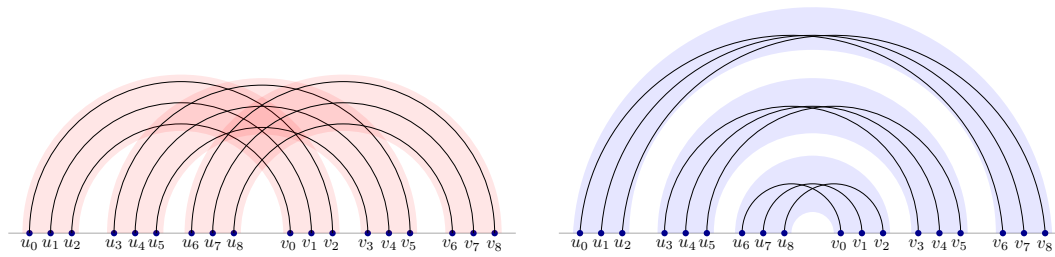
These theorems are useful both for upper bounds (as an explicit assignment of edges to stacks or queues is not needed) and for lower bounds (as a tedious case distinction which edge could go to which stack or queue gets superfluous). Unfortunately, no analogous characterization is known for mixed linear layouts. As a consequence, all known results on mixed linear layouts rely on an explicit assignment of edges to stacks and queues. In this paper, we aim to close this gap and find a set of patterns whose absence characterizes mixed linear layouts similarly to twists and rainbows in Theorems 1 and 2. The following open question asks whether there is a theorem similar to Theorems 1 and 2 for mixed linear layouts, where stack/queue is replaced by mixed page number and the largest twist/rainbow is replaced by the largest pattern in \mathcal{P} .

► **Open Problem 3.** *Do there exist finite sets, $\mathcal{P}_1, \mathcal{P}_2, \dots$, of patterns and a binding function, $f : \mathbb{N} \rightarrow \mathbb{N}$, such that for all $k \in \mathbb{N}$ and every ordered graph, G , the following holds:*

- $\text{mn}(G) < f(k)$ if G avoids all patterns in \mathcal{P}_k , and
- $\text{mn}(G) \geq k$ if G contains some pattern in \mathcal{P}_k ?

Indeed, an affirmative answer would give a theorem of the form of Theorems 1 and 2: There is a set \mathcal{P} of patterns, namely the union of $\mathcal{P}_1, \mathcal{P}_2, \dots$, such that a family of ordered graphs has bounded mixed page number if and only if the size of the largest pattern of \mathcal{P} that occurs in a graph of the family is bounded. To see the implication, note that there are only finitely many patterns of a certain size, so if the size of the largest pattern of \mathcal{P} is bounded, then the first property of Open Problem 3 gives that the mixed page number is bounded. And conversely, a family whose mixed page number is at most k contains no patterns of \mathcal{P} that are larger than the largest pattern in $\mathcal{P}_1, \dots, \mathcal{P}_k$ by the second property.

Note that for pure stack/queue layouts, the answer to the question is positive, since the corresponding sets \mathcal{P}_k of forbidden patterns (also called *obstruction sets*) contain a single element, a k -twist and a k -rainbow, respectively. Indeed, a more technical formulation of Theorem 1 would be that for every ordered graph G and every $k \in \mathbb{N}$, it holds that the stack number of G is less than $14k \log k$ if G avoids a k -twist, and is at least k if G contains a k -twist [17, 40]. We remark that in contrast to stack layouts, queue layouts even admit the



■ **Figure 1** The two 3-thick patterns: Three pairwise crossing 3-rainbows (left) and three pairwise nesting 3-twists (right).

identity as binding function [44]. To complement this, we answer the problem affirmatively for mixed linear layouts of bounded-degree graphs and provide negative results for an exact characterization with the identity binding function.

The remainder of this section is organized as follows. First, we present our main results and describe technical contributions with concrete bounds that guide through the subsequent sections. Then, we relate our findings to the state-of-the-art. Finally, we discuss linear layouts from various perspectives targeted to readers not familiar with the topic.

1.1 Main Results

For an explicit description of the set \mathcal{P} of patterns, we define a k -thick pattern to be obtained either from a k -twist by replacing each edge by a k -rainbow, or from a k -rainbow by replacing each edge by k -twist; see Figure 1 and Section 2.2 for a more elaborate introduction.

► **Theorem 4.** *For every family of ordered graphs with bounded maximum degree, the size of the largest thick pattern is bounded if and only if the mixed page number is bounded.*

Note that Open Problem 3 asks for a characterization up to a function bounding the mixed page number depending on the size of the largest pattern occurring in the graph. That is, even if we know the size of the largest pattern, we do not learn the exact mixed page number but only an upper bound. A more granular characterization would guarantee the exact mixed page number given that a set of forbidden patterns is excluded. As a negative result, we present an infinite family of patterns that needs to be included in an obstruction set of graphs with mixed page number 2. Thus, Open Problem 3 does not admit such a precise answer.

► **Theorem 5.** *Let \mathcal{P} be the set of patterns such that an ordered matching has mixed page number at most 2 if and only if it contains no pattern of \mathcal{P} . Then \mathcal{P} is infinite.*

1.2 Summary

When answering Open Problem 3, which we do positively for bounded-degree graphs, there is a trade-off between three objectives: First, the number of patterns should be small since each of them needs to be excluded to prove upper bounds. Second, the bound we obtain on the mixed page number should be small. And third, we want our results to hold for graph classes that are as large as possible, while smaller graph classes have the potential of better results in the first two objectives. In this subsection, we present explicit bounds on the mixed page number that are different trade-offs between these three objectives.

45:4 Forbidden Patterns in Mixed Linear Layouts

As a base for subsequent results, we start with (ordered) matchings having a separated layout and a relatively large obstruction set. Here, a layout of a bipartite graph is called *separated* if we first have all vertices of one part of the bipartition, and then all vertices of the other part. We provide an explicit list of the patterns in Section 2.1.

► **Theorem 6** (Section 2.1). *There is a set \mathcal{P} of patterns such that for every matching M with a separated layout, if the largest pattern of \mathcal{P} in M is of size k^2 , then the mixed page number of M is at least k and at most $2k$.*

For stack layouts and queue layouts, it turned out to be a powerful tool to have only one pattern, namely twists, respectively rainbows. Close enough, we present two patterns, namely the two options for thick patterns, to play the same role for mixed linear layouts by combining twists and rainbows.

► **Theorem 7** (Section 2.2). *If the largest thick pattern in a matching M with a separated layout is a k -thick pattern, then the mixed page number of M is at least k and at most $2k^7$.*

We finish the matching case by generalizing the separated layout to an arbitrary vertex ordering. We remark that we expect the patterns to always be matchings, as is the case with twists and rainbows, and even conjecture that the same set of patterns also works for general ordered graphs.

► **Theorem 8** (Section 3). *If the largest thick pattern in an ordered matching M is a k -thick pattern, then the mixed page number of M is at least k and at most $k^{O(k)}$.*

To prove this, we work with so-called quotients of linear layouts that can capture the global structure of an (ordered) graph and show how to transfer linear layouts of a quotient to the initial graph (Lemma 17), which might be of independent interest.

With Vizing's theorem [76], all our results on matchings generalize to bounded-degree graphs, in particular we obtain the following bounds for Theorem 4.

► **Theorem 9** (Section 3). *If the largest thick pattern in an ordered graph G with maximum degree Δ is a k -thick pattern, then the mixed page number of G is at least k and at most $\Delta k^{O(k)}$.*

Note that this is indeed a specification of Theorem 4 since the upper bound shows that if there is no large thick pattern, then the mixed page number is bounded, and conversely, if the mixed page number is bounded, there is no large thick pattern due to the lower bound.

Finally, we aim for the second objective to be optimal, i.e., we aim for an obstruction set such that the mixed page number is at most k if and only if none of the forbidden patterns is contained. For $k = 2$, we show that such a finite obstruction set exists if the ordered graph is bipartite and has a separated layout, but not in the general case.

► **Theorem 10** (Section 4.1). *There is a finite set \mathcal{P} of patterns such that a bipartite graph with separated layout has mixed page number at most 2 if and only if it contains no pattern of \mathcal{P} .*

For the negative side, we remark that it is not surprising that such an exact characterization is infeasible as no such characterization is known for stack layouts. Here, we know that an ordered graph without k -twist has stack number at most $O(k \log k)$ [17] and that this bound is tight [55]. However, one might think that with a larger obstruction set, an exact characterization is possible. For $k \geq 4$ this is not true [74] but to the best of our knowledge no explicit infinite family of patterns was known to be in the obstruction set. We provide

constructions for the following theorem, which in particular proves Theorem 5 as all minimal patterns need to be included in the obstruction set. Here, minimal refers to edge-minimal for a given mixed page number (stack number), i.e., if any edge is removed, then the mixed page number (stack number) decreases.

► **Theorem 11** (Section 4.2). *There is an infinite family of minimal ordered matchings with mixed page number > 2 and an infinite family of minimal ordered matchings with stack number k for all $k > 2$.*

As some theorems have only a proof sketch, we refer to [41] for the full paper.

1.3 Related Work

Building on earlier notions [50, 64], the concepts of stack and queue number were first investigated by Bernhart and Kainen [12] in 1979 and Heath and Rosenberg [44] in 1992, respectively. Over the last three decades, there has been extensive research on these concepts leading to numerous results, primarily for planar graphs [2, 8, 9, 20, 22, 35, 65, 80, 81], but also for 1-planar graphs [10] and graphs with bounded genus [42, 57] or bounded treewidth [56]. In light of our results it is worth mentioning that there is a particular interest in bounded-degree graphs [11, 25, 78]. In this context, the vertex ordering is usually not given, so there are two steps to solve: First one needs to find a good vertex ordering and second, the resulting ordered graph is analyzed, often using the characterization with twists and rainbows. In addition, the stack and queue number for directed acyclic graphs [43, 48, 63], where the vertex ordering must respect the orientation of the edges, have been studied fruitfully, e.g., for (planar) posets [4, 32, 53, 62, 69], or upward planar graphs [34, 48, 49].

Although Heath and Rosenberg [44] already suggested a study of mixed linear layouts back in 1992, specifically conjecturing that every planar graph has a 1-stack 1-queue layout, significant progress began only quite recently when Pupyrev [68] disproved their conjecture. Subsequently, other papers have followed strengthening Pupyrev’s result by showing that not even series-parallel [5] or planar bipartite graphs [35] admit a 1-stack 1-queue layout. Further investigations into more general mixed linear layouts have often been proposed [9, 11, 63]. Results on this include complete and complete bipartite graphs [3], subdivisions [24, 60, 68], and computational hardness results [19]. Very recently, Katheder, Kaufmann, Pupyrev, and Ueckerdt [51] related the queue, stack, and mixed page number to each other and asked for an improved understanding of graphs with small mixed page number both in the separated and in the general setting as, together with their results, this would fully reveal the connection between these three concepts. All these investigations have in common that it is quite difficult to analyze mixed linear layouts due to the lack of a simple characterization similar to rainbows and twists for stack and queue layouts, which is the gap we address in this paper.

An explicit investigation of rainbows and twists in linear layouts can be found from early on. In their seminal paper on queue layouts, Heath and Rosenberg [44] identified rainbows as a simple characterization for queue layouts. They showed that the queue number with respect to a fixed vertex order always equals the size of the largest rainbow, a result that has often been used to derive bounds on the queue number [2, 4, 8, 11, 22, 27, 32, 53, 69]. Similarly, twists offer a straight-forward characterization for stack layouts, although the stack number of an ordered graph is not exactly the size of the largest twist. While the size of the largest twist is clearly a lower bound on the stack number, Davies [17] recently showed an upper bound of $\mathcal{O}(k \log k)$ on the stack number, where k denotes the size of the largest twist. This very recent and asymptotically tight [55] bound, along with earlier larger upper bounds [15, 18, 40, 55] has proven useful for bounding the stack number of various graph

classes [34, 48, 49, 63]. For small k , it is known that an order without a 2-twist (that is, when $k = 1$) corresponds to an outerplanar drawing of a graph, which is a 1-stack layout. For $k = 2$ (that is, an order without a 3-twist), five stacks are sufficient and sometimes necessary [1, 54]; for $k = 3$, it is known that 19 stacks suffice [17].

Similarly, for the mixed page number we do not expect an exact characterization, but rather a characterization up to a function. For general graphs, previous results [19, 74] have shown that deciding whether an ordered matching has an s -stack q -queue layout is NP-complete for all $s \geq 4$ and $q \geq 0$, making the question for a finite set of patterns exactly characterizing s -stack q -queue layouts obsolete. However, in some special cases, such as for separated layouts of bipartite graphs or for smaller s and q , there is still potential for an exact characterization, while in the general case we aim to find a finite characterization up to a function that is as small as possible.

Our investigations start with bipartite graphs having a separated layout, which is a setting that has been widely studied (sometimes, implicitly) under different names, such as *2-track layouts*, *2-layer drawings*, or *partitioned drawings* [3, 6, 16, 21, 23, 24, 28, 29, 61, 67, 70, 79]. Initially, we further narrow down our focus to matchings with a separated layout. Separated matchings can also be viewed as permutations of the right endpoints of the edges, relative to the order of the left endpoints. In this context, the mixed page number of the ordered matching equals the minimum number of monotone subsequences into which the permutation can be partitioned. For this problem it is known that permutations that can be partitioned into a fixed number of monotone subsequences, and thus matchings with a fixed mixed page number, are characterized by a finite, though potentially large set of forbidden subsequences [31, 52, 77].

Another related topic is the extremal theory of ordered graphs [66, 72], which has also been studied from the perspective of 0-1-matrices in the case of bipartite graphs with a separated layout [36, 38, 47, 58, 71]. In this field, the focus is on determining how many edges (or 1-entries) an ordered graph (or a 0-1-matrix) can have while avoiding a specified family of patterns (or submatrices). A necessary condition for a set of patterns to characterize a fixed mixed page number is that these patterns must enforce the number of edges to be linear in the number of vertices. If the patterns are matchings, this necessary condition is met by the proof of the Füredi-Hajnal conjecture [36], which was proven by Marcus and Tardos [58].

1.4 Connections to Other Fields

Before proving our results in the subsequent sections, let us review linear layouts from different perspectives. We briefly state some connections here and refer to the full version [41] for a more detailed discussion.

Stack/Queue Perspective. Stack layouts and queue layouts capture how well an ordered graph can be processed by stacks and queues, which can be best seen in the case of an ordered matching (but also works for general ordered graphs). Here, the vertices are considered in the given order, and an edge is pushed into the data structure when its first endpoint is reached, and it is popped when its second endpoint is reached. With a stack, an ordered matching can be processed if and only if the edges obey a first-in-last-out order, i.e., if and only if no two edges have alternating endpoints. Similarly, an ordered matching can be processed with a queue if and only if the edges obey a first-in-first-out order, which is equivalent to having no two nesting edges. Thus, the minimum number of stacks or queues equals the stack number, respectively queue number, and allowing to use both as needed represents mixed linear layouts. That is, investigating linear layouts improves our understanding of how three very fundamental data structures, namely graphs, stacks, and queues, interact.

Coloring Perspective. Finding a stack layout for an ordered graph is equivalent to coloring a circle graph [17]. A *circle graph* is the intersection graph of chords of a circle, where two chords intersect if they cross but not if they share an endpoint. Hence, all findings on colorings of circle graphs [15, 18, 37, 40, 55, 74, 75] also apply to stack layouts and vice versa. Most importantly, Theorem 1 is proved in the language of circle graphs by showing that they are χ -bounded, that is their chromatic number is bounded by a function of their clique number [17, 40]. A twist in an ordered graph corresponds to pairwise crossing chords, and similarly the size of the largest rainbow is the maximum number of parallel chords, up to a factor of 2. Together, a mixed linear layout asks for a partition of the chords into two groups: The first group is supposed to have only few pairwise crossing chords and corresponds to the set of stacks in the mixed linear layout. And the second group should have only small sets of parallel chords and thereby corresponds to the set of queues.

Upward Stack Number Perspective. One of the most prominent open questions in the field of linear layouts is whether or not planar posets and upward planar graphs have bounded stack number [62]. A directed acyclic graph is called *upward planar* if it can be drawn in the plane such that the edges are crossing-free and y -monotone. Despite intensive research in this direction [13, 14, 34, 46, 48, 49, 59, 63], this problem is still widely open. Hoping for a positive answer, bounding the mixed page number is an intermediate step before answering the question for pure stack layouts. In light of this, our results provide tools in this direction.

2 Separated Layouts

We start with matchings having a separated layout, which is the base for proving Theorem 4 in Section 3. Recall that a linear layout of a bipartite graph $G = (V_1 \cup V_2, E)$ is *separated* if all vertices of V_1 precede all vertices of V_2 in the vertex ordering. First, we give a linear upper bound in Section 2.1 using a comparably large obstruction set of patterns with mixed page number k . We then reduce the number of patterns to 2 in Section 2.2 and still obtain a polynomial dependency on k .

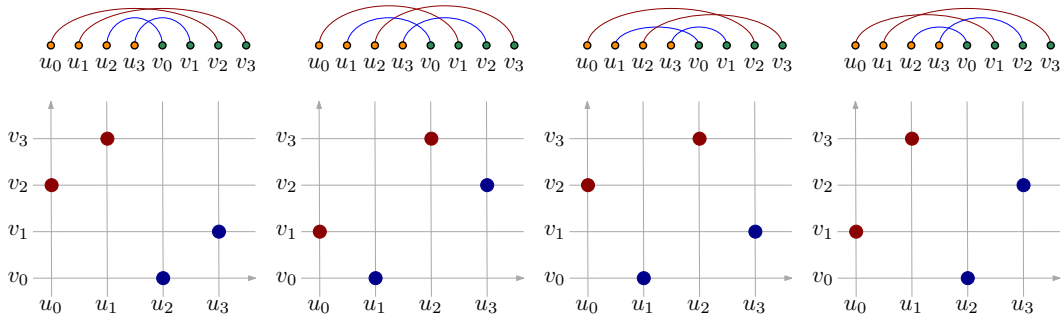
2.1 \diamond -Patterns

Consider the *grid representation* of a bipartite graph G with a fixed separated layout. That is, one part of the bipartition is represented by columns, the other by rows, and each edge is described by an x - and a y -coordinate indicating the column, respectively the row, of its endpoints. As we consider matchings, we have only one edge in each row and in each column. For two edges e_1, e_2 of G we write $e_1 \nearrow e_2$ if $x(e_1) < x(e_2)$ and $y(e_1) < y(e_2)$, i.e., if e_1 and e_2 cross. Similarly, we write $e_1 \searrow e_2$ if $x(e_1) < x(e_2)$ and $y(e_1) > y(e_2)$, i.e., if e_1 and e_2 nest. Consider a set of k^2 edges indexed by $e_{i,j}$ for $1 \leq i \leq k$ and $1 \leq j \leq k$. We say that the edges form a k - \diamond -*pattern* if the following holds:

- (i) $e_{i,j} \nearrow e_{i,j+1}$ for every $1 \leq i \leq k$ and $1 \leq j < k$, and
- (ii) $e_{i,j} \searrow e_{i+1,j}$ for every $1 \leq i < k$ and $1 \leq j \leq k$.

That is, in the grid representation we have k increasing sequences of length k , where the j -th elements in each chain together form a decreasing sequence for each $j = 1, \dots, k$. We denote the *size* of a k - \diamond -pattern by $k \times k$ to indicate that we have k^2 edges, where k is the parameter we are usually interested in. Notice that there are exactly four \diamond -patterns of size 2×2 ; see Figure 2.

As it turns out, \diamond -patterns are deeply connected to mixed linear layouts. Our main result on \diamond -patterns is summarized in the following theorem.



■ **Figure 2** 2- \diamond -patterns for Theorem 12.

► **Theorem 12.** *Let M be a matching with a separated layout. If the largest \diamond -pattern in M has size $k \times k$, then the separated mixed page number of G is at least k and at most $2k$.*

We translate Theorem 12 to the language of posets so we can use a result by Greene [39] to prove it in this regime. For this, we interpret a \diamond -pattern, or more generally any matching M with a separated vertex ordering, as a poset $P(M)$ whose elements are the edges of M . Since M is a matching, for every two elements, e_1, e_2 of $P(M)$ with $x(e_1) < x(e_2)$, it holds that either $e_1 \nearrow e_2$ or $e_1 \searrow e_2$. In the former case, we set $e_1 < e_2$ in $P(M)$ and in the latter we make the two elements incomparable, for which we write $e_1 \parallel e_2$. We call $P(M)$ the *poset of M* . Observe that a chain of $P(M)$ is increasing in the grid representation and corresponds to a twist in M , and an antichain is decreasing in the grid representation and corresponds to a rainbow. That is, to bound the mixed page number, we aim to cover each element of the poset by a chain or an antichain (or both) and thereby minimize the number of chains plus antichains.

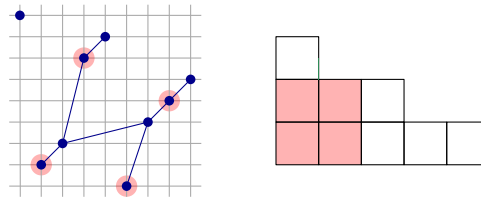
► **Observation 13.** *For the mixed page number $\text{mn}(M)$ of a matching M with separated vertex ordering we have $\text{mn}(M) \leq m$ if and only if its poset $P(M)$ can be covered by m chains and antichains.*

Now, the lower bound of Theorem 12 asks for a proof that $P(M)$ cannot be covered by less than k chains and antichains. The omitted proof of the following slightly stronger statement is straight-forward by counting the maximum number of elements in an (anti)chain.

► **Lemma 14.** *Let M be a k - \diamond -pattern and let $P(M)$ be its poset. Then every decomposition of $P(G)$ into a minimum number of chains and antichains consists either of k chains or of k antichains.*

That is, \diamond -patterns are particularly hard graphs for mixed linear layouts as they do not profit from the possibility to mix chains and antichains, respectively queues and stacks, which makes them a suitable candidate for a characterization. The remainder of this subsection is devoted to the upper bound of Theorem 12, which confirms that these candidates are, up to a factor of 2, the only reason for a large mixed page number.

The upper bound relies on insights into a Ferrer’s diagram, a standard combinatorial tool [30], that represents how many elements can be covered by a certain number of chains, respectively antichains. In general, a Ferrer’s diagram represents a partition of an integer by left-aligned rows of cells, where a summand s is represented by a row of length s and the rows are ordered by decreasing length from bottom to top. In our context, we partition the number $|P|$ of elements of a poset. Roughly speaking, the Ferrer’s diagram of a poset P consists of $|P|$ cells arranged in such a way that the number of cells in the first i rows is the



■ **Figure 3** Left: A grid representation of a matching M with edges indicating the comparabilities in the poset $P(M)$ of M . The 2- \diamond -pattern is highlighted red. Right: The Ferrer's diagram of $P(M)$ showing the partition $|P(M)| = 5 + 3 + 1$ by rows of length 5, 3 and 1. The diagram expresses that one chain can cover five elements (bottommost row), two chains can cover $5 + 3 = 8$ chains (two bottommost rows), and that three chains can cover all $5 + 3 + 1 = 9$ elements (all three rows). Note, however, that eight elements cannot be covered with two chains of length 5 and 3.

number of elements in P that can be covered by i chains, and the number of cells in the first i columns equals the number of elements that can be covered by i antichains. It is proved by Greene [39] that such diagrams exist. We refer readers not familiar with this kind of Ferrer's diagrams, sometimes also called Greene's diagrams, to the full version [41, Section 2.1] and also point to Figure 3 (right) for an example.

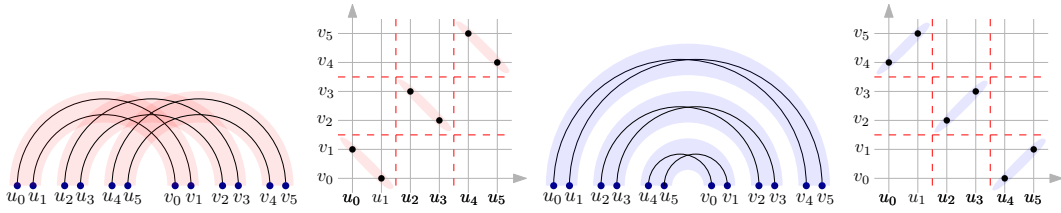
For Theorem 12, we want a bound depending on the size of the largest \diamond -pattern, so we first need to identify these patterns in the Ferrer's diagrams. Let us first discuss why this is not a trivial task. Recall that the Ferrer's diagram only indicates the number of elements that can be covered by a certain number of (anti)chains but does not associate the cells with elements of the poset as this is not always possible. For an example, consider Figure 3 where we have three rows of size 5, 3, and 1 but there is no chain decomposition with chains of lengths 5, 3, 1. That is, there is no way of writing elements into all cells of the diagram such that elements in the same row form a chain. Therefore, we cannot simply use the “elements in the $k \times k$ -square” to identify a k - \diamond -pattern.

Instead, besides the translation of the linear layout problem to posets and its Ferrer's diagrams, the core of the proof of Theorem 12 is a counting argument that allows to identify a \diamond -pattern of the same size as the largest square in the Ferrer's diagram. Having this, we may use $2k$ (anti)chains, where k is the size of the largest square instead of the \diamond -pattern, which can be done using the properties of a Ferrer's diagram (see full version [41, Section 2.1]). Finally, we remark that Theorem 12 is tight (see full version [41, Lemma 18]) and yields a 2-approximation in $O(n^{3/2} \log n)$ [33, 73].

2.2 Connection to Twists and Rainbows

Having Theorem 12, we can characterize which patterns cause a large mixed page number up to a factor of 2. In contrast to stack layouts and queue layouts, however, we have not only a single pattern like twists or rainbows, respectively, but a family of patterns. Note that all k - \diamond -patterns are necessary to forbid if we want an exact characterization as they all have mixed page number k . In this section, we give up the idea of an (almost) exact characterization in favor of reducing the number of patterns we forbid. More precisely, we point out two specific \diamond -patterns whose absence characterizes the mixed page number up to a polynomial factor.

As we may use both stacks and queues in mixed linear layouts, we combine twists and rainbows to obtain two new patterns for characterizing the mixed page number. A t -thick k -twist is obtained from a k -twist by replacing each edge by a t -rainbow. That is, we have



■ **Figure 4** From left to right: A 2-thick 3-twist with its grid representation, and a 2-thick 3-rainbow with its grid representation.

k pairwise vertex-disjoint t -rainbows, where each edge nests with the edges belonging to the same rainbow but crosses the edges of all other rainbows, see Figure 4 (left). Similarly, a t -thick k -rainbow is obtained from a k -rainbow by replacing each edge by a t -twist, see Figure 4 (right). In both cases, we call t the *thickness* and write the *size* as $k \times t$ to emphasize that we have kt edges organized as k smaller groups of size t . Throughout the paper, we often have $t = k$ and simply write k -thick rainbow, respectively k -thick twist. If we mean any of the two patterns, we say k -thick pattern.

In this section, we investigate the relation between thick patterns and \diamond -patterns. Specifically, we show that thick patterns occur if and only if \diamond -pattern occur, where the sizes are tied by a polynomial function. Together with Theorem 12 we obtain:

► **Theorem 15.** *Let M be a matching with a separated layout. If the largest thick pattern in M has size $k \times k$, then the mixed page number of M is at least k and at most $2k^7$.*

Note that the lower bound is already given by Theorem 12, so our task is to bound the mixed page number in terms of the largest thick pattern. Theorem 12 also gives an upper bound using \diamond -patterns instead of thick patterns, so we aim to show that if there is a large \diamond -pattern, than there also is a large thick pattern.

► **Proposition 16.** *Let M be a k^7 - \diamond -pattern, then M contains a k -thick pattern.*

The proof makes use of the grid representation, which is subdivided repeatedly. First, after each subdivision, we find a sufficiently large \diamond -pattern, and second these smaller patterns can be combined to a thick pattern. See the full version [41, Lemma 19] for the proof.

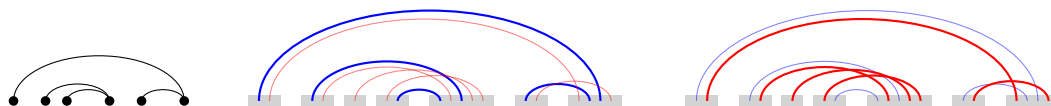
3 Non-separated Matchings and Bounded-Degree Graphs

To transfer our findings for separated layouts to the general matching case, we first need the notion of quotients. For this, consider a matching G with a fixed vertex ordering \prec together with a partition \mathcal{I} of the vertices into *intervals*, i.e., the vertices in each part are consecutive. The *quotient* G/\mathcal{I} of G and \mathcal{I} with respect to \prec is the graph obtained from contracting each interval together with the inherited vertex ordering. The following lemma shows that mixed linear layouts of quotients can be transferred to the original matching.

► **Lemma 17.** *Let $k \geq 1$, let G be a matching with a fixed vertex ordering without a $(k + 1)$ -thick pattern, let \mathcal{I} be a partition of the vertices into intervals, and let $H = G/\mathcal{I}$ be the quotient. Then we have*

$$\text{mn}(G) \leq 6\ell \cdot 2k^7(1 + 14(k + 1) \log(k + 1) + k) + 2m \in O(\ell k^8 \log(k) + m),$$

where $\ell = \text{mn}(H)$ is the mixed page number of the quotient and $m = \max_{I \in \mathcal{I}}(\text{mn}(G[I]))$ is the maximum mixed page number induced by some interval.



■ **Figure 5** A stack of the quotient H consisting of one-sided stars (left) and the matching induced in G (middle and right). Each star induces a separated pattern admitting a page assignment α with at most $2k^7$ stacks (blue) and queues (red). The set Q is formed by one queue per star (red). The stacks can be reused for distinct stars, while the edges in the queues of all stars together are partitioned into two groups – one having only small twists and one having only small rainbows. The intervals of \mathcal{I} are indicated by gray boxes.

Note that the lemma is not false for $k = 0$ but does not make sense to state as a 1-thick pattern is a 1-thick 1-rainbow or a 1-thick 1-twist, i.e., a single edge. Thus, forbidding a 1-thick pattern means that G has no edges.

Proof. First, observe that all edges with both endpoints in the same interval I can be covered by at most $\text{mn}(G[I])$ stacks plus the same number of queues. As the pages can be reused for all intervals, we have $2 \max_{I \in \mathcal{I}} (\text{mn}(G[I])) = 2m$ pages for all edges with both endpoints in the same interval.

Thus, our main task is to deal with edges in G between distinct intervals, that is with the edges of G induced by some edge of the quotient H . We only consider a single page of H , which we pay with a factor of ℓ . Now having a stack or a queue, it is well known that it can be partitioned into six one-sided star forests, i.e., for each part either all stars have their center to the left of their leaves or all to the right. That is, at the cost of a factor of 6, we may now consider a stack, respectively a queue, of H consisting of one-sided stars. Without loss of generality, we may assume that the stars have their center as their rightmost vertex. Note that a one-sided star in H induces a separated pattern in G consisting of a single interval (corresponding to the center of the star) on one side and possibly several intervals on the other. This is particularly convenient as we already know how to deal with separated matchings by Theorem 15. We fix a $2k^7$ -page assignment α for each subgraph of G induced by some one-sided star.

Stacks of H . We are now ready to construct a mixed linear layout for a subgraph of G induced by a one-sided star forest forming a stack in the layout of H , see Figure 5. First, edges in G that are assigned to a stack by α are easy to handle: As the stars do not cross, we may reuse the same $2k^7$ stacks. That is, we are left with the edges of G assigned to queues by α . Consider only one out of the up to $2k^7$ queues of each star and let Q denote the set of these edges. We pay this with a factor of $2k^7$. As the stars may nest, we cannot simply join the queues but need a more involved strategy. Observe that the edges of Q belonging to the same star in H are separated but do not nest, i.e., they form a twist. Next we aim to use these twists to either cover the edges with few pages or to find a large thick rainbow. For each twist, take the k leftmost edges (or all if there are less than k), and denote the union of all these edges of all twists by L . As the stars do not cross, the edges in L form twists of size at most k and can be covered by $14k \log(k)$ stacks [17].

For the remaining edges, we show that a rainbow implies a thick rainbow of the same size in G , and thus they can be covered by k queues. Indeed, consider a rainbow e_1, e_2, \dots in $Q - L$, where the edges nest above each other in this order. We refer to Figure 6 for an illustration of the upcoming argument. Recall that the edges of Q belonging to the same star form a twist and thus the edges in the rainbow belong to pairwise distinct stars. Let T_i



■ **Figure 6** Left: Non-crossing one-sided stars in H . Right: A rainbow $e_1, e_2, e_3 \in Q - L \subseteq E(G)$ together with the twist T_2 . Recall that e_2 is chosen such that it is the rightmost edge of T_2 . The right endpoints of T_2 are consecutive as they are in a common interval that is contracted to the center of the star in H . The left endpoints are consecutive as the respective edges in the stars do not cross and thus the edges of T_2 do not cross any edges of other stars like e_3 .

denote the $(k + 1)$ -twist consisting of e_i and the k edges in L induced by the same star. Also recall that the stars are in a stack of H and as their edges do not cross, the same holds for edges in G belonging to distinct stars. In particular, the edges of the twist T_i do not cross e_{i+1} nor e_{i-1} . Therefore, the left endpoints of T_i are between the left endpoints of e_i and e_{i+1} . For the right endpoints of T_i , recall that the stars are one-sided with the center to the very right. As such, the right endpoints of T_i belong to a common interval of \mathcal{I} , and thus are to the right of e_{i-1} . Since T_i is a twist, its right endpoints are between the right endpoints of e_{i-1} and e_i . This yields $(k + 1)$ -twists that are pairwise nesting, i.e., if there is a $(k + 1)$ -rainbow in $Q - L$, then we obtain a $(k + 1)$ -thick rainbow in G , a contradiction. Hence, there is no $(k + 1)$ -rainbow in $Q - L$ and k queues suffice for these edges.

To sum up, we have $2k^7$ stacks for the edges assigned to stacks by α , plus $14k \log(k)$ stacks for L and k queues for $Q - L$, which we do $2k^7$ times as this is the number of queues α may use. This yields $2k^7 + (14k \log(k) + k) \cdot 2k^7$ for all edges induced by one star forest in H that is in a stack in the layout of H .

Queues of H . Although not completely symmetric, the approach for a queue of H is fairly similar, which is why we refer to the full version [41, Lemma 20] here. ◀

Next we use Lemma 17 to prove our main result of this section, namely that it suffices to bound the size of the largest thick pattern to obtain bounded mixed page number. Recall that this is also necessary as k -thick patterns have mixed page number k .

► **Theorem 18.** *Let G be a matching with a fixed vertex ordering. If G does not contain a k -thick pattern, then the mixed page number of G is in $k^{O(k)}$.*

Proof. The idea is to start with G and apply Lemma 17 repeatedly, i.e., we define suitable intervals, contract them, and repeat the procedure on the quotient graph. After at most $k - 1$ steps, we show that the mixed page number of the obtained quotient graph is bounded. Applying Lemma 17 to the levels of contractions then gives that the mixed page number of G is bounded.

More detailed, we define a sequence of graphs H_1, H_2, \dots , starting with $H_1 = G$. If for some $i \geq 1$, there is no $(k + 1)$ -twist in H_i , the stack number, and thus the mixed page number of H_i is bounded by $\mathcal{O}(k \log k)$ [17] and we stop the procedure. Otherwise, we define H_{i+1} as the quotient of H_i and the partition \mathcal{I}_i into intervals I_1^i, I_2^i, \dots of H_i as follows. The first interval I_1^i starts with the first vertex of H_i in the given vertex ordering. From left to right, we add vertices to I_1^i until it contains a $(k + 1)$ -twist. In particular, the last vertex of I_1^i is the rightmost right endpoint of a $(k + 1)$ -twist and is followed by the first vertex of I_2^i . The next interval I_2^i is then defined in the same way, i.e., starting from its first vertex, it includes the minimum number of vertices such that it contains a $(k + 1)$ -twist (or all if there

is no $(k + 1)$ -twist in the remaining vertices). We continue until every vertex is in one of the intervals. Then, H_{i+1} is defined as the quotient H_i/\mathcal{I}_i . Note that every interval, except possibly the last one, contains a $(k + 1)$ -twist. On the other hand, by construction, there is no $(k + 2)$ -twist in the subgraph of H_i induced by any of the intervals, so the mixed page number within the intervals is bounded by $\mathcal{O}(k \log k)$ [17].

Next we show that the procedure stops with H_k (or before). Suppose to the contrary that there is always a $(k + 1)$ -twist in H_i for $i = 1, \dots, k$. We show that this implies a k -thick rainbow in G , which is a contradiction. That is, we now look for k -twists, one in each H_i , that nest above each other. For this, consider a $(k + 1)$ -twist with vertices $u_1, \dots, u_{k+1}, v_1, \dots, v_{k+1}$ in H_i , for $2 \leq i \leq k$, and note that the first k edges form a k -twist which nests above u_{k+1} . Now recall that every vertex of H_i (except for the last) is the result of contracting an interval of H_{i-1} containing a $(k + 1)$ -twist. Thus, a $(k + 1)$ -twist in H_i corresponds to a k -twist nesting above a $(k + 1)$ -twist of H_{i-1} . By transitivity of the nesting relation, this gives a k -thick rainbow, a contradiction.

It follows that we make at most $k - 1$ steps until H_k is the last quotient we obtain and has no further $(k + 1)$ -twist. Then, the mixed page number of H_k is bounded by $\mathcal{O}(k \log k)$. Also recall that the mixed page number within the intervals is $\mathcal{O}(k \log k)$ and thus is dominated by the quotient. Hence for H_{k-1} , Lemma 17 yields $\text{mn}(H_{k-1}) \leq \text{mn}(H_k) \cdot c \cdot k^8 \log k + \max_j \text{mn}(I_j^{k-1}) \in \mathcal{O}(k \log k \cdot k^8 \log k)$, where $c \leq 6 \cdot 2 \cdot 14 = 168$ is the constant from the big-O notation of Lemma 17. Similarly, applying Lemma 17 to H_{i-1} and $H_i = H_{i-1}/\mathcal{I}_{i-1}$ for $2 \leq i \leq k$ gives us a factor of $c \cdot k^8 \log k + \mathcal{O}(k \log k)$ each time. After $k - 1$ steps, we obtain an upper bound of $\mathcal{O}(c^{k-1} \cdot k^{8(k-1)+1} \log^k(k)) \in k^{\mathcal{O}(k)}$ on the mixed page number of $H_1 = G$. ◀

With Vizings's theorem [76], Theorem 18 generalizes to bounded-degree graphs. Note that for both theorems, the size of the largest thick pattern gives a trivial linear lower bound.

► **Theorem 19.** *Let G be an ordered graph with maximum degree Δ . If G does not contain a k -thick pattern, then the mixed page number of G is in $\Delta k^{\mathcal{O}(k)}$.*

4 Critical Graphs

In this section, we focus on the existence of a one-to-one characterization of mixed linear layouts. Note that up until now, we only identified patterns characterizing the mixed page number up to a function, i.e., we only get an upper and lower bound on the mixed page number even if we know exactly the size of the largest pattern. Here, we are attacking the question for an exact characterization, which would guarantee the exact mixed page number, given that all of the specified patterns are excluded. To this end, we introduce the concept of *critical graphs* that are minimal graphs that do not admit a mixed linear layout with a certain number of pages. Suppose $G = (V, E)$ is a graph with a fixed vertex order \prec . For $s, q \geq 0$, we call G (s, q) -critical if it does not admit an s -stack q -queue layout under \prec but every subgraph $G - e$ for all edges e does. Similarly, we define k -critical graphs as minimal graphs not admitting a layout on mixed $k = s + q$ (for some $s, q \geq 0$) pages.

We begin by investigating critical graphs for separated layouts. In the case of separated matchings, prior results [31, 52, 77] imply that the number of critical graphs is finite. However, for separated non-matchings, we can only bound the number of k -critical graphs when $k = 2$, and it remains open for larger k . In the non-separated case, prior hardness results [19, 74] imply that the number of (s, q) -critical graphs is infinite for $s \geq 4$ and $q \geq 0$, even for matchings. For $k = 2$ and for $s \geq 2$ we construct infinite sets of matchings that are k - and

	matching	non-matching
separated	finite	$k = 2$: finite (Theorem 21); $k > 2$: open
non-separated	$k = 2$: infinite (Theorem 26); $k \neq 2$: open; $0 < s < 4, q > 0$: open; $s = 0, q \geq 0$: finite; $s \geq 2, q = 0$: infinite (Theorem 25); $s \geq 4, q \geq 0$: infinite	

■ **Figure 7** An overview visualizing whether the number of k -critical and (s, q) -critical graphs is finite or infinite in the cases of separated/non-separated layouts of matchings/non-matchings. The blue cases follow immediately from previous results, which we discuss in Section 4.1 and Section 4.2.

$(s, 0)$ -critical, respectively. For all other cases, particularly for all $k \neq 2$, the question of whether there exists a finite set of patterns that exactly characterizes a mixed page number of k remains unresolved. Our results are summarized in Figure 7.

4.1 Critical Graphs for Separated Layouts

A separated matching G corresponds to a permutation $\pi(G)$. The separated mixed page number of a matching G equals the minimum number of monotone subsequences that $\pi(G)$ can be partitioned into. In this setting the following is known: permutations that can be partitioned into a fixed number of monotone subsequences are characterized by a finite set of forbidden subsequences [31, 52, 77]. Thus, for every pair (s, q) , there exists a finite number of (s, q) -critical matchings with a fixed separated layout. For separated non-matchings our main result is that there is only a finite number of 2-critical graphs.

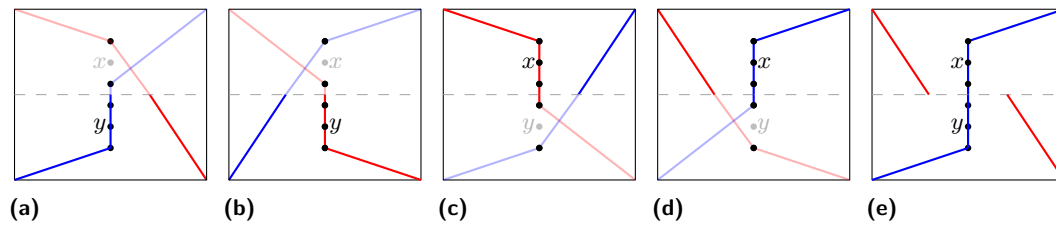
► **Theorem 20.** *There exists a finite number of separated $(1, 1)$ -critical and 2-critical graphs.*

On the way to proving this, we additionally show that if the maximum degree of all (s, q) -critical graphs is bounded, then the total number of (s, q) -critical graphs must be finite. Further, we show that if the number of (s, q) -critical graphs is bounded for all pairs (s, q) with $s + q = k$, then the number of k -critical graphs is bounded. Therefore, to show that the number of k -critical graphs is finite, not only for $k = 2$ but for arbitrary k , it suffices to show that the maximum degree of all (s, q) -critical graphs is bounded. Then, for $(1, 1)$ -critical graphs, we show that the maximum degree is indeed bounded, so we can conclude that the number of $(1, 1)$ -critical graphs is finite.

Before proving Theorem 20, we remark that a similar characterization of graphs with a bounded separated pure stack (queue) number is straightforward. In fact, there exists a single $(s, 0)$ -critical graph and a single $(0, q)$ -critical graph for all $s, q \geq 1$. In contrast, characterizations for mixed linear layouts are significantly more complex. We computationally identified all 9 graphs that are 1-critical, all 20 graphs that are $(1, 1)$ -critical, and all 3128 graphs that are 2-critical.

Our proof uses essentially the same arguments as [77] to show that if the maximum degree in an (s, q) -critical graph G is bounded, then the total number of edges of G is bounded. The key in the arguments in [77] is that partial Boolean functions, interpreted as an assignment of the edges of a subgraph to the set of stacks or to the set of queues, can be combined to a total Boolean function corresponding to the given graph. See the full version [41, Section 4.1] for details.

The following lemma is the result of these arguments and potentially useful for proving that the number of (s, q) -critical graphs is finite for arbitrary s and q . For this, it now suffices to show that the maximum degree in every (s, q) -critical graph is bounded.



■ **Figure 8** There are two cases each for the 1-stack 1-queue layouts of $G - x$ and $G - y$: Either the increasing sequence (blue) contains all points below x , resp. y , while the decreasing sequence (red) covers all points above (a and c), or the other way around (b and d). (e) visualizes the combination of (a) and (d) into a 1-stack 1-queue layout of G .

► **Lemma 21.** *There is a function $N^*(s, q, \Delta)$ such that for every s, q it holds that every separated (s, q) -critical graph with maximum degree Δ contains at most $N^*(s, q, \Delta)$ edges.*

Additionally, the following lemma allows us to extend results on (s, q) -critical graphs to more general k -critical graphs. Specifically, it shows that if the number of (s, q) -critical graphs is bounded for all pairs (s, q) with $s + q = k$, then the number of k -critical graphs is also bounded. The proof is not surprising and also inspired by [77] so we refer to the full version [41, Lemma 22]

► **Lemma 22.** *Suppose that the number of edges in an (s, q) -critical graph is bounded by some function $m(s, q)$ for every s, q . Then, for any k , the number of edges in a k -critical graph is bounded by $\sum_{s+q=k} m(s, q)$.*

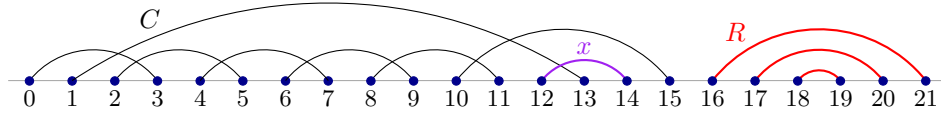
Note that in the separated case there is exactly one $(2, 0)$ -critical graph – a 3-twist – and exactly one $(0, 2)$ -critical graph – a 3-rainbow. Moreover, the following lemma, together with Lemma 21, shows that the number of $(1, 1)$ -critical graphs is also bounded. Applying Lemma 22, it then follows that the number of 2-critical graphs is finite. With Lemma 21, the only thing that is left to finish Theorem 20 is to prove that the maximum degree in any $(1, 1)$ -critical graph is bounded.

► **Lemma 23.** *Let G be a separated $(1, 1)$ -critical graph. Then $\Delta(G) < 6$.*

The following proof sketch is worked out more detailed in the full version [41, Lemma 24].

Proof Sketch. Consider the grid representation of G , i.e., vertices are represented by columns and rows, with a point in the intersection if the two vertices share an edge. Suppose that G has a vertex of degree 6, say represented by a column. Since G is critical, removing any point yields a 1-stack 1-queue graph, i.e., all remaining points can be covered by an increasing and a decreasing sequence together. We choose two such points x and y and obtain two pairs of sequences that we combine to one increasing and one decreasing sequence covering all points, i.e., all edges of G . This contradicts G being $(1, 1)$ -critical.

Figure 8 shows how two such layouts are recombined. To see that the situation indeed is as illustrated, observe that none of the sequences contains points above and below the removed point as then the point could simply be added. For the combination to work, it is crucial to choose x and y such that at least two vertices are between them and they are not the topmost or bottommost vertex. Together with the first observation, this guarantees that the order in which the sequences reach the cutting line fit together. ◀



■ **Figure 9** A forbidden edge pattern with $k = 6$ for $mn = 2$.

4.2 Critical Graphs for Non-Separated Layouts

Based on the literature, we first observe that the number of critical graphs characterizing non-separated s -stack q -queue layouts is expected to be infinite even for matchings, for $s \geq 4$. To this end, we use a hardness result stating that it is NP-complete to recognize 4-stack layout for matchings with a fixed layouts (from coloring circle graphs) [74]. Moreover, an already NP-complete mixed linear layout recognition problem with given vertex ordering remains NP-complete under addition of a stack or queue [19]. Thus, deciding whether a matching with a given vertex ordering admits an s -stack q -queue layout is NP-complete for all $s \geq 4$ and $q \geq 0$. On the other hand, a finite forbidden set of critical graphs implies a poly-time recognition, which would contradict the two mentioned hardness results under the assumption that $P \neq NP$. However, it is still interesting to construct an explicit infinite set of critical graphs, for fixed $k = s + q$, especially for $k < 4$ where the state of the art does not give whether or not the obstruction set is finite. In the following, we construct such an infinite set of $(s, 0)$ - and 2-critical graphs for $s \geq 2$.

As a first step, we give an infinite obstruction set for layouts with $s = 2, q = 0$. We use edges such that the conflicting edges (that cannot share a stack) form an odd-length cycle; clearly, the cycle requires three colors (stacks). In the full version [41, Lemma 25] this is generalized to an infinite obstruction set for all pure stack layouts with stack number at least 2.

► **Theorem 24.** *For every integer $s \geq 2$ and every $n \geq 3$, there exists an $(s, 0)$ -critical matching with at least n vertices.*

Observe that Theorem 24 does not rule out a possibility of a finite obstruction set for mixed linear layouts with $mn = 2$. The next theorem closes this gap.

► **Theorem 25.** *For every $n \geq 3$, there exists a 2-critical matching with at least n vertices.*

Proof. Let $k \geq 2$ be an even integer. We build a matching with $n = 2(k + 2) + 6$ vertices (starting with 0) having three types of edges (see Figure 9):

- $C = \{(2i, 2i + 3), 0 \leq i < k - 1\} \cup \{(2k - 2, 2k + 3)\} \cup \{(1, 2k + 1)\}$;
- a single edge $x = (2k, 2k + 2)$;
- $R = \{(n - 6, n - 1)\} \cup \{(n - 5, n - 2)\} \cup \{(n - 4, n - 3)\}$.

We claim that the graph $G_k = (\{0, \dots, n - 1\}, C \cup \{x\} \cup R)$ is 2-critical, that is, $mn(G_k) = 3$ but $mn(G_k - e) = 2$ for every edge e . First, we show that $G_k - e$ admits a mixed linear layout on two pages for every edge e , then that G_k requires at least 3 pages. We consider the three cases $e \in C$, $e = x$, and $e \in R$ separately. First, if we remove an edge from C , then the graph admits a 2-stack layout: The edges of $C \setminus \{e\}$ can be assigned to two stacks, while x and edges from R are assigned to one of the two stacks. Second, if we remove edge x , then the graph admits a 1-stack 1-queue layout: The “long” edge $(1, 2k + 1)$ of C together with edges from R are assigned to a stack, while the “short” edges $(2i, 2i + 3)$ of C are assigned to a queue. And third, if we remove an edge from R , then the graph admits a 2-queue layout: One queue contains an edge from R , edge x , and the “long” edge $(1, 2k + 1)$ from C . Another queue contains an edge from R along with the “short” edges $(2i, 2i + 3)$ from C .

Finally, graph G_k does not admit a layout on two (mixed) pages, since on one hand (i) it cannot be assigned to two queues (R forms a 3-rainbow), and (ii) it cannot be assigned to two stacks (C is an odd cycle). On the other hand, (iii) G_k cannot be assigned to a stack and a queue, since otherwise the “long” edge $(1, 2k + 1)$ of C is in the queue (it crosses two edges forming a 2-rainbow) and hence, all edges covered by the “long” edge would be in a stack, which implies a crossing between two such stack edges. ◀

To summarize, we know that the number of (s, q) -critical graphs is infinite if $s \geq 4$, even for matchings, due to previous hardness results [19, 74]. Furthermore, Theorem 24 and Theorem 25 show that for $s \geq 2$ the numbers of $(s, 0)$ - and 2-critical matchings are also infinite, providing a construction for an infinite, though not necessarily complete, set of such graphs. Recall that the $(0, q)$ -critical graphs are exactly the $(q + 1)$ -rainbows [44], i.e., there is exactly one such graph for every $q \in \mathbb{N}$. What remains open is the case $s < 4$ and $q \geq 1$, as well as the more general k -critical graphs for any $k \neq 2$. Notably, it is even unknown whether the set of 1-critical graphs is finite. For $s = q = 1$, we conjecture that there there is a finite number of critical graphs and present candidates in the full version [41, Conjecture 27].

5 Conclusions

In this paper we made the first steps towards characterizing mixed linear layouts of ordered graphs via forbidden patterns. The most prominent open question for fully resolving Open Problem 3 is to transfer Theorem 4 to general graphs with unbounded maximum degree. We remark that the proofs in Section 3 work similarly for general ordered graphs; hence, the challenge is to bound the mixed page number of bipartite graphs in the separated settings. We expect that thick patterns is the correct choice even for large-degree graphs.

Another interesting question is whether separated mixed linear layouts are characterized by a finite obstruction set; that is, whether the statement of Section 4.1 holds for all k . Again we expect a positive answer here, and observe that for a proof, it is sufficient to bound the maximum degree of separated k -critical graphs for $k \geq 3$, that is, extend Lemma 23 for the case.

Finally, we highlight a possible application of the studied characterizations. Is the mixed page number of upward planar (possibly, bounded-degree) graphs bounded by a constant? To answer the question affirmatively, it is sufficient for a graph to construct a topological ordering containing no k -thick pattern for some $k \in \mathbb{N}$.

References

- 1 A. A. Ageev. A triangle-free circle graph with chromatic number 5. *Discrete Mathematics*, 152(1-3):295–298, May 1996. doi:10.1016/0012-365X(95)00349-2.
- 2 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. *Algorithmica*, 82(9):2564–2585, September 2020. doi:10.1007/s00453-020-00697-4.
- 3 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. The mixed page number of graphs. *Theoretical Computer Science*, 2022. doi:10.1016/j.tcs.2022.07.036.
- 4 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Lazy queue layouts of posets. *Algorithmica*, 85(5):1176–1201, May 2023. doi:10.1007/s00453-022-01067-y.



- 5 Patrizio Angelini, Michael A. Bekos, Philipp Kindermann, and Tamara Mchedlidze. On mixed linear layouts of series-parallel graphs. *Theoretical Computer Science*, 936:129–138, November 2022. doi:10.1016/j.tcs.2022.09.019.
- 6 Patrizio Angelini, Giordano Da Lozzo, Henry Förster, and Thomas Schneck. 2-layer k-planar graphs: Density, crossing lemma, relationships, and pathwidth. In *Graph Drawing and Network Visualization: 28th International Symposium, GD 2020*, pages 403–419, 2020. doi:10.1007/978-3-030-68766-3_32.
- 7 Michael J Bannister, William E Devanny, Vida Dujmović, David Eppstein, and David R Wood. Track layouts, layered path decompositions, and leveled planarity. *Algorithmica*, 81:1561–1583, 2019. doi:10.1007/s00453-018-0487-5.
- 8 Michael Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. An improved upper bound on the queue number of planar graphs. *Algorithmica*, 85(2):544–562, February 2023. doi:10.1007/s00453-022-01037-4.
- 9 Michael Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi Raftopoulou, and Torsten Ueckerdt. Four pages are indeed necessary for planar graphs. *Journal of Computational Geometry*, pages 332–353 Pages, August 2020. doi:10.20382/JOCG.V11I1A12.
- 10 Michael A. Bekos, Till Bruckdorfer, Michael Kaufmann, and Chrysanthi N. Raftopoulou. The book thickness of 1-planar graphs is constant. *Algorithmica*, 79(2):444–465, October 2017. doi:10.1007/s00453-016-0203-2.
- 11 Michael A. Bekos, Henry Förster, Martin Gronemann, Tamara Mchedlidze, Fabrizio Montecchiani, Chrysanthi Raftopoulou, and Torsten Ueckerdt. Planar graphs of bounded degree have bounded queue number. *SIAM Journal on Computing*, 48(5):1487–1502, January 2019. doi:10.1137/19M125340X.
- 12 Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, December 1979. doi:10.1016/0095-8956(79)90021-2.
- 13 Sujoy Bhore, Giordano Da Lozzo, Fabrizio Montecchiani, and Martin Nöllenburg. On the upward book thickness problem: Combinatorial and complexity results. *European Journal of Combinatorics*, 110:103662, 2023. doi:10.1016/j.ejc.2022.103662.
- 14 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward book embeddability of st-graphs: Complexity and algorithms. *Algorithmica*, 85(12):3521–3571, 2023. doi:10.1007/S00453-023-01142-Y.
- 15 Jakub Černý. Coloring circle graphs. *Electronic notes in Discrete mathematics*, 29:457–461, 2007. doi:10.1016/j.endm.2007.07.072.
- 16 Sabine Cornelsen, Thomas Schank, and Dorothea Wagner. Drawing graphs on two and three lines. *Journal of Graph Algorithms and Applications*, 8(2):161–177, January 2004. doi:10.7155/jgaa.00087.
- 17 James Davies. Improved bounds for colouring circle graphs. *Proceedings of the American Mathematical Society*, July 2022. doi:10.1090/proc/16044.
- 18 James Davies and Rose McCarty. Circle graphs are quadratically χ -bounded. *Bulletin of the London Mathematical Society*, 53(3):673–679, 2021. doi:10.1112/blms.12447.
- 19 Philipp de Col, Fabian Klute, and Martin Nöllenburg. Mixed linear layouts: Complexity, heuristics, and experiments. In *Graph Drawing and Network Visualization: 27th International Symposium, GD 2019*, pages 460–467, Berlin, Heidelberg, 2019. Springer-Verlag. doi:10.1007/978-3-030-35802-0_35.
- 20 H. de Fraysseix, P. O. de Mendez, and J. Pach. A left-first search algorithm for planar graphs. *Discrete & Computational Geometry*, 13(3):459–468, June 1995. doi:10.1007/BF02574056.
- 21 Emilio Di Giacomo, Walter Didimo, Peter Eades, and Giuseppe Liotta. 2-layer right angle crossing drawings. *Algorithmica*, 68:954–997, January 2014. doi:10.1007/S00453-012-9706-7.
- 22 Vida Dujmović, Gwenaél Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R Wood. Planar graphs have bounded queue-number. *Journal of the ACM (JACM)*, 67(4):1–38, 2020. doi:10.1145/3385731.

- 23 Vida Dujmović, Attila Pór, and David R Wood. Track layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):497–522, 2004. doi:10.46298/dmtcs.315.
- 24 Vida Dujmović and David R Wood. Stacks, queues and tracks: Layouts of graph subdivisions. *Discrete Mathematics and Theoretical Computer Science*, 7:155–202, 2005. doi:10.46298/dmtcs.346.
- 25 Vida Dujmović, Pat Morin, and David R. Wood. Queue layouts of graphs with bounded degree and bounded genus, 2019. arXiv:1901.05594, doi:10.48550/arXiv.1901.05594.
- 26 Vida Dujmović, Pat Morin, and Céline Yelle. Two results on layered pathwidth and linear layouts. *Journal of Graph Algorithms and Applications*, 25(1):43–57, 2021. doi:10.7155/jgaa.00549.
- 27 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, Vol. 6 no. 2:317, 2004. doi:10.46298/dmtcs.317.
- 28 Peter Eades and Sue Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374, 1994. doi:10.1016/0304-3975(94)90179-1.
- 29 Peter Eades and Nicholas C Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.
- 30 Martin J. Erickson. *Introduction to Combinatorics*. John Wiley & Sons, Ltd, 1996. doi:10.1002/9781118032640.
- 31 Tomás Feder and Pavol Hell. Matrix partitions of perfect graphs. *Discrete Mathematics*, 306(19-20):2450–2460, 2006. doi:10.1016/j.disc.2005.12.035.
- 32 Stefan Felsner, Torsten Ueckerdt, and Kaja Wille. On the queue-number of partial orders. In Helen C. Purchase and Ignaz Rutter, editors, *Graph Drawing and Network Visualization: 29th International Symposium, GD 2021*, pages 231–241, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-92931-2_17.
- 33 Stefan Felsner and Lorenz Wernisch. Maximum k-chains in planar point sets: Combinatorial structure and algorithms. *SIAM Journal on Computing*, 28(1):192–209, 1998. doi:10.1137/S0097539794266171.
- 34 Fabrizio Frati, Radoslav Fulek, and Andres Ruiz-Vargas. On the page number of upward planar directed acyclic graphs. *Journal of Graph Algorithms and Applications*, 17(3):221–244, March 2013. doi:10.7155/jgaa.00292.
- 35 Henry Förster, Michael Kaufmann, Laura Merker, Sergey Pupyrev, and Chrysanthi Raftopoulou. Linear layouts of bipartite planar graphs. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures*, pages 444–459, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-38906-1_29.
- 36 Zoltán Füredi and Péter Hajnal. Davenport-Schinzel theory of matrices. *Discrete Mathematics*, 103(3):233–251, May 1992. doi:10.1016/0012-365X(92)90316-8.
- 37 M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic Discrete Methods*, 1(2):216–227, 1980. doi:10.1137/0601025.
- 38 Jesse Geneson. Almost all permutation matrices have bounded saturation functions. *The Electronic Journal of Combinatorics*, 28(P2.16), May 2021. doi:10.37236/10124.
- 39 Curtis Greene. Some partitions associated with a partially ordered set. *Journal of Combinatorial Theory, Series A*, 20(1):69–79, 1976. doi:10.1016/0097-3165(76)90078-9.
- 40 A. Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Mathematics*, 55(2):161–166, July 1985. doi:10.1016/0012-365X(85)90044-5.
- 41 Deborah Haun, Laura Merker, and Sergey Pupyrev. Forbidden patterns in mixed linear layouts, 2024. doi:10.48550/arXiv.2412.12786.
- 42 Lenwood S. Heath and Sorin Istrail. The pagenumber of genus g graphs is $o(g)$. *J. ACM*, 39(3):479–501, July 1992. doi:10.1145/146637.146643.
- 43 Lenwood S. Heath, Sriram V. Pemmaraju, and Ann N. Trenk. Stack and queue layouts of directed acyclic graphs: Part i. *SIAM Journal on Computing*, 28(4):1510–1539, January 1999. doi:10.1137/S0097539795280287.

- 44 Lenwood S Heath and Arnold L Rosenberg. Laying out graphs using queues. *SIAM Journal on Computing*, 21(5):927–958, 1992. doi:10.1137/0221055.
- 45 Robert Hickingbotham and David R. Wood. Shallow minors, graph products, and beyond-planar graphs. *SIAM Journal on Discrete Mathematics*, 38(1):1057–1089, 2024. doi:10.1137/22M1540296.
- 46 Le Tu Quoc Hung. A planar poset which requires 4 pages. *Ars Combinatoria*, 35:291–302, 1993.
- 47 Barnabás Janzer, Oliver Janzer, Van Magnan, and Abhishek Methuku. Tight general bounds for the extremal numbers of 0–1 matrices. *International Mathematics Research Notices*, 2024(15):11455–11463, June 2024. doi:10.1093/imrn/rnae129.
- 48 Paul Jungeblut, Laura Merker, and Torsten Ueckerdt. Directed acyclic outerplanar graphs have constant stack number. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1937–1952, November 2023. doi:10.1109/FOCS57990.2023.00118.
- 49 Paul Jungeblut, Laura Merker, and Torsten Ueckerdt. A sublinear bound on the page number of upward planar graphs. *SIAM Journal on Discrete Mathematics*, 37(4):2312–2331, 2023. doi:10.1137/22M1522450.
- 50 Paul C. Kainen. Some recent results in topological graph theory. In Ruth A. Bari and Frank Harary, editors, *Graphs and Combinatorics*, pages 76–108, Berlin, Heidelberg, 1974. Springer Berlin Heidelberg. doi:10.1007/BFb0066436.
- 51 Julia Katheder, Michael Kaufmann, Sergey Pupyrev, and Torsten Ueckerdt. Transforming stacks into queues: Mixed and separated layouts of graphs. In *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, 2024. doi:10.48550/arXiv.2409.17776.
- 52 André E Kézdy, Hunter S Snevily, and Chi Wang. Partitioning permutations into increasing and decreasing subsequences. *Journal of Combinatorial Theory, Series A*, 73(2):353–359, 1996. doi:10.1016/S0097-3165(96)80012-4.
- 53 Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. The queue-number of posets of bounded width or height. In Therese Biedl and Andreas Kerren, editors, *Graph Drawing and Network Visualization: 26th International Symposium, GD 2018*, pages 200–212, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-030-04414-5_14.
- 54 Alexandr Kostochka. Upper bounds on the chromatic number of graphs. *Trudy Inst. Mat. (Novosibirsk)*, 10(Modeli i Metody Optim.):204–226, 1988.
- 55 Alexandr Kostochka and Jan Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163(1-3):299–305, 1997. doi:10.1016/S0012-365X(96)00344-5.
- 56 Joseph L. Ganley and Lenwood S. Heath. The pagenumber of k-trees is $o(k)$. *Discrete Applied Mathematics*, 109(3):215–221, May 2001. doi:10.1016/S0166-218X(00)00178-5.
- 57 S. M. Malitz. Genus g graphs have pagenumber $O(\sqrt{g})$. *Journal of Algorithms*, 17(1):85–109, 1994. doi:10.1006/jagm.1994.1028.
- 58 Adam Marcus and Gábor Tardos. Excluded permutation matrices and the Stanley–Wilf conjecture. *Journal of Combinatorial Theory, Series A*, 107(1):153–160, July 2004. doi:10.1016/j.jcta.2004.04.002.
- 59 Tamara Mchedlidze and Antonios Symvonis. Crossing-free acyclic hamiltonian path completion for planar st-digraphs. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation (ISAAC 2009)*, volume 5878 of *Lecture Notes in Computer Science*, pages 882–891, 2009. doi:10.1007/978-3-642-10631-6_89.
- 60 Miki Miyauchi. Topological stack-queue mixed layouts of graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E103.A(2):510–522, 2020. doi:10.1587/transfun.2019EAP1097.
- 61 Hiroshi Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discrete & Computational Geometry*, 33(4):569–591, 2005. doi:10.1007/s00454-005-1168-0.

- 62 Richard Nowakowski and Andrew Parker. Ordered sets, pagenumbers and planarity. *Order*, 6(3):209–218, September 1989. doi:10.1007/BF00563521.
- 63 Martin Nöllenburg and Sergey Pupyrev. On families of planar dags with constant stack number. In *Graph Drawing and Network Visualization: 31st International Symposium, GD 2023*, pages 135–151, Berlin, Heidelberg, January 2024. Springer-Verlag. doi:10.1007/978-3-031-49272-3_10.
- 64 L. Taylor Ollmann. On the book thicknesses of various graphs. In *Proc. 4th Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume 8, 1973.
- 65 Shannon Overbay. *Generalized book embeddings*. Phd thesis, Colorado State University, USA, November 1998.
- 66 János Pach and Gábor Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel Journal of Mathematics*, 155:359–380, 2006. doi:10.1007/BF02773960.
- 67 Sriram V Pemmaraju. *Exploring the powers of stacks and queues via graph layouts*. PhD thesis, Virginia Tech, 1992.
- 68 Sergey Pupyrev. Mixed linear layouts of planar graphs. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization: 25th International Symposium, GD 2017*, pages 197–209, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-73915-1_17.
- 69 Sergey Pupyrev. Queue layouts of two-dimensional posets. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Graph Drawing and Network Visualization: 30th International Symposium, GD 2022*, pages 353–360, Cham, 2023. Springer International Publishing. doi:10.1007/978-3-031-22203-0_25.
- 70 Matthew Suderman. Pathwidth and layered drawings of trees. *International Journal of Computational Geometry & Applications*, 14(03):203–225, 2004. doi:10.1142/S0218195904001433.
- 71 Gábor Tardos. On 0–1 matrices and small excluded submatrices. *Journal of Combinatorial Theory, Series A*, 111(2):266–288, August 2005. doi:10.1016/j.jcta.2004.11.015.
- 72 Gábor Tardos. Extremal theory of ordered graphs. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3235–3243, Rio de Janeiro, Brazil, May 2019. WORLD SCIENTIFIC. doi:10.1142/9789813272880_0179.
- 73 Alexander Tiskin. Fast RSK correspondence by doubling search. 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 86:1–86:10, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.86.
- 74 Walter Unger. On the k-colouring of circle-graphs. In Robert Cori and Martin Wirsing, editors, *STACS 88*, pages 61–72, Berlin, Heidelberg, 1988. Springer. doi:10.1007/BFb0035832.
- 75 Walter Unger. The complexity of colouring circle graphs. In *STACS 92: 9th Annual Symposium on Theoretical Aspects of Computer Science Cachan, France, February 13–15, 1992 Proceedings 9*, pages 389–400. Springer, 1992. doi:10.1007/3-540-55210-3_199.
- 76 V. G. Vizing. The chromatic class of a multigraph. *Cybernetics*, 1(3):32–41, May 1965. doi:10.1007/BF01885700.
- 77 David Wörn. Partitioning permutations into monotone subsequences. *Electron. J. Comb.*, 28(3), 2021. doi:10.37236/10267.
- 78 David R Wood. Bounded-degree graphs have arbitrarily large queue-number. *Discrete Mathematics & Theoretical Computer Science*, 10(1), 2008. doi:10.46298/dmtcs.434.
- 79 David R Wood. 2-layer graph drawings with bounded pathwidth. *Journal of Graph Algorithms and Applications*, 27(9):843–851, November 2023. doi:10.7155/jgaa.00647.
- 80 Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36–67, February 1989. doi:10.1016/0022-0000(89)90032-9.
- 81 Mihalis Yannakakis. Planar graphs that need four pages. *Journal of Combinatorial Theory, Series B*, 145:241–263, November 2020. doi:10.1016/j.jctb.2020.05.008.

Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

Úrsula Hébert-Johnson  

University of California, Santa Barbara, CA, USA

Daniel Lokshtanov  

University of California, Santa Barbara, CA, USA

Abstract

We design an algorithm that generates an n -vertex unlabeled chordal graph uniformly at random in expected polynomial time. Along the way, we develop the following two results: (1) an FPT algorithm for counting and sampling labeled chordal graphs with a given automorphism π , parameterized by the number of moved points of π , and (2) a proof that the probability that a random n -vertex labeled chordal graph has a given automorphism $\pi \in S_n$ is at most $1/2^{c \max\{\mu^2, n\}}$, where μ is the number of moved points of π and c is a constant. Our algorithm for sampling unlabeled chordal graphs calls the aforementioned FPT algorithm as a black box with potentially large values of the parameter μ , but the probability of calling this algorithm with a large value of μ is exponentially small.

2012 ACM Subject Classification Theory of computation \rightarrow Generating random combinatorial structures; Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases Chordal graphs, graph sampling, graph counting, unlabeled graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.46

Related Version *Full Version:* <https://arxiv.org/abs/2501.05024>

Funding Úrsula Hébert-Johnson: Supported by NSF grant CCF-2147094.

Acknowledgements We would like to thank Eric Vigoda for discussions that led to a concise proof of an important step in the running time analysis.

1 Introduction

A graph is *chordal* if it has no induced cycles of length at least 4. The term was coined by Gavril in 1972 [12], more than fifty years ago, but the notion of chordal graphs in fact goes as far back as 1958 [13]. In the early papers, chordal graphs were referred to by other names, such as *triangulated* graphs. By now, many structural results have been proved about chordal graphs, and there are many algorithms that take a chordal graph as input. Thus it would be useful to have an efficient algorithm for generating random chordal graphs, both for the practical purpose of software testing, as well as the more mathematical purpose of testing conjectures.

In [14], Hébert-Johnson et al. designed an algorithm that generates n -vertex labeled chordal graphs uniformly at random. This algorithm runs in polynomial time, using at most $O(n^7)$ arithmetic operations for the first sample and $O(n^4)$ arithmetic operations for each subsequent sample. However, when discussing the performance of an algorithm that is being tested, the correct output typically does not depend on the labeling of the vertices. If we use a labeled-graph sampling algorithm to generate random test cases, then asymmetric graphs will be given too much weight/probability compared to those that happen to have many automorphisms. This naturally leads to the question of efficiently generating *unlabeled* chordal graphs uniformly at random. In this paper, we present an algorithm that solves this problem and runs in expected polynomial time.



© Úrsula Hébert-Johnson and Daniel Lokshtanov;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 46; pp. 46:1–46:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



► **Theorem 1.** *There is a randomized algorithm that given $n \in \mathbb{N}$, generates a graph uniformly at random from the set of all unlabeled chordal graphs on n vertices. This algorithm uses $O(n^7)$ arithmetic operations in expectation.*

It is worth mentioning the difference between the running times for labeled vs. unlabeled chordal graph sampling. The sampling algorithm of Hébert-Johnson et al. generates a random n -vertex labeled chordal graph in polynomial time, even in the worst case. However, obtaining a worst-case polynomial-time algorithm for sampling *unlabeled* chordal graphs – or an expected-polynomial-time counting algorithm for such graphs – appears to be very difficult since these questions remain open even for general graphs. This is relevant because the class of chordal graphs is known to be GI-complete. While there exist classes of graphs (which we discuss below) for which efficient unlabeled sampling algorithms are known, we are not aware of any GI-complete graph class with such an algorithm.

Our algorithm for sampling unlabeled chordal graphs builds upon an algorithm of Wormald that generates n -vertex unlabeled graphs uniformly at random in expected time $O(n^2)$ [20]. This in turn builds upon a related algorithm by Dixon and Wilf [7] that solves the same problem but assumes that the exact number of n -vertex unlabeled graphs has already been computed. In [20], the algorithm of Wormald follows a somewhat similar structure but removes that assumption. As mentioned above, the question of computing the exact number of n -vertex unlabeled graphs in expected polynomial time remains open to this day.

It often happens that we wish to sample from a particular graph class (e.g., chordal graphs). For unlabeled trees, there is a uniform sampling algorithm that runs in polynomial time [19]. One can also count the exact number of unlabeled trees on n vertices in polynomial time [17, A000055]. On the topic of counting, an algorithm for counting unlabeled k -trees is presented in [9], but the running time is not stated. An expected-polynomial-time algorithm for uniform sampling of 2-connected unlabeled planar graphs was presented by Bodirsky et al. in 2005 [4], followed by the same result for connected unlabeled cubic planar graphs in 2008 [6]. For the class of connected unlabeled bipartite permutation graphs, a uniform sampling algorithm was designed by Saitoh et al. that runs in $O(n)$ time [18].

Although extensive research has been done on the topic of labeled graph sampling [5, 8, 10, 11], to the best of our knowledge, the literature on sampling *unlabeled* graphs from a given graph class is relatively sparse. As is the case for chordal graphs, the corresponding labeled sampling problem tends to be solved first for a given graph class, and then perhaps one can address the problem of efficiently sampling unlabeled graphs from the same graph class.

1.1 Methods

The sampling algorithm of Wormald [20] is based on the fact that unlabeled graphs correspond to orbits of the following group action: the symmetric group S_n acts on the set of labeled graphs by permuting the vertex labels. This correspondence follows from the Frobenius-Burnside lemma. Therefore, to sample a random unlabeled graph, it is enough to sample a random orbit of this group action.

To make this approach work for chordal graphs, we need two ingredients: (1) an algorithm for counting and sampling labeled chordal graphs with a given automorphism π , and (2) a proof that the probability that a random n -vertex labeled chordal graph has a given automorphism $\pi \in S_n$ is at most $1/2^{c \max\{\mu^2, n\}}$, where μ is the number of moved points of π and c is a constant.

For (1), we design an FPT (fixed-parameter tractable) counting algorithm that is parameterized by μ , the number of moved points of π . This algorithm uses $O(2^{7\mu}n^9)$ arithmetic operations. Using the standard sampling-to-counting reduction of [15], we also obtain a corresponding sampling algorithm with the same running time. Our main algorithm (for sampling *unlabeled* chordal graphs) calls each of these FPT algorithms as a black box with potentially large values of the parameter μ . Nevertheless, using the bound from (2), we are able to show that the probability of using a large value of μ is exponentially small, so the expected running time is not significantly affected.

To design the counting algorithm for (1), we rewrite each of the recurrences from the algorithm of Hébert-Johnson et al., now carrying around information about the automorphism π and its moved points. The original algorithm is a dynamic-programming algorithm in which there is a constant number of types of vertices (X , L , etc.) in each graph that we wish to count. In our updated version, we now keep track of the type of each vertex that is moved by π .

To prove the bound for (2), we distinguish between the cases when μ is small and μ is large (μ is either less than or greater than $\frac{n}{d \log n}$, where d is a constant). When μ is small, we use the fact that almost every chordal graph is a split graph [1], and we strengthen this by showing that in fact, almost every chordal graph is a balanced split graph. For the case of balanced split graphs, the argument is easy and is similar to the proof of the bound for general graphs [16]. When μ is large, the argument is more complicated. We observe that the vast majority of n -vertex labeled chordal graphs have maximum clique size close to $n/2$. Along the way, we also use the fact that for every chordal graph G , there exists a PEO (perfect elimination ordering) of G such that some maximum clique appears at the tail end of that PEO.

2 Preliminaries

Let \mathbb{N} be the set of natural numbers, not including 0. For $n \in \mathbb{N}$, we use the notation $[n] := \{1, 2, \dots, n\}$. For a graph G and vertex subsets $S, T \subseteq V(G)$, we say S *sees all of* T if $T \subseteq N(S)$.

► **Definition 2.** Let $A = \{a_1, \dots, a_r\}$ and $B = \{b_1, \dots, b_r\}$ be finite subsets of \mathbb{N} such that $|A| = |B|$, where the elements a_i and b_i are listed in increasing order. We define $\phi(A, B): A \rightarrow B$ as the bijection that maps a_i to b_i for all $i \in [r]$.

2.1 Permutations and labeled graphs

For $n \in \mathbb{N}$, let S_n denote the group of all permutations of $[n]$. For a permutation $\pi \in S_n$, we define $M_\pi := \{i \in [n] : \pi(i) \neq i\}$ to be the set of points moved by π . For $n \in \mathbb{N}$, $[n]_0 := \{0, 2, 3, \dots, n\}$ denotes the set of all possible values of $|M_\pi|$ for $\pi \in S_n$.

Suppose $\pi \in S_n$, $C \subseteq [n]$. We write $\pi(C) := \{\pi(i) : i \in C\}$ to denote the image of C under π . We say C is *invariant* under π if $\pi(i) \in C$ for all $i \in C$. For a set C that is invariant under π , we write $\pi|_C$ to denote the permutation π restricted to the domain C .

A *labeled graph* is a pair $G = (V, E)$, where the vertex set V is a finite subset of \mathbb{N} and the edge set E is a set of two-element subsets of V . For a permutation $\pi \in S_n$ and a labeled graph G such that $V(G) \subseteq [n]$ is invariant under π , we say $\pi|_{V(G)}$ is an *automorphism* of G if for all $u, v \in V(G)$, u and v are adjacent if and only if $\pi(u)$ and $\pi(v)$ are adjacent.

2.2 Chordal graphs and related notions

A vertex v in a graph G is *simplicial* if its neighborhood $N(v)$ is a clique. A *perfect elimination ordering* (PEO) of a graph G is an ordering v_1, \dots, v_n of the vertices of G such that for all $i \in [n]$, v_i is simplicial in the subgraph induced by the vertices v_i, \dots, v_n . A graph is chordal if and only if it has a perfect elimination ordering [3]. The following two lemmas are well-known facts about chordal graphs. The proof of the first can be found in [3].

► **Lemma 3.** *Every chordal graph G contains a simplicial vertex. If G is not a complete graph, then G contains two non-adjacent simplicial vertices.*

► **Lemma 4.** *A chordal graph G on n vertices has at most n maximal cliques.*

Proof. Let C be a maximal clique in G , and let v_i be the leftmost vertex of C in a given perfect elimination ordering v_1, \dots, v_n of G . We claim that C is equal to the closed neighborhood to the right of v_i , i.e., $C = N[v_i] \cap \{v_i, \dots, v_n\}$. It is clear that C is contained in $N[v_i] \cap \{v_i, \dots, v_n\}$ since v_i has no other neighbors to its right, so we indeed have $C = N[v_i] \cap \{v_i, \dots, v_n\}$ by maximality of C . Therefore, there are at most n maximal cliques. ◀

► **Definition 5.** *Let G_1, G_2 be two graphs, and suppose $C := V(G_1) \cap V(G_2)$ is a clique in both G_1 and G_2 . When we say we **glue** G_1 and G_2 together at C to obtain G , this means G is the union of G_1 and G_2 : the vertex set is $V(G) = V(G_1) \cup V(G_2)$, and the edge set is $E(G) = E(G_1) \cup E(G_2)$.*

As is shown in [14], if G_1 and G_2 are both chordal, then the resulting graph G is chordal.

2.3 Evaporation sequences

Our algorithm for counting the number of labeled chordal graphs with a given automorphism will use the notion of evaporation sequences from [14].

Suppose we are given a chordal graph G and a clique $X \subseteq V(G)$. The *evaporation sequence* of G with *exception set* X is defined as follows: If $X = V(G)$, then the evaporation sequence of G is the empty sequence. If $X \subsetneq V(G)$, then let \tilde{L}_1 be the set of all simplicial vertices in G , and let $L_1 = \tilde{L}_1 \setminus X$. Suppose L_2, \dots, L_t is the evaporation sequence of $G \setminus L_1$ (with exception set X). Then L_1, L_2, \dots, L_t is the evaporation sequence of G . As is shown in [14], the fact that X is a clique implies that all vertices outside of X eventually evaporate, so this is well-defined.

If the evaporation sequence L_1, L_2, \dots, L_t of G has length t , then we say G *evaporates* at time t with *exception set* X , and t is called the *evaporation time*. We define $L_G(X) := L_t$ to be the last set in the evaporation sequence of G , and we let $L_G(X) = \emptyset$ if the sequence is empty. Similarly, we define the evaporation time of a vertex subset. Suppose G has evaporation sequence L_1, L_2, \dots, L_t with exception set X , and suppose $S \subseteq V(G) \setminus X$ is a nonempty vertex subset. Let t_S be the largest index i such that $L_i \cap S \neq \emptyset$. We say S *evaporates* at time t_S in G with exception set X .

3 Sampling unlabeled chordal graphs

In [20], Wormald presented an algorithm that generates an n -vertex unlabeled graph uniformly at random in expected time $O(n^2)$. In this paper, we achieve a similar result for chordal graphs:

► **Theorem 1.** *There is a randomized algorithm that given $n \in \mathbb{N}$, generates a graph uniformly at random from the set of all unlabeled chordal graphs on n vertices. This algorithm uses $O(n^7)$ arithmetic operations in expectation.*

The sampling algorithm of Wormald makes use of the fact that unlabeled graphs correspond to orbits of a particular group action. Since our algorithm will follow a similar outline, we begin by discussing some of the ideas behind the algorithm of Wormald.

Suppose we have been given $n \in \mathbb{N}$ as input, and let Ω be the set of all labeled graphs with vertex set $[n]$. The symmetric group S_n acts on Ω in the following way: For each $\pi \in S_n$ and $G \in \Omega$, $\pi \cdot G$ is the graph that results from permuting the vertex labels of G according to π . The orbits of Ω under the action of S_n are the isomorphism classes of labeled graphs, each of which corresponds to an unlabeled graph. Let

$$\Gamma = \{(\pi, G) \in S_n \times \Omega : \pi \text{ is an automorphism of } G\}.$$

Suppose we fix an n -vertex unlabeled graph H , and let the corresponding isomorphism class of labeled graphs be \mathcal{H} . As is shown in [20], the number of pairs $(\pi, G) \in \Gamma$ such that $G \in \mathcal{H}$ is equal to $|S_n| = n!$. This follows from the Frobenius-Burnside lemma. Therefore, if we sample a random pair $(\pi, G) \in \Gamma$ uniformly at random, and then we forget the labels on the graph G , this amounts to sampling an n -vertex unlabeled graph uniformly at random.

In the case of chordal graphs, the same statements hold true. Let Ω_{chord} be the set of all labeled chordal graphs with vertex set $[n]$. The symmetric group S_n acts on Ω_{chord} in the same way as above, by permuting the vertex labels. The set of orbits of this group action corresponds to the set of unlabeled chordal graphs. Let

$$\Gamma_{\text{chord}} = \{(\pi, G) \in S_n \times \Omega_{\text{chord}} : \pi \text{ is an automorphism of } G\}.$$

Let H_c be an unlabeled chordal graph, and let \mathcal{H}_c be the corresponding isomorphism class of labeled graphs. Applying the Frobenius-Burnside lemma to the orbit corresponding to \mathcal{H}_c shows that the number of pairs $(\pi, G) \in \Gamma_{\text{chord}}$ such that $G \in \mathcal{H}_c$ is equal to $n!$.

In [20], Wormald describes an algorithm for sampling a random pair $(\pi, G) \in \Gamma$ in order to sample a random unlabeled graph. The same outline can be used to sample a random pair $(\pi, G) \in \Gamma_{\text{chord}}$. However, there are two key points where some difficulty arises. First of all, in one of the steps of the algorithm that samples from Γ , it is necessary to count the number of n -vertex labeled graphs with a given automorphism (and sample from the set of such graphs). This is easy to do for general graphs but becomes more complicated for chordal graphs (see Section 4). Second, the algorithm of Wormald uses the fact that the number of labeled graphs with a given automorphism π is at most $2^{\binom{n}{2} - \mu n/2 + \mu(\mu+2)/4}$, where $\mu = |M_\pi|$ is the number of moved points of π . To transform this into an algorithm for sampling unlabeled chordal graphs, it is necessary to prove similar bounds on the number of labeled chordal graphs with a given automorphism. These will be the bounds B_μ in our algorithm.

3.1 Algorithm for sampling unlabeled chordal graphs

Let $\text{COUNT_CHORDAL_LAB}(n)$ stand for the counting algorithm in [14] that computes the number of n -vertex labeled chordal graphs. In the full version of the paper, we prove the following two theorems.

► **Theorem 6.** *There is a deterministic algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, computes the number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu} n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

► **Theorem 7.** *There is a randomized algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, generates a graph uniformly at random from the set of all labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu}n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

See Section 4 for the details of the counting algorithm of Theorem 6 along with some intuition behind the proof of correctness. In our algorithm for sampling unlabeled chordal graphs, $\text{COUNT_CHORDAL_LAB}(n, \pi)$ stands for the algorithm of Theorem 6 and $\text{SAMPLE_CHORDAL_LAB}(n, \pi)$ stands for the algorithm of Theorem 7.

Recall that $[n]_0 = \{0, 2, 3, \dots, n\}$. For $\mu \in [n]_0$, let $R_\mu := \{\pi \in S_n : |M_\pi| = \mu\}$ be the set of permutations with exactly μ moved points. For $\pi \in S_n$, let $\text{Fix}(\pi)$ denote the set of n -vertex labeled chordal graphs G such that π is an automorphism of G . To follow the same approach as the algorithm of Wormald, for each $\mu \in [n]_0$, we need an upper bound B_μ that satisfies $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$. Let B_0 be the number of n -vertex labeled chordal graphs, which is exactly equal to $|R_0| |\text{Fix}(\text{id})|$. For $2 \leq \mu \leq \frac{n}{200 \log n}$, let

$$B_\mu = \left(B_0 (9/10)^n + 2^n 2^{2n^2/9} + n \cdot 2^{n^2/4+n/2} \right) n^\mu \mu!, \quad (1)$$

and for $\frac{n}{200 \log n} < \mu \leq n$, let

$$B_\mu = n^{2n+1} 2^{n^2/4-f(\mu)} n^\mu \mu!, \quad (2)$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$. In Section 3.2, we will prove that we indeed have $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$, when n is sufficiently large. Let $B = \sum_{\mu \in [n]_0} B_\mu$.

Our algorithm for sampling a random n -vertex unlabeled chordal graph is given in Algorithm 1. The general idea is as follows. For $\mu \in [n]_0$, let $\Gamma_\mu = \{(\pi, G) \in \Gamma_{\text{chord}} : |M_\pi| = \mu\}$. We choose μ such that the probability of each value of μ is B_μ/B , which is approximately equal to $\Gamma_\mu/\Gamma_{\text{chord}}$. (We do not know how to efficiently compute the exact value of $\Gamma_\mu/\Gamma_{\text{chord}}$ since we do not know the exact number of unlabeled chordal graphs.) Since B_μ/B is not exactly equal to $\Gamma_\mu/\Gamma_{\text{chord}}$, we adjust for this by restarting with a certain probability in Step 11. We then proceed to select a random pair $(\pi, G) \in \Gamma_\mu$, and we output the graph G without labels.

■ **Algorithm 1** Unlabeled chordal graph sampler.

```

1: procedure SAMPLE_CHORDAL_UNLABELED( $n$ )
2:   ▷ Setup
3:    $B_0 \leftarrow \text{COUNT\_CHORDAL\_LAB}(n)$ 
4:   Let  $B_\mu$  be given by Equation (1) for  $2 \leq \mu \leq \frac{n}{200 \log n}$ 
5:   Let  $B_\mu$  be given by Equation (2) for  $\frac{n}{200 \log n} < \mu \leq n$ 
6:    $B \leftarrow \sum_{\mu \in [n]_0} B_\mu$ 
7:
8:   ▷ Main algorithm
9:   Choose  $\mu \in [n]_0$  at random such that  $\mu = i$  with probability  $B_i/B$  for each  $i \in [n]_0$ 
10:  Choose  $\pi \in R_\mu$  uniformly at random
11:  Restart (go back to Step 9) with probability
       $1 - |R_\mu| \cdot \text{COUNT\_CHORDAL\_LAB}(n, \pi) / B_\mu$ 
12:   $G \leftarrow \text{SAMPLE\_CHORDAL\_LAB}(n, \pi)$ 
13:  Forget the labels on the vertices of  $G$ 
14:  return  $G$ 
15: end procedure

```

Step 10 can easily be implemented in $O(n)$ time in the following way. If $\mu = 0$, let $\pi = \text{id}$. For $\mu \geq 2$, we can repeatedly choose a random permutation of $[\mu]$ until we obtain a derangement. The expected number of trials for this is a constant (see [20]).

In Step 11, to compute $|R_\mu|$, we observe that $|R_0| = 1$ and $|R_\mu| = !\mu \binom{n}{\mu}$ for $\mu \geq 2$. Here $!\mu$ is the number of derangements of μ . We compute $!\mu$ using the formula $!m = m! \sum_{i=0}^m (-1)^i / i!$ for $m \in \mathbb{N}$, which can be derived using the inclusion-exclusion principle.

3.2 Correctness of Algorithm 1

The correctness of `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π) is proved in the full version of the paper.

We need to show that the output graph is chosen uniformly at random. One “iteration” refers to one run of Steps 9 to 11 or 9 to 14. In a given iteration, we say the pair (π, G) was “chosen” if π was chosen from R_μ and G was chosen by `SAMPLE_CHORDAL_LAB`(n, π). We claim that for all $(\pi, G) \in \Gamma_{\text{chord}}$, in any given iteration, the probability that (π, G) is chosen is $1/B$. Indeed, this probability is equal to

$$\frac{B_\mu}{B} \frac{1}{|R_\mu|} \frac{|R_\mu| |\text{Fix}(\pi)|}{B_\mu} \frac{1}{|\text{Fix}(\pi)|} = \frac{1}{B},$$

where $\mu = |M_\pi|$, since the probability of choosing G in Step 12 is $1/|\text{Fix}(\pi)|$. Therefore, we output all n -vertex unlabeled chordal graphs with equal probability.

Next, we need to show that $B_\mu \geq |R_\mu| |\text{Fix}(\pi)|$ for all $\pi \in R_\mu$ to verify that the probability $|R_\mu| |\text{Fix}(\pi)| / B_\mu$ in Step 11 is at most 1. When $\mu = 0$ this is an equality, so suppose $\mu \geq 2$. Clearly $|R_\mu| \leq n^\mu \mu!$, so we just need to prove that the number of n -vertex labeled chordal graphs with automorphism π is at most $B_\mu / (n^\mu \mu!)$ for all $\pi \in S_n$ with μ moved points.

In the case when B_μ is defined according to Equation (2), this follows from Theorem 8. The proof this theorem can be found in the full version of the paper. (The bound in Theorem 8 is in fact true for all values of μ – the reason why we define B_μ differently for smaller values of μ will become clear when we discuss the running time in Section 3.3.)

► **Theorem 8.** *Let $n \in \mathbb{N}$, $\pi \in S_n$, and let $\mu = |M_\pi|$. The number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism is at most*

$$n^{2n+1} 2^{n^2/4 - f(\mu)},$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$.

For the other case, suppose $2 \leq \mu \leq \frac{n}{200 \log n}$, and suppose $\pi \in S_n$ is a permutation with μ moved points. We need to show that the number of n -vertex labeled chordal graphs with automorphism π is at most $B_\mu / (n^\mu \mu!)$. We begin by reducing to the case of split graphs. A *split* graph is a graph whose vertex set can be partitioned into a clique and an independent set, with arbitrary edges between the two parts. It is easy to see that every split graph is chordal. Furthermore, the following result by Bender et al. [1] shows that a random n -vertex labeled chordal graph is a split graph with probability $1 - o(1)$.

► **Proposition 9** (Bender et al. [1]). *If $\alpha > \sqrt{3}/2$, n is sufficiently large, and G is a random n -vertex labeled chordal graph, then*

$$\Pr(G \text{ is a split graph}) > 1 - \alpha^n.$$

Applying this proposition with $\alpha = \frac{9}{10}$ tells us that the number of n -vertex labeled chordal graphs that are *not split* is at most $B_0 \left(\frac{9}{10}\right)^n$. To bound the number of n -vertex labeled split graphs with automorphism π , we consider two cases: balanced split graphs and unbalanced split graphs. We say a partition of the vertex set of a split graph G is a *split partition* if it partitions G into a clique and an independent set; i.e., a split partition is a partition that demonstrates that G is split. We denote a split partition that consists of the clique C and the independent set I by the ordered pair (C, I) . We say an n -vertex split graph is *balanced* if $|C| \geq \frac{n}{3}$ and $|I| \geq \frac{n}{3}$ for every split partition (C, I) of G . It is easy to bound the number of unbalanced split graph as follows.

► **Lemma 10.** *The number of labeled split graphs on n vertices that are not balanced is at most $2^n 2^{2n^2/9}$.*

Proof. Suppose (C, I) is a partition of $[n]$ into two parts such that $|C| < \frac{n}{3}$ or $|I| < \frac{n}{3}$. The number of labeled split graphs with this particular split partition is at most $2^{|I||C|} \leq 2^{\frac{n}{3} \cdot \frac{2n}{3}} = 2^{2n^2/9}$. Therefore, the number of unbalanced labeled split graphs on n vertices is at most $2^n 2^{2n^2/9}$ since there are at most 2^n possible partitions. ◀

The following lemma will be useful for bounding the number of balanced split graphs.

► **Lemma 11.** *Let $\pi \in S_n$, and let G be an n -vertex labeled split graph. If π is an automorphism of G , then there exists a split partition (C, I) of G such that C and I are invariant under π .*

Proof. Let \hat{C} be the set of vertices that belong to the clique in every split partition of G , let \hat{I} be the set of vertices that belong to the independent set in every split partition of G , and let $\hat{Q} = V(G) \setminus (\hat{C} \cup \hat{I})$. By Observation 7.3 in [14], every vertex in \hat{Q} is adjacent to every vertex in \hat{C} and is non-adjacent to every vertex in \hat{I} . By Lemma 7.4 in [14], \hat{Q} is either a clique or an independent set. Suppose π is an automorphism of G . If \hat{Q} is a clique, then the split partition $(\hat{C} \cup \hat{Q}, \hat{I})$ has the desired property. If \hat{Q} is an independent set, then the split partition $(\hat{C}, \hat{I} \cup \hat{Q})$ has the desired property. ◀

► **Lemma 12.** *Let $\pi \in S_n$, and suppose $|M_\pi| \geq 2$. The number of balanced labeled split graphs G on n vertices such that π is an automorphism of G is at most*

$$n \cdot 2^{n^2/4+n/2}.$$

Proof. Suppose (C, I) is a partition of $[n]$ into two parts. Let $i = |I|$ and $c = |C|$. The number of labeled split graphs with this particular split partition is 2^{ic} . Therefore, the number of labeled split graphs with automorphism π for which (C, I) has the property from Lemma 11 is at most 2^{ic} . Let $Z_{(C, I)}$ be the number of such graphs. Whenever two vertices $u, v \in I$ (resp. C) belong to the same cycle in the cycle decomposition of π , u and v must have the same relationship as each other to each of the vertices in C (resp. I). Therefore, we in fact have an upper bound of $\max\{2^{(i-1)c}, 2^{i(c-1)}\}$ on $Z_{(C, I)}$, since π has at least two moved points. If $i \geq \frac{n}{3}$ and $c \geq \frac{n}{3}$, then we have $\max\{2^{(i-1)c}, 2^{i(c-1)}\} \leq 2^{n^2/4-n/2}$.

By Lemma 11, every balanced labeled split graph on n vertices with automorphism π has a split partition (C, I) such that C and I are invariant under π . Furthermore, this split partition is balanced (i.e., both parts have size at least $\frac{n}{3}$). Therefore, the number of balanced labeled split graphs on n vertices with automorphism π is at most

$$\sum_{\lceil \frac{n}{3} \rceil \leq i \leq \lfloor \frac{2n}{3} \rfloor} 2^n \cdot 2^{n^2/4-n/2} \leq n \cdot 2^{n^2/4+n/2}$$

since the number of partitions (C, I) with $|I| = i$ is certainly at most 2^n . ◀

Putting together Proposition 9 and Lemmas 10 and 12, we can see that

$$\left(B_0(9/10)^n + 2^n 2^{2n^2/9} + n \cdot 2^{n^2/4+n/2} \right) n^\mu \mu!$$

is an upper bound on $|R_\mu| |\text{Fix}(\pi)|$ when n is sufficiently large.

Let N_0 be the cutoff such that this works for $n \geq N_0$. To be precise, when implementing this algorithm, we would solve the problem by brute force if $n < N_0$ (by generating all possible n -vertex chordal graphs and then selecting one at random), and we would run Algorithm 1 as written if $n \geq N_0$.

3.3 Running time of Algorithm 1

The running time of Steps 1-6 is $O(n^7)$ arithmetic operations since that is the running time of `COUNT_CHORDAL_LAB`(n). In this section, we will show that the rest of the algorithm uses only $O(n^7)$ arithmetic operations in expectation.

If we choose $\mu = 0$ in Step 9 of Algorithm 1, then the algorithm is guaranteed to terminate in that iteration. Thus the expected number of iterations is at most B/B_0 . Let T be the expected running time of Algorithm 1 after completing Step 6 (this is where we choose μ at random and the loop begins). In Lemma 13, we show that $\mathbf{E}[T]$ is at most the product of B/B_0 and the expected running time of one iteration.

Let N be the number of iterations of Algorithm 1, and let T_j be the time spent in iteration j for each $j \in [N]$. We have $T = \sum_{j=1}^N T_j$.

► **Lemma 13.** *We have $\mathbf{E}[T] \leq \frac{B}{B_0} \mathbf{E}[T_1]$.*

Proof. Clearly $\mathbf{E}[T]$ is finite since the expected number of iterations is finite and the procedures `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π) have worst-case running time bounds. Therefore, we can solve for $\mathbf{E}[T]$ in the following way.

The steps that we run in one iteration (Steps 9-14) do not depend on j – they are always the same, regardless of how many iterations have happened so far. Thus we have $\mathbf{E}\left[\sum_{j=2}^N T_j \mid N > 1\right] = \mathbf{E}[T]$, which implies $\mathbf{E}[T \mid N > 1] = \mathbf{E}[T_1 \mid N > 1] + \mathbf{E}[T]$. Therefore, we have

$$\begin{aligned} \mathbf{E}[T] &= \mathbf{E}[T \mid N = 1] \Pr(N = 1) + \mathbf{E}[T \mid N > 1] \Pr(N > 1) \\ &= \mathbf{E}[T_1 \mid N = 1] \Pr(N = 1) + (\mathbf{E}[T_1 \mid N > 1] + \mathbf{E}[T]) \cdot \Pr(N > 1) \end{aligned}$$

$$\begin{aligned} \implies \mathbf{E}[T](1 - \Pr(N > 1)) &= \mathbf{E}[T_1 \mid N = 1] \Pr(N = 1) + \mathbf{E}[T_1 \mid N > 1] \Pr(N > 1) \\ &= \mathbf{E}[T_1] \end{aligned}$$

$$\implies \mathbf{E}[T] = \frac{\mathbf{E}[T_1]}{\Pr(N = 1)} \leq \frac{B}{B_0} \mathbf{E}[T_1]. \quad \blacktriangleleft$$

For $\mu \in [n]_0$, let $T(n, \mu)$ be an upper bound on the time it takes to run one iteration, assuming we have chosen this particular value of μ in Step 9. By the running time of `COUNT_CHORDAL_LAB`(n, π) and `SAMPLE_CHORDAL_LAB`(n, π), we can assume $T(n, \mu) = O(2^{7\mu} n^9)$. We have

$$\mathbf{E}[T_1] = \sum_{\mu \in [n]_0} \frac{B_\mu}{B} T(n, \mu).$$

46:10 Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

To prove a bound on $\mathbf{E}[T_1]$, we will start by proving a bound on B_μ/B for all $\mu \in [n]_0$. Since $B_0 \leq B$, it is sufficient to prove a bound on B_μ/B_0 for all $\mu \in [n]_0$. The following lemma gives us a lower bound on B_0 .

► **Lemma 14.** *For $n \geq 2$, the number of n -vertex labeled chordal graphs is at least*

$$\frac{2^n 2^{n^2/4}}{n^2}.$$

Proof. It is enough to just consider n -vertex labeled split graphs with a split partition (C, I) such that $|C| = \lfloor \frac{n}{2} \rfloor$. Since $\binom{n}{\lfloor \frac{n}{2} \rfloor} \geq 2^n/n$ for $n \geq 2$, the number of such graphs is at least

$$\binom{n}{\lfloor \frac{n}{2} \rfloor} \frac{2^{n^2/4}}{n} \geq \frac{2^n 2^{n^2/4}}{n^2}.$$

We divide by n in the first expression since each split graph can have up to n distinct split partitions in which C is of a given size [2]. ◀

► **Lemma 15.** *Suppose $n \geq 13$. If $2 \leq \mu \leq \frac{n}{200 \log n}$, then*

$$\frac{B_\mu}{B_0} \leq \frac{3}{n^{16\mu}}.$$

Proof. Let $B_\mu^{(1)}$, $B_\mu^{(2)}$, and $B_\mu^{(3)}$ be the three terms that are added together in Equation (1), in order, so that $B_\mu = (B_\mu^{(1)} + B_\mu^{(2)} + B_\mu^{(3)}) n^\mu \mu!$. We have

$$\frac{B_\mu^{(1)} n^\mu \mu!}{B_0} = \left(\frac{9}{10}\right)^n n^\mu \mu!,$$

and we claim that this is at most $1/n^{16\mu}$. Since $\mu! \leq n^\mu$, it is sufficient to show $n^{18\mu} \leq (10/9)^n$, which is true when $\mu \leq \frac{n}{200 \log n}$.

For the next term, by Lemma 14 we have

$$\frac{B_\mu^{(2)} n^\mu \mu!}{B_0} \leq n^2 2^{-n^2/36} n^\mu \mu!.$$

To see that this is at most $1/n^{16\mu}$, it is sufficient to show $n^{19/200} \leq 2^{n/36}$ since $\mu \leq \frac{n}{200}$. This is indeed true for $n \geq 13$.

For the third term, by Lemma 14 we have

$$\frac{B_\mu^{(3)} n^\mu \mu!}{B_0} \leq n^3 2^{-n/2} n^\mu \mu!.$$

To see that this is at most $1/n^{16\mu}$, it is sufficient to show $n^{20\mu} \leq 2^{n/2}$, which is true when $\mu \leq \frac{n}{40 \log n}$. Adding together these three terms, we obtain $B_\mu/B_0 \leq 3/n^{16\mu}$. ◀

► **Lemma 16.** *For sufficiently large n , if $\frac{n}{200 \log n} < \mu \leq n$, then*

$$\frac{B_\mu}{B_0} \leq \frac{1}{n^{16\mu}}.$$

Proof. Lemma 14 implies $B_0 \geq 2^{n^2/4}$ for $n \geq 4$, so we have

$$\frac{B_\mu}{B_0} \leq n^{2n+1} 2^{-f(\mu)} n^\mu \mu!,$$

where $f(\mu) = \frac{\mu^2}{900} - \frac{\mu}{10}$. We claim that this expression is at most $1/n^{16\mu}$. Since $\mu! \leq n^\mu$, it is sufficient to show $n^{2n+1} n^{18\mu} \leq 2^{f(\mu)}$. When n is sufficiently large,¹ we have $\log^3 n \leq \frac{1}{200^2 \cdot 1800} \frac{n^2}{2n+1}$. Since $\mu \geq \frac{n}{200 \log n}$, this implies

$$n^{2n+1} \leq 2^{\mu^2/1800}. \quad (3)$$

When n is sufficiently large,² we also have $n \geq 18 \cdot 900 \cdot 200 \log^2 n + 90 \cdot 200 \log n$. Since $\mu \geq \frac{n}{200 \log n}$, this implies

$$n^{18\mu} \leq 2^{\mu^2/1800 - \mu/10}. \quad (4)$$

Multiplying Equations (3) and (4) gives us the desired bound of $n^{2n+1} n^{18\mu} \leq 2^{f(\mu)}$. ◀

By Lemmas 15 and 16, the above summation for $\mathbf{E}[T_1]$ is at most $T(n, 0) + O(1)$ since $T(n, \mu)$ is certainly at most $O(n^{16\mu})$. For an iteration in which we have chosen $\mu = 0$, when counting and sampling n -vertex labeled chordal graphs with automorphism $\pi = \text{id}$, we can simply run the counting and sampling algorithms of [14], rather than passing in $\pi = \text{id}$ as an input. Thus the expected running time $\mathbf{E}[T_1]$ of one iteration is at most $O(n^7)$ arithmetic operations. Lemmas 15 and 16 also immediately give us a bound on B/B_0 , since we have

$$\frac{B}{B_0} = \frac{B_0}{B_0} + \frac{B_2}{B_0} + \dots + \frac{B_n}{B_0} = O(1).$$

Therefore, by Lemma 13, the overall running time is at most $O(n^7)$ arithmetic operations in expectation.

4 Counting labeled chordal graphs with a given automorphism

In this section, we describe the algorithm for `COUNT_CHORDAL_LAB`(n, π), which counts the number of labeled chordal graphs with a given automorphism. In the full version of the paper, we prove correctness, analyze the running time, and derive the corresponding sampling algorithm (Theorem 7).

► **Theorem 6.** *There is a deterministic algorithm that given $n \in \mathbb{N}$ and $\pi \in S_n$, computes the number of labeled chordal graphs with vertex set $[n]$ for which π is an automorphism. This algorithm uses $O(2^{7\mu} n^9)$ arithmetic operations, where $\mu = |M_\pi|$.*

There is a known dynamic-programming algorithm for computing the number n -vertex labeled chordal graphs that uses $O(n^7)$ arithmetic operations, if we do not require the graphs to have a particular automorphism [14]. Our algorithm is closely based on that one, but we add more arguments and more details to each of the recurrences to keep track of the behavior of the automorphism. As was done in [14], we evaluate the recurrences top-down using memoization.

¹ This holds for $n \geq 2.6 \cdot 10^5$.

² This holds for $n \geq 3.3 \cdot 10^9$.

4.1 Reducing from counting chordal graphs to counting connected chordal graphs

For $k \in \mathbb{N}$, let $a(k)$ denote the number of labeled chordal graphs with vertex set $[k]$, and let $c(k)$ denote the number of *connected* labeled chordal graphs with vertex set $[k]$. The algorithm of [14] begins by reducing from counting chordal graphs to counting connected chordal graphs via the following recurrence, which appears as Lemma 3.13 in [14]. (We omit the “ ω -colorable” requirement since we will not need that here.)

► **Lemma 17** ([14]). *The number of labeled chordal graphs with vertex set $[k]$ is given by*

$$a(k) = \sum_{k'=1}^k \binom{k-1}{k'-1} c(k') a(k-k')$$

for all $k \in \mathbb{N}$.

Here k' stands for the number of vertices in the connected component that contains the label 1. The remaining connected components have a total of $k - k'$ vertices. Since this recurrence is relatively simple, most of the difficulty in the algorithm of [14] lies in the recurrences for counting *connected* chordal graphs. However, when counting graphs with a given automorphism, the step of reducing to connected graphs is already quite a bit more involved.

Suppose we are given $n \in \mathbb{N}$ and $\pi \in S_n$ as input. From now on, whenever we refer to n or π , we mean these particular values from the input.

We will first define $a(k, p, M)$ and $c(k, p, M)$, which are variations of $a(k)$ and $c(k)$ that only count graphs for which $\pi^p|_{V(G)}$ is an automorphism (see Definition 18). Here π^p stands for π to the power p , i.e., the permutation that arises from applying π a total of p times. The reason for raising π to a power will be apparent in the recurrence for $a(k, p, M)$. In the initial, highest-level recursive call, we will have $p = 1$ and thus $\pi^p = \pi$.

In [14], when counting the number of possibilities for a subgraph of size k' (for example, a connected component), the authors essentially relabel that subgraph to have vertex set $[k']$, so that one can count the number of possibilities using, for example, $c(k')$. In our algorithm, we want to relabel the vertices of each subgraph in a similar way. However, this time, we do not relabel the vertices that are moved by π . This ensures that the automorphism in each later recursive call will still be π , or a permutation closely related to π .

As a consequence, the vertex sets of the subgraphs that we wish to count become slightly more complicated. For example, suppose we wish to count the number of possibilities for a 5-vertex subgraph that originally contains the moved vertices $2, 8, 9 \in M_\pi$. Since we do not relabel the moved vertices, the resulting vertex set after relabeling is $\{1, 2, 3, 8, 9\}$ rather than $\{1, 2, 3, 4, 5\}$. More formally, suppose we have been given $k \in \mathbb{N}$ and $M \subseteq M_\pi$, where $|M| \leq k$. Let V be the set of the first $k - |M|$ natural numbers in $\mathbb{N} \setminus M_\pi$. We define $V_{k,M} := V \cup M$. For example, if $k = 5$ and $M = M_\pi = \{2, 8, 9\}$, then $V_{k,M} = \{1, 2, 3, 8, 9\}$. Intuitively, $V_{k,m}$ is the label set of size k whose intersection with M_π is M and that otherwise contains labels that are as small as possible.

For $\hat{\pi} \in S_n$ and $M \subseteq [n]$, recall that M is *invariant* under $\hat{\pi}$ if $\hat{\pi}(i) \in M$ for all $i \in M$. For $M', M \subseteq [n]$, we write $M' \subseteq_{\hat{\pi}} M$ to indicate that $M' \subseteq M$ and M' is invariant under $\hat{\pi}$. Intuitively, M' is a subset of M that respects the cycles of $\hat{\pi}$ by taking all or nothing of each cycle.

► **Definition 18.** *Suppose $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. Let $a(k, p, M)$ (resp. $c(k, p, M)$) denote the number of labeled chordal graphs (resp. **connected** labeled chordal graphs) with vertex set $V_{k,M}$ for which $\pi^p|_{V(G)}$ is an automorphism.*

Our algorithm returns $a(n, 1, M_\pi)$. This is precisely the number of labeled chordal graphs with vertex set $[n]$ and automorphism π since $V_{n, M_\pi} = [n]$.

Suppose we have been given fixed values of the arguments k, p, M of $a(k, p, M)$. Let s be the smallest label in the vertex set $V_{k, M}$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M' \subseteq_{\pi^p} M$ such that $|M| - k + k' \leq |M'| \leq k'$ and such that the following condition holds: if $s \in M$, then $s \in M'$, and otherwise, $|M'| \leq k' - 1$. This last condition will ensure that s belongs to the connected component of size k' in the recurrence for $a(k, p, M)$.

Suppose $C \subseteq [n]$, $\sigma \in S_n$. For an element $i \in C$, we say the *period* of i with respect to (C, σ) is the smallest positive integer j such that $\sigma^j(i) \in C$. Let \mathcal{Q} be the family of sets $C \subseteq M$ such that all elements of C have the same period $j \geq 2$ with respect to (C, π^p) , and such that $s \in C$. For a set $C \in \mathcal{Q}$, we write p_C to denote the period of the elements of C (with respect to (C, π^p)), and we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$ denote the union of the sets that C is mapped to by powers of σ , where $\sigma = \pi^p$.

To compute $a(k, p, M)$, we use the following recurrence. The dot in front of the curly braces denotes multiplication. Note that we have $|M'| \leq k'$ and $|M \setminus M'| \leq k - k'$ by the definition of $\mathcal{P}_{k'}$.

► **Lemma 19.** *Let $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. We have*

$$a(k, p, M) = \sum_{\substack{1 \leq k' \leq k \\ M' \in \mathcal{P}_{k'}}} c(k', p, M') a(k - k', p, M \setminus M') \cdot \begin{cases} \binom{k - |M|}{k' - |M'|} & \text{if } s \in M \\ \binom{k - 1 - |M|}{k' - 1 - |M'|} & \text{otherwise} \end{cases} \\ + \sum_{C \in \mathcal{Q}} c(|C|, p \cdot p_C, C) a(k - p_C |C|, p, M \setminus C_{\pi^p}).$$

For some intuition, suppose G is a graph counted by $a(k, p, M)$, and let C be the connected component of G that contains s . The first line of the recurrence for $a(k, p, M)$ covers the case when C is invariant under π^p . This case is analogous to Lemma 17. In the first summation, k' stands for $|C|$ and M' stands for the set of vertices in C that can be moved by π^p . If $s \in M$, then in addition to the vertices in M' , we need to choose $k' - |M'|$ additional vertices for C so that $|C| = k'$. For these, we must choose non-moved vertices, so there are $k - |M|$ possible vertices to choose from. If $s \notin M$, then we subtract 1 from each of the numbers in the binomial coefficient since we already know that s is a non-moved vertex in C .

The second line covers the case when C is not invariant under π^p , which means all of C is mapped to some other connected component of G by π^p . In this case, we have p_C components of size $|C|$ that are all isomorphic to C , and the rest of the graph has $k - p_C |C|$ vertices. We do not need a binomial coefficient in this case because all of the vertices in C are moved. There are $c(|C|, p \cdot p_C, C)$ possibilities for the edges of C since $\pi^{p \cdot p_C}$ is an automorphism of C . As an example, suppose $p = 1$. If we apply $\pi^p = \pi$ to the vertices of C a total of p_C times, then the image $\pi^{p_C}(C)$ is equal to C , although these two sets might not match up pointwise. To ensure that the image $\pi^{p_C}(C)$ matches up with the edges of C , we require that π^{p_C} is an automorphism of C . This is why we need the argument p in $a(k, p, M)$ and $c(k, p, M)$.

Note that for a graph G counted by $a(k, p, M)$, we have $M_{\pi^p} \cap V(G) \subseteq M_\pi \cap V(G) \subseteq M$ since $V(G) = V_{k, M}$. This means no vertex in $V(G) \setminus M$ is moved by π^p , so we are free to relabel these vertices in the proof of Lemma 19 without changing the automorphism. In the initial recursive call $a(n, 1, M_\pi)$, we have $p = 1$ and $M = M_\pi$, so $M_{\pi^p} \cap V(G) = M$. Later on in the algorithm, it is possible that some vertices in M are not actually moved by π^p . For example, in the previous paragraph, it could happen that $\pi^{p \cdot p_C}$ is the identity on C (and

then $p \cdot p_C$ becomes the new value of p). However, we always have $M \subseteq M_\pi$ by the definition of $a(k, p, M)$, so if $|M_\pi| = \mu$, then $|M| \leq \mu$. This fact will be useful for the running time analysis.

The proof of Lemma 19, along with the proofs of all of the following recurrences, can be found in the full version of the paper.

4.2 Recurrences for counting connected chordal graphs

To compute $c(k, p, M)$, we will define various counter functions that are analogous to those in [14]. First, we recall the counter functions from [14]. We refer to functions 1-4 in Definition 20 as g -functions, and we refer to the others as f -functions. See Figure 1 in [14] for an illustration of all of these functions.

► **Definition 20** ([14]). *The following functions count various subclasses of chordal graphs. The arguments t, x, ℓ, k, z are nonnegative integers, where $t \leq n$, $z \leq n$, $x + k \leq n$ for g -functions, and $x + \ell + k \leq n$ for f -functions. These also satisfy the domain requirements listed below.*

1. $g(t, x, k, z)$ is the number of labeled connected chordal graphs G with vertex set $[x + k]$ that evaporate in time at most t with exception set $X := [x]$, where X is a clique, with the following property: every connected component of $G \setminus X$ (if any) has at least one neighbor in $X \setminus [z]$. **Domain:** $t \geq 0$, $x \geq 1$, $z < x$.
2. $\tilde{g}(t, x, k, z)$ is the same as $g(t, x, k, z)$, except every connected component of $G \setminus X$ (if any) evaporates at time exactly t in G . Note: A graph with $V(G) = X$ would be counted because in that case, \tilde{g} is the same as g . **Domain:** $t \geq 1$, $x \geq 1$, $z < x$.
3. $\tilde{g}_p(t, x, k, z)$ is the same as $\tilde{g}(t, x, k, z)$, except no connected component of $G \setminus X$ sees all of X . **Domain:** $t \geq 1$, $x \geq 1$, $z < x$.
4. $\tilde{g}_1(t, x, k)$ and $\tilde{g}_{\geq 2}(t, x, k)$ are the same as $\tilde{g}(t, x, k, z)$, except every connected component of $G \setminus X$ sees all of X (hence we no longer require every component of $G \setminus X$ to have a neighbor in $X \setminus [z]$), and furthermore, for \tilde{g}_1 we require that $G \setminus X$ has exactly one connected component, and for $\tilde{g}_{\geq 2}$ we require that $G \setminus X$ has at least two components. **Domain for \tilde{g}_1 :** $t \geq 1$, $x \geq 0$. **Domain for $\tilde{g}_{\geq 2}$:** $t \geq 1$, $x \geq 1$.
5. $f(t, x, \ell, k)$ is the number of labeled connected chordal graphs G with vertex set $[x + \ell + k]$ that evaporate at time exactly t with exception set $X := [x]$, such that $G \setminus X$ is connected, $L_G(X) = \{x + 1, \dots, x + \ell\}$, and $X \cup L_G(X)$ is a clique. **Domain:** $t \geq 1$, $x \geq 0$, $\ell \geq 1$.
6. $\tilde{f}(t, x, \ell, k)$ is the same as $f(t, x, \ell, k)$, except every connected component of $G \setminus (X \cup L_G(X))$ evaporates at time exactly $t - 1$ in G , and there exists at least one such component, i.e., $X \cup L_G(X) \subsetneq V(G)$. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$.
7. $\tilde{f}_p(t, x, \ell, k)$ is the same as $\tilde{f}(t, x, \ell, k)$, except no connected component of $G \setminus (X \cup L_G(X))$ sees all of $X \cup L_G(X)$. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$.
8. $\tilde{f}_p(t, x, \ell, k, z)$ is the same as $\tilde{f}_p(t, x, \ell, k)$, except rather than requiring that $G \setminus X$ is connected, we require that $G \setminus [z]$ is connected. **Domain:** $t \geq 2$, $x \geq 0$, $\ell \geq 1$, $z \leq x$.

The counter functions for our algorithm will be similar to these, except we only want to count graphs with a particular automorphism. Therefore, we will have several more arguments in addition to the usual ones t, x, ℓ, k, z from Definition 20. As above, the argument p will indicate that π^p is the current automorphism. We also introduce the arguments M_X, M_L, M_Z , and M_K , each of which is a subset of M_π . Roughly, these sets specify which vertices – in $X, L := L_G(X), Z := [z]$, and the rest of the graph, respectively – can be moved by the current permutation (but the sets X, L , and Z will be modified slightly).

In Definition 20, the g -functions count graphs with vertex set $[x + k]$, and the f -functions count graphs with vertex set $[x + \ell + k]$. For our algorithm, we will make some adjustments to these vertex sets to ensure that the vertices moved by the current permutation still appear in the graph. This is similar to how we defined $V_{k,M}$ above. To do so, we define several symbols for these vertex sets and vertex subsets (V_{args} , etc.). Each of these depends on the list of arguments of the function in question, which we denote by $args$. For example, when defining g , we have $args = \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}$ (see Definition 21).

First, suppose $args$ comes from one of the g -functions in Definition 21. Let V_X be the set of the first $x - |M_X|$ natural numbers in $\mathbb{N} \setminus M_\pi$, and let V_K be the set of the first $k - |M_K|$ natural numbers in $\mathbb{N} \setminus (V_X \cup M_\pi)$. We define $X_{args} := V_X \cup M_X$ and $V_{args} := X_{args} \cup V_K \cup M_K$. Also, if z is included in $args$, then let V_Z be the set of the first $z - |M_Z|$ natural numbers in $\mathbb{N} \setminus M_\pi$. In this case, we define $Z_{args} := V_Z \cup M_Z$.

For the other case, suppose $args$ comes from one of the f -functions. Let V_X be the set of the first $x - |M_X|$ natural numbers in $\mathbb{N} \setminus M_\pi$, let V_L be the set of the first $\ell - |M_L|$ natural numbers in $\mathbb{N} \setminus (V_X \cup M_\pi)$, and let V_K be the set of the first $k - |M_K|$ natural numbers in $\mathbb{N} \setminus (V_X \cup V_L \cup M_\pi)$. We define $X_{args} := V_X \cup M_X$, $L_{args} = V_L \cup M_L$, and $V_{args} := X_{args} \cup L_{args} \cup V_K \cup M_K$. Also, if z is included in $args$, then let V_Z be the set of the first $z - |M_Z|$ natural numbers in $\mathbb{N} \setminus M_\pi$. In this case, we again define $Z_{args} := V_Z \cup M_Z$.

These vertex subsets still have the same sizes as they did in the original algorithm: the size of V_{args} is $x + k$ or $x + \ell + k$, $|X_{args}| = x$, $|L_{args}| = \ell$, the rest of the graph has size k , and $|Z_{args}| = z$. Just as we had $Z \subseteq X$ in the original algorithm, we now have $Z_{args} \subseteq X_{args}$. This is because we require $M_Z \subseteq M_X$ in Definition 21, and we also require $z - |M_Z| \leq x - |M_X|$, which implies $V_Z \subseteq V_X$.

For g -functions, we have $M_{\pi^p} \cap V_{args} \subseteq M_X \cup M_K$, and for f -functions, we have $M_{\pi^p} \cap V_{args} \subseteq M_X \cup M_L \cup M_K$, by the definition of V_{args} . We are now ready to define our new counter functions.

► **Definition 21.** *The following functions count various subclasses of chordal graphs. The arguments t, x, ℓ, k, z are nonnegative integers with the same domains as in Definition 20. We have $p \in [n]$, and the arguments M_X, M_L, M_K, M_Z are subsets of M_π . All other requirements for their domains are specified below.*

1. $g \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}, \tilde{g} \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix}, \tilde{g}_p \begin{pmatrix} t & x & k & z \\ p & M_X & M_K & M_Z \end{pmatrix},$
 $\tilde{g}_1 \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix},$ and $\tilde{g}_{\geq 2} \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix}$ are the same as $g(t, x, k, z), \tilde{g}(t, x, k, z),$
 $\tilde{g}_p(t, x, k, z), \tilde{g}_1(t, x, k),$ and $\tilde{g}_{\geq 2}(t, x, k),$ respectively, except we only count graphs for which $\pi^p|_{V(G)}$ is an automorphism, and we make the following changes to the vertices of the graph: the vertex set is V_{args} rather than $[x + k]$, the exception set is X_{args} rather than $[x]$, and $[z]$ is replaced by Z_{args} .

Domain: $M_X \subseteq_{\pi^p} M_\pi, M_K \subseteq_{\pi^p} M_\pi \setminus M_X, M_Z \subseteq_{\pi^p} M_X, |M_X| \leq x, |M_K| \leq k, |M_Z| \leq z,$ and $z - |M_Z| \leq x - |M_X|.$

2. $f \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix}, \tilde{f} \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix}, \tilde{f}_p \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix},$ and
 $\tilde{f}_p \begin{pmatrix} t & x & \ell & k & z \\ p & M_X & M_L & M_K & M_Z \end{pmatrix}$ are the same as $f(t, x, \ell, k), \tilde{f}(t, x, \ell, k), \tilde{f}_p(t, x, \ell, k),$ and
 $\tilde{f}_p(t, x, \ell, k, z),$ respectively, except we only count graphs for which $\pi^p|_{V(G)}$ is an automorphism, and we make the following changes to the vertices of the graph: the vertex set is

46:16 Sampling Unlabeled Chordal Graphs in Expected Polynomial Time

V_{args} rather than $[x + \ell + k]$, the exception set is X_{args} rather than $[x]$, the last set to evaporate is L_{args} rather than $\{x + 1, \dots, x + \ell\}$, and $[z]$ is replaced by Z_{args} .

Domain: $M_X \subseteq_{\pi^p} M_\pi$, $M_L \subseteq_{\pi^p} M_\pi \setminus M_X$, $M_K \subseteq_{\pi^p} M_\pi \setminus (M_X \cup M_L)$, $M_Z \subseteq_{\pi^p} M_X$, $|M_X| \leq x$, $|M_L| \leq \ell$, $|M_K| \leq k$, $|M_Z| \leq z$, and $z - |M_Z| \leq x - |M_X|$.

For a graph G counted by one of these functions, $\pi^p|_{V(G)}$ is indeed a permutation of $V(G)$ since M_X and M_K (and M_L if needed) are invariant under π^p .

To compute $c(k, p, M)$, we consider all possibilities for the evaporation time. In the following recurrence, \tilde{g}_1 (with those particular arguments) counts the number of labeled connected chordal graphs with vertex set $V_{args} = V_{k, M}$ and automorphism $\pi^p|_{V(G)}$ that evaporate at time exactly t with empty exception set.

► **Lemma 22.** *Let $k, p \in [n]$ and suppose $M \subseteq_{\pi^p} M_\pi$, where $|M| \leq k$. We have*

$$c(k, p, M) = \sum_{t=1}^k \tilde{g}_1 \begin{pmatrix} t & 0 & k \\ p & \emptyset & M \end{pmatrix}.$$

To compute \tilde{g}_1 , we consider all possibilities for the size ℓ of L_{args} . The set M in the inner sum stands for the set of vertices in L_{args} that can be moved by π^p . Note that we have $|M| \leq \ell$ and $|M_K \setminus M| \leq k - \ell$ by the definition of \mathcal{I}_ℓ . For \tilde{g}_1 and all of the following functions, we recommend reading the analogous recurrences in [14] for further insight into what is happening with the arguments t, x, ℓ, k, z in each recursive call.

► **Lemma 23.** *For \tilde{g}_1 , we have*

$$\tilde{g}_1 \begin{pmatrix} t & x & k \\ p & M_X & M_K \end{pmatrix} = \sum_{\ell=1}^k \sum_{M \in \mathcal{I}_\ell} \binom{k - |M_K|}{\ell - |M|} f \begin{pmatrix} t & x & \ell & k - \ell \\ p & M_X & M & M_K \setminus M \end{pmatrix},$$

where $\mathcal{I}_\ell = \{M \subseteq_{\pi^p} M_K : |M_K| - k + \ell \leq |M| \leq \ell\}$.

To compute f , we consider all possibilities for the set of labels that appear in connected components of $G \setminus (X_{args} \cup L_{args})$ that evaporate at time exactly $t - 1$. These components correspond to the recursive call to \tilde{f} . The set M stands for the set of vertices in components that evaporate at time exactly $t - 1$ that can be moved by π^p . Note that we have $|M| \leq k'$ and $|M_K \setminus M| \leq k - k'$ by the definition of $\mathcal{I}_{k'}$. Since $|M_L| \leq \ell$, we also have $x - |M_X| \leq x + \ell - |M_X \cup M_L|$, which is required by the domain of g .

► **Lemma 24.** *For f , we have*

$$f \begin{pmatrix} t & x & \ell & k \\ p & M_X & M_L & M_K \end{pmatrix} = \sum_{k'=1}^k \sum_{M \in \mathcal{I}_{k'}} \binom{k - |M_K|}{k' - |M|} \tilde{f} \begin{pmatrix} t & x & \ell & k' \\ p & M_X & M_L & M \end{pmatrix} g \begin{pmatrix} t - 2 & x + \ell & k - k' & x \\ p & M_X \cup M_L & M_K \setminus M & M_X \end{pmatrix},$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

To compute g , we consider all possibilities for the set of labels that appear in connected components of $G \setminus X_{args}$ that evaporate at time exactly t . These components correspond to the recursive call to \tilde{g} . The set M stands for the set of vertices in components that evaporate at time exactly t that can be moved by π^p .

► **Lemma 25.** *For g , we have*

$$g \binom{t \quad x \quad k \quad z}{p \quad M_X \quad M_K \quad M_Z} = \sum_{k'=0}^k \sum_{M \in \mathcal{I}_{k'}} \binom{k - |M_K|}{k' - |M|} \tilde{g} \binom{t \quad x \quad k' \quad z}{p \quad M_X \quad M \quad M_Z} g \binom{t-1 \quad x \quad k-k' \quad z}{p \quad M_X \quad M_K \setminus M \quad M_Z},$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

For the next recurrence, we will need a few more definitions. Suppose we have been given fixed values of the arguments of \tilde{g} . Let s be the smallest label in $V_{args} \setminus X_{args}$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M \subseteq_{\pi^p} M_K$ such that $|M_K| - k + k' \leq |M| \leq k'$ and such that the following condition holds: if $s \in M_K$, then $s \in M$, and otherwise, $|M| \leq k' - 1$. For $x' \in [x]$, let $\mathcal{I}_{x'} = \{M' \subseteq_{\pi^p} M_X : |M'| \leq x'\}$.

Let \mathcal{Q} be the family of pairs (C, M') , where $C \subseteq M_K$ and $M' \subseteq M_X$, such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , such that $s \in C$, and such that M' is invariant under $\pi^{p \cdot p_C}$. For a set C from such a pair, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

To compute \tilde{g} , we consider all possibilities for the label set of the connected component C of $G \setminus X_{args}$ that contains s . In the definition of \tilde{g} , the components of $G \setminus X_{args}$ are similar enough (since they all evaporate at time t) that they can potentially be mapped to one another by π^p . Thus we consider two cases: either C is invariant under π^p , or C is mapped to some other component of $G \setminus X_{args}$ by π^p . We add together the summations from these two cases, in a similar way to the recurrence for $a(k, p, M)$. In the recurrence for \tilde{g} , k' stands for $|C|$, and x' stands for $|N(C)|$. The set M stands for the set of vertices in C that can be moved by π^p , and M' stands for the set of vertices in $N(C)$ that can be moved by π^p .

► **Lemma 26.** *For \tilde{g} , we have*

$$\begin{aligned} \tilde{g} \binom{t \quad x \quad k \quad z}{p \quad M_X \quad M_K \quad M_Z} &= \sum_{\substack{1 \leq k' \leq k \\ 1 \leq x' \leq x \\ M \in \mathcal{P}_{k'} \\ M' \in \mathcal{I}_{x'}}} \tilde{g}_1 \binom{t \quad x' \quad k'}{p \quad M' \quad M} \tilde{g} \binom{t \quad x \quad k-k' \quad z}{p \quad M_X \quad M_K \setminus M \quad M_Z} \\ &\quad \cdot \begin{cases} \binom{k - |M_K|}{k' - |M|} & \text{if } s \in M_K \\ \binom{k-1 - |M_K|}{k'-1 - |M|} & \text{otherwise} \end{cases} \\ &\quad \cdot \begin{cases} \binom{x - |M_X|}{x' - |M'|} & \text{if } M' \not\subseteq M_Z \\ \binom{x - |M_X|}{x' - |M'|} - \binom{z - |M_Z|}{x' - |M'|} & \text{otherwise} \end{cases} \\ &+ \sum_{\substack{1 \leq x' \leq x \\ (C, M') \in \mathcal{Q}}} \tilde{g}_1 \binom{t \quad x' \quad |C|}{p \cdot p_C \quad M' \quad C} \tilde{g} \binom{t \quad x \quad k - p_C |C| \quad z}{p \quad M_X \quad M_K \setminus C_{\pi^p} \quad M_Z} \\ &\quad \cdot \begin{cases} \binom{x - |M_X|}{x' - |M'|} & \text{if } M' \not\subseteq M_Z \\ \binom{x - |M_X|}{x' - |M'|} - \binom{z - |M_Z|}{x' - |M'|} & \text{otherwise.} \end{cases} \end{aligned}$$

To compute \tilde{f} , we consider three cases: either no component of $G \setminus (X_{args} \cup L_{args})$ sees all of $X_{args} \cup L_{args}$, exactly one component sees all of $X_{args} \cup L_{args}$, or at least two components see all of $X_{args} \cup L_{args}$. The recursive calls to \tilde{g}_1 and $\tilde{g}_{\geq 2}$ correspond to the component/component(s) that see all of $X_{args} \cup L_{args}$. In the second and third cases, the set M stands for the set of vertices that can be moved by π^p in components that see all of $X_{args} \cup L_{args}$.

► **Lemma 27.** For \tilde{f} , we have

$$\begin{aligned} \tilde{f} \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} &= \tilde{f}_p \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} \\ &+ \sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{I}_{k'}}} \binom{k - |M_K|}{k' - |M|} \tilde{g}_1 \binom{t-1 \quad x+\ell \quad k'}{p \quad M_X \cup M_L \quad M} \tilde{f}_p \binom{t \quad x \quad \ell \quad k-k'}{p \quad M_X \quad M_L \quad M_K \setminus M} \\ &+ \sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{I}_{k'}}} \binom{k - |M_K|}{k' - |M|} \tilde{g}_{\geq 2} \binom{t-1 \quad x+\ell \quad k'}{p \quad M_X \cup M_L \quad M} \tilde{g}_p \binom{t-1 \quad x+\ell \quad k-k' \quad x}{p \quad M_X \cup M_L \quad M_K \setminus M \quad M_X}, \end{aligned}$$

where $\mathcal{I}_{k'} = \{M \subseteq_{\pi^p} M_K : |M_K| - k + k' \leq |M| \leq k'\}$.

For the next recurrence, suppose we have been given fixed values of the arguments of $\tilde{g}_{\geq 2}$. Let s be the smallest label in $V_{args} \setminus X_{args}$, and let $\mathcal{P}_{k'}$ be defined as it was in the recurrence for \tilde{g} . Let \mathcal{Q} be the family of sets $C \subseteq M$ such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , and such that $s \in C$. For a set $C \in \mathcal{Q}$, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

► **Lemma 28.** For $\tilde{g}_{\geq 2}$, we have

$$\begin{aligned} \tilde{g}_{\geq 2} \binom{t \quad x \quad k}{p \quad M_X \quad M_K} &= \\ &\sum_{\substack{1 \leq k' \leq k \\ M \in \mathcal{P}_{k'}}} \tilde{g}_1 \binom{t \quad x \quad k'}{p \quad M_X \quad M} \left(\tilde{g}_1 \binom{t \quad x \quad k-k'}{p \quad M_X \quad M_K \setminus M} + \tilde{g}_{\geq 2} \binom{t \quad x \quad k-k'}{p \quad M_X \quad M_K \setminus M} \right) \\ &\quad \cdot \begin{cases} \binom{k-|M_K|}{k'-|M|} & \text{if } s \in M_K \\ \binom{k-1-|M_K|}{k'-1-|M|} & \text{otherwise} \end{cases} \\ &+ \sum_{C \in \mathcal{Q}} \tilde{g}_1 \binom{t \quad x \quad |C|}{p \cdot p_C \quad M_X \quad C} \left(\tilde{g}_1 \binom{t \quad x \quad k-p_C|C|}{p \quad M_X \quad M_K \setminus C^\pi} + \tilde{g}_{\geq 2} \binom{t \quad x \quad k-p_C|C|}{p \quad M_X \quad M_K \setminus C^\pi} \right). \end{aligned}$$

To compute \tilde{g}_p , all we need to do is make a slight adjustment to the recurrence for \tilde{g} .

► **Lemma 29.** The recurrence for \tilde{g}_p is exactly the same as the recurrence for \tilde{g} in Lemma 26, except for the two sums over x' : In both summations, rather than summing over x' such that $1 \leq x' \leq x$, we sum over x' such that $1 \leq x' \leq x-1$.

To compute \tilde{f}_p , we first observe that when $z = x$, requiring $G \setminus Z_{args}$ to be connected is the same as requiring $G \setminus X_{args}$ to be connected.

► **Lemma 30.** We have $\tilde{f}_p \binom{t \quad x \quad \ell \quad k}{p \quad M_X \quad M_L \quad M_K} = \tilde{f}_p \binom{t \quad x \quad \ell \quad k \quad x}{p \quad M_X \quad M_L \quad M_K \quad M_X}$.

For the next \tilde{f}_p recurrence, we need one last round of definitions. Suppose we have been given fixed values of the arguments of \tilde{f}_p , including z and M_Z . Let s be the smallest label in $V_{args} \setminus (X_{args} \cup L_{args})$. For $k' \in [k]$, let $\mathcal{P}_{k'}$ be the family of sets $M \subseteq_{\pi^p} M_K$ such that $|M_K| - k + k' \leq |M| \leq k'$ and such that the following condition holds: if $s \in M_K$, then $s \in M$, and otherwise, $|M| \leq k' - 1$. For $0 \leq x' \leq x$, let $\mathcal{I}_{x'} = \{M' \subseteq_{\pi^p} M_X : |M'| \leq x'\}$. Also, for $0 \leq \ell' \leq \ell$, let $\mathcal{J}_{\ell'} = \{M' \subseteq_{\pi^p} M_L : |M'| \leq \ell'\}$.

Let \mathcal{Q} be the family of triples (C, M'_X, M'_L) , where $C \subseteq M_K$, $M'_X \subseteq M_X$, and $M'_L \subseteq M_L$, such that all elements of C have the same period $p_C \geq 2$ with respect to (C, π^p) , such that $s \in C$, and such that $M'_X \cup M'_L$ is invariant under π^{p_C} . For a set C from such a triple, we let $C_\sigma := C \cup \sigma(C) \cup \dots \cup \sigma^{p_C-1}(C)$, where $\sigma = \pi^p$.

► **Lemma 31.** For \tilde{f}_p with the argument z , we have

$$\begin{aligned}
 \tilde{f}_p \begin{pmatrix} t & x & \ell & k & z \\ p & M_X & M_L & M_K & M_Z \end{pmatrix} &= \sum_{\substack{1 \leq k' \leq k \\ 0 \leq x' \leq x \\ 0 \leq \ell' \leq \ell \\ 0 < x' + \ell' < x + \ell}} \sum_{\substack{M \in \mathcal{P}_{k'} \\ M'_X \in \mathcal{I}_{x'} \\ M'_L \in \mathcal{J}_{\ell'}}} \tilde{g}_1 \begin{pmatrix} t-1 & x'+\ell' & k' \\ p & M'_X \cup M'_L & M \end{pmatrix} \\
 &\cdot \begin{pmatrix} \ell - |M_L| \\ \ell' - |M'_L| \end{pmatrix} \cdot \begin{cases} \binom{k-|M_K|}{k'-|M|} & \text{if } s \in M_K \\ \binom{k-1-|M_K|}{k'-1-|M|} & \text{otherwise} \end{cases} \\
 &\cdot \begin{cases} \binom{x-|M_X|}{x'-|M'_X|} & \text{if } \ell' > 0 \text{ or } M'_X \not\subseteq M_Z \\ \binom{x-|M_X|}{x'-|M'_X|} - \binom{z-|M_Z|}{x'-|M'_X|} & \text{otherwise} \end{cases} \\
 &\cdot \begin{cases} \tilde{f}_p \begin{pmatrix} t & x+\ell' & \ell-\ell' & k-k' & z \\ p & M_X \cup M'_L & M_L \setminus M'_L & M_K \setminus M & M_Z \end{pmatrix} & \text{if } \ell' < \ell \\ \tilde{g}_p \begin{pmatrix} t-1 & x+\ell & k-k' & z \\ p & M_X \cup M'_L & M_K \setminus M & M_Z \end{pmatrix} & \text{otherwise} \end{cases} \\
 + \sum_{\substack{0 \leq x' \leq x \\ 0 \leq \ell' \leq \ell \\ 0 < x' + \ell' < x + \ell \\ (C, M'_X, M'_L) \in \mathcal{Q}}} \tilde{g}_1 \begin{pmatrix} t-1 & x'+\ell' & |C| \\ p \cdot p_C & M'_X \cup M'_L & C \end{pmatrix} \cdot \begin{pmatrix} \ell - |M_L| \\ \ell' - |M'_L| \end{pmatrix} \\
 \cdot \begin{cases} \binom{x-|M_X|}{x'-|M'_X|} & \text{if } \ell' > 0 \text{ or } M'_X \not\subseteq M_Z \\ \binom{x-|M_X|}{x'-|M'_X|} - \binom{z-|M_Z|}{x'-|M'_X|} & \text{otherwise} \end{cases} \\
 \cdot \begin{cases} \tilde{f}_p \begin{pmatrix} t & x+\ell' & \ell-\ell' & k-p_C|C| & z \\ p & M_X \cup M'_L & M_L \setminus M'_L & M_K \setminus C^\pi & M_Z \end{pmatrix} & \text{if } \ell' < \ell \\ \tilde{g}_p \begin{pmatrix} t-1 & x+\ell & k-p_C|C| & z \\ p & M_X \cup M'_L & M_K \setminus C^\pi & M_Z \end{pmatrix} & \text{otherwise.} \end{cases}
 \end{aligned}$$

See the full version of the paper for more intuition behind the recurrences for $\tilde{g}_{\geq 2}$ and \tilde{f}_p . The base cases for all of these counter functions are the same as in [14] since they only depend on the arguments t, x, ℓ, k, z .

5 Conclusion

We built upon the algorithm of Wormald for generating random unlabeled graphs to design an algorithm that, given n , generates a random unlabeled chordal graph on n vertices in expected polynomial time. This serves as a proof of concept that one can obtain a sampling algorithm for unlabeled graphs from a GI-complete graph class \mathcal{G} using the following two ingredients: (1) an FPT algorithm for counting labeled graphs in \mathcal{G} with a given automorphism π parameterized by the number of moved points of π and (2) a bound on the probability that a labeled graph in \mathcal{G} has a given automorphism. A few potential candidates for this are bipartite graphs, strongly chordal graphs, and chordal bipartite graphs, all of which are GI-complete. An additional open problem would be to design a uniform, or approximately uniform, sampling algorithm – either for unlabeled chordal graphs or general unlabeled graphs – that runs in expected polynomial time even when we condition on the output graph.

References

- 1 Edward A. Bender, L. Bruce Richmond, and Nicholas C. Wormald. Almost all chordal graphs split. *Journal of the Australian Mathematical Society*, 38(2):214–221, 1985.
- 2 Vladislav Bína and Jiří Přibíl. Note on enumeration of labeled split graphs. *Commentationes Mathematicae Universitatis Carolinae*, 56(2):133–137, 2015.
- 3 Jean R.S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.
- 4 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Sampling unlabeled biconnected planar graphs. In *Algorithms and Computation: 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005. Proceedings 16*, pages 593–603. Springer, 2005. doi:10.1007/11602613_60.
- 5 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating labeled planar graphs uniformly at random. *Theoretical Computer Science*, 379(3):377–386, 2007. doi:10.1016/J.TCS.2007.02.045.
- 6 Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating unlabeled connected cubic planar graphs uniformly at random. *Random Structures & Algorithms*, 32(2):157–180, 2008. doi:10.1002/RSA.20206.
- 7 John D. Dixon and Herbert S. Wilf. The random selection of unlabeled graphs. *Journal of Algorithms*, 4(3):205–213, 1983. doi:10.1016/0196-6774(83)90021-4.
- 8 Éric Fusy. Uniform random sampling of planar graphs in linear time. *Random Structures & Algorithms*, 35(4):464–522, 2009. doi:10.1002/RSA.20275.
- 9 Andrew Gainer-Dewar and Ira M. Gessel. Counting unlabeled k -trees. *Journal of Combinatorial Theory, Series A*, 126:177–193, 2014. doi:10.1016/J.JCTA.2014.05.002.
- 10 Pu Gao and Nicholas Wormald. Uniform generation of random regular graphs. *SIAM Journal on Computing*, 46:1395–1427, 2017. doi:10.1137/15M1052779.
- 11 Pu Gao and Nicholas Wormald. Uniform generation of random graphs with power-law degree sequences. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1741–1758, 2018. doi:10.1137/1.9781611975031.114.
- 12 Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972. doi:10.1137/0201013.
- 13 András Hajnal and János Surányi. Über die auflösung von graphen in vollständige teilgraphen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math.*, 1:113–121, 1958.
- 14 Úrsula Hébert-Johnson, Daniel Lokshtanov, and Eric Vigoda. Counting and sampling labeled chordal graphs in polynomial time, 2023. doi:10.48550/arXiv.2308.09703.
- 15 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986. doi:10.1016/0304-3975(86)90174-X.
- 16 Walter Oberschelp. Kombinatorische anzahlbestimmungen in relationen. *Mathematische Annalen*, 174:53–78, 1967.
- 17 OEIS Foundation Inc. Entry A058862 in the On-Line Encyclopedia of Integer Sequences, 2024. Published electronically at <http://oeis.org>.
- 18 Toshiki Saitoh, Yota Otachi, Katsuhisa Yamanaka, and Ryuhei Uehara. Random generation and enumeration of bipartite permutation graphs. *Journal of Discrete Algorithms*, 10:84–97, 2012. doi:10.1016/J.JDA.2011.11.001.
- 19 Herbert S. Wilf. The uniform selection of free trees. *Journal of Algorithms*, 2(2):204–207, 1981. doi:10.1016/0196-6774(81)90021-3.
- 20 Nicholas C. Wormald. Generating random unlabelled graphs. *SIAM Journal on Computing*, 16(4):717–727, 1987. doi:10.1137/0216048.

Minimizing the Number of Tardy Jobs with Uniform Processing Times on Parallel Machines

Klaus Heeger  

Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Hendrik Molter  

Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

In this work, we study the computational (parameterized) complexity of $P \mid r_j, p_j = p \mid \sum w_j U_j$. Here, we are given m identical parallel machines and n jobs with equal processing time, each characterized by a release date, a due date, and a weight. The task is to find a feasible schedule, that is, an assignment of the jobs to starting times on machines, such that no job starts before its release date and no machine processes several jobs at the same time, that minimizes the weighted number of tardy jobs. A job is considered tardy if it finishes after its due date.

Our main contribution is showing that $P \mid r_j, p_j = p \mid \sum U_j$ (the unweighted version of the problem) is NP-hard and W[2]-hard when parameterized by the number of machines. The former resolves an open problem in Note 2.1.19 by Kravchenko and Werner [Journal of Scheduling, 2011] and Open Problem 2 by Sgall [ESA, 2012], and the latter resolves Open Problem 7 by Mnich and van Bevern [Computers & Operations Research, 2018]. Furthermore, our result shows that the known XP-algorithm by Baptiste et al. [4OR, 2004] for $P \mid r_j, p_j = p \mid \sum w_j U_j$ parameterized by the number of machines is optimal from a classification standpoint.

On the algorithmic side, we provide alternative running time bounds for the above-mentioned known XP-algorithm. Our analysis shows that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is contained in XP when parameterized by the processing time, and that it is contained in FPT when parameterized by the combination of the number of machines and the processing time. Finally, we give an FPT-algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ parameterized by the number of release dates or the number of due dates. With this work, we lay out the foundation for a systematic study of the parameterized complexity of $P \mid r_j, p_j = p \mid \sum w_j U_j$.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms; Mathematics of computing \rightarrow Discrete mathematics; Computing methodologies \rightarrow Planning and scheduling

Keywords and phrases Scheduling, Identical Parallel Machines, Weighted Number of Tardy Jobs, Uniform Processing Times, Release Dates, NP-hard Problems, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.47

Funding Klaus Heeger: Supported by the ISF, grant No. 1070/20.

Hendrik Molter: Supported by the ISF, grant nr. 1470/24 and by the European Union's Horizon Europe research and innovation program under grant agreement 949707.

1 Introduction

Machine scheduling is one of the most fundamental application areas of combinatorial optimization [34]. In a typical scheduling problem, the task is to assign jobs to machines with the goal of maximizing a certain optimization objective while complying with certain constraints. Jobs are usually characterized by a *processing time*, a *release date*, a *due date*, and a *weight* (or a subset thereof). We consider the setting where we have access to several



© Klaus Heeger and Hendrik Molter;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 47; pp. 47:1–47:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



identical parallel machines that can each process one job (non-preemptively) at a time. One of the most fundamental optimization objectives is to minimize the weighted number of tardy jobs, where a job is considered *tardy* if it is completed after its due date.

The arguably simplest scheduling problem aims to minimize the (unweighted) number of tardy jobs on a single machine, where all jobs are released at time zero. In the standard three-field notation for scheduling problems by Graham et al. [15], this problem is called $1 \parallel \sum U_j$. It can be solved in polynomial time by a classic algorithm of Moore [33]. However, this problem becomes NP-hard when weights are introduced, the number of machines is increased, or release dates are added.

- The weighted version, $1 \parallel \sum w_j U_j$, is one of the first scheduling problems shown to be (weakly) NP-hard. It remains hard even if all jobs have the same due date, as in this case, it is equivalent to the well-known KNAPSACK problem, which was already included in Karp's famous list of 21 NP-hard problems [21]. The problem can be solved in pseudopolynomial time with a classic algorithm by Lawler and Moore [28].
- Adding a second machine leads to the problem $2 \parallel \sum_j U_j$, which is (weakly) NP-hard even if all jobs have the same due date, as it is a generalization of the well-known PARTITION problem, which was also already included in Karp's list of 21 NP-hard problems [21]. If the number of machines is unbounded, the problem is called $P \parallel \sum_j U_j$ and it is strongly NP-hard even if all jobs have the same due date, as it generalizes the well-known BIN PACKING problem [14].
- Introducing release times leads to the problem $1 \mid r_j \mid \sum_j U_j$, which is weakly NP-hard [30], even if there are only two different release dates and two different due dates. The reduction of Lenstra et al. [30] can be extended in a straightforward way to show that $1 \mid r_j \mid \sum U_j$ is strongly NP-hard.

Problem Setting and Motivation. We consider the case where release dates and weights are present and we have multiple identical parallel machines. However, we add the restriction that all processing times are the same. This problem is called $P \mid r_j, p_j = p \mid \sum w_j U_j$, we give a formal definition in Section 2. This problem arises naturally in manufacturing systems, where

- exact specifications by the customers for the product have negligible influence on the production time, but
- the specifications only become available at certain times and customers request the product to meet certain due dates.

As an illustrative example, consider the problem of scheduling burn-in operations in integrated circuit manufacturing. The specification of the layout of the circuit may only become available at a certain time, as it takes time to optimize it. At the same time, the specific layout has little to no influence on the processing time of the burn-in operation [31]. Furthermore, the customer may wish to have the circuit delivered at a certain due date.

To the best of our knowledge, the only known algorithmic result for $P \mid r_j, p_j = p \mid \sum w_j U_j$ is a polynomial-time algorithm by Baptiste et al. [2, 3] for the case where the number of machines is a constant. However, two special cases are known to be polynomial-time solvable. It is folklore that the case without release dates, $P \mid p_j = p \mid \sum_j w_j U_j$, and the case where the processing times equal one, $P \mid r_j, p_j = 1 \mid \sum_j w_j U_j$, can both be reduced to the LINEAR ASSIGNMENT problem in a straightforward manner. The LINEAR ASSIGNMENT problem is known to be solvable in polynomial time [27]. Furthermore, it is known that, given an instance of $P \mid r_j, p_j = p \mid \sum w_j U_j$, we can check in polynomial time whether all jobs can be scheduled such that *no* job is tardy [5, 37, 38].

Our Contribution. The complexity status of $P \mid r_j, p_j = p \mid \sum w_j U_j$ and its unweighted version $P \mid r_j, p_j = p \mid \sum U_j$ was a longstanding open problem. Kravchenko and Werner [25] pointed out that this question remains unanswered in Note 2.1.19 and Sgall [36] listed this issue as Open Problem 2. Our main contribution is to resolve the complexity status of $P \mid r_j, p_j = p \mid \sum U_j$ (and hence also $P \mid r_j, p_j = p \mid \sum w_j U_j$) by showing the following.

- $P \mid r_j, p_j = p \mid \sum U_j$ is NP-hard.

Having established NP-hardness, we focus on studying the parameterized complexity of $P \mid r_j, p_j = p \mid \sum w_j U_j$. As mentioned before, Baptiste et al. [2, 3] showed that the problem is in XP when parameterized by the number of machines. Mnich and van Bevern [32, Open Problem 7] asked whether this result can be improved to an FPT algorithm. We answer this question negatively by showing the following.

- $P \mid r_j, p_j = p \mid \sum U_j$ is W[2]-hard when parameterized by the number of machines.

On the positive side, we give several new parameterized tractability results. By providing an alternative running time analysis of the algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ by Baptiste et al. [2, 3], we show the following.

- $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in XP when parameterized by the processing time.
- $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the combination of the number of machines and the processing time.

Finally, we give a new algorithm based on a mixed integer linear program (MILP) formulation for $P \mid r_j, p_j = p \mid \sum w_j U_j$. We show the following.

- $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the number of different release dates.
- $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the number of different due dates.

We conclude by pointing to new future research directions. Most prominently, we leave open whether $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT or W[1]-hard when parameterized by the processing time.

Related Work. We give an overview of the literature investigating the parameterized complexity of minimizing the weighted number of tardy jobs in various related settings.

The problem of minimizing the weighted number of tardy jobs on a single machine, $1 \parallel \sum_j w_j U_j$ has been extensively studied in the literature under various aspects and constraints. Hermelin et al. [20] showed that the classical pseudopolynomial time algorithm by Lawler and Moore [28] can be improved in several special cases. Hermelin et al. [19] give an overview of the parameterized complexity of $1 \parallel \sum_j w_j U_j$ with respect to the parameters “number of processing times”, “number of due dates”, and “number of weights” (and their combinations). In particular, $1 \parallel \sum w_j U_j$ is in XP when parameterized by the number of different processing times [19]. This presumably cannot be improved to an FPT result as recently, it was shown that $1 \parallel \sum w_j U_j$ parameterized by the number of different processing times is W[1]-hard [16]. Faster algorithms are known for the case where the job weights equal the processing times [4, 12, 23, 35] and the problem has also been studied under fairness aspects [17]. Kaul et al. [22] extended the results of Hermelin et al [19] to $1 \mid r_j \mid \sum w_j U_j$, considering the “number of release times” as an additional parameter.

Minimizing the weighted number of tardy jobs on parallel machines has mostly been studied in the context of interval scheduling ($P \mid d_j - r_j = p_j \mid \sum_j w_j U_j$) and generalizations thereof [1, 18, 26, 39].

The setting where processing times are assumed to be uniform has been studied under various optimization criteria (different from minimizing the weighted number of tardy jobs) and constraints. For an overview, we refer to Kravchenko and Werner [25] and Baptiste et al. [3]. The even more special case of unit processing times has also been extensively studied. Reviewing the related literature in this setting is out of scope for this work.

2 Preliminaries

Scheduling. Using the standard three-field notation for scheduling problems by Graham et al. [15], the problem considered in this work is called $P \mid r_j, p_j = p \mid \sum w_j U_j$. In this problem, we have n jobs and m machines. Each machine can process one job at a time. Generally, we use the variable j to denote jobs and the variable i to denote machines. Each job j has a *processing time* $p_j = p$, a *release date* r_j , a *due date* d_j , and a *weight* w_j , where p , r_j , d_j , and w_j are nonnegative integers. We use $r_{\#}$, $d_{\#}$, and $w_{\#}$ to denote the number of different release dates, due dates, and weights, respectively.

A schedule maps each job j to a combination of a machine i and a starting time t , indicating that j shall be processed on machine i starting at time t . More formally, a *schedule* is a function $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, m\} \times \mathbb{N}$. If for job j we have $\sigma(j) = (i, t)$, then job j is scheduled to be processed on machine i starting at time t until time $t + p$. A schedule σ is *feasible* if there is no job j with $\sigma(j) = (i, t)$ and $t < r_j$ and if there is no pair of jobs j, j' with $\sigma(j) = (i, t)$ and $\sigma(j') = (i, t')$ such that $|t - t'| < p$. We say that a job j is *early* in a feasible schedule σ if $\sigma(j) = (i, t)$ and $t + p \leq d_j$, otherwise we say that job j is *tardy*. We say that machine i is *idle* at time t in a feasible schedule σ if there is no job j with $\sigma(j) = (i, t')$ and $t' \leq t \leq t' + p$. The goal is to find a feasible schedule that minimizes the weighted number of tardy jobs or, equivalently, maximizes the weighted number of early jobs $W = \sum_{j \mid \sigma(j) = (i, t) \wedge t + p \leq d_j} w_j$. We call a feasible schedule that maximizes the weighted number of early jobs *optimal*. Formally, the problem is defined as follows.

$$P \mid r_j, p_j = p \mid \sum w_j U_j$$

Input: A number n of jobs, a number m of machines, a processing time p , a list of release dates (r_1, r_2, \dots, r_n) , a list of due dates (d_1, d_2, \dots, d_n) , and a list of weights (w_1, w_2, \dots, w_n) .

Task: Compute a feasible schedule σ that maximizes $W = \sum_{j \mid \sigma(j) = (i, t) \wedge t + p \leq d_j} w_j$.

We use $P \mid r_j, p_j = p \mid \sum U_j$ to denote the unweighted (or, equivalently, uniformly weighted) version of $P \mid r_j, p_j = p \mid \sum w_j U_j$, that is, the case where $w_j = w_{j'}$ for every two jobs j and j' , or equivalently, $w_{\#} = 1$.

Note that given any feasible schedule of the early jobs, one can easily extend this to a feasible schedule of all jobs as tardy jobs can be scheduled arbitrarily late. Thus, when describing a schedule, we will only describe how the early jobs are scheduled.

Given an instance I of $P \mid r_j, p_j = p \mid \sum w_j U_j$, we can make the following observation, essentially implying that one can switch the roles of release dates and due dates.

► **Observation 1.** *Let I be an instance of $P \mid r_j, p_j = p \mid \sum w_j U_j$ and let d_{\max} be the largest due date of any job in I . Let I' be the instance obtained from I by setting $r'_j = d_{\max} - d_j$ and $d'_j = d_{\max} - r_j$. Then I admits a feasible schedule where the weighted number of early jobs is W if and only if I' admits a feasible schedule where the weighted number of early jobs is W .*

To see that Observation 1 is true note that a feasible schedule σ for I can be transformed into a feasible schedule σ' for I' (with the same weighted number of early jobs) by setting $\sigma'(j) = (i, d_{\max} - t - p)$, where $\sigma(j) = (i, t)$.

We now show that we can restrict ourselves to schedules where jobs may start only at “few” different points in time, which will be useful in our proofs. In order to do so, we define a set \mathcal{T} of *relevant starting time points*.

$$\mathcal{T} = \{t \mid \exists r_j \text{ and } \exists 0 \leq \ell \leq n \text{ s.t. } t = r_j + p \cdot \ell\}$$

It is known that there always exists an optimal schedule where the starting times of all jobs are in \mathcal{T} .

► **Lemma 2** ([2, 3]). *Let I be an instance of $P \mid r_j, p_j = p \mid \sum w_j U_j$. Then there exists a feasible schedule σ that maximizes the weighted number of early jobs such that for each job j we have $\sigma(j) = (i, t)$ for some $t \in \mathcal{T}$.*

Parameterized Complexity. We use the following standard concepts from parameterized complexity theory [7, 11, 13]. A *parameterized problem* $L \subseteq \Sigma^* \times \mathbb{N}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N}$, where k denotes the *parameter*. A parameterized problem L is in the class FPT (or *fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time for some computable function f that depends only on the parameter. A parameterized problem L is in the class XP if there is an algorithm that decides every instance (x, k) for L in $|x|^{f(k)}$ time for some computable function f that depends only on the parameter. If a parameterized problem L is W[1]-hard or W[2]-hard, then it is presumably not contained in FPT [7, 11, 13].

3 Hardness of $P \mid r_j, p_j = p \mid \sum U_j$

In this section, we present our main result, namely that the unweighted version of our scheduling problem, $P \mid r_j, p_j = p \mid \sum U_j$, is NP-hard and W[2]-hard when parameterized by the number m of machines. The former resolves an open problem in Note 2.1.19 by Kravchenko and Werner [25] and Open Problem 2 by Sgall [36], and the latter resolves Open Problem 7 by Mnich and van Bevern [32].

► **Theorem 3.** *$P \mid r_j, p_j = p \mid \sum U_j$ is NP-hard and W[2]-hard parameterized by the number m of machines.*

In order to show Theorem 3, we present a parameterized reduction from HITTING SET parameterized by solution size k , which is known to be NP-hard [21] (unparameterized) and W[2]-hard [10].

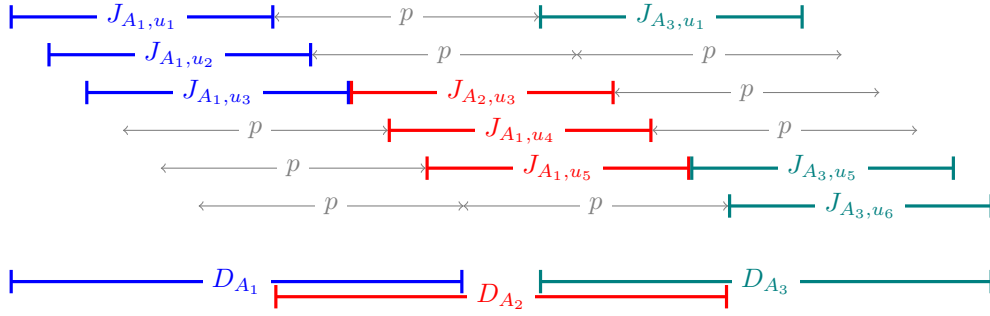
HITTING SET

Input: A finite set $U = \{u_0, \dots, u_{n-1}\}$, a set $\mathcal{A} = \{A_0, \dots, A_{m-1}\}$ of subsets of U , and an integer k .

Question: Is there a *hitting set* of size k , that is, a set $X \subseteq U$ with $|X| = k$ and $X \cap A_j \neq \emptyset$ for every $j \in \{0, \dots, m-1\}$?

Let $I = (U = \{u_0, \dots, u_{n-1}\}, \mathcal{A} = \{A_0, \dots, A_{m-1}\}, k)$ be an instance of HITTING SET. In order to ease the presentation, we give jobs names rather than identifying them with natural numbers. Furthermore, for a job J , we use $r(J)$ to denote the release date of J , and we use $d(J)$ to denote the deadline of J .

47:6 Minimizing the Number of Tardy Jobs with Uniform Processing Times



■ **Figure 1** The jobs of the first reduction approach for the HITTING SET instance $I = (\{u_1, u_2, u_3, u_4, u_5, u_6\}, \{A_1 = \{u_1, u_2, u_3\}, A_2 = \{u_3, u_4, u_5\}, A_3 = \{u_1, u_5, u_6\}, k = 2\})$.

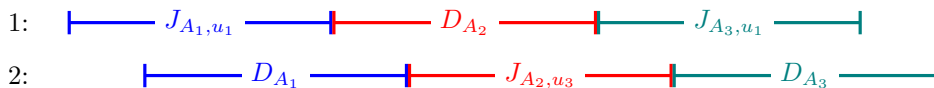
Our reduction will have k machines. The main idea behind the reduction is as follows. Each machine acts as a “selection gadget”, that is, we will interpret the jobs scheduled on each machine in an optimal schedule as the selection of a particular element of U to be included in the hitting set. As there are k machines, this ensures that the (hitting) set consisting of the selected elements has size at most k . Intuitively, we want that selecting element u_i on a machine corresponds to all jobs on this machine starting at time i modulo p in an optimal schedule. For each set $A_j \in \mathcal{A}$, there are two kinds of jobs.

- First, jobs J_{A_j, u_i} for each $u_i \in A_j$, where scheduling job J_{A_j, u_i} encodes that the element u_i is selected to be part of the hitting set.
- Second, there are $k - 1$ dummy jobs D_{A_j} which can be scheduled on the up to $k - 1$ machines not corresponding to elements of A_j .

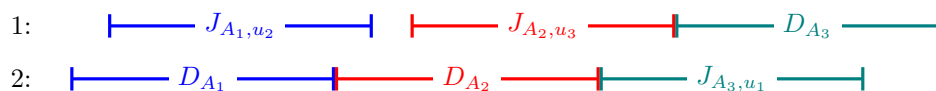
We give the jobs J_{A_j, u_i} and D_{A_j} release dates and due dates such that they are the only jobs that can be started in the interval $[j \cdot p, (j + 1) \cdot p - 1]$ and are early. See Figure 1 for an illustration. Intuitively, this makes sure that an optimal solution has to schedule one of these jobs on each machine. In particular, one job J_{A_j, u_i} is scheduled, implying that A_j is hit by one of the selected elements. See Figure 2 for an illustration.

There is, however, one problem with the reduction as sketched above. We do not ensure that all early jobs scheduled on some machine start at the same time modulo p . Thus, it is possible to schedule e.g. first job J_{A_1, u_2} on machine 1 and then job J_{A_2, u_3} such that the two jobs are both early. Machine 1 now does not encode the selection of a single element to the hitting set. We illustrate an example in Figure 3. However, note that it is only possible to increase the index of the “selected” element, and as there are only k machines, the total increase is bounded by $k \cdot (n - 1)$. Consequently, repeating the reduction sketched above $k \cdot (n - 1) + 1$ times ensures that at least one of the repeated instances will select only one item per machine, and then this instance correctly encodes a hitting set of size k .

We now describe the reduction in detail. Formally, given the instance I of HITTING SET, we construct an instance I' of $P \mid r_j, p_j = p \mid \sum U_j$ as follows. We set the processing time to $p = 2n$. We construct the following jobs for each $A_j \in \mathcal{A}$ and $\ell \in \{0, \dots, k \cdot (n - 1)\}$:



■ **Figure 2** An optimal schedule for the instance from Figure 1 representing the hitting set $\{u_1, u_3\}$.



■ **Figure 3** An optimal schedule for the instance from Figure 1 which does not represent a hitting set.

- one job J_{A_j, u_i}^ℓ for each $u_i \in A_j$ with release date $r(J_{A_j, u_i}^\ell) = (\ell \cdot m + j) \cdot p + i$ and due date $d(J_{A_j, u_i}^\ell) = r(J_{A_j, u_i}^\ell) + p$, and
- $k - 1$ dummy jobs $D_{A_j}^\ell$ with release date $r(D_{A_j}^\ell) = (\ell \cdot m + j) \cdot p$ and due date $d(D_{A_j}^\ell) = r(D_{A_j}^\ell) + p + n$.

Finally, we set the number of machines to k . This finishes the construction. For an overview of the due dates and release dates of the jobs, see Table 1. We can easily observe the following.

► **Observation 4.** *Given an instance I of HITTING SET, the above-described instance I' of $P \mid r_j, p_j = p \mid \sum U_j$ can be computed in polynomial time and has k machines.*

We continue by showing the correctness of the reduction. More specifically, we show that the HITTING SET I instance is a yes-instance if and only if the constructed instance I' of $P \mid r_j, p_j = p \mid \sum U_j$ admits a feasible schedule with $(k \cdot (n - 1) + 1) \cdot m \cdot k$ early jobs. We split the proof into the forward and backward direction. We start with the forward direction.

► **Lemma 5.** *If the HITTING SET instance I admits a hitting set of size k , then the $P \mid r_j, p_j = p \mid \sum U_j$ instance I' admits a feasible schedule with $(k \cdot (n - 1) + 1) \cdot m \cdot k$ early jobs.*

Proof. Let $X = \{u_{i_1}, \dots, u_{i_k}\}$ be a hitting set of size k for I . We construct a schedule for I' as follows. On machine q , for each $\ell \in \{0, \dots, k \cdot (n - 1)\}$ and $j \in \{0, \dots, m - 1\}$, we schedule one job to start at time $(\ell \cdot m + j) \cdot p + i_q$. This job is $J_{A_j, u_{i_q}}^\ell$ if $u_{i_q} \in A_j$, and $D_{A_j}^\ell$ otherwise. Because X is a hitting set, we have that $u_{i_q} \in A_j$ for some $q \in \{1, \dots, k\}$ and hence we schedule each dummy job $D_{A_j}^\ell$ at most $k - 1$ times, that is, at most its multiplicity times.

We can observe that all jobs scheduled so far are early. For each job $J_{A_j, u_{i_q}}^\ell$ that is scheduled on machine q , we have set its starting time to $(\ell \cdot m + j) \cdot p + i_q$ which equals this job's release date (cf. Table 1). Furthermore, job $J_{A_j, u_{i_q}}^\ell$ finishes at $(\ell \cdot m + j + 1) \cdot p + i_q$, its deadline. Each dummy job $D_{A_j}^\ell$ is early as well, since their release times are smaller or equal to the release time of $J_{A_j, u_{i_q}}^\ell$, and their due dates are larger or equal to the due date of $J_{A_j, u_{i_q}}^\ell$. Furthermore, we can observe that there is no overlap in the processing times between any two jobs scheduled on machine q .

It follows that we have feasibly scheduled $(k \cdot (n - 1) + 1) \cdot m \cdot k$ such that they finish early. We schedule the remaining jobs in some arbitrary way such that the schedule remains feasible. ◀

■ **Table 1** Overview of the release dates and due dates of the jobs created for each $A_j \in \mathcal{A}$ and $\ell \in \{0, \dots, k \cdot (n - 1)\}$.

job	release date	due date	multiplicity
J_{A_j, u_i}^ℓ	$(\ell \cdot m + j) \cdot p + i$	$(\ell \cdot m + j + 1) \cdot p + i$	1
$D_{A_j}^\ell$	$(\ell \cdot m + j) \cdot p$	$(\ell \cdot m + j + 1) \cdot p + n$	$k - 1$

Next, we continue with the backward direction.

► **Lemma 6.** *If the $P \mid r_j, p_j = p \mid \sum U_j$ instance I' admits a feasible schedule with $(k \cdot (n - 1) + 1) \cdot m \cdot k$ early jobs, then the HITTING SET instance I admits a hitting set of size k .*

Proof. Let σ be a feasible schedule for I' with $(k \cdot (n - 1) + 1) \cdot m \cdot k$ early jobs. Since the largest due date is $(k \cdot (n - 1) \cdot m + m) \cdot p + n$ and $p = 2n$, it follows that on each machine, at most $k \cdot (n - 1) \cdot m + m$ jobs can be scheduled in a feasible way such that they finish early. Consequently, on each machine, there are exactly $(k \cdot (n - 1) + 1) \cdot m$ early jobs.

More specifically, for each machine, each $\ell \in \{0, \dots, k \cdot (n - 1)\}$ and $j \in \{0, \dots, m - 1\}$, there must be one job which starts in the interval $[(\ell \cdot m + j) \cdot p, (\ell \cdot m + j) \cdot p + n]$. Otherwise, there would be less than $(k \cdot (n - 1) + 1) \cdot m$ early jobs on that machine. For machine q , let $x_\ell^q \in \{1, \dots, n\}$ such that there is a job on machine q which starts at time $\ell \cdot m \cdot p + x_\ell^q$ (the existence of such a job is guaranteed by the previous observation). Then we must have $1 \leq x_0^q \leq x_1^q \leq \dots \leq x_{k \cdot (n - 1)}^q \leq n$. This follows from the observation that if $x_\ell^q > x_{\ell + 1}^q$, then the starting time of the job corresponding to $x_{\ell + 1}^q$ would be earlier than the completion time of the job corresponding to x_ℓ^q and hence the schedule would be infeasible. Consequently, there are at most $n - 1$ values of ℓ such that $x_\ell^q \neq x_{\ell + 1}^q$. As there are k machines, this implies that there are at most $k \cdot (n - 1)$ values such that $x_\ell^q \neq x_{\ell + 1}^q$ for *some* machine q . We can conclude that there exists at least one $\ell \in \{0, \dots, k \cdot (n - 1)\}$ so that $x_\ell^q = x_{\ell + 1}^q$ for every machine q . This implies that for each $j \in \{0, \dots, m - 1\}$ and each machine q , there is one job starting at time $(\ell \cdot m + j) \cdot p + x_\ell^q$ on machine q . We fix such an ℓ for the rest of the proof and claim that $X = \{u_{x_\ell^1}, \dots, u_{x_\ell^k}\}$ is a hitting set of size at most k for I .

Clearly, X has size at most k . Consider some set $A_j \in \mathcal{A}$ with $j \in \{0, \dots, m - 1\}$. The only jobs which can start in the interval $[(\ell \cdot m + j) \cdot p, (\ell \cdot m + j) \cdot p + n]$ are the dummy jobs $D_{A_j}^\ell$ and the jobs J_{A_j, u_i}^ℓ for $u_i \in A_j$. Because there are only $k - 1$ dummy jobs $D_{A_j}^\ell$ but k machines, this implies that there exists at least one machine q such that job J_{A_j, u_i}^ℓ is scheduled at machine q for some $u_i \in A_j$. We know that on machine q one job starts in the interval $[(\ell \cdot m + j) \cdot p, (\ell \cdot m + j) \cdot p + n]$ and has starting time $(\ell \cdot m + j) \cdot p + x_\ell^q$. By construction, this job must be J_{A_j, u_i}^ℓ with $i = x_\ell^q$ which implies that $u_i = u_{x_\ell^q} \in X$. We can conclude that X is a hitting set for I . ◀

Now we have all the pieces to prove Theorem 3.

Proof of Theorem 3. Observation 4 shows that the described reduction can be computed in polynomial time and produces an instance of $P \mid r_j, p_j = p \mid \sum U_j$ with k machines. Lemmas 5 and 6 show that the described reduction is correct. Since HITTING SET is known to be NP-hard [21] and W[2]-hard when parameterized by k [10], the result follows. ◀

4 New Analysis of Known Algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$

With Theorem 3 we have established that $P \mid r_j, p_j = p \mid \sum U_j$ is NP-hard. Hence, it is natural to resort to parameterized algorithms for efficiently finding exact solutions in restricted cases. To the best of our knowledge, the only known parameterized algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ is an XP-algorithm for the number m of machines as a parameter by Baptiste et al. [2, 3].

► **Theorem 7** ([2, 3]). *$P \mid r_j, p_j = p \mid \sum w_j U_j$ can be solved in $n^{O(m)}$ time, where n is the number of jobs and m is the number of machines.*

Theorem 7 implies that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in XP when parameterized by the number m of machines. Since Theorem 3 also shows W[2]-hardness for $P \mid r_j, p_j = p \mid \sum U_j$ parameterized by the number m of machines, the algorithm behind Theorem 7 presumably cannot be improved to an FPT-algorithm.

However, as it turns out, we can upper-bound the running time of the algorithm behind Theorem 7 in different ways to obtain additional tractability results. In the remainder of this section, we show that the algorithm developed by Baptiste et al. [2, 3] additionally to Theorem 7 also implies the following.

► **Theorem 8.** $P \mid r_j, p_j = p \mid \sum w_j U_j$ can be solved in $p^{O(m)} \cdot n^{O(1)}$ time and in $m^{O(p)} \cdot n^{O(1)}$ time, where n is the number of jobs, m is the number of machines, and p is the processing time.

Theorem 8 implies that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in XP when parameterized by the processing time p and that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the combination of the number m of machines and the processing time p . In order to prove Theorem 8, we present the dynamic programming algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ by Baptiste et al. [2, 3]. For the correctness of this algorithm, we refer to their work. We give an alternative running time analysis that shows the claimed running time bounds.

To this end, we need to introduce some additional notation and terminology. Recall that \mathcal{T} denotes the set of relevant starting time points. The algorithm makes use of Lemma 2, that is, we can assume the starting times of all jobs in an optimal schedule are from \mathcal{T} . A *resource profile* is a vector $x = (x_1, x_2, \dots, x_m)$ with $x_1 \leq x_2 \leq \dots \leq x_m$, $x_m - x_1 \leq p$, and $x_i \in \mathcal{T}$ for all $i \in \{1, \dots, m\}$. Let \mathcal{X} denote the set of all resource profiles. Now we define the following dynamic program. We assume that the jobs are sorted according to their due dates, that is, $d_1 \leq d_2 \leq \dots \leq d_n$.

For two resource profiles $a, b \in \mathcal{X}$ and some $k \in \{1, \dots, n\}$ we define $W(k, a, b)$ to be the maximum weighted number of early jobs of any feasible schedule for the jobs $1, \dots, k$ such that

- sorting the starting times of the first jobs on each machine from smallest to largest yields a vector a' with $a \leq a'$, and
- sorting the completion times of the last jobs on each machine from smallest to largest yields a vector b' with $b' \leq b$,

where for two vectors a, b of length m we say that $a \leq b$ if and only if for all $i \in \{1, \dots, m\}$ we have that $a_i \leq b_i$.

From this definition, it follows that $W(n, (0, \dots, 0), (t_{\max}, \dots, t_{\max}))$, where t_{\max} is the largest element in \mathcal{T} , is the maximum weighted number of early jobs of any feasible schedule. Baptiste et al. [2, 3] proved the following.

► **Lemma 9** ([2, 3]). *For all $k \in \{1, \dots, n\}$ and all resource profiles $a, b \in \mathcal{X}$ with $a \leq b$ it holds that $W(k, a, b)$ equals $W(k-1, a, b)$ if $r_k \notin [a_1, b_m - p]$ and otherwise*

$$\max \left(W(k-1, a, b), \max_{\substack{x \in \mathcal{X}, r_k \leq x_1, x_1 + p \leq d_k, a \leq x, \\ x' = (x_2, x_3, \dots, x_m, x_1 + p) \leq b}} \left(W(k-1, a, x) + W(k-1, x', b) + w_k \right) \right),$$

where we define $W(0, a, b) = 0$ for all $a, b \in \mathcal{X}$ with $a \leq b$.

A straightforward running time analysis yields the following. We have that $|\mathcal{T}| \in O(n^2)$ and hence $|\mathcal{X}| \in O(n^{2m})$. It follows that the size of the dynamic programming table W is in $O(n^{4m+1})$ and the time to compute one entry is in $O(n^{2m})$. This together with Lemma 9 yields Theorem 7. In the remainder of the section, we give an alternative running time analysis to prove Theorem 8.

47:10 Minimizing the Number of Tardy Jobs with Uniform Processing Times

Proof of Theorem 8. To prove Theorem 8 we give a different bound for the size of \mathcal{X} . Recall that for all resource profiles $x \in \mathcal{X}$ we have that $x_m - x_1 \leq p$. It follows that there are $|\mathcal{T}|$ possibilities for the value of x_1 , and then for $2 \leq i \leq m$ we have that $x_1 \leq x_i \leq x_1 + p$. Hence, we get that $|\mathcal{X}| \in O(n^2 \cdot p^{m-1})$. This together with Lemma 9 immediately gives us that $P \mid r_j, p_j = p \mid \sum w_j U_j$ can be solved in $p^{O(m)} \cdot n^{O(1)}$ time.

Furthermore, we have $x_1 \leq x_2 \leq \dots \leq x_m$. Hence, the resource profile x can be characterized by counting how many times a value $t \in \mathcal{T}$ appears in x . Again, we can exploit that $x_m - x_1 \leq p$. There are $|\mathcal{T}|$ possibilities for the value of x_1 and given x_1 , each other entry x_i of x with $2 \leq i \leq m$ can be characterized by the amount $0 \leq y_i = x_i - x_1 \leq p$ by which it is larger than x_1 . Clearly, there are $p + 1$ different possible values for the amount y_i . It follows that given x_1 , we can characterize x by counting how often a value $0 \leq t \leq p$ appears as an amount. Hence, we get that $|\mathcal{X}| \in O(n^2 \cdot m^{p+1})$. This together with Lemma 9 immediately gives us that $P \mid r_j, p_j = p \mid \sum w_j U_j$ can be solved in $m^{O(p)} \cdot n^{O(1)}$ time. ◀

Lastly, we remark that with similar alternative running time analyses for the dynamic programming algorithm by Baptiste et al. [2, 3], one can show that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in XP when parameterized by the number of release dates or due dates, and that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the combination of the number of machines and the number of release dates or due dates. However, as we show in the next section, we can obtain fixed-parameter tractability by parameterizing only by the number of release dates or parameterizing only by the number of due dates.

5 FPT-Algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$

In this section, we present a new FPT algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ parameterized by the number of release dates or due dates. Formally, we show the following.

► **Theorem 10.** *$P \mid r_j, p_j = p \mid \sum w_j U_j$ can be solved in $2^{O(2^{r\#} \cdot r\#)} \cdot n^{O(1)}$ time and in $2^{O(2^{d\#} \cdot d\#)} \cdot n^{O(1)}$ time.*

Theorem 10 implies that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in FPT when parameterized by the number $r\#$ of different release dates and when parameterized by the number $d\#$ of different due dates. We prove the first running time upper-bound of Theorem 10, where we parameterize by the number $r\#$ of release dates. By Observation 1, the case for the number $d\#$ of deadlines is symmetric. We present a reduction from $P \mid r_j, p_j = p \mid \sum w_j U_j$ to MIXED INTEGER LINEAR PROGRAM (MILP).

MIXED INTEGER LINEAR PROGRAM (MILP)

Input: A vector x of n variables, a subset S of the variables which are considered integer variables, a constraint matrix $A \in \mathbb{R}^{m \times n}$, and two vectors $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

Task: Compute an assignment to the variables (if one exists) such that all integer variables in S are set to integer values, $Ax \leq b$, $x \geq 0$, and $c^\top x$ is maximized.

We give a reduction that produces an MILP instance with a small number of integer values. More precisely, the number of integer values will be upper-bounded by a function of the number of release dates of the $P \mid r_j, p_j = p \mid \sum w_j U_j$ instance. This allows us to upper-bound the running time necessary to solve the MILP instance using the following well-known result.

► **Theorem 11** ([8, 29]). *MILP can be solved in $2^{O(n_{int} \log n_{int})} \cdot |I|^{O(1)}$ time, where n_{int} is the number of integer variables and $|I|$ is the instance size.*

Furthermore, we construct the MILP instance in a way that ensures that there always exist optimal solutions where *all* variables are set to integer values. Informally, we ensure that the constraint matrix for the rational variables is totally unimodular¹. This allows us to use the following result.

► **Lemma 12** ([6]). *Let $A_{frac} \in \mathbb{R}^{m \times n_2}$ be totally unimodular. Then for any $A_{int} \in \mathbb{R}^{m \times n_1}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^{n_1+n_2}$, the MILP*

$$\max c^\top x \text{ subject to } (A_{int} \ A_{frac})x \leq b, x \geq 0,$$

where $x = (x_{int} \ x_{frac})^\top$ with the first n_1 variables (i.e., x_{int}) being the integer variables, has an optimal solution where all variables are integer.

Before we describe how to construct an MILP instance for a given instance of $P \mid r_j, p_j = p \mid \sum w_j U_j$, we make an important observation on optimal schedules. Intuitively, we show that we can assume that each job is scheduled as early as possible and idle times only happen directly before release dates.

► **Lemma 13**. *Let σ be a feasible schedule for an instance of $P \mid r_j, p_j = p \mid \sum w_j U_j$ such that the weighted number of early jobs is W . Then there exists a feasible schedule σ' such that*

- *the weighted number of early jobs is W ,*
- *for each job j with $\sigma'(j) = (i, t)$ for some $t \neq r_j$, machine i is not idle at time $t - 1$, and*
- *all starting times of σ' are in the set \mathcal{T} of relevant starting time points.*

Proof. Assume that there is a job j such that $\sigma(j) = (i, t)$ for some $t > r_j$ and machine i is idle at time $t - 1$. Assume that job j is the earliest such job, that is, the job with minimum t . Since machine i is idle at time $t - 1$ and $r_j < t$, we can create a new schedule σ' that is the same as σ except that $\sigma'(j) = (i, t - 1)$. Clearly, we have that σ' is feasible and has the same weighted number of early jobs as σ . By repeating this process, we obtain a feasible schedule σ'' with the same set of early jobs and such that for each job j with $\sigma''(j) = (i, t)$ and machine i is idle at time $t - 1$, it holds that $t = r_j$. Furthermore, we have that each starting point in σ'' is a release date r or a time t with $t = r + \ell \cdot p$ for some release date r and some integer ℓ . Hence, all starting times of σ'' are in the set \mathcal{T} of relevant starting time points. ◀

We call a feasible schedule σ *release date aligned* if the second condition of Lemma 13 holds, i.e., for each job j with $\sigma(j) = (i, t)$ for some $t > r_j$, the machine i is not idle at time $t - 1$. Note that Lemma 13 is stronger than Lemma 2 and implies that there always exists an optimal feasible schedule that is release date aligned.

Given a feasible schedule σ that is release date aligned, we say that a release date r_j is *active* on machine i if job j is scheduled to start at this release date, that is, $\sigma(j) = (i, r_j)$, and machine i is idle at time $r_j - 1$. Let T be a subset of all release dates, then we say that machine i has type T in σ if T is the set of active release dates on machine i .

Recall that $\mathcal{T} = \{t \mid \exists r_j \text{ and } \exists 0 \leq \ell \leq n \text{ s.t. } t = r_j + p \cdot \ell\}$ denotes the set of relevant starting time points. We say that a starting time $t \in \mathcal{T}$ is available on a machine with type T if $t = r + \ell \cdot p$ for some $r \in T$ and $t + p \leq r'$, where r' is the smallest release date in T that is larger than r .

¹ A matrix is *totally unimodular* if each of its square submatrices has determinant 0, 1, or -1 [9].

47:12 Minimizing the Number of Tardy Jobs with Uniform Processing Times

Given an instance I of $P \mid r_j, p_j = p \mid \sum w_j U_j$, we create an instance I' of MILP as follows. For each type T we create an integer variable x_T that quantifies how many machines have type T and create the constraint

$$\sum_T x_T \leq m. \quad (1)$$

For each starting time $t \in \mathcal{T}$ we create a fractional variable x_t that quantifies on how many machines the starting time t is available and create the following set of constraints.

$$\forall t \in \mathcal{T} : x_t = \sum_T t_T \cdot x_T, \quad (2)$$

where $t_T = 1$ if starting time t is available on a machine with type T and $t_T = 0$ otherwise.

For each combination of a job j and a starting time t , we create a fractional variable $x_{j,t}$ if job j can be scheduled to start at time t without violating j 's release date or due date. This variable indicates whether job j is scheduled to start at starting time t and is early. We create the following constraints.

$$\forall t \in \mathcal{T} : \sum_j x_{j,t} \leq x_t. \quad (3)$$

$$\forall j \in \{1, \dots, n\} : \sum_t x_{j,t} \leq 1. \quad (4)$$

Finally, we use the following function as the maximization objective.

$$\sum_{j,t} w_j \cdot x_{j,t} \quad (5)$$

This finishes the construction of the MILP instance I' . We can observe the following.

► **Observation 14.** *Given an instance I of $P \mid r_j, p_j = p \mid \sum w_j U_j$, the above described MILP instance I' can be computed in $O(2^{r\#} \cdot (m+n)^2)$ time and has $2^{r\#}$ integer variables.*

In the following, we prove the correctness of the reduction to MILP. We start with showing that if the $P \mid r_j, p_j = p \mid \sum w_j U_j$ instance admits a feasible schedule where the weighted number of early jobs is W , then the constructed MILP instance admits a feasible solution that has objective value W .

► **Lemma 15.** *If the $P \mid r_j, p_j = p \mid \sum w_j U_j$ instance I admits a feasible schedule where the weighted number of early jobs is W , then the MILP instance I' admits a feasible solution that has objective value W .*

Proof. Assume that we are given a feasible schedule σ for I such that the weighted number of early jobs is W . By Lemma 13 we can assume that σ is release date aligned.

We construct a solution for I' as follows. Consider job j and let $\sigma(j) = (i, t)$ such that $t + p \leq d_j$, that is, job j is early. We know that $t \in \mathcal{T}$. We set $x_{j,t} = 1$ and for all $t' \in \mathcal{T}$ with $t' \neq t$, we set $x_{j,t'} = 0$. Note that this guarantees that Constraints (4) are fulfilled. Furthermore, assuming we can set the remaining variables to values such that the remaining constraints are fulfilled, we have that the objective value of the solution to I' is W .

In the remainder, we show how to set the remaining variables such that all constraints are fulfilled. Initially, we set all variables x_T to zero. Next, we determine the type of each machine. Consider machine i . Then $T := \{r \mid \exists j \text{ s.t. } \sigma(j) = (i, r_j)\}$ is the type of machine i . Then we increase x_T by one. We do this for every machine. Clearly, afterwards Constraint (1) are fulfilled.

Next, we set $x_t := \sum_T t_T \cdot x_T$, where $t_T = 1$ if starting time t is available on a machine with type T and $t_T = 0$ otherwise. Clearly, this fulfills Constraints (2). It remains to show that Constraints (3) are fulfilled. So consider some time $t \in \mathcal{T}$. For each job j starting at time t on some machine i , we increased x_T by one for some T with $t_T = 1$ when processing machine i . Thus, we have $\sum_j x_{j,t} \leq \sum_T t_T \cdot x_T = x_t$. ◀

Before we continue with the other direction of the correctness, we prove that we can apply Lemma 12 to show that the MILP instance I' admits an optimal solution where all variables are set to integer values.

► **Lemma 16.** *The MILP instance I' admits an optimal solution where all variables are set to integer values. Such a solution can be computed in $2^{O(2^r \cdot r \cdot \#)} \cdot n^{O(1)}$ time.*

Proof. Notice that since the Constraints (2) are equality constraints, we have that in any feasible solution to I' , all variables x_t are set to integer values. Hence, I' is equivalent to the MILP I'' arising from I' by declaring x_t to be integer variables for every $t \in \mathcal{T}$ (in addition to the variables x_T), and it suffices to show that I'' has an integer solution.

We show that the constraint matrix for the fractional variables $x_{j,t}$ in I'' (i.e., the variables $x_{j,t}$) is totally unimodular. By Lemma 12 this implies that I'' and therefore also I' admits an optimal solution where all variables are set to integer values.

Note that the Constraints (3) partition the set of fractional variables $x_{j,t}$, that is, each fractional variable is part of exactly one of the Constraints (3). The same holds for the Constraints (4). Furthermore, the coefficients in the constraint matrix for each variable are either 1 (if they are part of a constraint) or 0. Hence, we have that the constraint matrix is a 0-1 matrix with exactly two 1's in every column. Moreover, in each column, one of the two 1's appears in a row corresponding to the Constraints (3) and the other 1 is in a row corresponding to the Constraints (4). This is a sufficient condition for the constraint matrix to be totally unimodular [9].

Finally, we argue that an optimal integer solution for I'' can be computed in the claimed running time upper-bound. We use the algorithm implicitly described by Chaudhary et al. [6] in their proof for Lemma 12. First, we compute an optimal solution for I' . By the arguments at the beginning of this proof, this is also an optimal solution for I'' . To transform this optimal solution into an optimal solution where every variable is set to an integer value, we fix all integer variables, resulting in an LP² instance I''' whose variables are precisely the fractional variables from I'' . Since the constraint matrix of this LP I''' is totally unimodular (as argued above), it is well-known that an optimal solution for I''' where all variables are set to integer values can be computed in polynomial time, see e.g. Korte and Vygen [24, Theorem 4.18]. Combining this integral optimal solution for I''' with the integral variables from I'' then yields an optimal solution for I'' . Now, with Theorem 11 and Observation 14, we get the claimed overall running time upper-bound. ◀

Now we proceed with showing that if the constructed MILP instance admits an optimal solution that has objective value W , then the original $P \mid r_j, p_j = p \mid \sum w_j U_j$ instance admits a feasible schedule where the weighted number of early jobs is W .

► **Lemma 17.** *If the MILP instance I' admits an optimal solution where all variables are set to integer values and that has objective value W , then the $P \mid r_j, p_j = p \mid \sum w_j U_j$ instance I admits a feasible schedule σ where the weighted number of early jobs is W . The schedule σ can be computed from the optimal solution to I' in polynomial time (in the size of I').*

² LINEAR PROGRAM (LP) is the special case of MILP where no variables are considered integer variables (that is, the set S in the definition of MILP is empty).

47:14 Minimizing the Number of Tardy Jobs with Uniform Processing Times

Proof. Assume we are given an optimal solution for I' where all variables are set to integer values and that has objective value W . We construct a feasible schedule σ as follows.

First, we assign a type to every machine. Iterate through the types T and, initially, set $i = 1$. If $x_T > 0$, then assign type T to machines i to $i + x_T - 1$. Afterwards, increase i by x_T . Since the solution to I' fulfills Constraint (1), we know that this procedure does not assign types to more than m machines.

Now iterate through the relevant starting times $i \in \mathcal{T}$. Let $J_t = \{j \mid x_{j,t} = 1\}$ and let $M_t = \{i \mid \text{starting time } t \text{ is available on machine } i\}$. By Constraints (2) and (3) we know that $|J_t| \leq |M_t| = x_t$. Hence, we can create a feasible schedule by setting $\sigma(j) = (i, t)$ for every job $j \in J_t$, where $i \in M_t$ and for all $j, j' \in J_t$ with $j \neq j'$ we have $\sigma(j) = (i, t)$ and $\sigma(j') = (i', t)$ with $i \neq i'$. In other words, we schedule each job in J_t to a distinct machine $i \in M_t$ with starting time t . The Constraints (4) ensure that we schedule each job at most once. For all jobs j that are not contained in any set J_t with $t \in \mathcal{T}$, we schedule j to an arbitrary machine i to a starting time that is later than r_j and later than the completion time of the last job scheduled on i . Clearly, we can compute σ in time polynomial in $|I'|$. By the definition of available starting times and the fact that we only create variable $x_{j,t}$ if $t \geq r_j$, this schedule is feasible.

It remains to show that the weighted number of early jobs is W . To this end, note that we only create variable $x_{j,t}$ if $r_j \leq t$ and $t + p \leq d_j$. Hence, for each job j with $x_{j,t} = 1$ for some $t \in \mathcal{T}$, we know that this job is early in the constructed schedule σ . It follows that the weighted number of early jobs is $\sum_{j,t} w_j \cdot x_{j,t}$, which equals the maximization objective of I and hence equals W . ◀

Now we have all the pieces to prove Theorem 10.

Proof of Theorem 10. Given an instance I of $P \mid r_j, p_j = p \mid \sum w_j U_j$ we create an MILP instance I' as described above and use Lemma 16 to compute an optimal solution for I where all variables are set to integer values. Lemmas 15 and 17 show that we can correctly compute an optimal schedule for I from the solution to I' in polynomial time (in the size of I'). Observation 14 together with Lemma 16 show that this algorithm has the claimed running time upper-bound. ◀

6 Conclusion and Future Work

In this work, we resolved open questions by Kravchenko and Werner [25], Sgall [36], and Mních and van Bevern [32] by showing that $P \mid r_j, p_j = p \mid \sum U_j$ is NP-hard and W[2]-hard when parameterized by the number of machines. The established hardness of the problem motivates investigating it from the viewpoint of exact parameterized or approximation algorithms. In this work, we focussed on the former, leaving the latter for future research. We provided a first step in systematically exploring the parameterized complexity of $P \mid r_j, p_j = p \mid \sum w_j U_j$. Our parameterized hardness result shows that the known XP-algorithm for the number of machines as a parameter is optimal from a classification standpoint. Furthermore, we showed that this known algorithm implies that the problem is also contained in XP when parameterized by the processing time, and that it is contained in FPT when parameterized by the combination of the number of machines and the processing time. Finally, we give an FPT-algorithm for $P \mid r_j, p_j = p \mid \sum w_j U_j$ parameterized by the number of release dates (or due dates). We leave several questions open, the most interesting one is the following.

- Is $P \mid r_j, p_j = p \mid \sum w_j U_j$ in FPT or W[1]-hard when parameterized by the processing time p of any job?

Other interesting parameters to consider might be the number of early jobs or the number of tardy jobs. It is easy to see that $P \mid r_j, p_j = p \mid \sum w_j U_j$ is in XP when parameterized by either one of those parameters, by some simple guess-and-check algorithm (recall that we can check in polynomial time whether all jobs can be scheduled early [5, 37, 38]). Hence, it remains open whether the problem is in FPT or W[1]-hard with respect to those parameters.

References




- 1 Esther M. Arkin and Ellen B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, 1987. doi:10.1016/0166-218X(87)90037-0.
- 2 Philippe Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103(1-3):21–32, 2000. doi:10.1016/S0166-218X(99)00238-3.
- 3 Philippe Baptiste, Peter Brucker, Sigrid Knust, and Vadim G Timkovsky. Ten notes on equal-processing-time scheduling: at the frontiers of solvability in polynomial time. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(2):111–127, 2004.
- 4 Karl Bringmann, Nick Fischer, Danny Hermelin, Dvir Shabtay, and Philip Wellnitz. Faster minimization of tardy processing time on a single machine. *Algorithmica*, 84(5):1341–1356, 2022. doi:10.1007/S00453-022-00928-W.
- 5 Peter Brucker and Svetlana A. Kravchenko. Scheduling jobs with equal processing times and time windows on identical parallel machines. *Journal of Scheduling*, 11(4):229–237, 2008. doi:10.1007/S10951-008-0063-Y.
- 6 Juhi Chaudhary, Hendrik Molter, and Meirav Zehavi. Parameterized analysis of bribery in challenge the champ tournaments. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2704–2712. ijcai.org, 2024. URL: <https://www.ijcai.org/proceedings/2024/299>.
- 7 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.
- 8 Daniel Dadush, Chris Peikert, and Santosh Vempala. Enumerative lattice algorithms in any norm via M -ellipsoid coverings. In *Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 580–589. IEEE, 2011. doi:10.1109/FOCS.2011.31.
- 9 George Bernard Dantzig. *Linear inequalities and related systems*. Number 38. Princeton University Press, 1956.
- 10 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. doi:10.1007/978-1-4612-0515-9.
- 11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. doi:10.1007/978-1-4471-5559-1.
- 12 Nick Fischer and Leo Wennmann. Minimizing tardy processing time on a single machine in near-linear time. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297 of *LIPICs*, pages 64:1–64:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.64.
- 13 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 2006. doi:10.1007/3-540-29953-X.
- 14 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 15 Ronald L. Graham, Eugene L. Lawler, Jan K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. Elsevier, 1979.
- 16 Klaus Heeger and Danny Hermelin. Minimizing the weighted number of tardy jobs is W[1]-hard. In *Proceedings of the 32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPICs*, pages 68:1–68:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ESA.2024.68.

47:16 Minimizing the Number of Tardy Jobs with Uniform Processing Times

- 17 Klaus Heeger, Danny Hermelin, George B Mertzios, Hendrik Molter, Rolf Niedermeier, and Dvir Shabtay. Equitable scheduling on a single machine. *Journal of Scheduling*, 26(2):209–225, 2023. doi:10.1007/S10951-022-00754-6.
- 18 Danny Hermelin, Yuval Itzhaki, Hendrik Molter, and Dvir Shabtay. On the parameterized complexity of interval scheduling with eligible machine sets. *Journal of Computer and System Sciences*, page 103533, 2024. doi:10.1016/J.JCSS.2024.103533.
- 19 Danny Hermelin, Shlomo Karhi, Michael L. Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298(1):271–287, 2021. doi:10.1007/S10479-018-2852-9.
- 20 Danny Hermelin, Hendrik Molter, and Dvir Shabtay. Minimizing the weighted number of tardy jobs via (max,+)-convolutions. *INFORMS Journal on Computing*, 2023.
- 21 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 22 Matthias Kaul, Matthias Mnich, and Hendrik Molter. Single-machine scheduling to minimize the number of tardy jobs with release dates. In *Proceedings of the 19th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 321 of *LIPICs*, pages 19:1–19:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.IPEC.2024.19.
- 23 Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ilps. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2947–2960. SIAM, 2023. doi:10.1137/1.9781611977554.CH112.
- 24 Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer, 6th edition edition, 2018.
- 25 Svetlana A Kravchenko and Frank Werner. Parallel machine problems with equal processing times: a survey. *Journal of Scheduling*, 14:435–444, 2011. doi:10.1007/S10951-011-0231-3.
- 26 Sven O Krumke, Clemens Thielen, and Stephan Westphal. Interval scheduling on related machines. *Computers & Operations Research*, 38(12):1836–1844, 2011. doi:10.1016/J.COR.2011.03.001.
- 27 Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- 28 Eugene L. Lawler and James M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16(1):77–84, 1969.
- 29 Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983. doi:10.1287/MOOR.8.4.538.
- 30 Jan K. Lenstra, A.H.G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
- 31 LL Liu, Chi To Ng, and TC Edwin Cheng. Bicriterion scheduling with equal processing times on a batch processing machine. *Computers & Operations Research*, 36(1):110–118, 2009. doi:10.1016/J.COR.2007.07.007.
- 32 Matthias Mnich and René van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, 2018. doi:10.1016/J.COR.2018.07.020.
- 33 James M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15:102–109, 1968.
- 34 Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems, 5th Edition*. Springer, 2016.
- 35 Baruch Schieber and Pranav Sitaraman. Quick minimization of tardy processing time on a single machine. In *Proceedings of the 18th International Symposium on Algorithms and Data Structures Symposium (WADS)*, volume 14079 of *Lecture Notes in Computer Science*, pages 637–643. Springer, 2023. doi:10.1007/978-3-031-38906-1_42.

- 36 Jiri Sgall. Open problems in throughput scheduling. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, volume 7501 of *Lecture Notes in Computer Science*, pages 2–11. Springer, 2012. doi:10.1007/978-3-642-33090-2_2.
- 37 Barbara B. Simons. Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines. *SIAM Journal on Computing*, 12(2):294–299, 1983. doi:10.1137/0212018.
- 38 Barbara B. Simons and Manfred K. Warmuth. A fast algorithm for multiprocessor scheduling of unit-length jobs. *SIAM Journal on Computing*, 18(4):690–710, 1989. doi:10.1137/0218048.
- 39 Shao Chin Sung and Milan Vlach. Maximizing weighted number of just-in-time jobs on unrelated parallel machines. *Journal of Scheduling*, 8(5):453–460, 2005. doi:10.1007/S10951-005-2863-7.

Subshifts Defined by Nondeterministic and Alternating Plane-Walking Automata

Benjamin Hellouin de Menibus   

Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France

Pacôme Perrotin   

Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France

*Il voulut être sofic,
il ne fut que pompé.*

Abstract

Plane-walking automata were introduced by Salo & Törma to recognise languages of two-dimensional infinite words (subshifts), the counterpart of 4-way finite automata for two-dimensional finite words. We extend the model to allow for nondeterminism and alternation of quantifiers. We prove that the recognised subshifts form a strict subclass of sofic subshifts, and that the classes corresponding to existential and universal nondeterminism are incomparable and both larger than the deterministic class. We define a hierarchy of subshifts recognised by plane-walking automata with alternating quantifiers, which we conjecture to be strict.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Tree languages; Theory of computation → Regular languages

Keywords and phrases Formal languages, Finite automata, Subshifts, Symbolic dynamics, Tilings

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.48

Funding The authors acknowledge financial support from the ANR-22-CE40-0011 project Inside Zero Entropy Systems.

Acknowledgements Many people have contributed to this article through discussions, suggestions and failed attempts; we wish to thank (in alphabetical order) Florent Becker, Amélia Durbec, Pierre Guillon and Guillaume Theyssier. We are grateful to Ville Salo for providing the proof of Theorem 8 and pointing us towards the Kari-Moore rectangles of Section 5.3, and to Denis Kuperberg and Thomas Colcombet for pointing us towards works on tree-walking automata cited in Section 5.2.

1 Introduction

One-dimensional finite automata are a classical model to recognise languages of finite words. They have been extended to recognise languages of finite patterns in two and more dimensions, often called *picture languages*, starting with the work of Blum and Hewitt in 1967 [3]. While the one-dimensional model is very robust to changes in definition, this is not the case in higher dimensions and many different models have been introduced with varying computational power; see [11] for a survey that focuses on the non-deterministic case.

Symbolic dynamics are concerned with subshifts, which are languages of infinite words or patterns. In dimension 1, sofic subshifts can be seen as the infinite counterparts to regular languages, and have three equivalent definitions: the set of infinite walks on a labelled graph (finite automaton without initial nor final states); the set of infinite words avoiding a regular set of forbidden subwords; the letter-to-letter projection of a subshift of finite type. The latter definition carries through to higher dimensions without difficulties to define higher-dimensional sofic subshifts.



© Benjamin Hellouin de Menibus and Pacôme Perrotin;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 48; pp. 48:1–48:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



There are many ways to extend regular languages to higher dimensions. Some models such as *4-way automata* keep the notion of a linear *run* where the head reads the input pattern, letter by letter, by moving (“walking”) over the pattern; this contrasts with models such as *recognisable languages* where acceptance consists in checking local constraints instead of a run. Broadly speaking, the latter models are more powerful than the former, and the same phenomenon arises for languages on trees: tree automata vs. tree-walking automata. [7] provides a catalogue of the different kinds of models, while a more recent survey on the “walking” models can be found in [11].

When applying these models to subshifts, recognisable languages yields an alternative definition of sofic subshifts; this was first done in [8] to our knowledge, and we present this construction in Section 2.3. Recently, Salo and Törma introduced *plane-walking automata* (PWA) [14], a particular case of graph-walking automata, which are a counterpart of 4-way automata for infinite patterns. In particular, they proved that deterministic plane-walking automata define a class of subshifts that is strictly between subshifts of finite type and sofic subshifts, extending to infinite patterns a result from [9] on 4-way automata. It is also proved in [9] that, for finite patterns, nondeterministic 4-way automata are strictly more powerful than the deterministic version and that languages of alternating 4-way automata are incomparable with recognisable languages, a sharp contrast with the one-dimensional case.

In this paper, we introduce nondeterminism and alternations to plane-walking automata and consider the classes of higher-dimensional subshifts obtained this way. We prove that subshifts accepted by Σ_1 PWA (existential nondeterminism) and Π_1 PWA (universal nondeterminism) form incomparable classes (Theorem 14), both strictly larger than the deterministic case, and that subshifts accepted by arbitrary alternating PWA still form a strict subclass of sofic subshifts (Theorem 8). This yields two new classes of higher-dimensional subshifts that are intermediate between finite type and sofic subshifts, and natural in that they generalise one-dimensional sofic subshifts. We introduce an alternating hierarchy of nondeterministic power from deterministic up to unbounded alternating plane-walking automata, and conjecture that this hierarchy of subshifts does not collapse; to our knowledge, this is not known even for finite words. Our proofs involve equivalents of the pumping lemma for two-dimensional infinite patterns.

Definitions and results are written with the two-dimensional case (\mathbb{Z}^2) in mind, although they extend easily to any higher dimension, and our definitions also extend to any finitely generated groups (see [15]).

2 Configurations and Subshifts

2.1 Symbolic Dynamics

We call *positions* the elements of \mathbb{Z}^2 , which is endowed with the Manhattan distance d . We use $\rightarrow = (1, 0)$ and $\uparrow = (0, 1)$, $\bullet = (0, 0)$ and so on; we often write 0 as a position instead of $(0, 0)$.

Let Σ be a finite set called *alphabet*. A *configuration* x is an element of $\Sigma^{\mathbb{Z}^2}$, while a *pattern* p is an element of Σ^S for some finite $S \subset \mathbb{Z}^2$, called the support of p and denoted $\text{supp}(p) = S$. For $i \in \mathbb{Z}^2$, $\sigma^i(x)$ is a new configuration shifted by i , i.e. $(\sigma^i(x))_j = x_{i+j}$. In dimension one, we call a pattern a *word*.

Given $\pi : \Sigma' \rightarrow \Sigma$, we extend π to $\Sigma'^{\mathbb{Z}^2}$ by putting $\pi(x)_p = \pi(x_p)$ for all $p \in \mathbb{Z}^2$.

A *subshift* X is a set of configurations that is closed in the product topology and invariant by all shifts; more concretely, it is defined as the set of configurations where no pattern from a set of forbidden patterns \mathcal{F} appears.

A subshift defined by a finite set of forbidden patterns is called *of finite type* (SFT for short). By a standard technique of higher block coding, replacing the alphabet Σ by $\Sigma^{[0,n] \times [0,n]}$ for n large enough, we can assume without loss of generality that forbidden patterns all have support $\{\bullet, \rightarrow\}$ or $\{\bullet, \uparrow\}$, and we do throughout the paper.

A subshift that can be written as $\pi(X)$, where X is an SFT on alphabet Σ' and $\pi : \Sigma' \rightarrow \Sigma$, is called *sofic*. In dimension one, sofic subshifts have alternative definitions as the set of bi-infinite walks on some labelled graph or as the set of bi-infinite words avoiding some regular set of forbidden words.

We denote the classes of SFT and sofic subshifts SFT and Sofic, respectively.

2.2 Two-dimensional Automata

We define an abstract model of two-dimensional automata. We use different notions of acceptance in the paper, and add additional restrictions to the model as necessary.

► **Definition 1.** *A two-dimensional automaton on \mathbb{Z}^2 is a labelled directed graph $A = (V, E, \Sigma, D, I)$, where*

- *V and E are finite sets of vertices and edges, respectively, where $E \subset V^2 \times \mathbb{Z}^2$ (we call the second component the direction);*
- *Σ is a finite alphabet and $D : V \rightarrow \Sigma$ associates a symbol to each vertex.*
- *$I \in V$ are the initial states and D is a bijection from I to Σ . For any $a \in \Sigma$, we denote i_a the only state in I such that $D(i_a) = a$.*

If the automaton is alternating, then there is additionally a map $Q : V \rightarrow \{\forall, \exists\}$ associating a quantifier to each vertex.

Notice that since E is finite, the set of possible directions is a finite subset of \mathbb{Z}^2 .

2.3 Recognisable Picture Languages

Recognisable languages, as introduced in [6], are a possible extension of regular languages to higher-dimensional words (or *picture languages*) as projections of local languages, which can be expressed using two-dimensional automata.

The same model applied on subshifts yields an alternative definition of sofic subshifts using two-dimensional automata.

► **Definition 2.** *Let A be a non-alternating two-dimensional automaton whose set of directions is $\{\uparrow, \rightarrow\}$. Given a pattern or a configuration x , an recognising run of A on x is a function $r : \text{supp}(x) \rightarrow V$ such that:*

- *for all $i \in \text{supp}(x)$, $D(r(i)) = x_i$;*
- *for all $i \in \text{supp}(x)$ such that $i + \rightarrow \in \text{supp}(x)$, there is an edge $(r(i), r(i + \rightarrow), \rightarrow) \in E$, and similarly for \uparrow .*

Then:

- *the recognised language $R_f(A)$ is the set of all patterns that admit a recognising run.*
- *the recognised subshift $R_\infty(A)$ is the set of all configurations that admit a recognising run.*

Notice that this definition is intrinsically nondeterministic in the choice of the run and makes no use of initial states, so we omit I in this section. The first equivalence of the following result appeared in [8, Proposition 7]; we provide a short proof in our framework.

► **Proposition 3.** *For a subshift X , the following are equivalent:*

1. X is sofic;
2. $X = R_\infty(A)$ for some automaton A ;
3. X is defined by a set of forbidden patterns $R_f(A)^c$ for some automaton A , where A is assumed to satisfy the hypotheses of Definition 2.

Proof. (1 \Rightarrow 2) Let Y be a SFT on alphabet Σ' given by a finite set \mathcal{F} of forbidden patterns and $\pi : \Sigma' \rightarrow \Sigma$ be such that $\pi(Y) = X$; remember that we assume that patterns in \mathcal{F} have support $\{\bullet, \uparrow\}$ or $\{\bullet, \rightarrow\}$. We define $A = (V, E, \Sigma', D)$ by setting $V = \Sigma'$ and $D = \pi$, and E is defined as follows: for all $p \in \Sigma'^{\{\bullet, \uparrow\}}$, $(p_\bullet, p_\uparrow, \uparrow) \in E$ if and only if $p \notin \mathcal{F}$, and similarly for \rightarrow . By construction, $y \in \Sigma'^{\mathbb{Z}^2}$ is a recognising run of A on x if and only if $y \in Y$ and $\pi(y) = x$, so $X = R_\infty(A)$.

(2 \Rightarrow 1). Let $A = (V, E, \Sigma', D)$ be an automaton and define the SFT $Y \subset V^{\mathbb{Z}^2}$ by the set of forbidden patterns:

$$\mathcal{F} = \{p \in V^{\{\bullet, \rightarrow\}} : (p_\bullet, p_\rightarrow, \rightarrow) \notin E\} \cup \{p \in V^{\{\bullet, \uparrow\}} : (p_\bullet, p_\uparrow, \uparrow) \notin E\}.$$

Again by construction, y is a recognising run of A on x if and only if $y \in Y$ and $D(y) = x$, so $X = D(Y)$ is sofic.

(2 \Leftrightarrow 3) The two statements are equivalent for any automaton A . If $x \in R_\infty(A)$ has a recognising run, then the restriction of this run to any finite pattern in x is recognising, so all patterns in x belong to $R_f(A)$. Conversely, if all patterns in x admit a recognising run, then x does as well by a standard compactness argument. ◀

Notice that the third condition involves complementation, in contrast with the one-dimensional case; recognisable languages are not closed by complement in dimension 2 [1].

3 Plane-walking Automata and Associated Subshifts

Plane-walking automata generalise the definition of one-dimensional sofic subshifts seen as the set of infinite walks on a labelled graph.

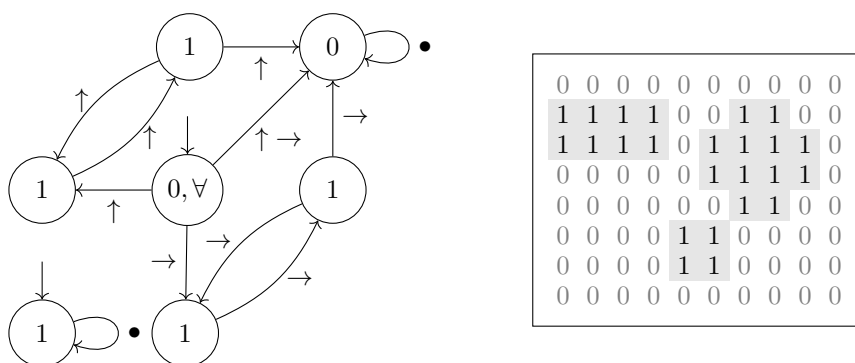
3.1 Definitions

The automata we use in this section correspond to the plane-walking automata (PWA) from Salo and Törnä [14] with one head and added nondeterminism. They are alternating two-dimensional automata with a specific acceptance condition, that we call alternating plane-walking automata to be consistent with the literature.

► **Definition 4** (Subshifts defined by plane-walking automata). *Let $A = (V, E, \Sigma, D, I, Q)$ be an alternating PWA. Given $x \in \Sigma^{\mathbb{Z}^2}$, $p \in \mathbb{Z}^2$ and $v \in V$, there is an accepting run on x starting from (p, v) if $D(v) = x_p$ and:*

- $Q(v) = \exists$ and there is an edge $(v, v', d) \in E$ and an accepting run starting from $(p + d, v')$, or
- $Q(v) = \forall$ and all edges $(v, v', d) \in E$ with $D(v') = x_{p+d}$ have an accepting run starting from $(p + d, v')$; furthermore, there must be one such edge.

A configuration x is accepted by A if, for every $p \in \mathbb{Z}^2$, there is an accepting run of A on x starting from (p, i_{x_p}) . The set of configurations accepted by A is a subshift, denoted $L_\infty(A)$.



■ **Figure 1** On the left, an example of an alternating plane-walking automaton. On the right, a finite pattern of a configuration accepted by the automaton. The associated subshift is the subshift where all vertical and horizontal runs of 1s are either of even size or infinite.

The lack of a base case in the previous definition means that an accepting run must be infinite; in other words, a run accepts if it never reaches a position and state with no possible move. The model of Salo and Törma used rejecting states, which we opted not to do by symmetry w.r.t the lack of accepting states; this is a stylistic choice as a rejecting state can be simulated by a state with no outgoing edge, and an accepting state by a state with a loop with direction \bullet .

An alternating plane-walking automaton and its associated subshift are illustrated in Figure 1. Without loss of generality, by adding additional states, the set of directions can be restricted to $\{\uparrow, \downarrow, \leftarrow, \rightarrow, \bullet\}$ which we assume in the rest of this article.

We denote Alt the class of subshifts X such that $X = L_\infty(A)$ for some alternating PWA A . It is not difficult, as in Proposition 3, to extend the notion of acceptance to finite patterns (considering a run as accepting when it leaves the support of the pattern), where it coincides with alternating 4-way picture-walking automata. Subshifts in Alt can equivalently be defined by a set of forbidden patterns that are the complement of the language of some alternating PWA. As for recognisable languages, these languages are not closed by complement [11, Theorem 7].

Plane-walking automata as considered by Salo & Törma [14] are deterministic, in the sense that there is only one outgoing edge from each state on which the run does not fail immediately; therefore the quantifiers in the definition are unused. We denote by Δ_1 the class of subshifts defined by such deterministic plane-walking automata, and we call deterministic every state where the quantifier is not relevant, so we omit it in the pictures for clarity.

► **Definition 5** (Branch, Footprint). *A run on x can be represented by a tree where each vertex corresponds to a current position and state.*

A branch of a run is a branch in that tree, in the usual sense.

The footprint of a subtree or a branch is the set of all visited positions.

3.2 Comparison with SFT and Sofic Subshifts

We show that subshifts defined by alternating plane-walking automata are an intermediate class between the two classical classes of SFT and sofic subshifts. Notice that, in dimension one, the classical powerset construction tells us that added nondeterminism does not impact the power of the finite automata, so every class defined in this article coincide with sofic subshifts.

The next result is proved in [14] (Inclusion is stated without proof, Strictness is Lemma 1; Inclusion also follows from Lemma 15 below).

► **Proposition 6.** $SFT \subsetneq \Delta_1 \subset Alt$.

The following result appeared in [9] (Theorem 1) for finite patterns. We translate the proof in our framework since our statements differ due to differing models.

► **Proposition 7.** $Alt \subset Sofic$.

Proof. Let $A = (V, E, \Sigma, D, I, Q)$ be an alternating PWA. Let $\Sigma' = \Sigma \times \mathcal{P}(V)$, where a symbol from Σ' encodes the set of states starting from which there is an accepting run in the current position. Let π_1 and π_2 be the projections on the first and second component, respectively.

We define a SFT $Y \subset \Sigma'^{\mathbb{Z}^2}$ by the following set \mathcal{F} of forbidden patterns: for $p \in \Sigma'^{\{\leftarrow, \rightarrow, \uparrow, \downarrow, \bullet\}}$, denote $p_\bullet = (s, S)$ where $s \in \Sigma$ and $S \in \mathcal{P}(V)$. Then $p \in \mathcal{F}$ if and only if:

1. $\exists v \in S, D(v) \neq s$, or
2. $S \cap I = \emptyset$, or
3. there is $v \in S$ such that
 - $Q(v) = \exists$ and for all $(v, v', d) \in E$, we have $p_d = (s', S')$ with $v' \notin S'$.
 - $Q(v) = \forall$ and there is $(v, v', d) \in E$ such that $p_d = (s', S')$ with $v' \notin S'$.

We claim that $L_\infty(A) = \pi_1(Y)$.

Given a configuration $y \in Y$, we prove inductively the following statement: there is an accepting run starting from (i, v) on $\pi_1(y)$ for all $i \in \mathbb{Z}^2$ and $v \in \pi_2(y_i)$. If $Q(v) = \exists$, then Condition 3 ensures we find an edge (v, v', d) with $v' \in \pi_2(y_{i+d})$. Condition 1 and iterating this argument yields an accepting run starting from $(i + d, v')$. A similar argument works for \forall . Condition 2 ensures that there is an initial state in $\pi_2(y_i)$, so $\pi_1(y)$ is accepted by A .

Conversely, given a configuration $x \in L_\infty(A)$, we define $y_i = (x_i, S_i)$ where $S_i \subset V$ is the set of states v such that there is an accepting run on x from (i, v) . Since x admits an accepting run from any position for some initial state, it is easy to see that all three conditions above are satisfied and $y \in Y$. ◀

The proof of the following result is due to Ville Salo (personal communication).

► **Theorem 8.** $Alt \subsetneq Sofic$.

► **Definition 9.** Given a one-dimensional subshift $X \subset \Sigma^{\mathbb{Z}}$, its two-dimensional lift is defined as

$$X^\uparrow = \{y \in \Sigma^{\mathbb{Z}^2} : \exists x \in X, \forall i, j, y_{i,j} = x_i\}.$$

By the constructions of Aubrun-Sablik [2] or Durand-Romaschenko-Shen [5], the lift of any one-dimensional subshift given by a computable set of forbidden patterns is a sofic subshift.

► **Lemma 10.** If X^\uparrow is accepted by an alternating PWA, then X is sofic.

Proof. Let A be the automaton that accepts X^\uparrow in the sense that $X^\uparrow = L_\infty(A)$, and A' be the automaton obtained by replacing every direction \uparrow or \downarrow by \bullet in A .

Since every configuration in X^\uparrow is constant in the vertical direction, A' accepts every configuration in X^\uparrow (as well as additional configurations that are not constant vertically). Since A' only travels horizontally, it can be seen as a two-way one-dimensional automaton

that accepts exactly the one-dimensional configurations that lift to X^\uparrow ; in other words, A' accepts X . It follows that X is defined by a regular set of forbidden patterns, so it is sofic. ◀

To prove Theorem 8, see that the lift X^\uparrow of any non-sofic one-dimensional subshift X given by a computable set of forbidden patterns is sofic and not accepted by any alternating PWA. It is well-known that such subshifts exist; see e.g. [12, Example 3.1.7].

4 Alternating Hierarchy for Plane-walking Automata

In this paper, we are interested in comparing the power of plane-walking automata with varying access to nondeterminism. We introduce an alternating hierarchy of subshifts, similar to the classic alternating hierarchies of propositional logic formulæ. We are not able to prove that this forms a strict hierarchy of subshifts, but we show that the hierarchy does not collapse at the first level.

4.1 Definitions

Starting from Δ_1 (deterministic) PWA, we define Σ_1 , resp. Π_1 , PWA as the existential, resp. universal, PWA, that is, all states are labelled by \exists quantifiers, resp. \forall quantifiers. We call Σ_1 and Π_1 the corresponding classes of subshifts.

By definition, $\Delta_1 \subset \Sigma_1 \cap \Pi_1$. The rest of this paper is dedicated to show that Σ_1 and Π_1 are incomparable sets (and thus strictly larger than Δ_1). First, we define the whole hierarchy inductively by using automata decompositions.

► **Definition 11** (Decomposition). *A two-dimensional automaton $A = (V, E, \Sigma, D, I, Q)$ has a decomposition $(S, V \setminus S)$ for $S \subseteq V$ if there exists no path from $V \setminus S$ to S in E . By extension, we call decomposition the pair of induced subautomata.*

► **Definition 12** (Σ_n and Π_n). *An alternating plane-walking automaton is Σ_{n+1} if it admits a decomposition into a Σ_1 -automaton and a Π_n -automaton. We denote Σ_{n+1} the class of subshifts X such that $X = L_\infty(A)$ for a Σ_{n+1} -automaton.*

The definitions for Π_{n+1} are symmetrical.

Equivalently, a Σ_n automata is such that the image by Q of any path starting from I is a word of $\exists^* \forall^* \dots$ with n blocks ($n - 1$ alternations). This justifies the following definition:

► **Definition 13** (Δ_n). *An alternating plane-walking automaton is Δ_n if the image by Q of any path starting from I is a word of $\{\exists, \forall\}^*$ with $n - 1$ blocks ($n - 2$ alternations). We denote by Δ_n the class of subshifts X such that $X = L_\infty(A)$ for some Δ_n PWA A .*

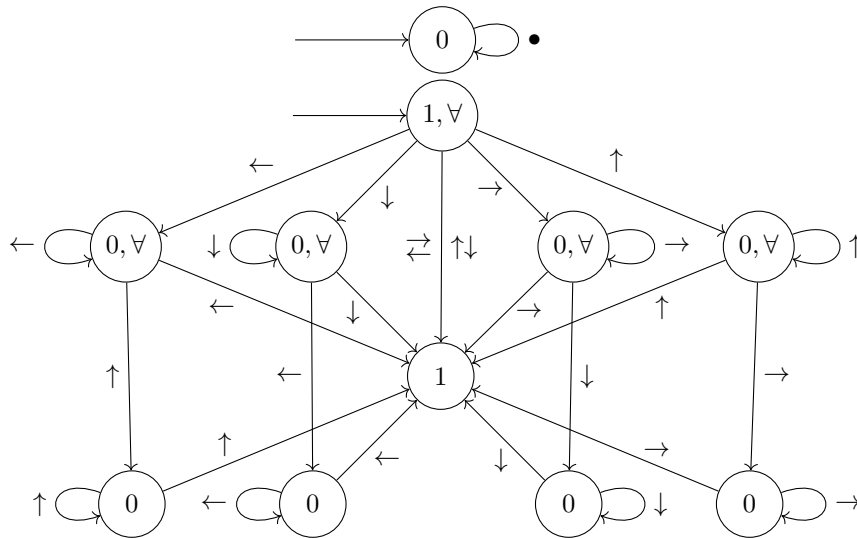
It is not difficult to see that $\Delta_n \subseteq \Sigma_n \subseteq \Delta_{n+1}$ and $\Delta_n \subseteq \Pi_n \subseteq \Delta_{n+1}$. We do not know whether $\Sigma_n \cap \Pi_n = \Delta_n$.

We are ready to state our main result:

► **Theorem 14.** $\Sigma_1 \not\subseteq \Pi_1$, and $\Pi_1 \not\subseteq \Sigma_1$. As a consequence, $\Delta_1 \subsetneq \Sigma_1$ and $\Delta_1 \subsetneq \Pi_1$.

In the next two subsections, we build two subshifts in $\Sigma_1 \setminus \Pi_1$ and $\Pi_1 \setminus \Sigma_1$, respectively. We begin by a technical lemma:

► **Lemma 15.** *Let $X \in \Sigma_n$ be a subshift and Y be a SFT. Then $X \cap Y \in \Sigma_n$. The same holds for Π and Δ .*



■ **Figure 2** A \forall -automaton accepting X_{ssu} . A branch visiting the central state has no next move, so it rejects.

Proof. Given a PWA A , define A' as follows. From the initial state, check the area $\{\bullet, \uparrow, \rightarrow\}$ around the initial position. If a forbidden pattern is found, reject; this is done deterministically. Otherwise, come back to the initial position and execute a run of A . A' belongs to the same class as A and accepts all configurations in $L_\infty(A)$ where no forbidden pattern appears. ◀

4.2 Sunny Side Up

► **Definition 16** (Sunny side up). *The sunny side up subshift X_{ssu} is the set of configurations $x \in \{0, 1\}^{\mathbb{Z}^2}$ with at most one $i \in \mathbb{Z}^2$ such that $x_i = 1$.*

The sunny side up subshift can be easily accepted using a \forall -automaton that has the ability of exploring an unbounded space and rejecting if any “problem” is found in any of the branches.

► **Proposition 17.** $X_{ssu} \in \Pi_1$.

Proof. Let A be the automaton represented in Figure 2. Every run starting from a position containing 0 accepts. Therefore every configuration with no 1 is accepted.

If A starts on a 1 at position i , it nondeterministically picks a quarter-plane to explore and rejects if this branch encounters another 1. A run never visits a given position twice, so if x contains a single 1, all runs accept.

If x contains two 1 at positions i and j , draw a path from i to j using at most two directions in $\{\rightarrow, \uparrow, \leftarrow, \downarrow\}$. This path yields a rejecting run starting from i . ◀

► **Proposition 18.** $X_{ssu} \notin \Sigma_1$.

The following proof is related to Proposition 1 in [14], which only holds for deterministic PWA (and a larger class of subshifts). Intuitively, Σ_1 -automata are ill-fitted to explore unbounded spaces, as a run may reject from a given starting point only if all branches reject, but every branch may only visit a small part of the space.

In the next proof, we use the notation $p \pm K = \{p \pm i : i \in K\}$ for $p \in \mathbb{Z}^2, K \subset \mathbb{Z}^2$.

Proof. Let A be a Σ_1 PWA with C states; we show that $X_{ssu} \neq L_\infty(A)$. For $\vec{v} \in \mathbb{Z}^2$ and $r \in \mathbb{N}$, we denote by $B^+(\vec{v}, r)$ the set $\{j \in \mathbb{Z}^2 : \exists k \in \mathbb{R}^+, d(j, k\vec{v}) \leq r\}$ (for the Manhattan distance on \mathbb{R}^2), and $B(\vec{v}, r) = B^+(\vec{v}, r) \cup -B^+(\vec{v}, r)$. In other words, this is the band of width r around the (half-)line of direction \vec{v} starting from 0; if $\vec{v} = 0$, $B(0, r) = B^+(0, r)$ is a ball of radius r . When r is not indicated, we assume that $r = C$.

Let A be a Σ_1 -automaton that accepts all configurations in X_{ssu} . Define $x \in X_{ssu}$ such that $x_i = 0$ for all i ; $y \in X_{ssu}$ such that $y_0 = 1$ and $y_i = 0$ for all other positions i ; and, for any $p \neq 0$, $z^p \notin X_{ssu}$ such that $z_0^p = z_p^p = 1$ and $z_i^p = 0$ for all other positions i . We find some p such that A accepts z^p , showing that A does not accept X_{ssu} .

To find some accepted z^p , we use the following property: if there is an accepting branch in x that visits neither 0 nor p , the same branch is accepting in z^p . Similarly, if an accepting branch in y does not visit p or an accepting branch in $\sigma^p(y)$ does not visit 0, then the same branch is accepting in z^p .

Let $S_1(r) \subset V \times B(0, r)$ be such that $(s, k) \in S_1$ if and only if there is an accepting run on y from (s, k) . For each $(s, k) \in S_1$, pick an arbitrary accepting branch b . It is described by a sequence $(s_n, p_n)_{n \in \mathbb{N}} \in (V \times \mathbb{Z}^2)^\mathbb{N}$. If b stays in $B(0)$, we find two steps $i < j$ such that $(s_i, p_i) = (s_j, p_j)$ and, by a pumping argument, we build another accepting branch b_p which is eventually periodic and stays in $B(0)$. If b leaves $B(0)$, we find two steps $i < j$ such that $s_i = s_j$ and $d(0, p_j) \geq d(0, p_i)$ and, by a pumping argument, we build another accepting branch b_p which is eventually periodic up to translation by the pumping vector $v^{(s,k)} = p_j - p_i$ every $j - i$ steps; in other words, b_p stays in some band $B(v^{(s,k)})$. Denote $P_1(r) = \{v^{(s,k)} : (s, k) \in S_1(r)\}$ the set of all such pumping vectors.

Let $S_0 \subset V$ be the set of states reachable from i_0 through a path labelled by 0. Since x is accepted, there must be a cycle that stays in S_0 . For any such simple cycle (without repeated states), the associated pumping vector is the sum of all directions. Denote P_0 the finite set of all such pumping vectors.

We distinguish two cases that we illustrate in Figure 3.

All vectors in P_0 are colinear to some v

We choose p such that $p \notin B(v) \cup \bigcup_{v_1 \in P_1(0)} B(v_1)$. This avoids a finite set of one-dimensional subsets, so such a choice is possible.

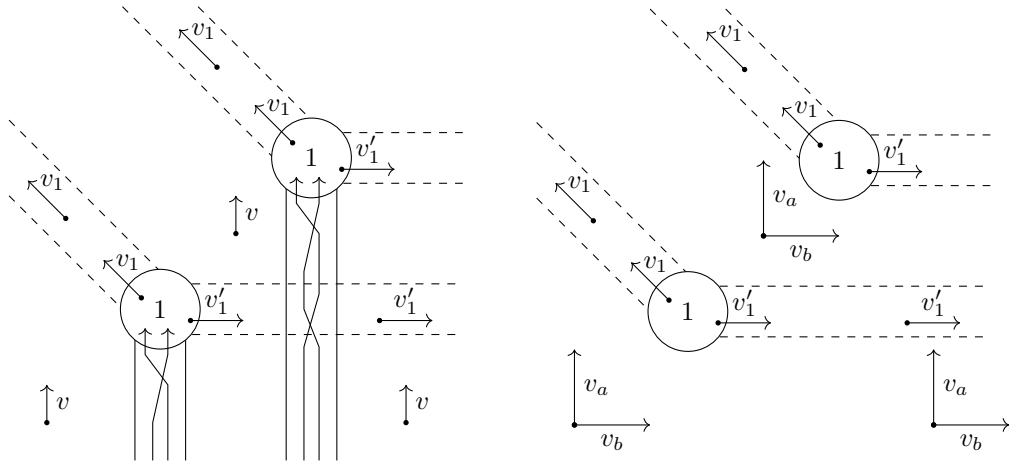
Take any position $k \in \mathbb{Z}^2$ and assume that $k \notin p - B^+(v)$. Pick an accepting branch b in y from k . As long as b never visits the 1 in position 0, it stays in a state of S_0 , so we can pump to build a periodic accepting branch that stays in $k + B^+(v)$ and does not visit p . If b visits 0, then it is in some state s such that $(s, 0) \in S_1(0)$, and so we can pump to build another accepting branch b_p that stays in $B(0)$ or in $B^+(v_1)$ for some $v_1 \in P_1(0)$. Either way, we built an accepting branch in y from k that does not visit p , so it is accepting in z^p .

If $k \in p - B^+(v)$, then $k \notin 0 - B^+(v)$ since we chose p in such a way that the sets are disjoint. With a similar argument, we build an accepting branch that does not visit 0 on $\sigma^p(y)$.

Every starting position in z^p has an accepting branch, so z^p is accepted by A .

There are two noncolinear vectors v_a, v_b in P_0

This means that, in x , the accepting run from $(i_0, 0)$ has two accepting branches that stay in $B^+(v_a)$ and $B^+(v_b)$, respectively. Since v_a and v_b are not colinear, there is $r > 0$ large enough that $B(v_a) \cap B(v_b) \subset B(0, r)$.



■ **Figure 3** Left: the first case when all vectors in P_0 are colinear to v . Right: the second case when $\{v_a, v_b\} \subset P_0$. In both cases, $P_1 = \{v_1, v'_1\}$ and the circles represent positions 0 and p . The proof shows that each initial position admits a path that never visits 0 or p and accepts in y or $\sigma^p(y)$.

We choose p such that $p \notin \bigcup_{v_1 \in P_1} B(v_1, r + C)$ and such that p is in the quarterplane generated by v_a, v_b , at distance at least $r + 2C$ from the border.

Take a position $k \in \mathbb{Z}^2$. If $k \in B(0, r)$, then we pump to build an accepting branch b_p in y that stays either in $B(0)$ or in $k + B^+(v_1)$ for some $v_1 \in P_1(r)$, so that b_p does not visit p . If $k \in B(p, r)$, the same argument applies on $\sigma^p(y)$.

If $k \notin B(0, r) \cup B(p, r)$, then one of the two bands $B_a = k + B^+(v_a)$ and $B_b = k + B^+(v_b)$ contains neither 0 nor p . Indeed,

- $B_a \cap B_b$ contains neither position by definition of r ;
- If both positions appeared, we would have $|p - 0 - \pm(av_a - bv_b)| \leq 2C$ for some $a, b \geq 0$, which contradicts the choice of p .

Assuming that it is B_a , we pump to build an accepting branch in x from k that stays in B_a and visits neither 0 nor p .

Every position in z^p has an accepting branch, so z^p is accepted by A . ◀

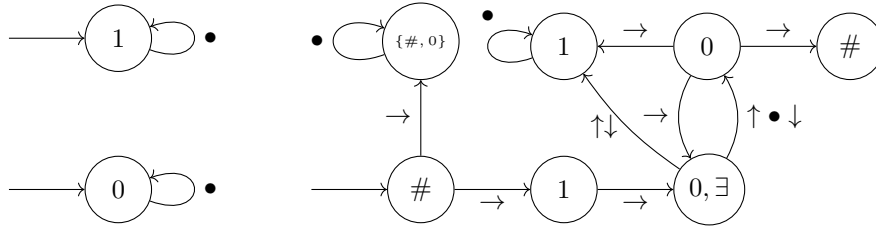
4.3 The Cone Labyrinth

► **Definition 19.** Let $\Sigma = \{0, 1, \#\}$. The cone labyrinth subshift (denoted X_{cl}) is the set of configurations x which contains none of the forbidden patterns $010, 11, \#1\#, \begin{smallmatrix} 0 \\ \# \end{smallmatrix}, \begin{smallmatrix} 1 \\ \# \end{smallmatrix}, \begin{smallmatrix} \# \\ 0 \end{smallmatrix}, \begin{smallmatrix} \# \\ 1 \end{smallmatrix}$ and such that from any position with a pattern $\#1$, there is a path to a position with a pattern $1\#$ using steps $\nearrow, \rightarrow, \searrow$ that only visits positions with symbols 0.

In a configuration $x \in \{0, 1, \#\}^{\mathbb{Z}^2}$ of X_{cl} , every column contains either only $\#$ symbols, or only 0 and 1 symbols. The $\#$ marks the outside areas, the 0 the inside areas, and 1 corresponds to entrances / exits to change areas. In particular, a 1 can only be between a 0 and a $\#$. Furthermore, every entrance $\#1$ can be matched to at least one exit $1\#$, in the sense that one can walk from $\#1$ to $1\#$ through zeroes using steps $\nearrow, \rightarrow, \searrow$.

In other words, if the width of the inside area is k , then every entrance must be matched to an exit at a vertical distance at most k .

► **Proposition 20.** $X_{cl} \in \Sigma_1$.



■ **Figure 4** \exists -automaton accepting X_{cl} . We assumed for readability that the patterns from Definition 19 have already been forbidden.

Proof. We build a \exists automaton that accepts X_{cl} assuming that patterns from Definition 19 do not appear; we can do this assumption by Lemma 15. The automaton is represented in Figure 4.

If the initial position is not the entrance of a labyrinth #1, accept (by looping). Otherwise, walk nondeterministically in all directions $\nearrow, \rightarrow, \searrow$. Keep going as long as you see 0, accept if you find a 1, reject if you find a #. Therefore the automata accepts if and only if, starting from every entrance, one branch found a matching exit. ◀

► **Proposition 21.** $X_{cl} \notin \Pi_1$.

Intuitively, in order for a run to reject a labyrinth with no exit, some individual branch must reject, which requires visiting a region of unbounded size (depending on the width on the labyrinth). By making the width large enough, we disorient this run into rejecting a valid configuration because it cannot check all required cells.

Proof. Let A be a Π_1 -automaton that rejects all configurations not in X_{cl} . We build a configuration $y \in X_{cl}$ that A rejects, the process being illustrated in Figure 5.

First define a configuration $x^n \notin X_{cl}$ for some $n > |Q|$.

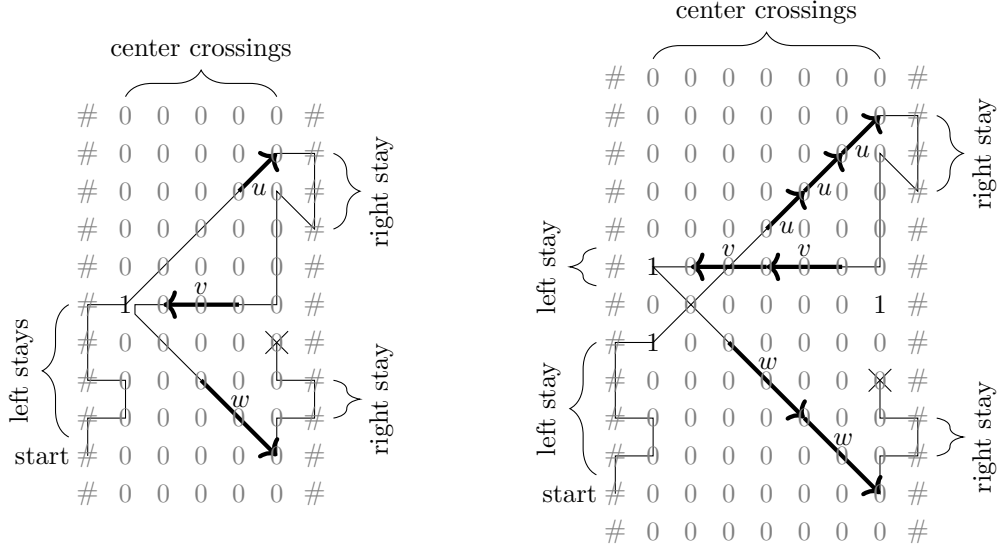
$$x^n_{(i,j)} = \begin{cases} 1 & \text{if } (i, j) = (0, 0) \\ 0 & \text{otherwise, if } 0 \leq i \leq n \\ \# & \text{otherwise} \end{cases}$$

Since A rejects x^n , there exists a position from which there is no accepting run of A ; that is, some branch b from some position p rejects (in finite time). The configuration would be valid if we switched the symbols at positions $(0, 0)$ or (n, k) for any $-n \leq k \leq n$; therefore b must visit all these positions, since it would otherwise reject a valid configuration.

Within x^n , we call “the left” the half-plane $i \leq 0$, “the right” the half-plane $i \geq n$, and “the center” the band $0 < i < n$. For simplicity, we assume that b starts in the left and ends in the right. We decompose b as $\ell_0 c_0^{\ell \rightarrow r} r_0 c_0^{r \rightarrow \ell} \dots \ell_T c_T^{\ell \rightarrow r} r_T$, where for all $k = 0, \dots, T$:

- every subsequence is nonempty;
- ℓ_k starts in the left, stays in the left and center, and ends in the left (*left stays*);
- r_k starts in the right, stays in the right and center, and ends in the right (*right stays*);
- $c_k^{\ell \rightarrow r}$ and $c_k^{r \rightarrow \ell}$ stay in the center (*center crossings*).

Consider the first crossing $c_0^{\ell \rightarrow r}$. Crossing takes at least n steps, so we find two positions (i, j) and $(i + a_0, j + b_0)$ with $a_0 > 0$ that $c_0^{\ell \rightarrow r}$ visited in that order in the same state; (a_0, b_0) is the pumping vector. Similarly, we find such pumping vectors (a_k, b_k) with $a_k > 0$ for all $c_k^{\ell \rightarrow r}$ and (α_k, β_k) with $\alpha_k < 0$ for all $c_k^{r \rightarrow \ell}$.



■ **Figure 5** On the left, a rejecting run over the invalid labyrinth x^4 , which has no exit. Bolded parts \vec{u} , \vec{v} and \vec{w} represent pumping vectors which start and end in the same state. These vectors are repeated on the right to provide another rejecting run of the same automata over the valid labyrinth y .

By pumping on these vectors on each crossing, we build a branch b' that will be valid on some other configuration $y \neq x^n$; for the time being, we only pay attention to the positions in the branch and not to the underlying configuration.

Let C be the lowest common multiple of all a_k and $-\alpha_k$ and let $m > 0$ to be fixed later. We define $b' = \ell'_0 c_0^{\ell \rightarrow r} r'_0 c_0^{r \rightarrow \ell} \dots \ell'_T c_T^{\ell \rightarrow r} r'_T$ step by step.

- $\ell'_0 = \ell_0$ is unchanged;
- $c_0^{\ell \rightarrow r}$ is $c_0^{\ell \rightarrow r}$ that we pump $\frac{mC}{a_0}$ times along the vector (a_0, b_0) .
- $r'_0 = \sigma^{(mC, \frac{mC}{a_1} b_1)}(r_0)$; in other words, r'_0 is shifted relative to r_0 so that the positions are consistent with the previous part.
- $c_0^{r \rightarrow \ell}$ is $c_0^{r \rightarrow \ell}$ shifted by $(mC, \frac{mC}{a_1} b_1)$ and pumped $\frac{mC}{-\alpha_0}$ times along the vector (α_0, β_0) .

In a similar manner, we pump every crossing in the center and shift:

- ℓ'_k and $c_k^{\ell \rightarrow r}$ by $(0, o_k^\ell(m))$, where $o_k^\ell(m) = \sum_{i=0}^{k-1} \frac{mC}{a_i} b_i + \sum_{i=0}^{k-1} \frac{mC}{-\alpha_i} \beta_i$;
- r'_k and $c_k^{r \rightarrow \ell}$ by $(mC, o_k^r(m))$ where $o_k^r(m) = \sum_{i=0}^k \frac{mC}{a_i} b_i + \sum_{i=0}^{k-1} \frac{mC}{-\alpha_i} \beta_i$.

These values are chosen so that the positions are locally consistent with allowed transitions in A . Notice that $o_k^\ell(m) - o_{k'}^\ell(m)$ is either always 0 or tends to $\pm\infty$ as $m \rightarrow \infty$.

Denoting φ the footprint, we see that $\varphi(\ell'_k) = \varphi(\ell_k) + o_k^\ell(m)$. Because every $\varphi(\ell_k)$ is finite, we can find m large enough such that, for all k and k' , $\varphi(\ell'_k) \cap \varphi(\ell'_{k'}) = \emptyset$ or $\varphi(\ell'_k) \cap \varphi(\ell'_{k'}) = \varphi(\ell_k) \cap \varphi(\ell_{k'})$, and similarly for right stays. In other words, two stays either visit no cell in common or they cross in exactly the same manner as the original branch.

If needed, we increase m so that $m > |\varphi(b)|$.

Now we build a configuration y so that b' is a branch of the run starting at the same position p . Start by putting $y = x^{n+mC}$, and do the following modifications:

1. set $y_{(0, o_k^\ell(m))} = 1$ for all k ;
2. choose some i such that $|i| < n + Cm$ and $(n + Cm, i) \notin \bigcup_k \varphi(r'_k)$. Then set $y_{(n, i + o_k^\ell(m))} = 1$ for all k .

Condition 1 adds entrances at such positions that the left stay ℓ'_k “sees” an entrance exactly when the original left stay ℓ_k saw an entrance at position $(0, 0)$. This does not impact other left stays by the argument above.

Condition 2 adds exits at positions that are not visited by b' , which is possible because we chose m to be larger than the footprint of b , but satisfy the definition for a valid labyrinth. This ensures that $y \in X_{cl}$ and that b' rejects.

By construction, the branch b' is a branch for the run on y starting at p . It follows that $y \in X_{cl}$ is rejected by A , a contradiction. ◀

5 Conclusion

5.1 Summary and open questions

We proved that subshifts accepted by alternating plane-walking automata form a strict subset of sofic subshifts, and that the first level of the hierarchy with bounded alternations is strict: Σ_1 and Π_1 are incomparable, and thus the inclusion of Δ_1 in both of them is strict.

We sum up our open questions:

1. Is the hierarchy strict for all n or does it collapse at some level? For example, can we find a subshift in $\Sigma_2 \setminus \Sigma_1$?
2. If a subshift is accepted by a universal PWA and an existential PWA, is it also accepted by a deterministic PWA? More generally, is it the case that $\Delta_n = \Sigma_n \cap \Pi_n$?
3. Is there an equivalent definition for subshifts accepted by alternating plane-walking automata by forbidden patterns accepted by plane-walking automata that do not require taking a complement?

Pumping arguments are tedious even in the first floors of the hierarchy, and we would like to find other tools, for example (as suggested by Guillaume Theyssier and inspired by [17]) from communication complexity.

This is only one of multiple possible hierarchies on walking automata. We mention n -nested automata in the next section. Automata with the ability to leave up to n pebbles (non-moving marks used as memory) during a run have also been considered. Salo and Törma studied automata with multiple heads and this hierarchy collapses at $n = 3$ [14, 13].

Definitions 2 and 4 (recognised / accepted subshifts) extend to higher dimensions \mathbb{Z}^d , $d \geq 2$, or any finitely generated group by replacing the set of directions by S and $S \cup S^{-1} \cup \{\bullet\}$, respectively, where S is a set of generators of the group. While our main results extend directly to \mathbb{Z}^d by considering lifts of our two-dimensional examples (see Definition 9), we make no guesses as to the situation in more complicated groups.

5.2 Strict Hierarchy and Tree-walking Automata

► **Conjecture 22.** *The hierarchy is strict, that is, $\Sigma_n \subsetneq \Sigma_{n+1}$ for all n (and the same is true for all combinations of Π, Σ and Δ).*

We present some elements supporting this conjecture. Tree-walking automata (TWA) is a similar model that recognise words written on (finite or infinite) trees. For example, Theorem 8 can be seen as the translation of [4].

In the context of tree-walking automata, we could not find any work on a similar Σ/Π hierarchy. However, [16] considers k -nested TWA, defined intuitively as follows: 0-nested TWA are (existential) nondeterministic TWA; k -nested TWA are nondeterministic TWA where, at each step, the set of available transitions is determined by foreseeing the behaviour of two $k - 1$ -nested TWA A and \bar{A} , in the following sense: the next transition is chosen nondeterministically among transitions after which A would accept and \bar{A} would reject.

The class of (tree-walking or plane-walking) k -nested automata seems to be related to Σ_{k+1} automata, although we do not have a proof of either direction. For tree-walking automata, Theorem 29 in [16] proves that the hierarchy of languages recognized by k -nested TWA is strict. This is to us a strong indication that the alternating hierarchy is strict on trees (free groups).

We believe that these results can be brought to plane-walking automata by considering subshifts where trees are drawn on a background on zeroes (this can be ensured with finitely many forbidden patterns), and every tree must belong to L_k , where L_k is a tree language that is Σ_{k+1} but not Σ_k (alternatively, k -nested and not $k-1$ -nested). This is not straightforward as plane-walking automata have the ability to walk out of the tree, which should not provide additional recognition power, but pumping arguments are very tedious for alternating plane-walking automata. We leave this as an open question for future research.

5.3 An Alternative Approach: Kari-Moore's Rectangles

We mention an alternative approach to prove that $\Sigma_1 \neq \Pi_1$ that was used in [10] for finite patterns. The following statements are only translations to our model and from finite patterns to periodic configurations, and we do not vouch for their correctness. This is another suggestion for future work generalising Theorem 14.

Let $f : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$. Let $X_f \subset \{0, 1\}^{\mathbb{Z}^2}$ be the smallest subshift containing the strongly periodic configurations generated by the rectangles:

$$\{r \in \{0, 1\}^{[0, n] \times [0, k]} : k \in f(n) \text{ and for all } i, j > 0, r_{0,0} = r_{i,0} = r_{0,j} = 1, r_{i,j} = 0\}.$$

If X_f is accepted by a Σ_1 plane-walking automaton with k states, then for every n , the language $\{1^j : j \in f(n)\}$ is recognised by a two-way nondeterministic finite automaton with kn states, and hence regular [10].

The following example of a function f is such that $X_f \in \Sigma_1 \setminus \Pi_1$, while $X_{f^c} \in \Pi_1 \setminus \Sigma_1$:

$$f(n) = \{in + j : i, j \in \mathbb{N}, j < i\}.$$

Indeed, $f^c(n)$ is a finite set whose largest element is $(n-2) \cdot n + (n-1) \approx n^2$. If it is accepted by an automaton with kn states, this would be a contradiction for $n > k$ by pumping.

References



- 1 Marcella Anselmo and Maria Madonia. Classes of two-dimensional languages and recognizability conditions. *RAIRO Theor. Informatics Appl.*, 44(4):471–488, 2010. doi:10.1051/ita/2011003.
- 2 Nathalie Aubrun and Mathieu Sablik. Simulation of effective subshifts by two-dimensional subshifts of finite type. *CoRR*, abs/1602.06095:35–63, 2016. doi:10.48550/arXiv.1602.06095.
- 3 Manuel Blum and Carl Hewitt. Automata on a 2-dimensional tape. In *8th Annual Symposium on Switching and Automata Theory, Austin, Texas, USA, October 18-20, 1967*, pages 155–160. IEEE, IEEE Computer Society, 1967. doi:10.1109/F0CS.1967.6.
- 4 Mikolaj Bojanczyk and Thomas Colcombet. Tree-walking automata do not recognize all regular languages. 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 234–243. ACM, 2005. doi:10.1145/1060590.1060626.
- 5 Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *Journal of Computer and System Sciences*, 78(3):731–764, 2012. doi:10.1016/j.jcss.2011.11.001.

- 6 Dora Giammarresi and Antonio Restivo. Recognizable picture languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 6(2&3):241–256, 1992. doi:10.1142/S021800149200014X.
- 7 Dora Giammarresi and Antonio Restivo. Two-dimensional languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 215–267. Springer, 1997. doi:10.1007/978-3-642-59126-6_4.
- 8 Nataša Jonoska and Joni B. Pirnot. Finite state automata representing two-dimensional subshifts. *Theoretical computer science*, 410(37):3504–3512, 2009. doi:10.1016/j.tcs.2009.03.015.
- 9 Jarkko Kari and Cristopher Moore. New results on alternating and non-deterministic two-dimensional finite-state automata. In Afonso Ferreira and Horst Reichel, editors, *Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 396–406. Springer, 2001. doi:10.1007/3-540-44693-1_35.
- 10 Jarkko Kari and Cristopher Moore. Rectangles and squares recognized by two-dimensional automata. In Juhani Karhumäki, Hermann A. Maurer, Gheorghe Paun, and Grzegorz Rozenberg, editors, *Theory is Forever: Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday*, volume 3113 of *Lecture Notes in Computer Science*, pages 134–144. Springer, 2004. doi:10.1007/978-3-540-27812-2_13.
- 11 Jarkko Kari and Ville Salo. A survey on picture-walking automata. In Werner Kuich and George Rahonis, editors, *Algebraic Foundations in Computer Science – Essays Dedicated to Symeon Bozapalidis on the Occasion of His Retirement*, volume 7020 of *Lecture Notes in Computer Science*, pages 183–213. Springer, 2011. doi:10.1007/978-3-642-24897-9_9.
- 12 Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, 1995.
- 13 Ville Salo. Four heads are better than three. In Hector Zenil, editor, *Cellular Automata and Discrete Complex Systems – 26th IFIP WG 1.5 International Workshop, AUTOMATA 2020, Stockholm, Sweden, August 10-12, 2020, Proceedings*, volume 12286 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2020. doi:10.1007/978-3-030-61588-8_9.
- 14 Ville Salo and Ilkka Törmä. Plane-walking automata. In Teijiro Isokawa, Katsunobu Imai, Nobuyuki Matsui, Ferdinand Peper, and Hiroshi Umeo, editors, *Cellular Automata and Discrete Complex Systems – 20th International Workshop, AUTOMATA 2014, Himeji, Japan, July 7-9, 2014, Revised Selected Papers*, volume 8996 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 2014. doi:10.1007/978-3-319-18812-6_11.
- 15 Ville Salo and Ilkka Törmä. Group-walking automata. In Jarkko Kari, editor, *Cellular Automata and Discrete Complex Systems: 21st IFIP WG 1.5 International Workshop, AUTOMATA 2015, Turku, Finland, June 8-10, 2015. Proceedings 21*, volume 9099 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 2015. doi:10.1007/978-3-662-47221-7_17.
- 16 Balder ten Cate and Luc Segoufin. Transitive closure logic, nested tree walking automata, and xpath. *J. ACM*, 57(3):18:1–18:41, 2010. doi:10.1145/1706591.1706598.
- 17 Véronique Terrier. Communication complexity tools on recognizable picture languages. *Theoretical Computer Science*, 795:194–203, 2019. doi:10.1016/j.tcs.2019.05.040.


Cycle Counting Under Local Differential Privacy for Degeneracy-Bounded Graphs

Quentin Hillebrand  

The University of Tokyo, Japan

Vorapong Suppakitpaisarn  

The University of Tokyo, Japan

Tetsuo Shibuya  

The University of Tokyo, Japan

Abstract

We propose an algorithm for counting the number of cycles under local differential privacy for degeneracy-bounded input graphs. Numerous studies have focused on counting the number of triangles under the privacy notion, demonstrating that the expected ℓ_2 -error of these algorithms is $\Omega(n^{1.5})$, where n is the number of nodes in the graph. When parameterized by the number of cycles of length four (C_4), the best existing triangle counting algorithm has an error of $O(n^{1.5} + \sqrt{C_4}) = O(n^2)$. In this paper, we introduce an algorithm with an expected ℓ_2 -error of $O(\delta^{1.5}n^{0.5} + \delta^{0.5}d_{\max}^{0.5}n^{0.5})$, where δ is the degeneracy and d_{\max} is the maximum degree of the graph. For degeneracy-bounded graphs ($\delta \in \Theta(1)$) commonly found in practical social networks, our algorithm achieves an expected ℓ_2 -error of $O(d_{\max}^{0.5}n^{0.5}) = O(n)$. Our algorithm's core idea is a precise count of triangles following a preprocessing step that approximately sorts the degree of all nodes. This approach can be extended to approximate the number of cycles of length k , maintaining a similar ℓ_2 -error, namely $O(\delta^{(k-2)/2}d_{\max}^{0.5}n^{(k-2)/2} + \delta^{k/2}n^{(k-2)/2})$ or $O(d_{\max}^{0.5}n^{(k-2)/2}) = O(n^{(k-1)/2})$ for degeneracy-bounded graphs.

2012 ACM Subject Classification Security and privacy \rightarrow Privacy-preserving protocols; Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases Differential privacy, triangle counting, degeneracy, arboricity, graph theory, parameterized accuracy

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.49

Funding *Quentin Hillebrand*: partially supported by KAKENHI Grant 20H05965, and by JST SPRING Grant Number JPMJSP2108.

Vorapong Suppakitpaisarn: partially supported by KAKENHI Grant 21H05845 and 23H04377.

Tetsuo Shibuya: partially supported by KAKENHI Grant 20H05967, 21H05052, and 23H03345.

Acknowledgements The authors wish to express their thanks to the anonymous reviewers whose valuable feedback greatly enhanced the quality of this paper.

1 Introduction

In recent years, differential privacy [13, 15] has become the gold standard for providing strong privacy guarantees while enabling meaningful data analysis. Differential privacy ensures that the output of a computation does not significantly change when any single individual's data is modified, thus safeguarding individual privacy. While much of the initial work in differential privacy focused on traditional tabular data [14, 26], there is increasing interest in extending these privacy guarantees to graph data [31, 35], which presents its own unique set of challenges.



© Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 49; pp. 49:1–49:22



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Differential privacy has evolved into numerous variants to accommodate different scenarios, as detailed in [8]. Of particular interest to us is the concept of local differential privacy [7, 17]. This variant is unique in that it does not rely on the assumption of a trusted central server. Instead, users must obfuscate their private data before sharing it with an untrusted computing entity. In the context of graph data, the most commonly adopted notion is edge local differential privacy [29], where the sensitive information of each user pertains to their connections with others.

A widely used obfuscation method is randomized response [33, 32]. In this approach, users invert each bit of their adjacency vector with a certain probability. The server then collects this distorted information to construct an obfuscated graph. Although it is possible to publish various graph statistics from the obfuscated graph, the resulting information tends to be imprecise. Algorithms specifically designed to publish particular statistics typically yield more accurate and useful graph information.

■ **Table 1** Upper and lower bounds of the expected ℓ_2 -error for triangle and k -cycle counting under the local differential privacy. “Interactive” refers to a scenario in which multiple rounds of communication between the server and the clients are permitted.

	Upper Bound	Lower Bound
Triangle	$O(n^2)$ ([22], general graphs)	$\Omega(n^{1.5})$ (non-interactive) [24]
	$O(d_{\max}^{1.5}n^{0.5})$ ([16], general graphs)	$\Omega(n^{1.5})$ (interactive) [16]
	$O(d_{\max}^{0.5}n^{0.5})$ (this work , degeneracy-bounded graph)	$\Omega(n^2)$ (non-interactive) [16]
Odd length cycles C_k	$O(n^{k-1})$ (folklore, general graphs)	
	$O(n^{(k-1)/2})$ (this work , degeneracy-bounded graph)	

One graph statistic frequently considered by researchers in local differential privacy is the number of subgraphs [22, 20]. Specifically, many studies have focused on the publication of triangle counts [22, 23, 20, 16]. Theoretical analysis results on the ℓ_2 -error are summarized in Table 1. Unfortunately, to date, when n is the number of nodes and d_{\max} is the maximum degree of the input graphs, the best algorithm has an expected ℓ_2 -error of $O(n^2)$ or $O(d_{\max}^{1.5}n^{0.5})$. We believe that this error is too large for many applications and should be improved. On the other hand, it has been shown that for all locally differentially private algorithms, there exists a class of graphs where the ℓ_2 -error is $\Omega(n^{1.5})$ [16]. This lower bound implies that the expected ℓ_2 -error cannot be significantly improved.

1.1 Our Contribution

This motivates us to consider a specific class of graphs. In this paper, we specifically focus on graphs with bounded degeneracy, as most social graphs exhibit degeneracy values that are substantially smaller than both the number of vertices and the maximum degree. The table in the Appendix of [12] provides statistics on a diverse range of graphs, detailing the number of nodes, degeneracy, and maximum degree. As shown in the paper, for sufficiently large graphs, degeneracy is consistently at least an order of magnitude smaller than the maximum degree, and in some instances, several orders of magnitude smaller.

Additionally, several synthetic graph models commonly considered realistic naturally produce graphs with low degeneracy. Examples include preferential attachment graphs [1] and bounded expansion graphs [27].

Degeneracy is particularly significant in parameterizing the complexity of subgraph counting algorithms, as demonstrated in [6, 2, 5]. Given the relationship between computational complexity and estimation error in triangle counting algorithms, degeneracy is an important parameter for characterizing accuracy.

Let the graph degeneracy be δ . We propose a locally differentially private algorithm with an expected ℓ_2 -error of $O(\delta^{1.5}n^{0.5} + \delta^{0.5}d_{\max}^{0.5}n^{0.5})$. When the graph degeneracy is bounded ($\delta = O(1)$), the expected ℓ_2 -error becomes $O(d_{\max}^{0.5}n^{0.5}) = O(n)$. This result implies that our expected error for the degeneracy-bounded graphs can be smaller than the lower bound for general graphs.

We also extend our results to count the number of cycles with odd lengths in degeneracy-bounded graphs. To our knowledge, there are only two local differentially private algorithms proposed for counting subgraphs of more than three nodes. The first algorithm [24] is designed to count the number of four-length cycles but operates within the shuffle model, which is weaker than the original local differential privacy model. The second algorithm counts the number of walks of length k [3]. This field has limited work due to the significant noise introduced to ensure user privacy, which accumulates as the subgraph size increases. This accumulation results in unacceptable errors for differential privacy in larger subgraphs. For instance, while the expected ℓ_2 -error from triangle counting algorithms based on randomized response is $O(n^2)$ [24], the expected ℓ_2 -error for similar algorithms estimating the number of C_k is as high as $O(n^{k-1})$. In other words, the error increases by a factor of n with each increment in cycle length.

In this work, we propose an algorithm that significantly reduces the expected ℓ_2 -error to $O(n^{(k-1)/2})$ in degeneracy-bounded graphs. We believe that this error is much smaller than the actual number of cycles in most graphs. Consequently, our algorithm is the first to publish a meaningful number of large cycles under local differential privacy.

1.2 Technical Overview

In this section, we provide an overview of the technical concepts behind our triangle counting algorithm. The algorithm for counting odd-length cycles, for $k \geq 5$, extends these ideas but requires a more intricate and detailed analysis.

Let the input graph be $G = (V = \{\nu_1, \dots, \nu_n\}, E)$. In prior work [22], they apply a randomized response mechanism that flips each bit in the adjacency matrix with a certain probability. Let the resulting graph after applying the randomized response be $G' = (V, E')$. In the local differential privacy setting, each node ν_i knows whether it is connected to another node ν_j (where $\nu_j \neq \nu_i$) if $\{\nu_i, \nu_j\} \in E$. For the triangle counting method, node ν_i considers $(\nu_i, \nu_j, \nu_\kappa)$ as a triangle if $\{\nu_i, \nu_j\} \in E$, $\{\nu_i, \nu_\kappa\} \in E$, and $\{\nu_j, \nu_\kappa\} \in E'$. Define $e_{i,j,\kappa} = 1$ if node ν_i considers $(\nu_i, \nu_j, \nu_\kappa)$ as a triangle, otherwise set $e_{i,j,\kappa} = 0$. Define $S_i = \{(j, \kappa) : \{\nu_i, \nu_j\}, \{\nu_i, \nu_\kappa\} \in E \text{ and } j < \kappa\}$. The estimated number of triangles for node ν_i , reported by the user, is $\tilde{t}_i = \sum_{(j,\kappa) \in S_i} e_{i,j,\kappa}$. The total estimated number of triangles in the graph is then $\tilde{f}_\Delta(G) = \frac{1}{3} \sum_i \tilde{t}_i = \frac{1}{3} \sum_i \sum_{(j,\kappa) \in S_i} e_{i,j,\kappa}$, where dividing by three corrects for the fact that each triangle is counted once by each of the three users forming it (i.e., triple-counted), ensuring each triangle is counted only once.

The ℓ_2 -error of the estimated triangle count $\tilde{f}_\Delta(G)$ mostly arises from the variance in the estimation. A significant portion of this variance comes from the covariance between pairs of variables in the summation $\frac{1}{3} \sum_i \sum_{(j,\kappa) \in S_i} e_{i,j,\kappa}$. Two variables, $e_{i,j,\kappa}$ and $e_{i',j',\kappa'}$, are dependent if $(j, \kappa) = (j', \kappa')$. The number of dependent pairs in the counting process is equivalent to the number of tuples $(\nu_i, \nu_j, \nu_{i'}, \nu_\kappa)$ such that $(j, \kappa) \in S_i \cap S_{i'}$, which corresponds to the number of 4-cycles in the input graph G . Therefore, the squared ℓ_2 -error is approximately proportional to the number of 4-cycles in the graph, which is $O(n^4)$.

Let us assume that the indices of all users are predetermined and publicly known before the counting process begins. Define $S'_i = \{(j, \kappa) : \{\nu_i, \nu_j\}, \{\nu_i, \nu_\kappa\} \in E \text{ and } j < i < \kappa\}$. If node i only considers the pairs (j, κ) within S'_i , then each triangle is counted exactly once. The estimated number of triangles, $\hat{f}_\Delta(G)$, can be calculated as $\hat{f}_\Delta(G) = \sum_i \hat{t}_i$, where $\hat{t}_i = \sum_{(j, \kappa) \in S'_i} e_{i, j, \kappa}$. In this counting method, the number of dependent variable pairs is at most the number of 4-cycles that contain the three nodes ν_i, ν_j, ν_κ with $j < i < \kappa$.

Let δ represent the degeneracy of the input graph G , and for each $\nu \in V$, let $d(\nu)$ denote the degree of ν . Assume that the degrees of all nodes are publicly known, and the nodes are indexed in non-decreasing order of their degree, i.e., if $i > j$, then $d(\nu_i) \leq d(\nu_j)$. Referring to the bound established by Chiba and Nishizeki [6], which states that $\sum_{(\nu_i, \nu_j) \in E} \min(d_i, d_j) \leq O(\delta \cdot |E|)$, we demonstrate in this paper that the number of such cycles is $O(\delta^3 n)$. Consequently, the squared ℓ_2 -error is reduced from $O(n^4)$ in previous work to $O(\delta^3 n)$.

However, we cannot assume that the degrees of all nodes are publicly known, as this information is sensitive. To address this issue, we use local Laplacian queries, allowing each user to publish a noisy version of their degree. Let the noisy degree of $\nu \in V$ be denoted as $\tilde{d}(\nu)$. We then assign indices to users based on these noisy degrees, such that if $i > j$, then $\tilde{d}(\nu_i) \leq \tilde{d}(\nu_j)$. Afterward, we run the protocol described in the previous paragraph. We show that even with noisy degrees, the expected number of such cycles remains bounded by $O(\delta^3 n)$.

In summary, our mechanism involves two steps. First, users publish their noisy degrees using the local Laplacian mechanism, and the server assigns indexes based on these noisy values. In the second step, using the results of randomized response, each user ν_i estimates the number of triangles $(\nu_i, \nu_j, \nu_\kappa)$ where $j < i < \kappa$. This method significantly reduces the number of dependent triangle pairs in degeneracy-bounded graphs, which in turn lowers the variance of the estimation.

1.3 Related Works

The field of graph data mining under local differential privacy is relatively new. In contrast, differential privacy has been studied for many years by various researchers, including works like [18, 28]. According to [22], local differential privacy typically only hides edges or relationships, except in special cases like [36]. Differential privacy, on the other hand, can hide whether an individual or node is part of a social network, as shown in [19, 30]. Therefore, while both edge and node differential privacy exist, node differential privacy does not apply in the context of local differential privacy.

Recent works have proposed methods to estimate the densest subgraph, k -core decomposition, and degeneracy under local differential privacy [10, 9, 11]. However, since we are focused on estimating different graph statistics in graphs, we do not use or extend the ideas from these works. Instead, the estimation of degeneracy can be used to approximate the ℓ_2 -error of our algorithm.

2 Preliminaries

2.1 Notations

For $V = \{\nu_1, \dots, \nu_n\}$ a set of vertices and $E \subseteq V^2$ a set of edges, we denote by $G = (V, E)$ the graph on V . We consider simple undirected graphs, meaning that for $\nu, \nu' \in V$, $(\nu, \nu) \notin E$ and $(\nu, \nu') \in E \implies (\nu', \nu) \in E$. We denote by $n = |V|$ the size of the graph and $m = |E|$ its number of edges.

For each $i \in [1, n]$, we introduce $a_i = [a_{i,1}, \dots, a_{i,n}]$, the adjacency list of user ν_i , where for any $j \in [1, n]$, $a_{i,j} = 1$ if the edge (ν_i, ν_j) is in E and $a_{i,j} = 0$ otherwise. Additionally, we introduce d_i , the degree of node ν_i , which corresponds to the number of edges incident to ν_i .

We call a path of length $k \in \mathbb{N}$, denoted P_k , any tuple $(\nu_{l_1}, \dots, \nu_{l_k})$ such that, for all $i \in [1, k]$, $(\nu_{l_i}, \nu_{l_{i+1}}) \in E$, and, for all $i \neq j$, $\nu_{l_i} \neq \nu_{l_j}$. We also use $\#P_k(G)$ to refer to the number of paths of length k in G . Similarly, a cycle of length $k \in \mathbb{N}$, or C_k , is a tuple $(\nu_{l_1}, \dots, \nu_{l_k})$ that forms a path and satisfies $(\nu_{l_k}, \nu_{l_1}) \in E$. We will also use $\#C_k(G)$ to refer to the number of cycles of length k in G .

2.2 Edge Local Differential Privacy

We say that two adjacency lists a and a' are neighboring if they differ by one bit, i.e. if we can go from one to the other by adding or removing an edge to node ν_i . If a' is a neighbor of a , we write that $a \sim a'$. The notion of edge local differential privacy is as follows:

► **Definition 1** (ε -edge local differentially private query). *Let $\varepsilon > 0$. A randomized algorithm \mathcal{R} is a ε -edge local differentially private query on the node ν_i if, for all neighboring bit strings $a \sim a'$, and for all S , it holds that*

$$\mathbb{P}[\mathcal{R}(a) \in S] \leq e^\varepsilon \mathbb{P}[\mathcal{R}(a') \in S].$$

► **Definition 2** (ε -edge local differentially private algorithm [29]). *Let \mathcal{A} be an algorithm that generates multiple randomized queries for each user, has each user apply these queries to their adjacency vector, and then estimates some graph statistics based on the results. We say \mathcal{A} is an ε -edge local differentially private algorithm if, for all users ν_i and for all possible sets of queries $\mathcal{R}_1, \dots, \mathcal{R}_k$ inquired to ν_i (where for each $1 \leq j \leq k$, \mathcal{R}_j is an ε_j -edge local differentially private query), it holds that $\varepsilon_1 + \dots + \varepsilon_k \leq \varepsilon$.*

2.3 Laplacian Query and Restricted Sensitivity

Next, we introduce queries that are ε -edge local differentially private. We first consider a query which aims to give an estimate of a real number statistics of the adjacency vector.

► **Definition 3** (Edge local Laplacian query [21]). *For a function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ on adjacency lists, and $a \sim a'$ denoting neighboring adjacency lists, the global sensitivity of f is defined as $\Delta_f = \max_{a \sim a'} |f(a) - f(a')|$. For $\varepsilon > 0$, the query that outputs $f(a) + \text{Lap}(\Delta_f/\varepsilon)$ is ε -edge local differentially private, where $\text{Lap}(b)$ represents noise drawn from the Laplacian distribution with parameter b .*

Global sensitivity in Definition 3 is designed to handle the worst-case scenario, which can lead to large amounts of noise being added to the data when using the Laplacian mechanism. However, if the data is known to belong to a specific set, restricted sensitivity allows us to adjust the noise according to the sensitivity within that set, resulting in more tailored and potentially lower noise levels.

► **Definition 4** (Restricted sensitivity (Definition 8 of [4])). *Let $a = (a_1, \dots, a_n)$, $a' = (a'_1, \dots, a'_n) \in \{0, 1\}^n$ and $d(a, a')$ be the Hamming distance between a and a' . The restricted sensitivity of f over a set of possible output \mathcal{H} is*

$$RS_f(\mathcal{H}) = \max_{a, a' \in \mathcal{H}} \left(\frac{|f(a) - f(a')|}{d(a, a')} \right).$$

We can use restricted sensitivity to publish data even if it is not initially in the set. To do this, we first need to define a projection method to map the data to the set. In this work, we will consider \mathcal{H}_d , the class of adjacency list with a maximum degree of d , for calculating restricted sensitivity. We assume that the order of all nodes is fixed, and if a node ν_i is adjacent to more than d nodes, we retain only the first d nodes according to this order. The map can be considered as an operation on each adjacency vector a_i . We denote the mapping result on a_i as $\mu_d(a_i)$.

► **Definition 5** (Edge local Laplacian query with restricted sensitivity on \mathcal{H}_d [4]). *For any f queried to a user i , the query that answers $f(\mu_d(a_i)) + \text{Lap}(3 \cdot RS_f(\mathcal{H}_d)/\varepsilon)$ is called edge local Laplacian query with restricted sensitivity on \mathcal{H}_d , and provides ε -edge local differential privacy.*

2.4 Unbiased Randomized Response

In this subsection, we consider the randomized response query, which aims to publish an obfuscated adjacency vector.

► **Definition 6** (Randomized response query [33, 32]). *For $\varepsilon > 0$, the randomized response mechanism takes an adjacency list $a = (a_1, \dots, a_n)$ as input and outputs an obfuscated list $\tilde{a} = (\tilde{a}_1, \dots, \tilde{a}_n)$. For i , the probability that \tilde{a}_i is set to 1 is given by:*

$$\mathbb{P}[\text{RR}(\tilde{a}_i) = 1] = \begin{cases} \frac{e^\varepsilon}{1+e^\varepsilon} & \text{if } a_i = 1 \\ \frac{1}{1+e^\varepsilon} & \text{if } a_i = 0. \end{cases}$$

With this definition, randomized response provides ε -edge local differential privacy.

We can construct a graph \tilde{G} based on the collection of obfuscated adjacency vectors obtained from all users. Using the statistics of the obfuscated graph \tilde{G} , we can then publish various information, including the number of subgraphs [34, 22, 23, 20]. However, randomized response produces biased results, making it less suitable for counting queries. This bias can be fixed by the subsequent definition.

► **Definition 7** (Unbiased randomized response query [16]). *Let $\varepsilon > 0$ and \tilde{a}_i be the adjacency vector published through randomized response with budget ε by user ν_i . Then, for all $(i, j) \in [1, n]^2$,*

$$\hat{a}_{i,j} = \frac{e^\varepsilon + 1}{e^\varepsilon - 1} \tilde{a}_{i,j} - \frac{1}{e^\varepsilon - 1}$$

is an unbiased estimator of $a_{i,j}$. Additionally, for $(i, j) \neq (i', j')$, $\hat{a}_{i,j}$ is independent of $\hat{a}_{i',j'}$, and $\text{Var}(\hat{a}_{i,j}) = \frac{e^\varepsilon}{(e^\varepsilon - 1)^2}$. We refer to a query that publishes \hat{a}_i as the unbiased randomized response query. This query is ε -edge locally differentially private.

We can use the results from the unbiased randomized response query to calculate the number of subgraphs. For example, without privacy constraints, the number of triangles can be calculated as $\sum_{i < j < k} a_{i,j} \cdot a_{j,k} \cdot a_{k,i}$. To privately estimate the number of triangles, we use $\sum_{i < j < k} \hat{a}_{i,j} \cdot \hat{a}_{j,k} \cdot \hat{a}_{k,i}$. It is theoretically shown in [16] that the estimator $\sum_{i < j < k} \hat{a}_{i,j} \cdot \hat{a}_{j,k} \cdot \hat{a}_{k,i}$ has a smaller ℓ_2 -error compared to the estimator obtained from the randomized response query, $\sum_{i < j < k} \tilde{a}_{i,j} \cdot \tilde{a}_{j,k} \cdot \tilde{a}_{k,i}$.

2.5 Graph Arboricity and Degeneracy

Graph arboricity and degeneracy can be defined as follows:

► **Definition 8** (Arboricity). *The arboricity of a graph G is the minimal number $\alpha(G)$ such that the edges of G can be partitioned into $\alpha(G)$ forests.*

► **Definition 9** (Degeneracy). *The degeneracy of a graph G is the smallest number $\delta(G)$ such that any subgraph of G , contains at least one node with induced degree at most $\delta(G)$.*

We observe that the variable δ is frequently used as a privacy parameter in differential privacy. However, since we do not consider that parameter in this paper, we choose to use δ to represent degeneracy, which is also a common convention. When the context is clear, we will drop the G of the notation and simply write α and δ . The two quantities are linked by the following theorem.

► **Theorem 10** (equation 3 and lemma 2.2 of [37]). *In any graph G , degeneracy and arboricity satisfy $\alpha \leq \delta \leq 2\alpha - 1$.*

The arboricity has previously been used outside of the differential private community to bound some graph statistics. A folklore useful result is that the number of edges in a graph is smaller than δn . Another well known result is as follows:

► **Theorem 11** (Chiba-Nishizeki Bound [6]). *With $m = |E|$ and d_i the degree of node ν_i , then*

$$\sum_{(\nu_i, \nu_j) \in E} \min(d_i, d_j) \leq m\alpha.$$

3 Node-Reordered Graphs and Their Properties

The first step of our mechanism is to order the vertices based on their estimated degree. The algorithm for this step is shown in Algorithm 1. At Line 2 of the algorithm, we privately publish the estimated degree. Under edge local differential privacy, the global sensitivity of the degree is 1. Therefore, we can use the Laplacian query (Definition 3) with noise scaled to $1/\varepsilon_0$ to publish the degree, where ε_0 is the privacy budget allocated to this step. We denote the estimated degree as $\tilde{d}_i = d_i + \text{Lap}(1/\varepsilon_0)$.

■ **Algorithm 1** Calculate a low degree ordering of a graph with respect to the estimated degree.

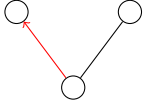
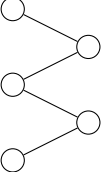
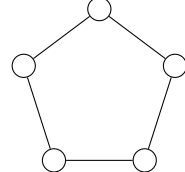
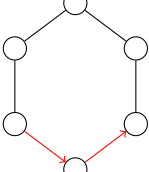
```

1 Function GetOrdering
   Input: Graph  $G = (V, E)$ , privacy budget  $\varepsilon_0$ 
   Output: A low degree ordering  $\phi$  of  $G$  with respect to the estimated degree
2   [User  $i$ ] Calculate and send  $\tilde{d}_i \leftarrow d_i + \text{Lap}(\frac{1}{\varepsilon_0})$  to the central server
3   [Server] Let  $\phi(i) = j$  if  $\tilde{d}_i$  is the  $j$ -the largest number in  $\tilde{d}_1, \dots, \tilde{d}_n$ . Calculate
    $\phi(i)$  for all  $i$ 
4   return  $\phi$ ;

```

After publishing the estimated degrees, in Line 3, we assign an order ϕ to the nodes based on their degrees, which we refer to as a *low degree ordering*. For $G = (\{\nu_1, \dots, \nu_n\}, E)$, we denote the reordered graph as $G^\phi = (V^\phi, E^\phi)$, where $V^\phi = \{\eta_i \mid i \in [1, n]\}$ and $\nu_i = \eta_{\phi(i)}$ for all i . The edge set E^ϕ is defined as $\{(\eta_{\phi(i)}, \eta_{\phi(j)}) \mid (\nu_i, \nu_j) \in E\}$. We note that G and G^ϕ are isomorphic, and thus have the same number of subgraphs. We denote by $d_i(G^\phi)$ the degree of η_i in G^ϕ and $d_i^-(G^\phi)$ the number of neighbors of node η_i in the set $\{\eta_1, \dots, \eta_{i-1}\}$.

■ **Table 2** List of subgraphs analyzed in Section 3, including their representations and bounds on their counts in the graph produced by Algorithm 1. Oriented edges indicate directionality, with an arrow from ν_j to ν_i signifying that $j > i$. The bound on S_2^* aligns with the bound on S_2 presented in [6], but it constitutes a distinct contribution as it is established for imperfectly ordered graphs. In contrast, the result on C_{2k}^* is entirely novel to this work and serves as the primary result of our proof.

Symbol	S_2^*	P_k	C_k	C_{2k}^*
Representation				
Bound	$\mathcal{O}(\delta^2 n)$	$\mathcal{O}(\delta^{\lceil \frac{k}{2} \rceil} n^{\lfloor \frac{k}{2} \rfloor + 1})$	$\mathcal{O}(\delta^{\lceil \frac{k}{2} \rceil} n^{\lfloor \frac{k}{2} \rfloor})$ [25]	$\mathcal{O}(\delta^{k+1} n^{k-1})$

In the remainder of this section, we analyze the properties of graphs produced by the reordering. Specifically, our focus is on bounding the frequency of certain substructures within the reordered graph. A summary of the results from this section is provided in Table 2.

► **Definition 12** (low star). For $k \in \mathbb{N}^*$, a low- k -star is a subgraph consisting of a central node and k neighboring nodes, where at least one of the neighboring nodes has an index smaller than that of the central node. We denote by $S_k^*(G)$ the number of such subgraphs contained in a graph G .

► **Theorem 13.** $\mathbb{E}[S_2^*(G^\phi)] \leq \mathcal{O}(\delta^2 n)$.

Proof. Let $\mathcal{N}_i(G^\phi)$ be the set of neighbors of η_i in G^ϕ . We have that:

$$\begin{aligned}
 S_2^*(G^\phi) &= \sum_{i=1}^n d_i^-(G^\phi)(d_i(G^\phi) - 1) \leq \sum_{i=1}^n d_i(G^\phi) \times d_i^-(G^\phi) = \sum_{i=1}^n d_i(G^\phi) \sum_{\eta_j \in \mathcal{N}_i(G^\phi)} \mathbb{1}_{j < i} \\
 &= \sum_{(\eta_i, \eta_j) \in E^\phi} d_{\max(i, j)}(G^\phi)
 \end{aligned}$$

Let τ_i denote the noise added to the estimated degree of user i . For each edge (η_i, η_j) , their ranks can only be exchanged if the sum of the errors in both degree estimations exceeds the gap between the two degrees. Therefore, the quantity $d_{\max(i, j)}(G^\phi)$ satisfies

$$d_{\max(i, j)}(G^\phi) \leq \min(d_i, d_j) + |\tau_i| + |\tau_j|.$$

Using this inequality, we can rewrite the count of $S_2^*(G^\phi)$ as

$$S_2^*(G^\phi) \leq \sum_{(\eta_i, \eta_j) \in E^\phi} \min(d_i, d_j) + \sum_{i=1}^n |\tau_i| d_i.$$

Since τ_i is sampled from $\text{Lap}(1/\varepsilon_0)$, we have that $|\tau_i|$ follows an exponential law of expectation $1/\varepsilon_0$. Hence,

$$\mathbb{E}[S_2^*(G^\phi)] \leq \sum_{(\nu_i, \nu_j) \in E^\phi} \min(d_i, d_j) + \frac{m}{\varepsilon_0}.$$

Since G is isomorphic to G^ϕ , $\alpha(G) = \alpha(G^\phi)$ and using Theorem 11 it follows that

$$\sum_{(\nu_i, \nu_j) \in E^\phi} \min(d_i, d_j) \leq m \cdot \alpha(G).$$

Since $m \leq n\delta$ and $\alpha(G) = O(\delta)$, this gives $\mathbb{E}[S_2^*(G^\phi)] \leq O(\delta^2 n)$. \blacktriangleleft

In addition to the ordered stars we just discussed, arboricity can also be used to bound the number of paths and cycles in a graph, as demonstrated in the following lemma and theorem. Recall that $\#P_k(G)$ is the number of paths with length k in the graph G .

► **Lemma 14.** *For any positive integer k , $\#P_{2k}(G) = O(\delta^k n^{k+1})$, $\#P_{2k+1} = O(\delta^{k+1} n^{k+1})$.*

Proof. We first consider $\#P_{2k+1}(G)$. Let f be a function that maps a path of length $2k+1$ to a tuple of $k+1$ edges, defined as $f(e_1, \dots, e_{2k+1}) = (e_1, e_3, \dots, e_{2k+1})$. We observe that, for any tuple of $k+1$ edges denoted by $\mathcal{E} = (e'_1, \dots, e'_{k+1})$, $f^{-1}(\mathcal{E})$ is either a set containing one path or an empty set. There is at most one path that uses e'_i as the $(2i-1)$ -th edge of the path for all i . Thus, we can conclude that the number of paths of length $2k+1$ is at most the number of sets of $k+1$ edges, which is $m^{k+1} = O(\delta^{k+1} n^{k+1})$.

Next, let us consider $\#P_{2k}(G)$. Let f be a function that maps a path of length $2k$ to a tuple of k edges, defined as $f(e_1, \dots, e_{2k}) = (e_1, e_3, \dots, e_{2k-1})$. We observe that, for any tuple of k edges denoted by $\mathcal{E} = (e'_1, \dots, e'_k)$, $f^{-1}(\mathcal{E})$ is a set of size no larger than n . There is at most one path of length $2k-1$ that uses e'_i as the $(2i-1)$ -th edge of the path, and there are at most n possible ways to extend a path of length $2k-1$ to a path of length k . Hence, $\#P_{2k}(G) \leq n \cdot m^k = O(\delta^k n^{k+1})$. \blacktriangleleft

Recall that $\#C_k(G)$ is the number of cycles with size k in the graph G . We obtain the following theorem.

► **Theorem 15.** *For any $k \geq 1$, $\#C_{k+2}(G) \leq \frac{2}{k} \alpha(G) \#P_k(G)$.*

Proof. Let us denote $\#P_k^{(i)}$ the number of paths of length k that have node ν_i as an extremity and $\#C_k^{(i,j)}$ the number of cycles of length k containing edge (ν_i, ν_j) . Using these notations, we have $\#C_{k+2} = \frac{1}{k} \sum_{(\nu_i, \nu_j) \in E} \#C_{k+2}^{(i,j)}$. Consider the number $\#C_{k+2}^{(i,j)}$. For a path of length k that has a node ν_i as a terminal, there is at most one cycle of length $k+2$ which includes this path and the edge (ν_i, ν_j) . Therefore, we conclude that $\#C_{k+2}^{(i,j)} \leq \#P_k^{(i)}$. Similarly, we have $\#C_{k+2}^{(i,j)} \leq \#P_k^{(j)}$. Hence,

$$\#C_{k+2} \leq \frac{1}{k} \sum_{(\nu_i, \nu_j) \in E} \min(\#P_k^{(i)}, \#P_k^{(j)}).$$

For any function $h : E \rightarrow \{1, \dots, n\}$ such that for all $e = (\nu_i, \nu_j) \in E$, $h(e)$ is equal to either i or j , $\min(\#P_k^{(i)}, \#P_k^{(j)}) \leq \#P_k^{(h(e))}$. By definition of the arboricity, there exists a set of disjoint forests $\{F_l\}_{l=1, \dots, \alpha(G)}$ such that $E = \bigcup_{l=1}^{\alpha(G)} F_l$. By choosing a root for each tree of these forests, we can introduce a function h such that each edge has its child node as an image. In this way, each node can only be the image of one edge per forest. This leads to

$$\begin{aligned} \#C_{k+2} &\leq \frac{1}{k} \sum_{l=1}^{\alpha(G)} \sum_{(\nu_i, \nu_j) \in F_l} \min(\#P_k^{(i)}, \#P_k^{(j)}) \leq \frac{1}{k} \sum_{l=1}^{\alpha(G)} \sum_{e \in F_l} \#P_k^{(h(e))} \leq \frac{1}{k} \sum_{l=1}^{\alpha(G)} \sum_{i \in V} \#P_k^{(i)} \\ &= \frac{2}{k} \alpha(G) \#P_k. \end{aligned}$$

The last step is justified by the fact that each path having two extremities, the sum of all the paths of length k starting with node ν_i is twice the number of paths of length k . ◀

Combining Lemma 14 and Theorem 15, we obtain the following corollary¹.

► **Corollary 16.** For $k \geq 1$, $\#C_{2k+2} = \mathcal{O}(\delta^{k+1}n^{k+1})$ and $\#C_{2k+1} = \mathcal{O}(\delta^{k+1}n^k)$.

Next, we focus on the number of cycles of length $2k$ for any $k \geq 2$, in which three consecutive vertices of the cycle exhibit monotonic ranks C_{2k}^* , as illustrated in Table 2. Throughout the rest of this article, we will denote the count of such subgraphs in G by $\#C_{2k}^*(G)$, omitting G from the notation when the context is clear. In the following theorem, for simplicity, we extend the notation by assuming $\#P_{-1}(G) = 1$ and $\#P_0(G) = n$ for every graph G .

► **Theorem 17.** For $k \geq 2$, $\#C_{2k}^*(G) \leq 2\alpha(G)S_2^*(G)\#P_{2k-5}(G)$.

Proof. Let $\#C_{2k}^{*(i,j)}(G)$ represent the number of subgraphs in G where three consecutive vertices exhibit monotonic ranks, with (ν_i, ν_j) being the edge immediately following these consecutive vertices. Also, for $k \geq 2$, let the number of paths of length p with a low-2-star as one of its extremities be denoted as $\#P_p^*$. Since we can construct at most one path included in $\#P_p^*$ where a low-2-star and a path of length $p - 3$ are its extremities, we obtain the inequality $\#P_p^* \leq S_2^* \cdot \#P_{p-3}$.

Let $C_{2k}^{*(i,j)}$ be a cycle which is counted in $\#C_{2k}^{*(i,j)}$. Consider the path in $C_{2k}^{*(i,j)}$ of length $2k - 2$ starting from ν_i that does not pass through ν_j and the other path in $C_{2k}^{*(i,j)}$ of the same length starting from ν_j that does not pass through ν_i . We observe that one extremity of the two paths is a low-2-star. Hence, $\#C_{2k}^{*(i,j)} \leq \min(\#P_{2k-2}^{*(i)}, \#P_{2k-2}^{*(j)})$ when $\#P_p^{*(i)}$ is the number of paths in the count of $\#P_p^*$ that have ν_i as an extremity. Using the same definition of h as in the proof of Theorem 15, we have

$$\begin{aligned} \#C_{2k}^*(G) &\leq \sum_{(\nu_i, \nu_j) \in E} \#C_{2k}^{*(i,j)} \leq \sum_{(\nu_i, \nu_j) \in E} \min(\#P_{2k-2}^{*(i)}, \#P_{2k-2}^{*(j)}) \\ &\leq \sum_{l=1}^{\alpha(G)} \sum_{(\nu_i, \nu_j) \in F_l} \min(\#P_{2k-2}^{*(i)}, \#P_{2k-2}^{*(j)}) \leq \sum_{l=1}^{\alpha(G)} \sum_{e \in F_l} \#P_{2k-2}^{*(h(e))} \\ &\leq \sum_{l=1}^{\alpha(G)} \sum_{i \in V} \#P_{2k-2}^{*(i)} \leq 2\alpha(G)\#P_{2k-2}^* \leq 2\alpha(G)S_2^*\#P_{2k-5}. \end{aligned} \quad \blacktriangleleft$$

The next corollary follows Theorem 13, 17, and Lemma 14.

► **Corollary 18.** For $k \geq 2$, $\mathbb{E}[C_{2k}^*(G^\phi)] = \mathcal{O}(\delta^{k+1}n^{k-1})$.

The next corollary considers the number of edge sets in G^ϕ with specific properties.

► **Corollary 19.** For any $p \in \mathbb{N}$, we consider edge sets $E \subseteq E^\phi$ of size $2p$ such that 1) for some $c > 0$, there exists a set of cycles C_1, \dots, C_c in G^ϕ where $C_1 \cup \dots \cup C_c = E$ and $C_i \cap C_j = \emptyset$ for $i \neq j$, and 2) at least one of C_1, \dots, C_c contains three consecutive vertices of monotonic index. The number of such edge sets is $\mathcal{O}(\delta^{p+1}n^{p-1})$.

¹ We note that this result was independently established in [25] by a different proof. We are grateful to the anonymous reviewer for bringing this to our attention.

Proof. Consider a partition of $2p$, denoted by (p_1, \dots, p_c) , where $p_1 + \dots + p_c = 2p$. The number of such partitions is a function of p and can be considered constant. We will demonstrate that the number of cycle sets C_1, \dots, C_c satisfying the conditions in the corollary statement, with $|C_i| = p_i$, is at most $\mathcal{O}(\delta^{p+1}n^{p-1})$. Therefore, the number of cycle sets satisfying the corollary statement is no more than $\mathcal{O}(\delta^{p+1}n^{p-1})$.

To prove the bound, we will consider two cases: either all the cycles have even lengths, or at least two of them have odd lengths, given that the total number of edges is even.

If all the cycles are of even length, then, for some $q > 0$ one of them is of length $2q$ and includes 3 consecutive vertices of monotonic index. By Corollary 18, there are $\mathcal{O}(\delta^{q+1}n^{q-1})$ possibilities for this cycle. For the remaining cycles, Corollary 16 tells us that the number of admissible configurations is bounded by $\mathcal{O}(\delta^{p-q}n^{p-q})$. In total, this gives a $\mathcal{O}(\delta^{p+1}n^{p-1})$ bound. If at least two cycles have odd lengths, say $2q+1$ and $2r+1$, then by Corollary 16, the number of possible configurations for these cycles can be bounded by $\mathcal{O}(\delta^{q+1}n^q)$ for the first cycle and $\mathcal{O}(\delta^{r+1}n^r)$ for the second cycle, and $\mathcal{O}(\delta^{p-q-r-1}n^{p-q-r-1})$ for the remaining cycles. Overall, this results in a bound of $\mathcal{O}(\delta^{p+1}n^{p-1})$. ◀

4 Triangle Counting Algorithm

We propose Algorithm 2 to count the number of triangles based on the ordering and properties discussed in the previous section. First, we execute Algorithm 1 at Line 2. Next, at Line 3, we use the randomized response query to obtain an obfuscated graph. From Lines 4 to 8, we employ the Laplacian query with restricted sensitivity on \mathcal{H}_d (Definition 5) to estimate the number of triangles associated with User i . Finally, at Line 9, we sum all the estimates and report the total as the estimated triangle count. We adopt the concept from [22] of distributing randomized response results to all nodes and having each node estimate its number of triangles. However, the other algorithmic ideas presented in this work are novel. In the following theorem, we demonstrate that our algorithm is differentially private.

■ **Algorithm 2** Our algorithm for estimating the number of triangles in degeneracy-bounded graphs.

```

1 Function TriangleCounting
   Input: Graph  $G = (V, E)$ , privacy budget  $\varepsilon = \varepsilon_0 + \varepsilon_1 + \varepsilon_2$ , parameter  $\zeta$ 
   Output: Estimation of the number of triangles in  $G$ 
2   [All Users and Server]  $\phi \leftarrow \text{GetOrdering}(G, \varepsilon_0)$  (Algorithm 1);
3   [All Users and Server] Inquire the unbiased randomized response query with privacy
   budget  $\varepsilon_1$  to all users. Let  $(\hat{a}_{j,k}^\phi)$  represent the results collected from this query. The
   server then distributes  $(\hat{a}_{j,k}^\phi)$  to all users.
4   [User  $i$ ]  $\hat{d}_i^\phi \leftarrow \hat{d}_i^\phi + \frac{1}{\varepsilon_0} \ln(n/\zeta)$ ;
5   [User  $i$ ]  $a_i^\phi \leftarrow \mu_{\hat{d}_i^\phi}(a_i^\phi)$  (The function  $\mu_d$  is defined before Definition 5.);
6   [User  $i$ ]  $S_i \leftarrow \{(j, k) \mid a_{i,j}^\phi = a_{i,k}^\phi = 1, j < i < k\}$ ;
7   [User  $i$ ]  $\hat{t}_i \leftarrow \sum_{(j,k) \in S_i} \hat{a}_{j,k}^\phi$ ;
8   [User  $i$ ]  $\tilde{t}_i \leftarrow \hat{t}_i + 3 \cdot \text{Lap}\left(\frac{e^{\varepsilon_1} + 1}{e^{\varepsilon_1} - 1} \cdot \frac{\hat{d}_i^\phi}{\varepsilon_2}\right)$ ;
9   [User  $i$ ] Upload  $\tilde{t}_i$  to the central server;
10  [Server]  $\hat{f}_\Delta(G) \leftarrow \sum_{v_i \in V} \tilde{t}_i$ ;
11  return  $\hat{f}_\Delta(G)$ ;

```

► **Theorem 20.** *Algorithm 2 provides $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$ -edge local differential privacy.*

Proof. For all possible executions of Algorithm 2, it inquires three queries to all users. They are 1) the Laplacian query with privacy budget ε_0 inside the `GetOrdering` function at Line 2, 2) the unbiased randomized response query with privacy budget ε_1 at Line 3, and 3) the Laplacian query with restricted sensitivity on \mathcal{H}_d at Lines 4-8.

To prove this theorem, we only need to show that the query at Lines 4-8 is ε_2 -edge local differentially private. The query aims to publish $f(a_i^\phi) = \sum_{(j,k) \in S_i} \hat{a}_{j,k}^\phi$. By the unbiased randomized response in Line 3, we have that, for any j, k, j', k' , $|\hat{a}_{j,k}^\phi - \hat{a}_{j',k'}^\phi| \leq \frac{e^{\varepsilon_1} + 1}{e^{\varepsilon_1} - 1}$. It can be shown that, for $a_i^\phi, a_i'^\phi \in \mathcal{H}_{\hat{d}_i^\phi}$ (defined in Definition 5) such that $d(a_i^\phi, a_i'^\phi) \leq d$, the number of different elements in the set S_i obtained from $a_i^\phi, a_i'^\phi$ at Line 6 is at most $d \cdot \hat{d}_i^\phi$. Therefore, the restricted sensitivity of the function f (denoted by $RS_f(\mathcal{H}_{\hat{d}_i^\phi})$ in Definition 5) is not larger than $d \cdot \hat{d}_i^\phi \cdot \frac{e^{\varepsilon_1} + 1}{e^{\varepsilon_1} - 1} \cdot \frac{1}{d} = \hat{d}_i^\phi \cdot \frac{e^{\varepsilon_1} + 1}{e^{\varepsilon_1} - 1}$. Hence, by Definition 5, the publication of \tilde{t}_i at Line 8 is ε_2 -edge local differentially private. \blacktriangleleft

We now discuss the accuracy of our estimation and its relation with the parameter ζ appearing at Line 4 the algorithm. We will see that ζ controls the trade off between the bias and the accuracy. The smaller ζ is, the smaller the average noise gets, but the larger the probability of bias and its expected magnitude is.

In the following lemma, we discuss that the projection $\mu_{\hat{d}_i^\phi}$ applied at Line 5 changes the adjacency vector a_i^ϕ only with small probability.

► **Lemma 21.** *For any $\zeta > 0$, with probability at least $1 - \zeta$, $|\tilde{d}_i - d_i| < (\ln \frac{n}{\zeta})/\varepsilon_0$ for all i .*

Proof. Using the cumulative distribution function of the Laplacian random variable, we have $\mathbb{P}\left[|\tilde{d}_i - d_i| \geq \varepsilon_0 \ln \frac{n}{\zeta}\right] \leq \frac{\zeta}{n}$. Thus, by taking this inequality for all $i \in [1, n]$, and using the union bound, we obtain $\mathbb{P}\left[\exists i \in [1, n], |\tilde{d}_i - d_i| \geq \varepsilon_0 \ln \frac{n}{\zeta}\right] \leq \zeta$. \blacktriangleleft

We show that our estimation has no bias with high probability in the subsequent theorem.

► **Theorem 22.** *With probability at least $1 - \zeta$, algorithm 2 provides an unbiased estimate of the number of triangles in the graph, i.e. $\mathbb{E}\left[\hat{f}_\Delta(G)\right] = \#C_3(G)$.*

Proof. As discussed in Definition 7, we have that $\mathbb{E}(\hat{a}_{j,k}^\phi) = a_{j,k}^\phi$. Using Lemma 21, with probability at least $1 - \zeta$, \tilde{d}_i^ϕ is larger than d_i^ϕ for all $i \in [1, n]$, and the function $\mu_{\hat{d}_i^\phi}$ has no effect. Consequently, S_i precisely represents the set of forks centered on node ν_i , encompassing all possible triangles. Therefore, \hat{t}_i is an unbiased estimate of the number of triangles (ν_i, ν_j, ν_k) such that $j < i < k$. Given that Laplace noise is centered and triangles can be decomposed accordingly, $\hat{f}_\Delta(G)$ is an unbiased estimation of $f_\Delta(G)$. \blacktriangleleft

Corollary 23 ensures that even in the unlikely event of some clipping occurring, the resulting bias would still represent only a small fraction of the actual count.

► **Corollary 23.** *The expected value of the bias of Algorithm 2 is bounded by $\mathcal{O}\left(\frac{\zeta}{\varepsilon_0 n} \#C_3\right)$.*

Proof. When the corrected estimated degree \hat{d}_i^ϕ is smaller than the actual degree d_i , $d_i - \hat{d}_i^\phi$ edges are excluded. This exclusion introduces a bias because the potential triangles involving these excluded edges are not counted. For each user i and their neighbor j , let $t_i^{(j)}$ denote the number of triangles counted by user i that involve the edge (ν_i, ν_j) . We also define $t_i^{\max} = \max_j t_i^{(j)}$. Then, the maximum bias resulting from a single clipped edge can be bounded by t_i^{\max} .

The expected number of clipped edges for user i is determined by evaluating the following integral, where $\beta = \frac{\ln(n/\zeta)}{\varepsilon_0}$ serves as the correction term for the degree:

$$\int_{-\infty}^{-\beta} \frac{\varepsilon_0}{2} e^{-\varepsilon_0|x|} (-x - \beta) dx = -\frac{\varepsilon_0}{2} \left[\frac{x - \beta}{\varepsilon_0} e^{-\varepsilon_0 x} + \frac{1}{\varepsilon_0^2} e^{-\varepsilon_0 x} \right]_{\beta}^{\infty} = \frac{1}{2\varepsilon_0} e^{-\varepsilon_0 \beta} = \frac{\zeta}{2\varepsilon_0 n}.$$

We obtain the final result by combining these elements and observing that $\sum_i t_i^{\max} \leq \sum_{i,j} t_i^{(j)} \leq 2f_{\Delta}(G)$. \blacktriangleleft

The accuracy of our estimation is demonstrated in the subsequent theorem.

► **Theorem 24.** *When $\zeta \leq \varepsilon_0$, the squared expected ℓ_2 -error of algorithm 2 is bounded by*

$$\mathcal{O} \left(\frac{\delta^3 n}{\varepsilon_1^2} + \frac{\delta d_{\max} n}{\varepsilon_1^2 \varepsilon_2^2} + \frac{n \ln^2(n/\zeta)}{\varepsilon_0^2 \varepsilon_1^2 \varepsilon_2^2} \right).$$

Proof. The squared ℓ_2 -error can be decomposed into the square of the bias plus the variance. We have established in Corollary 23 that the bias of the algorithm is bounded by $\mathcal{O} \left(\frac{\zeta}{\varepsilon_0 n} \#C_3 \right) = \mathcal{O}(\delta^2)$. We will now focus on bounding the variance of the algorithm. This variance arises from two distinct sources: the randomized response query and the Laplacian query with restrictive sensitivity.

Regarding the noise introduced by the Laplacian query with restrictive sensitivity, its variance is simply the sum of the variances of each term, which is

$$9 \left(\frac{e^{\varepsilon_1} + 1}{e^{\varepsilon_1} - 1} \right)^2 \sum_{\nu_i \in V} \frac{\hat{d}_i^2}{\varepsilon_2^2} = \mathcal{O} \left(\frac{\varepsilon_0^2 \delta d_{\max} n + n \ln^2(n/\zeta)}{\varepsilon_2^2 \varepsilon_1^2 \varepsilon_0^2} \right).$$

Next, we consider the variance from the randomized response query. In the following equations, we use the notation $\mathcal{N}_{j,k}^*$ to denote the set of neighbors ν_i of both ν_j and ν_k such that $j < i < k$. Note that by including one node from $\mathcal{N}_{j,k}^*$ along with ν_j and ν_k , a triple in S_2^* is formed. Similarly, including two nodes from $\mathcal{N}_{j,k}^*$ along with ν_j and ν_k results in a quadruplet in $\#C_4^*$. We also notice from Definition 7 that, for $(j, k) \neq (j', k')$, $\hat{a}_{j,k}^{\phi}$ is independent to $\hat{a}_{j',k'}^{\phi}$ and $\text{Cov} \left(\hat{a}_{j,k}^{\phi}, \hat{a}_{j',k'}^{\phi} \right) = 0$. Hence,

$$\begin{aligned} \text{Var} \left(\sum_{\nu_i \in V^{\phi}} \sum_{(j,k) \in S_i} \hat{a}_{j,k}^{\phi} \right) &= \sum_{(\nu_j, \nu_k) \in (V^{\phi})^2} \left[\sum_{\nu_i \in \mathcal{N}_{j,k}^*} \text{Var} \left(\hat{a}_{j,k}^{\phi} \right) + \sum_{\nu_i, \nu_{i'} \in \mathcal{N}_{j,k}^*} \text{Cov} \left(\hat{a}_{j,k}^{\phi}, \hat{a}_{j,k}^{\phi} \right) \right] \\ &= \mathcal{O} \left((S_2^* + \#C_4^*) / \varepsilon_1^2 \right) \end{aligned}$$

By Theorem 13 and Collorary 18, $\text{Var} \left(\hat{f}(G) \right) = \mathcal{O} \left(\frac{\delta^3 n}{\varepsilon_1^2} + \frac{\delta d_{\max} n}{\varepsilon_1^2 \varepsilon_2^2} + \frac{n \ln^2(n/\zeta)}{\varepsilon_0^2 \varepsilon_1^2 \varepsilon_2^2} \right)$. \blacktriangleleft

In the previous work [16], the number of terms in the variance calculation is bounded by the number of cycles of length four, which is $\mathcal{O}(d_{\max}^3 n)$. We reduce that number to $\#C_4^* = \mathcal{O}(\delta^3 n)$ using the `GetOrdering` function in Line 2 and by including only pairs (j, k) such that $j < i < k$. It is known that $\delta \leq d_{\max}$ and, in many practical graphs, the degeneracy is much smaller than the maximum degree.

5 Odd Length Cycle Counting

In this section, we will describe how to utilize low-degree ordering to accurately count odd-length cycles in graphs with bounded degeneracy. Some concepts are extended from the previous section. As shown in Algorithm 3, the algorithm for estimating the number of odd-length cycles is similar to Algorithm 2, except that the restricted sensitivity at Line 9 is larger, and at Line 8, we replace $\hat{a}_{i,j}^\phi$ with an estimate for the number of paths under specific constraints. We discuss the privacy of the algorithm in the subsequent theorem. The main challenge of the proof is to demonstrate that the Laplacian query under restricted sensitivity at Lines 5-9 is ε_2 -differentially private.

Algorithm 3 Our algorithm for estimating the number of odd-length cycles in degeneracy-bounded graphs.

1 **Function** OddCycleCounting

Input: Graph $G = (V, E)$, privacy budget $\varepsilon = \varepsilon_0 + \varepsilon_1 + \varepsilon_2$, k an odd number not smaller than 5, parameter ζ

Output: Estimation of the number of k -cycles in G

2 **[All Users and Server]** $\phi \leftarrow \text{GetOrdering}(G, \varepsilon_0)$ (Algorithm 1);

3 **[All Users and Server]** Inquire the unbiased randomized response query with privacy budget ε_1 to all users.

4 **[Server]** Let $(\hat{a}_{i,j}^\phi)$ represent the results collected from this query. The server then distributes $(\hat{a}_{i,j}^\phi)$ to all users.

5 **[Server]** Calculate

$$\#\hat{P}_{k-4} := \sum_{(l_1, \dots, l_{k-3}) \in V^{k-3}} \prod_{q \in [1, k-4]} \hat{a}_{l_q, l_{q+1}}^\phi,$$

then send this information to all users;

6 **[User i]** $\hat{d}_i^\phi \leftarrow \bar{d}_i^\phi + \frac{1}{\varepsilon_0} \ln(n/\zeta)$;

7 **[User i]** $a_i^\phi \leftarrow \mu_{\bar{d}_i^\phi}(a_i^\phi)$;

8 **[User i]** $S_i \leftarrow \{(j, \kappa) \mid a_{i,j}^\phi = a_{i,\kappa}^\phi = 1, j < i < \kappa\}$;

9 **[User i]** $\hat{c}_i \leftarrow \sum_{(j, \kappa) \in S_i} \#\hat{P}_{k-2}^{(i)}(j, \kappa)$ when

$$\#\hat{P}_{k-2}^{(i)}(j, \kappa) = \sum_{(l_1, \dots, l_{k-1}) \in X_{k-2}^{(i)}(j, \kappa)} \prod_{q \in [1, k-2]} \hat{a}_{l_q, l_{q+1}}^\phi$$

and $X_{k-2}^{(i)}(j, \kappa)$ is a set of non-repeating combination of $k-1$ vertices in G^ϕ with endpoints ν_j and ν_κ , such that, for any three consecutive nodes (ν_q, ν_r, ν_s) in the path with monotonic ranks, the node ν_i has a lower rank than ν_r ;

10 **[User i]** $\tilde{c}_i \leftarrow \hat{c}_i + \text{Lap}\left(3 \cdot \left(\frac{\varepsilon_1+1}{\varepsilon_1-1}\right)^2 \cdot \hat{d}_i^\phi \cdot \#\hat{P}_{k-4}/\varepsilon_2\right)$;

11 **[User i]** Upload \tilde{c}_i to the central server;

12 **[Server]** $\hat{f}_k(G) \leftarrow \sum_{\nu_i \in V} \tilde{c}_i$;

13 **return** $\hat{f}_k(G)$;

► **Theorem 25.** *Algorithm 3 provides $(\varepsilon_0 + \varepsilon_1 + \varepsilon_2)$ -edge local differential privacy.*

Proof. We need to demonstrate that Lines 5-9 of the algorithm, involving the Laplacian query with restricted sensitivity on $\mathcal{H}_{\hat{d}_i^\phi}$, ensure ε_2 -differential privacy. Following the arguments of Theorem 20, we assert that altering d entries of $a_{i,j}^\phi$ changes the set S_i by at most $d \cdot \hat{d}_i^\phi$ elements. A single element change in S_i can alter the value of \hat{c}_i by

$$\#\hat{P}_{k-2}^{(i)}(j, \kappa) = \sum_{(l_1, \dots, l_{k-1}) \in X_{k-2}^{(i)}(j, \kappa)} \prod_{q \in [1, k-2]} \hat{a}_{l_q, l_{q+1}}^\phi \leq \left(\frac{e^\varepsilon + 1}{e^\varepsilon - 1} \right)^2 \#\hat{P}_{k-4}.$$

Therefore, the restricted sensitivity of \tilde{c}_i is $\mathbf{d} \cdot \hat{d}_i^\phi \left(\frac{e^\varepsilon + 1}{e^\varepsilon - 1} \right)^2 \#\hat{P}_{k-4} / \mathbf{d} = \hat{d}_i^\phi \left(\frac{e^\varepsilon + 1}{e^\varepsilon - 1} \right)^2 \#\hat{P}_{k-4}$. Consequently, the publication of \tilde{c}_i at Line 9 is ε_2 -differentially private. ◀

The bias of the algorithm is given in the following theorem.

► **Theorem 26.** *With a probability of at least $1 - \zeta$, Algorithm 3 provides an unbiased estimate of the number of k -cycles in any graph G for any odd integer k .*

Proof. Since we publish $\hat{a}_{l_q, l_{q+1}}^\phi$ using the unbiased randomized response query, the publication is an unbiased estimation of $a_{l_q, l_{q+1}}^\phi$. Furthermore, as those estimators are independent from one another, for each $(j, \kappa) \in S_i$ and $\{l_1, \dots, l_{k-1}\} \in X_{k-2}^{(i)}(j, \kappa)$, $\prod_{q \in [1, k-2]} \hat{a}_{l_q, l_{q+1}}^\phi$ is an unbiased estimate of $\prod_{q \in [1, k-2]} a_{l_q, l_{q+1}}^\phi$. It results from this that $\#\hat{P}_{k-2}^{(i)}(j, \kappa)$ is an unbiased estimator of the number of paths between j and κ with length $k - 2$ such that, for any three consecutive nodes (ν_q, ν_r, ν_s) with monotonic ranks, the node ν_i has a lower rank than ν_r . We denote the number of such paths as $\#P_{k-2}^{(i)}(j, \kappa)$.

Let us introduce $C_k^{(i)} = \sum_{(j, \kappa) \in S_i} \#P_{k-2}^{(i)}(j, \kappa)$. Assuming no clipping occurs, which happens with a probability of at least $1 - \zeta$, we have by linearity of expectation that both \hat{c}_i and \tilde{c}_i are unbiased estimators of $C_k^{(i)}$. Therefore, all that remains to be proven is that the number of k -cycles in G is equal to $\sum_{\nu_i \in V^\phi} C_k^{(i)}$. It is evident that for each element counted in $\sum_{\nu_i \in V^\phi} C_k^{(i)}$, there is a corresponding cycle $(\nu_i, l_1, \dots, l_{k-1})$ in G^ϕ and, also, in G .

Conversely, consider a cycle of length k in G . Since it is also a cycle in G^ϕ , we can represent it in G^ϕ as (ν_1, \dots, ν_k) . Because the cycle is of odd length, there exist three consecutive nodes with a monotonic rank. Among all possible triplets, consider the one where the central node has the smallest rank, denoted as $(\nu_j, \nu_i, \nu_\kappa)$ with $j < i < \kappa$. Furthermore, let $j = l_1$ and $\kappa = l_{k-1}$, and assign the indices of the other nodes in the cycle to l_2 through l_{k-2} in the order they appear in the cycle. Thus, the cycle is counted in $\sum_{\nu_i \in V^\phi} C_k^{(i)}$. Furthermore, if any other node in the cycle were chosen as ν_i , the remaining path would not be part of $X_{k-2}^{(i)}(j, \kappa)$. This ensures that each cycle is counted exactly once in $\sum_{\nu_i \in V^\phi} C_k^{(i)}$. ◀

Finally, the ℓ_2 -error of Algorithm 3 is proven in the next theorem. The most challenging aspect of this theorem is to bound the covariance in the summation at Lines 8 and 10. We assert that any two dependent elements of $X_{k-2}^{(i)}(j, \kappa)$ can be considered as a set containing an even number of edges which forms multiple disjoint cycles with specific properties. Consequently, we can utilize our results from Corollary 19 to bound the number of such pairs. The proof of the theorem is given in the appendix of this paper.

► **Theorem 27.** *When $\zeta \leq \varepsilon_0$, the expected squared ℓ_2 -error of algorithm 3 is bounded by*

$$\mathcal{O} \left(\frac{\delta^3}{\varepsilon_1^2} \left(\frac{1}{\varepsilon_1^2} + \delta \right)^{k-3} n^{k-2} + \frac{\delta^{k-2} d_{max} n^{k-2}}{\varepsilon_2^2 \varepsilon_1^4} + \frac{\delta^{k-3} n^{k-2} \ln^2(n/\zeta)}{\varepsilon_2^2 \varepsilon_1^4 \varepsilon_0^2} \right).$$

Before proving Theorem 27, we demonstrate the following lemma.

► **Lemma 28.** *The expected value of the bias of Algorithm 3 is bounded by $\mathcal{O} \left(\frac{\zeta}{\varepsilon_0 n} \#C_k \right)$.*

Proof. We have already seen the proof of Corollary 23 that the expected value of the number of clipped edges for user i was bounded by $\frac{\zeta}{2\varepsilon_0 n}$. We now have to bound the bias created by one edge removal, i.e. the maximal number of cycles one edge can part of.

With $c_i^{(j)}$ the number of cycles counted by i that involve edge (i, j) , the maximal bias for user i is bounded by $\sum_j c_i^{(j)}$, and the bias of the algorithm by $\frac{\zeta}{2\varepsilon_0 n} \sum_{i,j} c_i^{(j)} \leq \mathcal{O}\left(\frac{\zeta}{\varepsilon_0 n} \#C_k\right)$. \blacktriangleleft

Now, we are ready to prove Theorem 27.

Proof of Theorem 27. The squared ℓ_2 -error can be decomposed into the square of the bias plus the variance. In Lemma 28, we established that the bias of the algorithm is bounded by $\mathcal{O}\left(\frac{\zeta}{\varepsilon_0 n} \#C_k\right) = \mathcal{O}\left(\delta^{\frac{k+1}{2}} n^{\frac{k-3}{2}}\right)$. We will now focus on bounding the variance of the algorithm.

Let the indicator variable $\mathbb{1}_{(l_1, \dots, l_p)}$ be 1 if the path $(\nu_{l_1}, \dots, \nu_{l_p})$ exists in G^ϕ , and 0 otherwise. We also denote the random variable $\prod_{q \in [1, p]} \hat{a}_{l_q, l_{q+1}}^\phi$ by $Z_{(l_1, \dots, l_{p+1})}$. Finally, we define $U_{(l_1, \dots, l_{p+1})} = Z_{(l_1, \dots, l_{p+1})} - \mathbb{1}_{(l_1, \dots, l_{p+1})}$. This random variable $U_{(l_1, \dots, l_{p+1})}$ has the properties that $\mathbb{E}[U_{(l_1, \dots, l_{p+1})}] = 0$ and $\text{Var}(U_{(l_1, \dots, l_{p+1})}) = \text{Var}(Z_{(l_1, \dots, l_{p+1})})$.

Similar to the case with triangles, the variance of Algorithm 3 arises from both the unbiased randomized response query and the Laplacian query with restricted sensitivity.

Concerning the variance term coming from the randomized response, we have to compute the variance of

$$\hat{C} = \sum_{\nu_i \in V} (\hat{c}_i - c_i) = \sum_{\nu_i \in V} \sum_{(j, \kappa) \in S_i} \sum_{\{l_1, \dots, l_{k-1}\} \in X_{k-2}^{(i)}(j, \kappa)} U_{(l_1, \dots, l_{k-1})}.$$

We have to take into account the term that comes from the sum of the variances of the U as well as the one coming from the covariances between them.

To compute the sum of variances, we start with:

$$\text{Var}(U_{(l_1, \dots, l_{k-1})}) = \prod_{q \in [1, k-2]} \text{Var}(\hat{a}_{l_q, l_{q+1}}^\phi) = \mathcal{O}\left(\frac{1}{\varepsilon_1^{2k-4}}\right).$$

Additionally, for each i and $(j, \kappa) \in S_i$, the cardinality of $X_{k-2}^{(i)}(j, \kappa)$ is bounded by n^{k-3} , and the number of ways to choose (i, j, κ) is bounded by S_2^* , which is $\mathcal{O}(\delta^2 n)$ by Theorem 13. This contributes a term in the variance from the sum of variances bounded by $\mathcal{O}(\delta^2 n^{k-2} / \varepsilon_1^{2k-4})$.

To analyze the term arising from the covariances, we first examine the covariance between $U_{(l_1, \dots, l_{k-1})}$ and $U_{(l'_1, \dots, l'_{k-1})}$. In the following equations, let A be the set of edges that appear only in (l_1, \dots, l_{k-1}) or (l'_1, \dots, l'_{k-1}) , and let B be the set of edges that appear in both. Recall that, for any (i, j) $\mathbb{E}[\hat{a}_{i,j}^\phi] = 0$ and $\mathbb{E}[\hat{a}_{i,j}^\phi] = \text{Var}(\hat{a}_{i,j}^\phi)$.

$$\begin{aligned} & \text{Cov}\left(U_{(l_1, \dots, l_{k-1})}, U_{(l'_1, \dots, l'_{k-1})}\right) \\ &= \mathbb{E}\left[\prod_{q \in [1, k-2]} \hat{a}_{l_q, l_{q+1}}^\phi \prod_{q \in [1, k-2]} \hat{a}_{l'_q, l'_{q+1}}^\phi\right] - \mathbb{1}_{(l_1, \dots, l_{k-1})} \mathbb{1}_{(l'_1, \dots, l'_{k-1})} \\ &= \prod_{(i,j) \in A} \mathbb{1}_{(i,j)} \prod_{(i,j) \in B} \text{Var}(a_{i,j}^\phi) - \prod_{q \in [1, k-2]} \mathbb{1}_{(l_q, l_{q+1})} \mathbb{1}_{(l'_q, l'_{q+1})}. \end{aligned}$$

We observe that the covariance between $U_{(l_1, \dots, l_{k-1})}$ and $U_{(l'_1, \dots, l'_{k-1})}$ is zero if the paths $(\nu_{l_1}, \dots, \nu_{l_{k-1}})$ and $(\nu_{l'_1}, \dots, \nu_{l'_{k-1}})$ do not share at least one common edge or if the edges present in only one of the paths are not present in the original graph. Now consider the situation where the covariance is non-zero. We have that $|B| > 0$. Additionally, we will denote ν_i the node responsible for counting this instance of $U_{(l_1, \dots, l_{k-1})}$ and $\nu_{i'}$ the one responsible for $U_{(l'_1, \dots, l'_{k-1})}$.

Let $\mathbf{V} := \{\nu_i, \nu_{l_1}, \dots, \nu_{l_{k-1}}, \nu_{i'}, \nu_{l'_1}, \dots, \nu_{l'_{k-1}}\}$, and let

$$\mathbf{E} := \bigcup_{1 \leq q \leq k-2} \{(\nu_{l_q}, \nu_{l_{q+1}}), (\nu_{l'_q}, \nu_{l'_{q+1}})\} \cup \{(\nu_{l_{k-1}}, \nu_i), (\nu_i, \nu_{l_1}), (\nu_{l'_{k-1}}, \nu_{i'}), (\nu_{i'}, \nu_{l'_1})\}.$$

In other words, the set \mathbf{E} consists of the edges in the paths $\{l_1, \dots, l_{k-1}\}$ and $\{l'_1, \dots, l'_{k-1}\}$, along with the additional edges $(\nu_{l_{k-1}}, \nu_i)$, (ν_i, ν_{l_1}) , $(\nu_{l'_{k-1}}, \nu_{i'})$, and $(\nu_{i'}, \nu_{l'_1})$. Additionally, let

$$\begin{aligned} A' &:= A \cup \{(\nu_{l_{k-1}}, \nu_i), (\nu_{l'_{k-1}}, \nu_{i'}) \mid (\nu_{l_{k-1}}, \nu_i) \neq (\nu_{l'_{k-1}}, \nu_{i'})\} \\ &\quad \cup \{(\nu_i, \nu_{l_1}), (\nu_{i'}, \nu_{l'_1}) \mid (\nu_i, \nu_{l_1}) \neq (\nu_{i'}, \nu_{l'_1})\}. \end{aligned}$$

Similarly, let

$$B' := B \cup \{(\nu_{l_{k-1}}, \nu_i) \mid (\nu_{l_{k-1}}, \nu_i) = (\nu_{l'_{k-1}}, \nu_{i'})\} \cup \{(\nu_i, \nu_{l_1}) \mid (\nu_i, \nu_{l_1}) = (\nu_{i'}, \nu_{l'_1})\}.$$

In other words, the sets A' and B' are the sets A and B extended to include the additional edges (ν_i, ν_{l_1}) , $(\nu_{l_{k-1}}, \nu_i)$, $(\nu_{i'}, \nu_{l'_1})$, and $(\nu_{l'_{k-1}}, \nu_{i'})$.

We introduce \mathbf{d} the difference between the cardinal of B' and B , $\mathbf{d} := |B'| - |B|$. Let $q \in [1, k-2]$ be the cardinality of B . In this case, the covariance is $\mathcal{O}\left(1/\varepsilon_1^{2q}\right)$. We have that $|A'| + 2|B'| = 2k$, which gives $|A'| = 2k - 2q - 2\mathbf{d}$.

In the next step, we will calculate the number of the pairs of paths with $|A'| = 2(k - q - \mathbf{d})$. Let us consider the degree of each node in (\mathbf{V}, \mathbf{E}) . It is clear that the degrees are neither greater than four nor less than two. A node has a degree of three only if one of the three edges incident to it belongs to B' and the other two to A' . A node has a degree of four if all four edges incident to it are in A' , and it has a degree of two if both edges incident to it are either in A' or in B' . Hence, if we consider the graph (\mathbf{V}, A') , we have a graph of degree two or four, which is a union of multiple disjoint cycles.

Let the number of those disjoint cycles be c and the size of those cycles be r_1, \dots, r_c . We have that $\sum_{t=1}^c r_t = 2k - 2q - 2\mathbf{d}$, i.e. (r_1, \dots, r_c) is a partition of $2k - 2q - 2\mathbf{d}$. We know that the number of such partitions is bounded by a function of k . Let suppose that the bound is $f(k)$.

Let us give the number of A' with cycle size (r_1, \dots, r_c) . We can use Corollary 16 to show that the number of such sets A' is $\mathcal{O}\left(\prod_{t=1}^c \delta^{r_t/2} n^{r_t/2}\right) = \mathcal{O}\left(\delta^{k-q-\mathbf{d}} n^{k-q-\mathbf{d}}\right)$. When $\mathbf{d} = 0$, we know that $\{\nu_i, \nu_j\}$ and $\{\nu_i, \nu_k\}$ are in A' . There are three consecutive nodes with monotonic ranks in the union of disjoint cycles (\mathbf{V}, A') . Hence, we can use Corollary 19 to show that the number of such sets A' is bounded by $\mathcal{O}\left(\delta^{k-q+1} n^{k-q-1}\right)$. By combining the two cases, we can conclude that the number of possible sets A' with cycle size (r_1, \dots, r_c) is at most $\mathcal{O}\left(\delta^{k-q+1-\mathbf{d}} n^{k-q-1}\right)$. The number of possible A' is then $f(k) \cdot \mathcal{O}\left(\delta^{k-q+1-\mathbf{d}} n^{k-q-1}\right)$. As k is a constant, the number is $\mathcal{O}\left(\delta^{k-q+1-\mathbf{d}} n^{k-q-1}\right)$.

We then consider the number of configurations for B' , which consists of a union of disjoint paths. Let the number of paths be c and their lengths be r_1, \dots, r_c . We have that $|r_1| + \dots + |r_c| = q$, and (r_1, \dots, r_c) forms a partition of q . The number of possible partitions is bounded by a function of k , denoted as $f(k)$. Each part must begin and end in the node set A' , where $|A'| \leq 2k$. Therefore, the number of possible paths r_t is at most $4k^2 n^{r_t-1}$, and the number of possible sets B' with the partition (r_1, \dots, r_c) is at most $\prod_{t=1}^c 4k^2 n^{r_t-1} = \mathcal{O}\left(n^{q-1}\right)$. Hence, the total number of possible sets B' is $f(k) \cdot \mathcal{O}\left(n^{q-1}\right) = \mathcal{O}\left(n^{q-1}\right)$.

Consequently, for each set A' , the number of possible configurations for B' is at most $\mathcal{O}(n^{q-1})$. The number of pairs of paths $\{l_1, \dots, l_{k-1}\}$ and $\{l'_1, \dots, l'_{k-1}\}$ with $|A'| = 2(k - q - \mathbf{d})$ is then at most $\mathcal{O}(\delta^{k-q+1-\mathbf{d}} n^{k-q-1} \cdot n^{q-1}) = \mathcal{O}(\delta^{k-q+1} n^{k-2})$. Each of these pairs contributes $\text{Var}(\hat{a}_{j,k}^\phi)^{2q} = \mathcal{O}(1/\varepsilon_1^{2q})$ to the covariance sum.

The covariance of \hat{C} can then be calculated as follows:

$$\mathcal{O}\left(\sum_{q=1}^{k-2} \delta^{k-q+1} n^{k-2} \frac{1}{\varepsilon_1^{2q}}\right) = \mathcal{O}\left(\frac{n^{k-2} \delta^3}{\varepsilon_1^2} \left(\delta + \frac{1}{\varepsilon_1^2}\right)^{k-3}\right).$$

Since this bound is larger than the one for the sum of variances, we can disregard the latter.

To compute the variance resulting from the Laplacian query with restricted sensitivity, we sum the variance of the Laplacian distribution for all nodes:

$$9 \left(\frac{e^\varepsilon + 1}{e^\varepsilon - 1}\right)^4 \mathbb{E}[\#\hat{P}_{k-4}^2] \sum_{\nu_i \in V} \frac{\hat{d}_i^2}{\varepsilon_2^2} = \mathcal{O}\left(\frac{\delta d_{\max} n}{\varepsilon_2^2 \varepsilon_1^4} + \frac{n \ln^2(n/\zeta)}{\varepsilon_2^2 \varepsilon_1^4 \varepsilon_0^2}\right) \mathbb{E}[\#\hat{P}_{k-4}^2]. \quad (1)$$

Let us now consider the expected value

$$\mathbb{E}[\#\hat{P}_{k-4}^2] = \mathbb{E}[\#\hat{P}_{k-4}]^2 + \text{Var}(\#\hat{P}_{k-4}) = \#P_{k-4}^2 + \text{Var}(\#\hat{P}_{k-4}). \quad (2)$$

By Lemma 14 and the fact that $k-4$ is an odd number, we have $\#P_{k-4}^2 = \mathcal{O}(\delta^{k-3} n^{k-3})$. The variance can be decomposed into the sum of the variances of each path, which is bounded by $\mathcal{O}(n^{k-3}/\varepsilon_1^{2k-8})$, and the sum of covariances.

The covariance is non-zero only if at least two edges are shared between the two paths and all edges that appear only once exist in the original graph. As previously discussed, this forms a cycle structure, except for the path extremities that do not need to be connected. Recall the definitions of the sets A and B from the previous paragraph.

The set A consists of two paths at the extremities and multiple disjoint cycles. Suppose the number of edges in A is $2p$, the number of edges in the two paths are q_1 and q_2 , and the number of disjoint cycles is c , with the number of edges in these cycles being r_1, \dots, r_c . This gives us $2p = q_1 + q_2 + \sum_{i=1}^c r_i$. In other words, $(q_1, q_2, r_1, \dots, r_c)$ forms a partition of $2p \leq 2k$. The number of such partitions is bounded by a function of k . Let the bound be $f(k)$.

We now discuss the number of possible configurations of A for the partition $(q_1, q_2, r_1, \dots, r_c)$. From Lemma 14 and Corollary 16, the number of cycles of length q is bounded by $\mathcal{O}(\delta^{q/2} n^{q/2})$, and the number of paths of length q is bounded by $\mathcal{O}(\delta^{q/2} n^{q/2+1})$. Thus, the number of configurations for the partition $(q_1, q_2, r_1, \dots, r_c)$ is:

$$\mathcal{O}\left(\delta^{q_1/2} n^{q_1/2+1} \cdot \delta^{q_2/2} n^{q_2/2+1} \cdot \prod_{t=1}^c \delta^{r_t/2} n^{r_t/2}\right) = \mathcal{O}(\delta^p n^{p+2}).$$

Hence, the number of possible configurations for A with $2p$ edges is no more than $f(k) \cdot \mathcal{O}(\delta^p n^{p+2}) = \mathcal{O}(\delta^p n^{p+2})$.

The number of edges in B is $(2k - 8 - 2p)/2 = k - p - 4$. Using the previous argument when calculating the number of possible set B' , we obtain that the number of configurations for B is $\mathcal{O}(n^{k-p-5})$. The number of configurations with $|A| = 2p$ is then $\mathcal{O}(\delta^p n^{p+2} \cdot n^{k-p-5}) = \mathcal{O}(\delta^p n^{k-3})$. Hence, the overall number of combinations is $\sum_{p=1}^{k-5} \mathcal{O}(\delta^p n^{k-3}) = \mathcal{O}(\delta^{k-5} n^{k-3})$.

From the previous paragraph, we observe that the covariance term outweighs the sum of the variances, leading to $\text{Var}(\#\hat{P}_{k-4}) = \mathcal{O}(\delta^{k-5}n^{k-3})$. Additionally, when calculating $\mathbb{E}[\#\hat{P}_{k-4}^2]$ in (2), it is evident that $\#P_{k-4}^2$ dominates $\text{Var}(\#\hat{P}_{k-4})$, resulting in $\mathbb{E}[\#\hat{P}_{k-4}^2] = \mathcal{O}(\delta^{k-3}n^{k-3})$. Substituting $\mathbb{E}[\#\hat{P}_{k-4}^2]$ with $\mathcal{O}(\delta^{k-3}n^{k-3})$ in (1), we find that the variance from the Laplacian mechanism is bounded by

$$\mathcal{O}\left(\frac{\delta^{k-2}d_{max}n^{k-2}}{\varepsilon_2^2\varepsilon_1^4} + \frac{\delta^{k-3}n^{k-2}\ln^2(n/\zeta)}{\varepsilon_2^2\varepsilon_1^4\varepsilon_0^2}\right).$$

We obtain the theorem result by summing the variance from the unbiased randomized response query and the variance from the Laplacian query with restricted sensitivity. ◀

6 Conclusion

In this work, we introduced a private vertex ordering algorithm. The transformation on the graph induced by this ordering reduces the count of specific order-sensitive motifs while preserving the overall graph structure. Due to its reliance on the Laplacian mechanism, the algorithm performs well even in high-privacy settings, making it an excellent preprocessing step for subgraph counting queries.

Within this framework, we first propose a new triangle counting algorithm whose accuracy depends on the count of specific ordered subgraphs. By combining this algorithm with the ordering preprocessing step, we achieve an expected error of $\mathcal{O}(n)$ for graphs with bounded degeneracy, compared to the $\mathcal{O}(n^2)$ error seen in the current state of the art.

Subsequently, we extended the algorithm to address the more general case of odd-length cycle counting. We propose the first purely local differentially private counting algorithm specifically designed for cycles longer than triangles. Under the assumption of bounded degeneracy, the algorithm achieves an error of $\mathcal{O}(n^{(k-1)/2})$ for cycles of length k .

Due to the constraints of local differential privacy, it might be assumed that the range of tasks we can perform on graphs under this privacy notion is limited. However, in this work, we demonstrate that more precise information can be published under local differential privacy by restricting our inputs to certain types of graphs. We believe that parameterized algorithms under local differential privacy represent an intriguing research area that can contribute significantly to both algorithm design and information privacy.

One limitation of this method is that the relative error can become significantly large when the number of cycles is small (or even zero), even in cases where the graph's degeneracy – and consequently the ℓ_2 -error of our algorithm – is high. Identifying a class of graphs for which an algorithm with bounded relative error can be designed would be a direction for future research. Another question for future investigation is determining lower bounds for degeneracy-bounded graphs under the local differential privacy.

References

- 1 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.
- 2 Suman K. Bera, Lior Gishboliner, Yevgeny Levanzov, C. Seshadhri, and Asaf Shapira. Counting subgraphs in degenerate graphs. *ACM Journal of the ACM (JACM)*, 69(3):23:1–23:21, 2022. doi:10.1145/3520240.
- 3 Louis Betzer, Vorapong Suppakitpaisarn, and Quentin Hillebrand. Publishing number of walks and katz centrality under local differential privacy. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024. URL: <https://openreview.net/forum?id=76UkTmdmkB>.

- 4 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 87–96. ACM, 2013. doi:10.1145/2422436.2422449.
- 5 Marco Bressan. Faster algorithms for counting subgraphs in sparse graphs. *Algorithmica*, 83(8):2578–2605, 2021. doi:10.1007/s00453-021-00811-0.
- 6 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985. doi:10.1137/0214017.
- 7 Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1655–1658. ACM, 2018. doi:10.1145/3183713.3197390.
- 8 Damien Desfontaines and Balázs Pej6. Sok: Differential privacies. *Proc. Priv. Enhancing Technol.*, 2020(2):288–313, 2020. doi:10.2478/popets-2020-0028.
- 9 Laxman Dhulipala, George Z. Li, and Quanquan C. Liu. Near-optimal differentially private k-core decomposition. *CoRR*, abs/2312.07706, 2023. doi:10.48550/arXiv.2312.07706.
- 10 Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, and Shangdi Yu. Differential privacy from locally adjustable graph algorithms: k-core decomposition, low out-degree ordering, and densest subgraphs. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 754–765. IEEE, 2022. doi:10.1109/FOCS54457.2022.00077.
- 11 Michael Dinitz, Satyen Kale, Silvio Lattanzi, and Sergei Vassilvitskii. Improved differentially private densest subgraph: Local and purely additive. *CoRR*, abs/2308.10316, 2023. doi:10.48550/arXiv.2308.10316.
- 12 Pål Grønås Drange, Patrick Greaves, Irene Muzi, and Felix Reidl. Computing complexity measures of degenerate graphs. In *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPICs*, pages 14:1–14:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.IPEC.2023.14.
- 13 Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. doi:10.1007/11787006_1.
- 14 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi:10.1007/11681878_14.
- 15 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi:10.1561/04000000042.
- 16 Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam D. Smith. Triangle counting with local edge differential privacy. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 52:1–52:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.52.
- 17 Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 211–222. ACM, 2003. doi:10.1145/773153.773174.
- 18 Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1106–1125. SIAM, 2010. doi:10.1137/1.9781611973075.90.

- 19 Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 169–178. IEEE Computer Society, 2009. doi:10.1109/ICDM.2009.11.
- 20 Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya. Communication cost reduction for subgraph counting under local differential privacy via hash functions. *CoRR*, abs/2312.07055, 2023. doi:10.48550/arXiv.2312.07055.
- 21 Quentin Hillebrand, Vorapong Suppakitpaisarn, and Tetsuo Shibuya. Unbiased locally private estimator for polynomials of laplacian variables. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 741–751. ACM, 2023. doi:10.1145/3580305.3599537.
- 22 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 983–1000. USENIX Association, 2021. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/imola>.
- 23 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-efficient triangle counting under local differential privacy. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 537–554. USENIX Association, 2022. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/imola>.
- 24 Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Differentially private triangle and 4-cycle counting in the shuffle model. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1505–1519. ACM, 2022. doi:10.1145/3548606.3560659.
- 25 George Manoussakis. Listing all fixed-length simple cycles in sparse graphs in optimal time. In *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 355–366. Springer, Springer, 2017. doi:10.1007/978-3-662-55751-8_28.
- 26 Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.41.
- 27 Jaroslav Nesetril 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.
- 28 Iyiola E. Olatunji, Thorben Funke, and Megha Khosla. Releasing graph neural networks with differential privacy guarantees. *Trans. Mach. Learn. Res.*, 2023, 2023. URL: <https://openreview.net/forum?id=wk8oXR0kFA>.
- 29 Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 425–438. ACM, 2017. doi:10.1145/3133956.3134086.
- 30 Sofya Raskhodnikova and Adam D. Smith. Differentially private analysis of graphs. In *Encyclopedia of Algorithms*, pages 543–547. Springer, 2016. doi:10.1007/978-1-4939-2864-4_549.
- 31 Sina Sajadmanesh and Daniel Gatica-Perez. Locally private graph neural networks. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2130–2145. ACM, 2021. doi:10.1145/3460120.3484565.
- 32 Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016*, volume 1558 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL: <https://ceur-ws.org/Vol-1558/paper35.pdf>.

- 33 Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
- 34 Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. Towards locally differentially private generic graph metric estimation. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 1922–1925. IEEE, 2020. doi:10.1109/ICDE48307.2020.00204.
- 35 Qingqing Ye, Haibo Hu, Man Ho Au, Xiaofeng Meng, and Xiaokui Xiao. LF-GDPR: A framework for estimating graph metrics with local differential privacy. *IEEE Trans. Knowl. Data Eng.*, 34(10):4905–4920, 2022. doi:10.1109/TKDE.2020.3047124.
- 36 Hailong Zhang, Sufian Latif, Raef Bassily, and Atanas Rountev. Differentially-private control-flow node coverage for software usage analysis. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1021–1038. USENIX Association, 2020. URL: <https://www.usenix.org/conference/usenixsecurity20/presentation/zhang-hailong>.
- 37 Xiao Zhou and Takao Nishizeki. Graph coloring algorithms. *IEICE Transactions on Information and Systems*, 83(3):407–417, 2000.

Designing Exploration Contracts

Martin Hoefer   

Department of Computer Science, RWTH Aachen University, Germany

Conrad Schecker  

Institute for Computer Science, Goethe University Frankfurt, Germany

Kevin Schewior   

Department of Mathematics and Computer Science, University of Cologne, Germany
University of Southern Denmark, Odense, Denmark

Abstract

We study a natural application of contract design in the context of sequential exploration problems. In our principal–agent setting, a search task is delegated to an agent. The agent performs a sequential exploration of n boxes, suffers the exploration cost for each inspected box, and selects the content (called the *prize*) of one inspected box as outcome. Agent and principal obtain an individual value based on the selected prize. To influence the search, the principal a-priori designs a contract with a non-negative payment to the agent for each potential prize. The goal of the principal is to maximize her expected reward, i.e., value minus payment. Interestingly, this natural contract scenario shares close relations with the *Pandora’s Box* problem.

We show how to compute optimal contracts for the principal in several scenarios. A popular and important subclass is that of *linear* contracts, and we show how to compute optimal linear contracts in polynomial time. For general contracts, we obtain optimal contracts under the standard assumption that the agent suffers cost but obtains value only from the transfers by the principal. More generally, for general contracts with non-zero agent values for outcomes we show how to compute an optimal contract in two cases: (1) when each box has only one prize with non-zero value for principal and agent, (2) for i.i.d. boxes with a single prize with positive value for the principal.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory and mechanism design

Keywords and phrases Exploration, Contract Design, Pandora’s Box Problem

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.50

Related Version *Full Version*: <https://arxiv.org/abs/2403.02317>

Funding *Martin Hoefer*: Supported by DFG Research Unit ADYN (project number 411362735) and DFG grant Ho 3831/9-1 (project number 514505843).

Kevin Schewior: Supported by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

1 Introduction

In many real-world situations, e.g., the search for a job candidate or house, a decision maker is faced with the choice between different alternatives of a-priori unknown value, which can be explored at some cost. Due to missing qualifications or time constraints, in many markets such exploration tasks are not executed directly by the decision maker. Rather, a decision maker (called *principal* \mathcal{P} in the following) can delegate the exploration to an *agent* \mathcal{A} , whose incentives are potentially misaligned with that of \mathcal{P} .

Indeed, suppose the principal intends to buy a house. They might know an inspection cost and have some prior knowledge about the value of each house. However, they might not be qualified to determine the exact market value by inspecting it. They decide to delegate the search to a real-estate agent. The agent can have an individual valuation for each house



© Martin Hoefer, Conrad Schecker, and Kevin Schewior;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 50; pp. 50:1–50:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



(e.g., a provision that is paid internally for selling the house). This individual value might also depend on the condition of the house, which is revealed only after inspection. Similar situations arise, e.g., when the principal wants to invest in a financial product and delegates this search to a financial agent.

In these settings, as a consequence of delegating the search, \mathcal{P} can merely observe the outcome of the search, not the other actions taken by \mathcal{A} . This leads to a possibly undesirable outcome for \mathcal{P} , even when they have the power to commit to accepting or rejecting certain outcomes [5].

A natural approach to align the incentives of an agent with the goals of the principal in such hidden-action settings are *contracts*. A contract allows for utility transfers to be made from the principal to the agent as a function of the outcome of the delegated task. While contract theory is a well-established [9] and celebrated [35] area in economics, the interest in *algorithmic* contract design has surged only relatively recently, see e.g., [14, 17, 20, 21, 27].

In this work, we initiate the study of algorithmic contract design for exploration problems. Specifically, we consider the following problem (for a fully formal definition, see Section 2): There are n boxes, the i -th of which contains a prize, drawn independently from a known probability distribution, and can be opened by paying a known exploration cost c_i . The j -th prize from box i has some utilities a_{ij} and b_{ij} for \mathcal{A} and \mathcal{P} , respectively. Based on that information, \mathcal{P} commits to a contract, specifying a transfer t_{ij} for each prize, where we assume non-negative transfers (limited liability). Given such a contract, \mathcal{A} picks an adaptive search strategy that allows to optimize the trade-off between value of the selected prize (for \mathcal{A}) and total exploration cost. \mathcal{A} uses the strategy to sequentially open boxes by paying the exploration cost. Depending on the observed prize, it determines the next box to open or to stop by selecting a prize from an open box. Assuming \mathcal{A} selects prize j from box i , the resulting final utility of \mathcal{P} is $b_{ij} - t_{ij}$. The goal is to efficiently compute a contract that maximizes the expectation of the latter quantity. Here we assume that ties in the optimization of \mathcal{A} are broken in favor of \mathcal{P} , which is a standard assumption that makes the maximization problem well-defined.

For the problem of computing optimal contracts, there is a solution by linear programming that runs in time polynomial in the number of outcomes and the number of actions of the agent (cf., e.g., [20]). Note that this result does not yield a polynomial-time algorithm for our problem since the action space is extremely succinctly represented – its size is lower-bounded by the number of opening orders, $n!$.

Our Contribution. We show that, in a variety of natural cases, optimal contracts can be computed in polynomial time (such as, e.g., optimal *linear* contracts, or optimal contracts when the agent has no intrinsic value). While our results leave open the question whether computing optimal contracts efficiently is always possible in general, our insights suggest that doing so is a highly non-trivial task and represents one (of several) very interesting avenue for future research.

Notably, \mathcal{A} faces an exploration problem known in the literature as the *Pandora’s Box Problem*. It was famously proposed and analyzed by Weitzman [39], who characterized optimal search strategies by a simple greedy rule. Starting from [33], this problem has recently received a lot of attention at the intersection of economics and computer science (see, e.g., [6, 8, 10, 25, 26, 38] and a recent survey [7]). We reformulate Weitzman’s characterization for \mathcal{A} ’s task: The *fair cap* of box i is a value such that the expected amount of $a_{ij} + t_{ij}$ exceeding the fair cap is precisely c_i . The boxes are considered in non-increasing order of fair caps. If the best $a_{ij} + t_{ij}$ (initially 0) observed so far is larger than the next fair cap, the next box is opened; if it is smaller, a corresponding prize is accepted. If neither is the case, \mathcal{A} is indifferent.

The first observation we make is that Weitzman’s policy does *not* solve \mathcal{A} ’s problem entirely. The reason is that \mathcal{A} breaks ties in favor of \mathcal{P} — recall that this is the standard assumption that makes the maximization problem well-defined. Note that the number of boxes that share some given fair cap φ may be linear in n , leading to a number of possible ways of choosing the order that is exponential in n . Indeed, the order of such boxes may matter to \mathcal{P} since different orders may lead to stopping with prizes that lead to vastly different utilities for \mathcal{P} . A complicating factor is that ties whether to select a prize with agent utility equal to φ have to be broken simultaneously. Interestingly, we show that the problem of breaking ties can be reduced – in polynomial time but in a non-obvious way – to solving *different instances* of the classic Pandora’s box problem.

With a solution of \mathcal{A} ’s problem at hand, the main trade-off inherent to an optimal contract is as follows: Transfers manipulate the fair caps, the resulting exploration order, and the acceptance decisions of \mathcal{A} in such a way that leads to a more favorable outcome for \mathcal{P} – but transfers are costly for \mathcal{P} which is unfavorable.

We start by considering *linear contracts* [20, 30], a class of simple and very intuitive contracts that has received a lot of attention due to its widespread use in applications. The agent receives a constant fraction of the principal’s revenue as commission. Formally, a linear contract is represented by a number $\alpha \in [0, 1]$, which defines transfers $t_{ij} = \alpha \cdot b_{ij}$.

To compute linear exploration contracts, we identify a polynomial-time computable set of *critical* values of α at which the set of \mathcal{A} -optimal policies (i.e., the set of policies that satisfy the above description) changes. By reasoning about the behavior of \mathcal{P} ’s utility between any two such values, we can argue that the optimal contract needs to be a critical value. This type of argument is reminiscent of arguments from the recent literature (e.g., [16, 17, 22]), but here it requires special care. Due to the tie breaking in favor of \mathcal{P} , which – as discussed above – is a key property here, we need to keep track of the *set* of \mathcal{A} -optimal policies.

For standard contract design, the authors in [20] describe a class of instances in which the restriction to linear contracts leads to a multiplicative loss of n in \mathcal{P} ’s achievable utility, where n is the number of actions. In this class of instances, action i with cost c_i deterministically leads to an individual outcome with value R_i . We can create one box for each action i , with opening cost c_i and a single outcome with agent utility 0 and principal utility R_i . This leads to a straightforward one-to-one correspondence between contracts in the two instances that preserves utilities and linearity. Thus, the multiplicative loss of n transfers to our model, which motivates our search for optimal *general* exploration contracts.

We first consider general contracts under the standard assumption that $a_{ij} = 0$, i.e., that the agent has no intrinsic motivation to explore any boxes and is only motivated by payments from \mathcal{P} . In contract design, this assumption is usually without loss of generality, since arbitrary intrinsic agent values $a_{ij} \neq 0$ can be accounted for with action costs to obtain an equivalent instance with agent values $a_{ij} = 0$. However, for the exploration contracts we consider, such an adjustment changes the costs of *exploration strategies* (rather than individual boxes). This substantially changes the structure of the problem. Indeed, as we see below, optimal policies for the problem with non-zero agent value have substantially different properties than the ones for zero value.

Clearly, the utility that \mathcal{P} can extract by solving the exploration problem themselves (i.e., ignoring the agent and paying the cost themselves) is a straightforward baseline; no policy can extract more value for \mathcal{P} . Solving the problem from \mathcal{P} ’s perspective yields fair caps $\varphi_i^{\mathcal{P}}$. If all $a_{ij} = 0$, then by transferring any value exceeding $\varphi_i^{\mathcal{P}}$ to \mathcal{A} , the \mathcal{P} -optimal policy is adopted by \mathcal{A} , without a loss in utility for \mathcal{P} . This implies, in particular, that delegating the exploration to \mathcal{A} using an optimal contract yields the same utility for \mathcal{P} as if she performed the exploration on her own. Formally, the idea is somewhat inspired by [33] for the original Pandora’s Box problem but different.

An orthogonal special case that has been considered in the literature (see, e.g., [5]) is that of *binary boxes*. That is, each box contains one of only two prizes, and one of them has 0 value for both \mathcal{P} and \mathcal{A} while the other one is arbitrary. Without any payments, the fair cap (from \mathcal{A} 's perspective again) of each box is its *basic fair cap*. The principal now faces the question of whether to lift (by transfers) the basic fair caps of favorable boxes above (or up to) the fair caps of less favorable boxes. While the latter is true in general, the special structure of binary boxes allows us to compute an optimal exploration contract by considering boxes in non-increasing order of their basic fair caps and moving their fair cap *greedily*. Note that this algorithm alone does not suffice to tackle more general cases. Indeed, when boxes are not binary, it may be profitable for the principal to choose different payments for two boxes with the same distribution and the same fair caps – a condition we encounter in the next and final case.

The final case we consider has a more general value structure, but we compromise on the boxes having different distributions. Specifically, we consider instances with only a single positive prize for \mathcal{P} , and all distributions are identical. Still, transfers can be made depending on the box from which the prize originates. Again, the complexity is reduced by the fact that only a single payment per box has to be decided. Through an intricate sequence of exchange arguments and the help of continuization, we establish that there is an optimal contract with a simple structure: (i) In a first phase, the prize with positive principal value gets immediately accepted, and all transfers are identical; (ii) all transfers for boxes in the second phase are also identical. These conditions allow to find the optimal contract by enumerating a polynomial-sized set of contracts.

In this final case, we also give a family of instances in which both the above phases exist and have substantial lengths in any optimal solution. Hence, perhaps surprisingly, such a relatively simple case already has optimal contracts with a fairly complicated structure.

Overview. After a review of related work in the subsequent section, we formally introduce our problem in Section 2 along with preliminary observations. Section 3 treats linear contracts. Section 4 deals with general contracts in several special cases (no agent value in Section 4.1 and binary boxes in Section 4.2). General contracts for i.i.d. boxes with a single positive prize are discussed in Section 4.3. Missing proofs are provided in the full version of the paper.

1.1 Related Work

Contract design has recently gained a lot of attention from a computational point of view. Duetting et al. [20] advocated the study of linear contracts as an alternative to the more complicated (and sometimes unintuitive) general contracts. They show robustness results and give parameterized approximation guarantees w.r.t. optimal (general) contracts. For settings in which the outcome space is succinctly represented, hardness results are shown in [21]. In a similar but different approach, *combinatorial contracts* [17] let the agent choose a subset of actions which stochastically determines the (binary) outcome. Important recent work in the area of combinatorial contracts includes [22], [16]. Contracts were also studied with multiple agents [16, 18], ambiguity [19], and private types [2] with connections to classic mechanism design [1]. More generally, contracts have been of interest in specific application domains, such as classification in machine learning [36].

Most closely related is prior work by Postl [34], who analyzes our contract-design problem for two boxes without intrinsic agent value. The paper provides an explicit characterization of the chosen box in the optimal contract, but neither considers algorithmic aspects nor the reduction to the standard Pandora's Box problem we discover in Sec. 4.1. Concurrent

to our work, Ezra et al. [23] study the special case without agent value. In this special case, they also give a polynomial-time algorithm for linear contracts (without discussing the issues of tie-breaking in favor of \mathcal{P}). For general contracts, the paper studies a technical approach when there is a constant number of prizes. Finally, the authors show NP-hardness for correlated distributions in the boxes.

Delegation is a related approach in the principal–agent framework and also received a lot of attention in the economics literature starting from seminal work of Holstrom [31]. Computational aspects of delegation are receiving interest recently, initiated by Kleinberg and Kleinberg [32]. In these models, the principal \mathcal{P} delegates a (search) task to an agent \mathcal{A} , again with potentially different interests, who has to inspect alternatives and *propose* an observed prize to \mathcal{P} . Instead of committing to a contract with payments, \mathcal{P} limits the set of prizes she is willing to accept upon proposal by \mathcal{A} . A prominent objective is to find good acceptance policy for \mathcal{P} and bound their performance by the *delegation gap*, which measures the multiplicative loss in utility for \mathcal{P} in comparison to the case when \mathcal{P} performs the (undelegated) search problem themselves. On a technical level, there are close connections to prophet inequalities [32] and online contention resolution schemes, which were established in different variants, such as multiple agents [37], stochastic probing [4, 5], as an online problem [11], or with limited information about agent utilities [13]. Perhaps most related is a version of the problem studied in [5]. Here, a principal delegates an instance of the Pandora’s Box problem by committing to a set of acceptable outcomes. An agent has to perform the costly inspection of alternatives. This standard delegation setting has some obvious limitations, e.g., the agent would never perform an inspection if his expected intrinsic value does not cover the inspection cost. A model variant with transfers is also studied in [5] in which the principal can reimburse the agent for exploration costs. Crucially, this is impossible in *hidden-action principal-agent* settings that we consider, where \mathcal{P} simply cannot directly reimburse \mathcal{A} for his expenses, since she cannot observe which inspections have been performed by \mathcal{A} .

In a more general context, online optimization in principal–agent settings has generated significant interest recently, e.g., in game-theoretic versions of the Pandora box problem [15], general optimal stopping problems [28, 29], or with unknown agent utilities [3, 12, 24, 40].

2 Preliminaries

There are two parties, a principal \mathcal{P} and an agent \mathcal{A} , and n boxes. Each box $i \in [n]$ has an opening cost $c_i \geq 0$ and contains one of m possible prizes. Prize $j \in [m]$ in box i yields a value-pair (a_{ij}, b_{ij}) , where $a_{ij} \geq 0$ is the value for \mathcal{A} and $b_{ij} \geq 0$ the value for \mathcal{P} . For each box $i \in [n]$, the prize in the box is distributed independently according to a known prior. We denote by $p_{ij} \geq 0$ the probability that prize j is in box i .

\mathcal{P} would like to motivate \mathcal{A} to open and inspect boxes in order to find and select a good prize (for \mathcal{P}). \mathcal{P} cannot observe the actions taken by \mathcal{A} (i.e., which boxes were opened in which order), only the final selected box with the prize inside is revealed. Instead, \mathcal{P} can commit to an *exploration contract* (or simply, *contract*) T : For each selected prize (i, j) , she specifies an amount $t_{ij} \in [0, b_{ij}]$ ¹ that she pays to \mathcal{A} . Then the final utility of \mathcal{P} is $b_{ij} - t_{ij}$.

¹ We make the natural assumption that $t_{ij} \leq b_{ij}$, i.e., every transfer in the contract is bounded by the prize value of \mathcal{P} . For all scenarios we study here (linear contracts, no agent value, binary boxes, i.i.d. with single positive prize for \mathcal{P}) it is straightforward to see that this assumption is w.l.o.g. – there is an optimal contract that satisfies $t_{ij} \leq b_{ij}$ for all $(i, j) \in [n] \times [m]$. It is a very interesting open problem to prove that this holds beyond the cases we study here.

Our model is an instance of the standard principal-agent model, where the hidden action of the agent \mathcal{A} is the policy to inspect and select from the boxes. Specifically, the task of \mathcal{A} is to open and inspect boxes to find and select at most one prize from an opened box. Formally, a *policy* specifies in every situation, based on the previously seen prizes, which box to open next, if any, or otherwise which prize to select from an opened box. Given a contract T , she is facing the classic Pandora's Box problem [39]: She obtains a utility of $a_{ij} + t_{ij}$ for the selected prize (0 if no prize is selected) minus the opening cost c_i for all opened boxes. Weitzman [39] shows that every optimal policy of \mathcal{A} that maximizes his expected total utility (called \mathcal{A} -optimal policy in the following) is in the form of the following index policy: For each box $i \in [n]$ compute an index or *fair cap* φ_i such that

$$\sum_{j \in [m]} p_{ij} \max\{0, a_{ij} + t_{ij} - \varphi_i\} = c_i.$$

The boxes are considered in non-increasing order of fair caps. Suppose box i is the next unopened box in this order, and the best prize \mathcal{A} has found thus far is v . If no box has been opened, then $v = 0$, and otherwise $v = a_{i'j'} + t_{i'j'}$ for some (i', j') where i' is a box that has already been opened, and prize j' has been found in it. There are three cases: (1) $v < \varphi_i$: \mathcal{A} opens box i ; (2) $v = \varphi_i$: \mathcal{A} is indifferent between opening box i and stopping opening boxes; and (3) $v > \varphi_i$: \mathcal{A} stops opening boxes. Note that any box i with $\varphi_i < 0$ is never opened. For each $(i, j) \in [n] \times [m]$ we define the capped value $\kappa_i = \min\{a_{ij} + t_{ij}, \varphi_i\}$. A policy as described above always selects the prize in box $i^* \in \arg \max_i \kappa_i$, i.e., the prize with the largest capped agent value.

Even under these conditions, there may still be several \mathcal{A} -optimal policies. Specifically,

- (i) the choice of φ_i is not unique when $c_i = 0$,
- (ii) the non-decreasing order of fair caps may not be unique,
- (iii) in the case $v = \varphi_i$ above, the choice of whether to stop or continue is not unique,
- (iv) among the observed prizes, the prize (i, j) maximizing $a_{ij} + t_{ij}$ may not be unique.

We assume that \mathcal{A} breaks ties in favor of \mathcal{P}^2 . Among the set of \mathcal{A} -optimal policies, she selects a \mathcal{P} -optimal one, i.e., a policy that maximizes the expected utility for \mathcal{P} . (Note that such a policy may be vastly different from an \mathcal{P} -optimal policy among *all* policies.) We call a policy that is selected in this way an *optimal policy (under contract T)*.

In case (iv), it is clear how to resolve the ambiguity in favor of \mathcal{P} : Simply select (i, j) maximizing $b_{ij} - t_{ij}$. The other cases, however, can give rise to a very large number of potential policies. Interestingly, in the full version of the paper we show that it is always possible to implement the tie-breaking in polynomial time and find an optimal policy under contract T . We prove this result through a reduction to the original Pandora's Box problem.

► **Theorem 1.** *Given a contract T , an optimal policy can be computed in polynomial time.*

We study contracts for \mathcal{P} that steer the index policy executed by \mathcal{A} towards good outcomes for \mathcal{P} . We explore both linear and general contracts. A *linear* contract is given by a single number $\alpha \in [0, 1]$, and $t_{ij} = \alpha \cdot b_{ij}$. Linear contracts are popular because of their simplicity, but they often suffer from substantial limitations of the achievable revenue. As such, we also explore *general* contracts, in which we only require every payment to be non-negative $t_{ij} \geq 0$ for every $i \in [n]$ and $j \in [m]$. A (linear) contract T that, among all possible (linear) contracts, achieves maximum expected utility for \mathcal{P} when \mathcal{A} executes an optimal policy under T , is called *optimal (linear) contract*.

² This is a standard assumption in bi-level problems. On a technical level, it ensures that the optimization problem of finding an optimal contract for \mathcal{P} is well-defined.

3 Optimal Linear Contracts

In this section we consider linear contracts. Recall that these are contracts characterized by a single $\alpha \in [0, 1]$ such that $t_{ij} = \alpha \cdot b_{ij}$ for all $(i, j) \in [n] \times [m]$. Our result is the following.

► **Theorem 2.** *An optimal linear exploration contract can be computed in polynomial time.*

Let $\alpha_1, \alpha_2 \in [0, 1]$ such that the set of \mathcal{A} -optimal policies (i.e., index policies) is the same for all $\alpha \in [\alpha_1, \alpha_2]$. Consider the expected utility of \mathcal{P} according to an optimal policy (i.e., an \mathcal{P} -optimal policy among this set of \mathcal{A} -optimal policies) as a function of α within the interval $[\alpha_1, \alpha_2]$. The following straightforward observation implies that this function is a linear and non-increasing function of α .

► **Observation 3.** *For every policy, there exists a constant c such that the expected utility of \mathcal{P} as a function of α is $(1 - \alpha) \cdot c$.*

Hence, within $[\alpha_1, \alpha_2]$, an optimal contract is α_1 . To find a polynomial-time algorithm for computing the global optimum α , it therefore suffices to show that, in polynomial time, one can find a partition of the interval $[0, 1]$ into (polynomially many) subintervals such that, in each subinterval, the set of \mathcal{A} -optimal policies is constant. (Strictly speaking, we will encounter a technicality at the endpoints of the subintervals.) In the following, we will define a (polynomial-time computable) set of *critical values* in $[0, 1]$ such that, in an interval (strictly) between any two consecutive critical values, the set of \mathcal{A} -optimal policies is constant.

Towards this definition, for any given $\alpha \in [0, 1]$ and every $(i, j) \in [n] \times [m]$, let $v_{ij}(\alpha) := a_{ij} + \alpha \cdot b_{ij}$ denote the value of prize j in box i for \mathcal{A} . Furthermore, let $\varphi_i(\alpha)$ denote the (unique) fair cap of box $i \in [n]$ with $c_i > 0$ as a function of α , i.e., it holds that

$$\sum_{j \in [m]} p_{ij} \max \{0, v_{ij}(\alpha) - \varphi_i(\alpha)\} = c_i$$

for all $\alpha \in [0, 1]$. Note that $\varphi_i(\alpha)$ is a continuous function of α in $[0, 1]$.

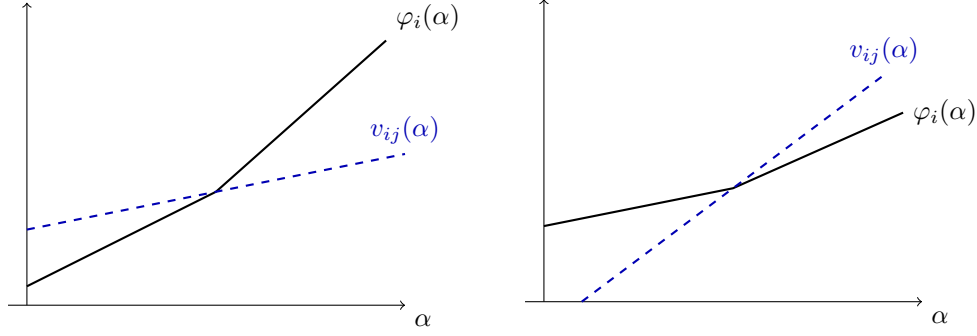
We call $\alpha \in [0, 1]$ a critical value if $\alpha \in \{0, 1\}$ or one of the following properties holds:

- (a) There exist boxes $i, i' \in [n]$ with $c_i > 0, c_{i'} > 0$ such that the order of their fair caps changes at α . Formally, $\varphi_i(\alpha) = \varphi_{i'}(\alpha)$, and there exists $\varepsilon > 0$ such that $\varphi_i(\alpha') \neq \varphi_{i'}(\alpha')$ for all $\alpha' \in (\alpha - \varepsilon, \alpha)$ or for all $\alpha' \in (\alpha, \alpha + \varepsilon)$.
- (b) There exist $i, i' \in [n]$ with $c_i > 0, c_{i'} > 0$ and $j \in [m]$ such that the order between the value of prize (i, j) for \mathcal{A} and the fair cap of box i' changes. Formally, $v_{ij}(\alpha) = \varphi_{i'}(\alpha)$, and there exists $\varepsilon > 0$ such that $v_{ij}(\alpha') \neq \varphi_{i'}(\alpha')$ for all $\alpha' \in (\alpha - \varepsilon, \alpha)$ or for all $\alpha' \in (\alpha, \alpha + \varepsilon)$.
- (c) There exists $i \in [n]$ with $c_i > 0$ such that $\varphi_i(\alpha) = 0$.

Observe that the critical values indeed have the property that, for any two such values α_1^c, α_2^c , the set of \mathcal{A} -optimal policies is constant within (α_1^c, α_2^c) . Note that any \mathcal{A} -optimal policy is also \mathcal{A} -optimal under both α_1^c and α_2^c , but there are potentially additional \mathcal{A} -optimal policies under α_1^c or under α_2^c . Together with Observation 3, this implies that the optimal contract is a critical value.

The following auxiliary lemma supports the analysis of the number of critical values.

► **Lemma 4.** *For each $i \in [n]$ with $c_i > 0$, the fair cap $\varphi_i : [0, 1] \rightarrow \mathbb{R}$ is a monotone convex piece-wise linear function with at most $2m + 1$ linear segments.*



■ **Figure 1** Two cases for an intersection between v_{ij} and φ_i . Left: The slope of v_{ij} was less than slope of φ_i before the intersection. By definition, it contributed to the weighted average slope that defines the slope of φ_i . After the intersection, it does not contribute anymore, and the weighted average only increases. Right: The slope of v_{ij} was greater than slope of φ_i before the intersection. Symmetrical arguments.

Proof. For any $\alpha \in [0, 1]$, it holds $\sum_{j \in [m]} p_{ij} \max\{0, v_{ij}(\alpha) - \varphi_i(\alpha)\} = c_i$ by definition. An equivalent formulation would be

$$\varphi_i(\alpha) = \frac{\sum_{j \in S_i(\alpha)} p_{ij} v_{ij}(\alpha) - c_i}{\sum_{j \in S_i(\alpha)} p_{ij}},$$

where $S_i(\alpha) := \{j \in [m] : v_{ij}(\alpha) \geq \varphi_i(\alpha)\}$. Hence, $\varphi_i(\alpha)$ is the weighted average (minus offset c_i) of all $v_{ij}(\alpha)$ that are in $S_i(\alpha)$, i.e., above $\varphi_i(\alpha)$. For a fixed set $S_i(\alpha)$, this means that the fair cap φ_i is linear in α , as every v_{ij} is linear in α . Therefore, the slope of φ_i only changes at intersections with some v_{ij} , where the set $S_i(\alpha)$ changes. We argue now that the fair cap can only increase at those intersections (cf. Fig. 1).

For every $(i, j) \in [n] \times [m]$ and $\alpha \in (0, 1]$ with $v_{ij}(\alpha) = \varphi_i(\alpha)$, but $v_{ij}(\alpha - \varepsilon) \neq \varphi_i(\alpha - \varepsilon)$ for some $\varepsilon > 0$, there are two cases:

1. $v_{ij}(\alpha - \varepsilon) > \varphi_i(\alpha - \varepsilon)$. Then $j \in S_i(\alpha - \varepsilon)$, but $j \notin S_i(\alpha + \varepsilon')$ for some $\varepsilon' > 0$. Hence the slope of φ_i was greater than the slope of v_{ij} before the intersection at α . The new weighted average of affine functions that are above the fair cap can only increase when v_{ij} is not contributing to that average anymore.
2. $v_{ij}(\alpha - \varepsilon) < \varphi_i(\alpha - \varepsilon)$. Then $j \notin S_i(\alpha - \varepsilon)$, but $j \in S_i(\alpha)$. Hence the slope of φ_i was less than the slope of v_{ij} before the intersection at α . The new weighted average of affine functions that are above the fair cap can only increase when v_{ij} is starting to contribute to that average.

Thus the slope of $\varphi_i(\alpha)$ is never decreasing when α increases, which makes φ_i convex. As v_{ij} is an affine function, every v_{ij} can intersect at most twice with φ_i . Therefore, there are at most $2m + 1$ linear segments of φ_i on the interval $[0, 1]$. ◀

According to Lemma 4, there are at most $\mathcal{O}(n)$ critical values induced by case (c). By the same lemma, there are at most $\mathcal{O}(nm)$ critical values for every box $i \in [n]$ with $c_i > 0$ induced by case (b), as φ_i is convex and there are nm affine functions v_{ij} , each of which intersects φ_i at most twice. Thus, $\mathcal{O}(n^2m)$ critical values in total are due to cases (b) and (c). Similarly, each linear segment of φ_i can intersect with another convex function $\varphi_{i'}$ at most twice. As there are at $\mathcal{O}(m)$ linear segments for $\alpha \in [0, 1]$ and n possible functions $\varphi_{i'}$, there are at most $\mathcal{O}(nm)$ such intersections for φ_i . Thus, case (a) induces at most $\mathcal{O}(n^2m)$ critical values. Overall, the number of critical values is polynomial in n and m , and the set of them can be computed in polynomial time.

By Theorem 1, we can compute an optimal contract for any given critical value in polynomial time. The expected utility of \mathcal{P} under this contract can also be computed in polynomial time. Thus, we can enumerate all critical values and take the best contract with respect to the expected utility for \mathcal{P} , implying Theorem 2.

4 Optimal General Contracts

4.1 No Intrinsic Agent Value

In the consideration of contract problems, it is often assumed that the agent has no intrinsic value (only cost) and receives benefit only via transfers from the principal. In this case, we have $a_{ij} = 0$ for all $(i, j) \in [n] \times [m]$. We show that under this assumption, we can compute an optimal contract in polynomial time.

► **Theorem 5.** *Suppose that $a_{ij} = 0$ holds for all $(i, j) \in [n] \times [m]$. Then an optimal exploration contract can be computed in polynomial time.*

Without any transfers from \mathcal{P} , \mathcal{A} has no intrinsic motivation to open any box – except the ones with inspection cost 0. Specifically, if the agent opens box $i \in [n]$, the (expected) payments from \mathcal{P} have to cover the inspection costs for \mathcal{A} , i.e., it holds that $\sum_{j \in [m]} p_{ij} t_{ij} \geq c_i$. As a direct consequence, \mathcal{P} cannot obtain more utility from a contract with \mathcal{A} over the one obtained by exploring boxes and paying inspection costs by herself (she can simply imitate \mathcal{A} 's behavior under the contract).

► **Definition 6.** *A given contract T implements a policy π if (1) π is optimal under T , and (2) $\sum_{j \in [m]} p_{ij} t_{ij} = c_i$ for all $i \in [n]$.*

As observed above, an optimal contract cannot yield more utility for \mathcal{P} than an optimal policy π^* that she can apply by herself (paying the costs herself). Hence, the next lemma implies Theorem 5.

► **Lemma 7.** *There exists a contract T^* that implements π^* .*

Proof. Recall that the following is an optimal policy π^* for \mathcal{P} (when she pays the costs herself). Define fair caps $\varphi_i^{\mathcal{P}}$ for each box $i \in [n]$ such that $\sum_{j \in [m]} p_{ij} \max\{0, b_{ij} - \varphi_i^{\mathcal{P}}\} = c_i$. Then \mathcal{P} opens boxes in non-increasing order of their fair caps as long as the best prize observed so far does not exceed the fair cap of the next box.

A contract T^* that implements π^* is given by payments $t_{ij} := \max\{0, b_{ij} - \varphi_i^{\mathcal{P}}\}$ for all $(i, j) \in [n] \times [m]$. Clearly, we have

$$\sum_{j \in [m]} p_{ij} t_{ij} = c_i, \quad \text{for all } i \in [n]. \quad (1)$$

Consequently, \mathcal{A} is faced with an instance of the Pandora's Box problem where her expected prize in each box i equals the opening cost. Thus, if \mathcal{A} applies the (\mathcal{A} -optimal) index policy for the emerging instance, the fair caps for \mathcal{A} are given by $\varphi_i^{\mathcal{A}} = 0$ for all $i \in [n]$.

This means that \mathcal{A} is entirely indifferent about the opening order. As long as there are only prizes with $t_{ij} = 0$, \mathcal{A} is also indifferent about stopping to open further boxes and selecting any previously observed prize at any point in time. Thus, \mathcal{A} 's behavior under these conditions can be assumed to be consistent with the one imposed by π^* .

Restrictions only arise from the fact that \mathcal{A} *must stop immediately* whenever a prize j with $t_{ij} > 0$ is drawn from box i , according to \mathcal{A} 's index policy. However, this is consistent with the behavior of the index policy π^* for \mathcal{P} : Note that $t_{ij} > 0$ if and only if $b_{ij} > \varphi_i^{\mathcal{P}}$. Hence, \mathcal{P} also stops when prize j is drawn from box i , because b_{ij} exceeds the fair cap of box i and, thus, the fair cap of the next box in the order.

Therefore, π^* is an optimal policy under contract T^* . Together with (1) this shows that T^* implements π^* . ◀

4.2 Binary Boxes

We study a subclass of the problem where every box $i \in [n]$ contains two prizes with positive probability. One of those prizes has value 0 for both \mathcal{P} and \mathcal{A} , and is called *0-prize*. The other prize (the *positive prize*) is denoted by the value pair (a_i, b_i) with $a_i, b_i \geq 0$ and has probability $p_i \in (0, 1]$. Consequently, the 0-prize is drawn from box i with probability $1 - p_i$. Note that there cannot be a positive payment for the 0-prize. Thus, payments for box i are given by a single $t_i \geq 0$, the payment for the positive prize.

Depending on a payment of $t \geq 0$, we define the fair cap $\varphi_i(t)$ of box i for \mathcal{A} by

$$\varphi_i(t) = t + a_i - \frac{c_i}{p_i}. \quad (2)$$

► **Lemma 8.** *Given a contract T , there is an optimal policy that has the following properties. \mathcal{A} opens boxes in order of the fair cap defined by (2). Boxes with the same fair cap are ordered according to the value $b_i - t_i$. \mathcal{A} accepts the first box with a positive prize (if any).*

Proof. As discussed in Section 2, the fair cap for boxes is unique whenever $c_i > 0$. For boxes with $c_i = 0$ we also assume that the fair cap is given as in (2). This is the smallest fair cap for this box and defers the inspection of box i to the latest point. Clearly, the agent is indifferent regarding this choice. Consequently, it is \mathcal{A} -optimal to stop immediately and accept as soon as \mathcal{A} opens a box with a positive prize in it, since the value is $a_i + t_i \geq \varphi_i(t_i) \geq \varphi_{i+1}(t_{i+1})$.

To argue that this behavior is also \mathcal{P} -optimal, suppose some box i^* with $c_{i^*} = 0$ gets a higher fair cap $\varphi_{i^*}(t_{i^*}) > a_{i^*} + t_{i^*}$ and gets opened earlier (in accordance with the resulting index policy of \mathcal{A}). With such fair cap for i^* , \mathcal{A} does not stop until all subsequent boxes i with fair cap $\varphi_i(t_i) > a_{i^*} + t_{i^*}$ are opened. As such, we can defer the opening of box i^* until after these boxes are opened. Now if there are several boxes with the same fair caps, it is optimal for \mathcal{P} if the opening is done in non-increasing order of $b_i - t_i$. This guarantees that there is no subsequent box with the same fair cap and a prize that has a better value for \mathcal{P} . Conversely, any box with the same fair cap and potentially better value for \mathcal{P} has been inspected before. As such, using minimal fair caps for boxes with $c_i = 0$, breaking ties w.r.t. $b_i - t_i$, and stopping as soon as a positive prize is observed is also \mathcal{P} -optimal. ◀

Let us define a *basic* fair cap of box i to be $\varphi_i(0)$. A box with negative basic fair cap would not be considered by \mathcal{A} unless the payment t_i is at least $c_i/p_i - a_i$, in which case the fair cap would be precisely 0. Note that boxes with prohibitively high cost $p_i(a_i + b_i) \leq c_i$ are w.l.o.g. never opened by \mathcal{A} under any contract T . In what follows, we exclude such boxes from consideration. For the remaining boxes $p_i(a_i + b_i) > c_i$, and, hence, $b_i > c_i/p_i - a_i =: \tilde{t}_i$. Whenever $\tilde{t}_i > 0$, the exploration cost exceeds the expected value for \mathcal{A} in box i . As such, a transfer of \tilde{t}_i is clearly necessary to motivate \mathcal{A} to inspect box i at all. We renormalize the box by adjusting the value of the positive prize to $(a_i + \tilde{t}_i, b_i - \tilde{t}_i)$. Then the basic fair cap becomes $\varphi_i(0) = 0$. Indeed, assuming $t_i \geq \tilde{t}_i$ is w.l.o.g.: Consider any contract T in which $t_i < \tilde{t}_i$ for all i in some non-empty set of boxes B , and the optimal policy π under T . Note

that choosing a contract T' where t_i is increased to \tilde{t}_i for such boxes does not hurt: This will increase the fair caps in B to 0. An \mathcal{A} -optimal policy π' for T' can be obtained from π by opening the boxes B in the end if no prize has been found yet, in any order. The resulting utility for \mathcal{P} in such cases is non-negative rather than 0 (as $\tilde{t}_i \leq b_i$); the utility in all other cases does not change.

In the remainder of the section, we consider the instance with renormalized boxes, i.e., each box i has a basic fair cap $\varphi_i(0) \geq 0$. Furthermore, we re-number the boxes and assume that the indices are assigned in non-increasing order of basic fair caps, i.e. $i < j \implies \varphi_i(0) \geq \varphi_j(0)$ for all $i, j \in [n]$. We break ties in the ordering w.r.t. non-increasing value of b_i .

Our next observation will allow us to restrict attention to the permutation and the fair caps in the policy of \mathcal{A} . By (2), fair caps are linear in payments. Clearly, for an optimal contract we strive to set smallest payments (and, hence, smallest fair caps) to induce a behavior of \mathcal{A} . For any given contract T , we say T implements an ordering σ of boxes if there is an optimal policy under T that considers boxes in the order of σ . For the reverse direction, we need a slightly more technical definition.

► **Definition 9.** For any given ordering σ of boxes, we define a contract $T(\sigma)$ as follows: For each position i (occupied with box $\sigma(i)$), let $j = \max(\arg \max_{j' \in \{i, i+1, \dots, n\}} \{\varphi_{\sigma(j')}(0)\})$ be the latest position of any subsequent box (including $\sigma(i)$ itself) with the highest basic fair cap of the subsequent boxes. In $T(\sigma)$, we assign payments to the unique positive prizes for the boxes on positions $i, i+1, \dots, j$ such that they all have a fair cap of exactly $\varphi_{\sigma(j)}(0)$. If $T(\sigma)$ has feasible payments and implements σ , we call $T(\sigma)$ the canonical contract for σ .

We have the following lemma concerning whether $T(\sigma)$ is the canonical contract for σ .

► **Lemma 10.** For any ordering σ , we can decide in polynomial time if $T(\sigma)$ is the canonical contract for σ (and compute it in this case). If $T(\sigma)$ is not the canonical contract for σ , then every contract T that implements σ is suboptimal for the given instance. Otherwise, the canonical contract has point-wise minimal payments among all contracts that implement σ .

Proof. The construction of $T(\sigma)$ and the subsequent feasibility check described in Definition 9 can be done in polynomial time. Note that if $T(\sigma)$ is the canonical contract for σ , it uses point-wise minimal payments for implementing σ : Having less payments for any box would directly contradict implementation of σ , since every optimal policy considers boxes in the order of their fair caps, which are lower bounded by the respective basic fair caps. Thus, if $T(\sigma)$ is not the canonical contract for σ due to infeasible payments, then implementing σ is impossible with any contract, because even higher payments would be necessary. If $T(\sigma)$ is not the canonical contract for σ , but the construction in Definition 9 defines feasible payments, it defines a canonical contract $T(\sigma')$ of some other ordering $\sigma' \neq \sigma$ using point-wise minimal payments. Note that σ' instead of σ only emerges because of tie-breaking in favor of \mathcal{P} , which implies that boxes with identical fair caps have to be considered in non-increasing order of $b_i - t_i$. Thus it could be possible to implement σ with another contract T , but additional payments would be required to remove (some of the) ties. However, since tie-breaking was already done in favor of \mathcal{P} , this contract T clearly yields less utility than $T(\sigma')$ and hence is not an optimal contract for the given instance. ◀

In the following, we will discuss how to compute the fair caps for the policy of \mathcal{A} in an optimal contract. By Lemma 10, the order σ that we compute in this way has to be implemented by $T(\sigma)$.

Let $e_{\mathcal{P}}(\varphi_1, \dots, \varphi_n)$ denote the expected utility for \mathcal{P} under a contract given by fair caps $\varphi_1, \dots, \varphi_n$. Algorithm 1 computes optimal fair caps for all boxes. Initially, the fair cap of each box is given by its basic fair cap. Then we calculate the optimal fair cap of boxes

Algorithm 1 Optimal Contract for Binary Boxes.

```

Data: Binary boxes  $1, \dots, n$  (non-increasing  $\varphi_i(0)$ , tie-break: non-increasing  $b_i$ )
Result: Contract  $(t_1, \dots, t_n)$ 
 $x_k \leftarrow \varphi_k(0)$  for all  $k \in [n]$ ; // Initialize with basic fair caps.
for  $k = 1, \dots, n$  do
  for  $j = 1, \dots, k$  do
    /* Compute expected utility for  $\mathcal{P}$  when box  $k$  had fair cap  $x_j$  */
     $\rho \leftarrow x_j$ ;
     $e_j \leftarrow e_{\mathcal{P}}(x_1, \dots, x_{k-1}, \rho, x_{k+1}, \dots, x_n)$ ;
  end
   $j^* \leftarrow \arg \max_{j \in \{1, \dots, k\}} e_j$ ; // Box  $k$  receives optimal fair cap  $x_{j^*}$ .
   $t_k \leftarrow x_{j^*} - \varphi_k(0)$ ;
end

```

$1, \dots, n$ iteratively. In iteration k of the outer for-loop, we find the optimal fair cap for box k among the fair caps of previous boxes. Crucially, it is sufficient to test which of those fair caps yield maximum utility, even if the final fair caps of subsequent boxes are not determined yet. Note that by (2), to raise the fair cap of any box $i \in [n]$ to $\varphi > \varphi_i(0)$, the required payment t_i is a constant independent of the probabilities and values in box i – more precisely, it is exactly $t_i = \varphi - \varphi_i(0)$.

► **Theorem 11.** *Algorithm 1 computes an optimal exploration contract for binary boxes in polynomial time.*

Proof. For the proof, we show inductively that the decision of the algorithm in each iteration of the outer for-loop regarding box k is optimal. Formally, there is an optimal ordering of all boxes that, when restricted to boxes $1, \dots, k$, is identical to the ordering computed by the algorithm after iteration k . By Lemma 10, this implies that the selected fair cap for box k is optimal (and, thus, the payment in T), as boxes are numbered and considered in a non-increasing order of their basic fair caps.

For the rest of the proof we show the following statement. For every $k \in [n]$, let σ_k denote the ordering of boxes after iteration k of the outer for-loop in Algorithm 1. There is an optimal ordering σ^* such that the relative order of boxes $1, \dots, k$ is identical in σ^* and σ_k .

We proceed by induction. The statement is trivially true for $k = 1$. Now suppose the statement holds for every of the first $k - 1$ iterations, and consider iteration k . Let i^* denote the position of box k w.r.t. boxes $1, \dots, k$ in σ^* , and let i denote its position σ_k . Note that position i^* corresponds to a position $i_n^* \geq i^*$ in the overall ordering σ^* (since for definition of i^* we ignore boxes $k + 1, \dots, n$). Clearly, the statement holds when $i^* = i$.

For the two remaining cases ($i^* > i$ and $i^* < i$) we show that the optimal ordering can be changed sequentially into one that has $i^* = i$ without decreasing the expected value for \mathcal{P} . We discuss the proof for case 1: $i^* > i$. The proof of the other case is very similar (as seen in the full version of the paper).

Consider $i^* > i$. Now suppose box k was brought to position i (w.r.t. first k boxes) in σ^* . The fair cap of k must be strictly increased, since otherwise k would receive the same payment and the same position in σ_k and σ^* (due to optimal tie-breaking by the agent in non-increasing order of principal value). Note that position i corresponds to a position $i_n < i_n^*$ in σ^* .

Consider the set L of boxes located between positions i_n and i_n^* in σ^* . Consider the first box $r \in L$ with $r > k$ and the position i_r in σ^* . Let L_1 be the boxes located before box r in positions $i_n, \dots, i_r - 1$. We use \tilde{p}_1 for the combined probability of selection in L_1 and \tilde{b}_1 for the combined expected value of a selected box in L_1 .

Let b_r^k be the remaining value of box r after a payment that lifts the fair cap of r to the basic fair cap of k . We split the analysis into two cases: $b_r^k \geq b_k$ and $b_r^k < b_k$. If $b_r^k \geq b_k$, consider the move of r to position i_n . Since there are only boxes $j < k$ in L_1 , we could place box k on position $i + |L_1| + 1$ in σ_k – right after boxes from L_1 , similar to the position i_r of box r in σ^* . We use Δ_{i_r} to denote the payment required to lift the fair cap of box k from its basic fair cap to the one required at position i_r . Moreover, Δ_{i_n, i_r} is the additional payment required to lift the fair cap of box k from the one at position i_r to the one at i_n . Since the algorithm places k at i_n instead of i_r , we know that

$$p_k(b_k - \Delta_{i_r} - \Delta_{i_n, i_r}) + (1 - p_k)\tilde{p}_1\tilde{b}_1 \geq \tilde{p}_1\tilde{b}_1 + (1 - \tilde{p}_1)p_k(b_k - \Delta_{i_r}),$$

which implies

$$b_k \geq \tilde{b}_1 + \Delta_{i_r} + \Delta_{i_n, i_r}/\tilde{p}_1.$$

Suppose now we move r from i_r to i_n , then we have the same terms with b_r^k instead of b_k . Clearly, since $b_r^k \geq b_k$ we know that the move is also profitable. This means that we can move r to position i_n without deteriorating the value of σ^* . Thereby we reduce the number of boxes $j \geq k$ between the positions of k in σ^* and σ_k .

For the second case consider $b_r^k < b_k$. We here consider two moves in σ^* – either swap box r right after box k , or swap box k right before box r . Neither of these moves shall maintain the value of σ^* , and we show that this is impossible.

Let L_2 be the boxes located between r and k in σ^* . We use \tilde{p}_2 and \tilde{b}_2 to denote selection probability and expected value of selected box, respectively. Now if we swap box r after box k , a strict decrease in value yields

$$\begin{aligned} & p_r(b_r^k - \Delta_{i_n^*} - \Delta_{i_n^*, i_r}) + (1 - p_r)\tilde{p}_2\tilde{b}_2 + (1 - p_r)(1 - \tilde{p}_2)p_k(b_k - \Delta_{i_n^*}) \\ & > \tilde{p}_2\tilde{b}_2 + (1 - \tilde{p}_2)p_k(b_k - \Delta_{i_n^*}) + (1 - \tilde{p}_2)(1 - p_k)p_r(b_r^k - \Delta_{i_n^*}). \end{aligned}$$

Note that ensuring the same fair cap as box k at position i_n^* is enough, since $b_r^k < b_k$ and box r is then sorted after box k . We obtain

$$\tilde{p}_2 b_r^k + p_k(1 - \tilde{p}_2)b_r^k > \tilde{p}_2\tilde{b}_2 + p_k(1 - \tilde{p}_2)b_k + \tilde{p}_2\Delta_{i_n^*} + \Delta_{i_n^*, i_r}$$

and, since $b_r^k < b_k$,

$$b_r^k > \tilde{b}_2 + \Delta_{i_n^*} + \Delta_{i_n^*, i_r}/\tilde{p}_2. \quad (3)$$

For the other move, we see that a strict decrease in value yields

$$\begin{aligned} & p_r(b_r^k - \Delta_{i_n^*} - \Delta_{i_n^*, i_r}) + (1 - p_r)\tilde{p}_2\tilde{b}_2 + (1 - p_r)(1 - \tilde{p}_2)p_k(b_k - \Delta_{i_n^*}) \\ & > p_k(b_k - \Delta_{i_n^*} - \Delta_{i_n^*, i_r}) + (1 - p_k)p_r(b_r^k - \Delta_{i_n^*} - \Delta_{i_n^*, i_r}) + (1 - p_k)(1 - p_r)\tilde{p}_2\tilde{b}_2. \end{aligned}$$

Note that ensuring the same fair cap as box r at position i_r is enough, since $b_r^k < b_k$ and box k is then sorted before box r . We obtain

$$(-p_r - \tilde{p}_2 + p_r\tilde{p}_2)b_k + (1 - p_r)\tilde{p}_2\Delta_{i_n^*} > -(1 - p_r)\Delta_{i_n^*, i_r} - p_r b_r^k - \tilde{p}_2\tilde{b}_2 + p_r\tilde{p}_2\tilde{b}_2,$$

so

$$(1 - p_r)\tilde{p}_2\tilde{b}_2 + (1 - p_r)\tilde{p}_2\Delta_{i_n^*} + (1 - p_r)\Delta_{i_n^*, i_r} > \tilde{p}_2(1 - p_r)b_k + p_r(b_k - b_r^k).$$

This implies $p_r < 1$, since otherwise we derive $b_k - b_r^k < 0$, a contradiction. Now, if $p_r \in (0, 1)$, $\tilde{p}_2 \in (0, 1]$, and $b_r^k < b_k$, the above implies

$$\tilde{b}_2 + \Delta_{i_n^*} + \Delta_2/\tilde{p}_2 > b_k. \quad (4)$$

Equations (3) and (4) imply a contradiction with $b_r^k < b_k$.

At least one swap (either moving r right after k , or moving k right before r) must be non-deteriorating in terms of solution value. Using this swap, we can change σ^* and decrease the set of agents $j \geq k$ between i_n^* and i_n without loss in value.

Now using these swaps (either the one for $b_r^k \geq b_k$ or one of the two for $b_r^k < b_k$) iteratively, we can turn σ^* into an ordering such that there are only boxes $j < k$ in L . Then moving box k from i_n^* to i_n is not harmful. We let p_ℓ denote the overall probability that a box in L is chosen and b_ℓ the expected value. Let Δ_{i, i^*} be the additional payment required to raise the fair cap from the one of position i^* to the one of position i .

Indeed, our algorithm could have placed box k at position i^* , but preferred to place k in position $i < i^*$. This means that

$$p_k(b_k - \Delta_{i, i^*}) + (1 - p_k)p_\ell b_\ell \geq p_\ell b_\ell + (1 - p_\ell)p_k b_k,$$

which also reflects the change in value if the move is executed in σ^* . This proves the inductive step when $i^* > i$. \blacktriangleleft

4.3 I.I.D. with Single Positive Prize for Principal

Finally, we study the subclass of the problem where all distributions are identical, with the further restriction that there is only a single positive prize for the principal. Even this restriction has a perhaps surprisingly complicated solution. We show the following result.

► **Theorem 12.** *There exists an algorithm that computes an optimal exploration contract for the i.i.d. case with a single positive prize for the principal in polynomial time.*

We introduce some simplified notation for this subclass. All of the n boxes have an identical distribution over prizes $\{0\} \cup [m] = \{0, 1, \dots, m\}$ and the same opening cost $c \geq 0$. For each prize $j \in \{0, 1, \dots, m\}$, the utilities for \mathcal{A} and \mathcal{P} are a_j and b_j , respectively. Without loss of generality, prize 0 is the only prize for which \mathcal{P} has a positive value, i.e., $b_j = 0$ for all $j \geq 1$. The probability that prize j is drawn when a box is opened is denoted by $q_j \in [0, 1]$. For the ease of notation, we define $p := q_0$ and $v := b_0$. For each box $i \in [n]$, \mathcal{P} defines a (non-negative) payment of $t_i \leq v$ for prize 0. We may assume $v > 0$ and $p > 0$ as otherwise $t_1 = \dots = t_n = 0$ would be the unique transfers.

We make two further assumptions that are without loss of generality. First, the values a_j for $j \geq 1$ are unique: If there are $j_1, j_2 \in [m]$ with $a_{j_1} = a_{j_2}$, replace both prizes by a new one with probability $q_{j_1} + q_{j_2}$. Second, we assume $a_1 \leq a_2 \leq \dots \leq a_m$.

Similarly as in Section 4.2, let $\varphi(t)$ be the fair cap of a box for \mathcal{A} when payment $t \geq 0$ is made for prize 0 of that box. We call $\varphi(0)$ the *basic fair cap*. Note that there is an optimal contract in which each fair cap is either $\varphi(0)$ or some value a_j , where $j \in [m]$, with $a_j > \varphi(0)$. The reason is that, if that were not the case, we could decrease the payment for each box, until that property is satisfied while keeping the policy optimal. Importantly, the agent value

of outcome 0 (which is given by $a_0 + t_i$) need not be considered here: Whenever outcome 0 is drawn from a box i with $\varphi_i > \varphi(0)$, the process stops immediately since, by definition of the fair cap and the index policy, it must hold $a_0 + t_i > \varphi_i \geq \varphi_{i+1}$, where φ_{i+1} is the fair cap of the next box.

Observe that $\varphi(0)$ can be achieved with different payments t , as long as $a_0 + t \leq \varphi(0)$. This may influence the stopping behavior of \mathcal{A} as the payment may determine the order of $a_0 + t$ and a_j for some prize j . To achieve some fair cap larger than $\varphi(0)$, however, there is a unique payment that achieves this fair cap because changing a values above the fair cap that occurs with non-zero probability always changes the fair cap by definition.

Now consider an optimal contract. Index the boxes in the order in which they are opened, and let $\varphi_1 \geq \varphi_2 \geq \dots \geq \varphi_n$ be the corresponding fair caps. We first concentrate on a potential set of *basic* boxes with $\varphi(0)$ in the final rounds of the process.

If $a_0 \geq \varphi(0)$, then every positive payment of \mathcal{P} would increase the fair cap to above $\varphi(0)$. Therefore, if basic boxes with fair cap $\varphi(0)$ exist in an optimal contract, $t_i = 0$ must hold for all these boxes.

Otherwise, if $a_0 < \varphi(0)$, suppose $j^* \in [m]$ is the best other prize for \mathcal{A} that is not exceeding the basic fair cap, i.e., $j^* := \arg \max_{j \in [m]} \{a_j : a_j \leq \varphi(0)\}$. Observe that for every basic box i , an optimal payment t_i either fulfills $a_0 + t_i = a_j$ for some $j \in [j^*]$, or $a_0 + t_i = \varphi(0)$. W.l.o.g., when prize 0 is drawn in a basic box i with $a_0 + t_i = \varphi(0)$, the process stops (by tie-breaking in favor of \mathcal{P} , who cannot improve), and when an outcome $j \geq 1$ with $a_j = \varphi(0)$ is drawn in any round before n , the process continues (again by tie-breaking in favor of \mathcal{P} , who gets 0 if a_j is accepted).

We next show that all basic boxes i with $a_0 + t_i = \varphi(0)$ can be assumed to be opened first among the set of basic boxes. The argument is an exchange argument and given in the full version of the paper.

► **Lemma 13.** *There is an optimal contract where all basic boxes ℓ with $a_0 + t_\ell = \varphi(0)$ are opened first among the set of basic boxes.*

We now divide the process into two phases. In every round i of Phase 1, $\varphi_i \geq \varphi(0)$ and prize 0 gets accepted by \mathcal{A} immediately if it is found. In Phase 2, $\varphi_i = \varphi(0)$ and prize 0 is not accepted immediately (only if in the last round a prize 0 represents the best one for \mathcal{A}). Suppose Phase 1 contains boxes $1, \dots, k$ and Phase 2 boxes $k + 1, \dots, n$. Note that either phase might be empty.

Before analyzing the process, we show an example that shows the perhaps counter-intuitive fact that the optimal contract might indeed involve two such phases, each of substantial length. Moreover, even though all boxes have the same distribution and the same fair cap, in the optimal contract they are assigned one of two different payments depending on their position in the inspection sequence.

► **Example 14.** Consider an instance with $m = 2$. Prize 0 has value $a_0 = 0$ and $b_0 = 1 + \alpha$ for some $\alpha > 0$ determined below, as well as $q_0 = 1/n$. The other prizes have $a_1 = 2$, $q_1 = 1/n$ and $a_2 = 0$, $q_2 = 1 - 2/n$. The inspection cost is $c = 1/n$, and the basic fair cap is $\varphi(0) = 1$.

Clearly, if prize 1 is found in a box, then \mathcal{A} always accepts. If prize 2 is found, \mathcal{A} continues to inspect further boxes. Suppose \mathcal{A} draws a prize 0 in the first round. It is rather unlikely that \mathcal{A} will run through all remaining $n - 1$ boxes without finding a prize 1 and eventually accept prize 0 from round 1. Specifically, this probability is $(1 - 1/n)^{n-1}$. Now \mathcal{P} can set $t_1 = 1 = \varphi(0)$, leading to direct acceptance by \mathcal{A} and immediate profit of α . Alternatively, \mathcal{P} can decide to set $t_1 = 0$ (and, in an optimal contract, then $t_i = 0$ throughout) gambling for a high profit of $1 + \alpha$ in the end. The gambling strategy is unprofitable if

50:16 Designing Exploration Contracts

$$\alpha \geq (1 + \alpha) \cdot (1 - 1/n)^{n-1} \quad \text{or, equivalently,} \quad \alpha \geq \frac{(1 - 1/n)^{n-1}}{1 - (1 - 1/n)^{n-1}} .$$

Suppose α satisfies this inequality, then round 1 is part of Phase 1 in an optimal contract. Via the same calculation, we can determine further rounds i of Phase 1. \mathcal{A} accepts prizes 0 and 1 in all earlier rounds $1, \dots, i-1$, so reaching round i is only possible if prize 2 has been found in all these rounds.

Now suppose for some $k \in \{1, \dots, n-1\}$

$$\alpha = \frac{(1 - 1/n)^k}{1 - (1 - 1/n)^k} .$$

Then \mathcal{P} wants to set $t_i = 1$ for all $i \leq k$, i.e., this becomes Phase 1 with direct acceptance of prize 0. For all rounds $i \geq k+1$, it is more profitable to gamble that no prize 1 will arrive in the remaining rounds and \mathcal{P} will secure a value of $1 + \alpha$. This represents Phase 2. \square

Turning to the analysis, we start by examining Phase 1. Using a relatively straightforward exchange argument given in the full version of the paper, we show the following.

► **Lemma 15.** *There is an optimal contract in which all boxes of Phase 1 have the same fair cap.*

In Phase 2, all boxes have fair cap $\varphi(0)$, but this does not exclude the possibility of different payments. If the first box of Phase 2, box $k+1$, is about to be opened, the highest agent value observed so far (if any) is at most $\varphi(0)$. In particular, prize 0 was never observed at that point, since this would have led to acceptance of \mathcal{A} in an earlier round.

During Phase 2, the process only stops early when an outcome $j > j^*$ is drawn. In this case, the utility for \mathcal{P} is 0. If the process does not stop early, \mathcal{P} receives a utility only if outcome 0 is the best outcome for the agent among all n outcomes.

\mathcal{P} could have different payments for different boxes as long as, for every box i , there is some $j \in [j^*]$ such that $a_0 + t_i = a_j$. If an outcome from a box with payment $t = a_j - a_0 \leq \varphi(0) - a_0$ (for some $j \in [j^*]$) is the maximum after n rounds, then

- outcome 0 must have been drawn from any number of boxes that have this payment t , and
- in all boxes with a payment of at most t , some outcome $0 \leq j' < j$ was drawn, and
- in all remaining boxes, some outcome $1 \leq j' < j$ was drawn.

Let $n_j := |\{i \in [n] : a_0 + t_i = a_j\}|$ denote the number of boxes of Phase 2 with such payment, for every $j \in [j^*]$. Furthermore, for every $j \in [j^*]$, let $Q_j := \sum_{j'=1}^j q_{j'}$ denote the combined probability of all outcomes (other than 0) that are not better for \mathcal{A} , and $N_j = \sum_{j'=1}^j n_{j'}$ denote the number of boxes with the according payment. The probability that outcome 0 with a payment of $t = a_j - a_0$ is the best outcome after n rounds is then given by

$$\begin{aligned} & \sum_{i=1}^{n_j} \binom{n_j}{i} \cdot p^i \cdot (p + Q_j)^{N_j-1} \cdot Q_j^{n-N_j-1-i} \\ &= \sum_{i=1}^{n_j} \binom{n_j}{i} \cdot p^i \cdot \left(\frac{p + Q_j}{Q_j}\right)^{N_j-1} \cdot Q_j^{n+n_j-n_j-i} \\ &= \sum_{i=1}^{n_j} \binom{n_j}{i} \cdot p^i \cdot \left(\frac{p}{Q_j} + 1\right)^{N_j-1} \cdot Q_j^{n-n_j} \cdot Q_j^{n_j-i} \end{aligned}$$

$$\begin{aligned}
&= \left(\frac{p}{Q_j} + 1\right)^{N_{j-1}} \cdot Q_j^{n-n_j} \cdot \left(\sum_{i=0}^{n_j} \binom{n_j}{i} \cdot p^i \cdot Q_j^{n_j-i} - Q^{n_j}\right) \\
&= \left(\frac{p}{Q_j} + 1\right)^{N_{j-1}} \cdot Q_j^{n-n_j} \cdot ((p + Q_j)^{n_j} - Q^{n_j}) \\
&= \left(\frac{p}{Q_j} + 1\right)^{N_{j-1}} \cdot Q_j^n \cdot \left(\left(\frac{p}{Q_j} + 1\right)^{n_j} - 1\right) \\
&= Q_j^n \cdot \left(\left(\frac{p}{Q_j} + 1\right)^{N_j} - \left(\frac{p}{Q_j} + 1\right)^{N_{j-1}}\right).
\end{aligned}$$

Thus, the expected utility for \mathcal{P} for Phase 2 is

$$\sum_{j=1}^{j^*} v_j \cdot Q_j^n \cdot \left(\left(\frac{p}{Q_j} + 1\right)^{N_j} - \left(\frac{p}{Q_j} + 1\right)^{N_{j-1}}\right).$$

With this at hand, we show the following lemma in the full version of the paper.

► **Lemma 16.** *There is an optimal contract in which all boxes of Phase 2 have the same payment for prize 0. There is $j^+ \in [j^*]$ such that $n_{j^+} = n - k$ (and $n_j = 0$ for all $j \neq j^+$).*

By the previous lemmas, it suffices to enumerate all combinations of the length of Phase 1, the fair cap in Phase 1, and the unique payment of prize 0 for Phase 2 (determined by a prize j to be targeted, a payment of 0, or a payment resulting in value $\varphi(0)$ for \mathcal{A} for prize 0). These are polynomially many combinations, and for each combination the expected utility of \mathcal{P} for the resulting contract can be computed in polynomial time. Theorem 12 follows.

References

- 1 Tal Alon, Paul Duetting, Yingkai Li, and Inbal Talgam-Cohen. Bayesian analysis of linear contracts. In *Proc. 24th Conf. Econ. Comput. (EC)*, page 66, 2023. doi:10.1145/3580507.3597795.
- 2 Tal Alon, Paul Dütting, and Inbal Talgam-Cohen. Contracts with private cost per unit-of-effort. In *Proc. 22nd Conf. Econ. Comput. (EC)*, pages 52–69, 2021. doi:10.1145/3465456.3467651.
- 3 Yakov Babichenko, Inbal Talgam-Cohen, Haifeng Xu, and Konstantin Zabarnyi. Regret-minimizing bayesian persuasion. *Games Econ. Behav.*, 136:226–248, 2022. doi:10.1016/J.GEB.2022.09.001.
- 4 Curtis Bechtel and Shaddin Dughmi. Delegated Stochastic Probing. In *Proc. Symp. Innov. Theoret. Comput. Sci. (ITCS)*, pages 37:1–37:19, 2021. doi:10.4230/LIPICS.ITCS.2021.37.
- 5 Curtis Bechtel, Shaddin Dughmi, and Neel Patel. Delegated Pandora’s box. In *Proc. 23rd Conf. Econ. Comput. (EC)*, pages 666–693, 2022. doi:10.1145/3490486.3538267.
- 6 Hedyeh Beyhaghi and Linda Cai. Pandora’s problem with nonobligatory inspection: Optimal structure and a PTAS. In *Proc. 55th Symp. Theory Comput. (STOC)*, pages 803–816, 2023. doi:10.1145/3564246.3585217.
- 7 Hedyeh Beyhaghi and Linda Cai. Recent developments in Pandora’s Box problem: Variants and applications. *ACM SIGecom Exchanges*, 21(1):20–34, 2023. doi:10.1145/3699814.3699817.
- 8 Hedyeh Beyhaghi and Robert Kleinberg. Pandora’s problem with nonobligatory inspection. In *Proc. 20th Conf. Econ. Comput. (EC)*, pages 131–132, 2019. doi:10.1145/3328526.3329626.
- 9 Patrick Bolton and Mathias Dewatripont. *Contract Theory*. MIT Press, 2005.
- 10 Shant Boodaghians, Federico Fusco, Philip Lazos, and Stefano Leonardi. Pandora’s box problem with order constraints. *Math. Oper. Res.*, 48(1):498–519, 2023. doi:10.1287/MOOR.2022.1271.



- 11 Pirmin Braun, Niklas Hahn, Martin Hoefer, and Conrad Schecker. Delegated online search. In *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, pages 2528–2536, 2023. doi:10.24963/IJCAI.2023/281.
- 12 Matteo Castiglioni, Andrea Celli, Alberto Marchesi, and Nicola Gatti. Online Bayesian persuasion. In *Proc. Conf. Adv. Neural Inf. Processing Syst. (NeurIPS)*, 2020.
- 13 Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Bayesian agency: Linear versus tractable contracts. In *Proc. 22nd Conf. Econ. Comput. (EC)*, pages 285–286, 2021. doi:10.1145/3465456.3467602.
- 14 Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Designing menus of contracts efficiently: The power of randomization. In *Proc. 23rd Conf. Econ. Comput. (EC)*, pages 705–735, 2022. doi:10.1145/3490486.3538270.
- 15 Bolin Ding, Yiding Feng, Chien-Ju Ho, Wei Tang, and Haifeng Xu. Competitive information design for pandora’s box. In *Proc. Symp. Discret. Algorithms (SODA)*, pages 353–381, 2023. doi:10.1137/1.9781611977554.CH15.
- 16 Shaddin Dughmi, Neel Bharatkumar Patel, Aditya Prasad, and Ram Deo-Campo Vuong. On supermodular contracts and dense subgraphs. In *Proc. Symp. Discret. Algorithms (SODA)*, pages 109–132, 2024. doi:10.1137/1.9781611977912.6.
- 17 Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Combinatorial contracts. In *Proc. Symp. Found. Comput. Sci. (FOCS)*, pages 815–826, 2021. doi:10.1109/FOCS52979.2021.00084.
- 18 Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Multi-agent contracts. In *Proc. Symp. Theory Comput. (STOC)*, pages 1311–1324, 2023. doi:10.1145/3564246.3585193.
- 19 Paul Dütting, Michal Feldman, and Daniel Peretz. Ambiguous contracts. In *Proc. 24th Conf. Econ. Comput. (EC)*, page 539, 2023.
- 20 Paul Dütting, Tim Roughgarden, and Inbal Talgam-Cohen. Simple versus optimal contracts. In *Proc. 20th Conf. Econ. Comput. (EC)*, pages 369–387, 2019. doi:10.1145/3328526.3329591.
- 21 Paul Dütting, Tim Roughgarden, and Inbal Talgam-Cohen. The complexity of contracts. *SIAM J. Comput.*, 50(1):211–254, 2021. doi:10.1137/20M132153X.
- 22 Paul Dütting, Michal Feldman, and Yoav Gal Tzur. Combinatorial contracts beyond gross substitutes. In *Proc. Symp. Discret. Algorithms (SODA)*, pages 92–108, 2024. doi:10.1137/1.9781611977912.5.
- 23 Tomer Ezra, Michal Feldman, and Maya Schlesinger. Sequential contracts. *CoRR*, abs/2403.09545, 2024. doi:10.48550/arXiv.2403.09545.
- 24 Yiding Feng, Wei Tang, and Haifeng Xu. Online Bayesian recommendation with no regret. In *Proc. 23rd Conf. Econ. Comput. (EC)*, pages 818–819, 2022. doi:10.1145/3490486.3538327.
- 25 Hu Fu, Jiawei Li, and Daogao Liu. Pandora box problem with nonobligatory inspection: Hardness and approximation scheme. In *Proc. 55th Symp. Theory Comput. (STOC)*, pages 789–802, 2023. doi:10.1145/3564246.3585229.
- 26 Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The markovian price of information. In *Proc. Int. Conf. Integer Prog. and Comb. Opt. (IPCO)*, pages 233–246, 2019. doi:10.1007/978-3-030-17953-3_18.
- 27 Guru Guruganesh, Jon Schneider, Joshua R. Wang, and Junyao Zhao. The power of menus in contract design. In *Proc. 24th Conf. Econ. Comput. (EC)*, pages 818–848, 2023. doi:10.1145/3580507.3597735.
- 28 Niklas Hahn, Martin Hoefer, and Rann Smorodinsky. Prophet inequalities for Bayesian persuasion. In *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, pages 175–181, 2020. doi:10.24963/IJCAI.2020/25.
- 29 Niklas Hahn, Martin Hoefer, and Rann Smorodinsky. The secretary recommendation problem. *Games Econ. Behav.*, 134:199–228, 2022. doi:10.1016/J.GEB.2022.05.002.
- 30 Bengt Holmstrom and Paul Milgrom. Aggregation and linearity in the provision of intertemporal incentives. *Econometrica*, 55(2):303–328, 1987.

- 31 Bengt Holmström. Moral hazard and observability. *Bell J. Econ.*, 10(1):74–91, 1979.
- 32 Jon M. Kleinberg and Robert D. Kleinberg. Delegated search approximates efficient search. In *Proc. 19th Conf. Econ. Comput. (EC)*, pages 287–302, 2018. doi:10.1145/3219166.3219205.
- 33 Robert D. Kleinberg, Bo Waggoner, and E. Glen Weyl. Descending price optimally coordinates search. In *Proc. 17th Conf. Econ. Comput. (EC)*, pages 23–24, 2016. doi:10.1145/2940716.2940760.
- 34 P. Postl. Delegated search: Procedure matters. Discussion Paper 04-17, Department of Economics, University of Birmingham, 2004.
- 35 Royal Swedish Academy of Sciences. Scientific background on the 2016 nobel prize in economic sciences, 2016.
- 36 Eden Saig, Inbal Talgam-Cohen, and Nir Rosenfeld. Delegated classification. In *Proc. Conf. Adv. Neural Inf. Processing Syst. (NeurIPS)*, 2023.
- 37 Suho Shin, Keivan Rezaei, and Mohammadtaghi Hajiaghayi. Delegating to multiple agents. In *Proc. 24th Conf. Econ. Comput. (EC)*, pages 1081–1126, 2023. doi:10.1145/3580507.3597669.
- 38 Sahil Singla. The price of information in combinatorial optimization. In *Proc. Symp. Discret. Algorithms (SODA)*, pages 2523–2532, 2018. doi:10.1137/1.9781611975031.161.
- 39 Martin L. Weitzman. Optimal Search for the Best Alternative. *Econometrica*, May, 47(3):641–654, 1979.
- 40 You Zu, Krishnamurthy Iyer, and Haifeng Xu. Learning to persuade on the fly: Robustness against ignorance. In *Proc. 22nd Conf. Econ. Comput. (EC)*, pages 927–928, 2021. doi:10.1145/3465456.3467593.

Protecting the Connectivity of a Graph Under Non-Uniform Edge Failures

Felix Hommelsheim  

University of Bremen, Germany



Zhenwei Liu  

Zhejiang University, Hangzhou, China

University of Bremen, Germany

Nicole Megow  

University of Bremen, Germany

Guochuan Zhang  

Zhejiang University, Hangzhou, China

Abstract

We study the problem of guaranteeing the connectivity of a given graph by protecting or strengthening edges. Herein, a protected edge is assumed to be robust and will not fail, which features a non-uniform failure model. We introduce the (p, q) -Steiner-Connectivity Preservation problem where we protect a minimum-cost set of edges such that the underlying graph maintains p -edge-connectivity between given terminal pairs against edge failures, assuming at most q unprotected edges can fail. We design polynomial-time exact algorithms for the cases where p and q are small and approximation algorithms for general values of p and q . Additionally, we show that when both p and q are part of the input, even deciding whether a given solution is feasible is NP-complete. This hardness also carries over to Flexible Network Design, a research direction that has gained significant attention. In particular, previous work focuses on problem settings where either p or q is constant, for which our new hardness result now provides justification.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Design and analysis of algorithms

Keywords and phrases Network Design, Edge Failures, Graph Connectivity, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.51

Related Version *Full Version*: <https://arxiv.org/abs/2501.04540>

Funding *Zhenwei Liu*: Research supported in part by NSFC (No. 12271477).

Guochuan Zhang: Research supported in part by NSFC (No. 12271477).

1 Introduction

In today's interconnected world, the robustness of infrastructures is crucial, particularly in the face of potential disruptions. This applies to road networks, communication grids, and electrical systems alike, where the ability to maintain functionality despite failures is paramount. Resilience in critical infrastructures requires the incorporation of redundancy to withstand unforeseen challenges. Survivable Network Design (SND) addresses the fundamental need of ensuring not just connectivity but robust connectivity. It goes beyond the conventional concept of linking two entities by recognizing the need for multiple, resilient connections.

Beyond its practical applications, SND is a fundamental problem in combinatorial optimization and approximation algorithms. In its classical setting, we are given a graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathbb{R}$ and connectivity requirements $k(s, t)$ for each vertex pair $s, t \in V$. The goal is to find a minimum-cost subset of edges $X \subseteq E$ such that $H = (V, X)$



© Felix Hommelsheim, Zhenwei Liu, Nicole Megow, and Guochuan Zhang;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 51; pp. 51:1–51:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



is $k(s, t)$ -connected for each $s, t \in V$, i.e., in H there are $k(s, t)$ many edge-disjoint paths for each vertex pair $s, t \in V$. This means that s and t are still connected in H after the deletion of any $k(s, t) - 1$ edges of H . SND is APX-hard and the current best approximation factor of 2 is achieved by Jain’s famous iterative rounding algorithm [32]. It is a long-standing open question whether this factor 2 can be improved, even for many special cases of SND.

Instead of building a new network from scratch, many real-world applications as well as the research community consider augmentation problems, in which we are already given some network, and the task is to robustify the network to withstand failures. The most common model is to increase the connectivity of a given network by adding more links [22, 25]. However, adding new links to a real-world network may be costly or even impractical.

This motivates the investigation of the problem of increasing the connectivity of a network by *protecting* or strengthening edges, as has been proposed by Abbas et al. [1] in a practical context. In this paper, we formally introduce the problem (p, q) -Steiner-Connectivity Preservation $((p, q)$ -SCP): Given an undirected graph $G = (V, E)$, possibly with parallel edges, nonnegative costs $c : E \rightarrow \mathbb{R}_+$ and k terminal pairs $(s_i, t_i) \in V \times V$. The task is to identify a minimum-cost set of edges $X \subseteq E$ such that for any edge set $F \subseteq E \setminus X$ with $|F| \leq q$, there are p edge-disjoint paths between any terminal pair (s_i, t_i) in $(V, E \setminus F)$. In other words, the task is to protect a minimum-cost subset of the edges $X \subseteq E$ such that, no matter which q unprotected edges from $E \setminus X$ are removed from G , there are still p edge-disjoint paths between any terminal pair. We refer to the special case with a single terminal pair (s, t) by (p, q) - s - t -Connectivity Preservation $((p, q)$ -stCP) and the other extreme case in which *each* pair of vertices is a terminal pair as (p, q) -Global-Connectivity Preservation $((p, q)$ -GCP). Using this notion, Abbas et al. [1] considered $(1, q)$ -GCP and proposed to start from a minimum spanning tree and remove unnecessary edges, which does not admit bounded approximation factors. Zhang et al. [44] used mixed-integer linear programming to solve $(1, q)$ -GCP and $(1, q)$ -SCP. Bienstock and Diaz [12] considered a special case of $(1, q)$ -SCP, the all-pair connectivity among a set of terminals, and showed a polynomial-time algorithm for $q \leq 2$ and the NP-hardness for $q = 8$.

The distinction between protected and unprotected edges has a similar flavor as the (p, q) -Flexible Network Design $((p, q)$ -FND) problem [2, 3, 6–9, 13, 17]. The input is an edge-weighted undirected graph together with a set of terminal pairs, where the edges are either safe or unsafe. The goal is to find a minimum-cost subgraph such that any terminal pair remains p -edge-connected after the failure of any q *unsafe* edges. Also here different versions have been studied, e.g., (p, q) -GFND, the global connectivity version (each pair of vertices is a terminal pair) or (p, q) -stFND, the s - t version (only one terminal pair). In contrast to their model, in our setting we strengthen *existing* edges rather than building a network from scratch. For $p = 1$, $(1, q)$ -SCP reduces to $(1, q)$ -FND. Given an instance of $(1, q)$ -SCP, we construct an instance of $(1, q)$ -FND as follows. We use the same underlying graph but replace each edge by two parallel edges: one unsafe edge of cost zero and one safe edge with cost equal to the cost of protecting the edge in the instance of $(1, q)$ -SCP. Since the unsafe edges have cost zero, we can assume that any feasible solution includes all unsafe edges. Then, a feasible solution to $(1, q)$ -FND can be transformed into a feasible solution to $(1, q)$ -SCP with the same cost by protecting the selected safe edges and vice versa. For $p > 1$, however, we are not aware of any reduction from (p, q) -SCP to (p, q) -FND.

1.1 Our Contribution

In this paper, we study Connectivity Preservation problems in terms of algorithms, complexity, and approximability.

The $(1, q)$ -Steiner-Connectivity Preservation problem is APX-hard if q is part of the input: When q is sufficiently large, say, larger than $|E|$, any feasible solution to $(1, q)$ -SCP includes at least one edge in any terminal-separating cut (precise definitions are given in Section 2). Hence, it is equivalent to Steiner Forest, which is APX-hard [11]. Similarly, (p, q) -GCP is APX-hard for any $p \geq 2$, as it contains the minimum p -edge-connected spanning subgraph problem [39] as a special case when q is sufficiently large. We strengthen this by showing that $(1, q)$ -GCP is also NP-hard when q is part of the input.

Motivated by the problem hardness, we first study cases when p or q is small. We obtain polynomial-time exact algorithms for $(p, 1)$ -SCP for any $p \geq 1$ and $(1, 2)$ -SCP as well as a polynomial-time exact algorithm for $(2, 2)$ -GCP. We emphasize that (p, q) -SCP generalizes (p, q) -GCP and hence any algorithm for (p, q) -SCP also works for (p, q) -GCP.

► **Theorem 1 (summarized).** *There are polynomial-time exact algorithms for $(p, 1)$ -Steiner-Connectivity Preservation for any $p \geq 1$ and $(1, 2)$ -Steiner-Connectivity Preservation. Furthermore, there is a polynomial-time exact algorithm for $(2, 2)$ -Global-Connectivity Preservation.*

The first result for $(p, 1)$ -SCP is quite easily obtained by observing that a solution is only feasible if every edge contained in some terminal-separating cut of size at most p is protected.

The polynomial-time algorithm for $(1, 2)$ -SCP is more involved and relies on a decomposition of terminal-separating cuts of size 2. We reduce the problem to the case in which G is assumed to be 2-edge-connected. Then, it remains to protect one edge in each terminal-separating cut of size 2. To do so, we decompose the problem into subproblems that consist either of a 2-edge-connected component or a cycle that can be solved independently.

The polynomial-time algorithm for $(2, 2)$ -GCP is the most involved exact algorithm. We first show that we can assume without loss of generality that G is 3-edge-connected, which reduces the problem to selecting a minimum-cost set of edges containing at least 2 edges from each 3-edge-cut. Equivalently, we select a maximum-weight set of edges such that we pick at most 1 edge from each 3-edge-cut. Using the well-known tree-representation of minimum cuts [20], we model this problem as a multi-commodity flow problem on a tree: given a tree and a set of weighted paths on the tree, find a maximum weighted subset of paths that are pairwise edge-disjoint, which was first introduced in [28] for the unweighted case. We solve the weighted problem via dynamic programming, which might be of independent interest.

We complement our exact algorithms for small p and q with hardness and approximation results for more general cases. For (p, q) -SCP, if both p and q are part of the input, we show that there is no polynomial-time algorithm verifying the feasibility of any given solution, unless $P=NP$, even in the case of s - t -connectivity. This rules out an α -approximation algorithm for any α . Our technique is based on a reduction from k -Clique on d -regular graphs. Note that the corresponding solution verifying problems of (p, q) -stCP and (p, q) -stFND are essentially the same: given sets of protected (resp. safe) and unprotected (resp. unsafe) edges, decide whether there is an s - t -cut that has no more than $p + q - 1$ edges and has less than p protected (resp. safe) edges. Hence, the hardness of verifying a solution also carries over to (p, q) - s - t -Flexible Network Design and justifies why previous work on this problem only discusses the cases where either p or q is constant [3, 17].

► **Theorem 2.** *When both p and q are part of the input, verifying the feasibility of a solution to (p, q) - s - t -Connectivity Preservation or (p, q) - s - t -Flexible Network Design is NP-complete, even in perfect graphs. Hence, they do not admit an α -approximation for any α unless $P = NP$.*

On the algorithmic side, if q is a constant, we can enumerate all “bad” edge sets whose removal destroy the p -edge-connectivity. Since any feasible solution intersects with or hits these “bad” sets, it reduces to the hitting set problem and admits a q -approximation. Then

we focus on the case where p is a constant. We first give a q -approximation for $(1, q)$ -SCP and extend it to (p, q) -SCP based on a primal-dual framework [29, 43]. In the framework, we iteratively protect one more edge in each *critical* cut (precise definition follows), which is very similar to the problem $(1, q)$ -SCP. We obtain the following result.

► **Theorem 3.** *There is a polynomial-time $\mathcal{O}((p + q) \log p)$ -approximation algorithm for (p, q) -Steiner-Connectivity Preservation when p is a constant.*

The global connectivity problem (p, q) -GCP has more symmetric structure, which enables us to remove the requirement of p being constant. Hence, the above result directly carries over to this case without the assumption on p . In addition, we design a different set-cover based augmentation process. This algorithm relies on the fact that there is only a polynomial number of critical cuts to be augmented, which is not true for (p, q) -SCP. Combining the two algorithms, we obtain the following result.

► **Theorem 4.** *There is a polynomial-time $\mathcal{O}(\log p \cdot \min\{p + q, \log n\})$ -approximation algorithm for (p, q) -Global-Connectivity Preservation.*

We obtain further results for special cases by showing reductions to known problems. Since the *Augmenting Small Cuts* problem [8] generalizes $(1, q)$ -Global-Connectivity Preservation, we obtain a 5-approximation building on [37]. Further, we show that $(1, q)$ -*s-t*-Connectivity Preservation is equivalent to the *Minimum Shared Edge* problem; formally defined in Section 4. This reduction implies, due to earlier work, a fixed-parameter tractable (FPT) algorithm parameterized by q if the graph is undirected [23] and an XP-algorithm (slice-wise polynomial) for directed graphs [5]. Further, for directed graphs the equivalence of the two problems implies a strong inapproximability bound of $\Omega(2^{\log^{1-\epsilon} \max\{q, n\}})$, unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$ [38]. Since $(1, q)$ -*s-t*-Connectivity Preservation is a special case of $(1, q)$ -*s-t*-Flexible Network Design, namely, $(1, q)$ -Flexible Network Design with only a single terminal pair, this strong hardness bound also holds for $(1, q)$ -stFND, where the best-known lower bound on the approximation ratio is $\Omega(\log^{2-\epsilon} q)$ unless $\text{NP} \subseteq \text{ZTIME}(n^{\text{polylog}(n)})$ [3].

1.2 Related Work

The (p, q) -Steiner-Connectivity Preservation problem generalizes many well-known problems from survivable network design (SND), which itself generalizes a collection of connectivity problems such as the minimum spanning tree problem, the Steiner tree and forest problem, or the minimum k -edge-connected spanning subgraph problem (k -ECSS).

Many special cases of SND remain APX-hard. This includes many augmentation problems, where typically the task is to increase the connectivity of a graph by at least 1. If the set of edges to be added is unrestricted, the problem can be solved even in near-linear time [15, 22], whereas the problem is APX-hard otherwise [25, 34]. Well-studied such problems include the Connectivity Augmentation problem [40, 42] and the Tree Augmentation problem [14, 41].

A problem of similar flavor was introduced by Adjashvili, Stiller and Zenklusen [4], who initiated the study of highly non-uniform failure models, called bulk-robustness. Therein, a family of edge sets \mathcal{F} is given and the goal is to find a minimum-cost spanning subgraph H such that $H \setminus F$ is connected for any $F \in \mathcal{F}$. They proposed an $\mathcal{O}(\log n + \log m)$ -approximation algorithm for a generalized matroid setting. They also studied an *s-t*-connectivity variant and obtained an $\mathcal{O}(w^2 \log n)$ -approximation algorithm, where $w = \max_{F \in \mathcal{F}} |F|$. Recently, Chekuri and Jain [18] considered the connectivity between multiple vertex pairs and achieved an $\mathcal{O}(w^2 \log^2 n)$ -approximation algorithm.

The aforementioned Flexible Network Design problem can be viewed as a problem between survivable network design and bulk-robustness, as it divides the edge set into safe and unsafe edges and only q unsafe edges can fail simultaneously. Since the work by Adjashvili, Hommelsheim and Mühlenhaller [2] for $(1, 1)$ -GFND, there has been a lot of work on (p, q) -GFND. Most research focused on the case where either p or q is a small constant. Boyd et al. [13] obtained $(q + 1)$ -approximation for $(1, q)$ -GFND, a 4-approximation for $(p, 1)$ -GFND and $\mathcal{O}(q \log n)$ -approximation for (p, q) -GFND. Very recently, Bansal et al. [9] showed an improved 7-approximation algorithm for $(1, q)$ -GFND. We refer to [6, 8, 9, 17] for a collection of results, including constant approximation for $(p, 2), (p, 3)$ -GFND, $\mathcal{O}(q)$ -approximation for $(2, q)$ -GFND and $\mathcal{O}(p)$ -approximation for $(p, 4)$ -GFND for even p . Parallel to (p, q) -GFND, Adjashvili et al. [3] considered the s - t -connectivity variant, to which some results in [17] translate.

Another closely related problem is the Capacitated k -Connected Subgraph problem (Cap- k -ECSS) [16]. In this problem, we are given an undirected graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathbb{R}_+$ and edge capacities $u : E \rightarrow \mathbb{Z}_+$. The goal is to find a minimum-cost spanning subgraph in which the capacity of any cut is at least k . Boyd et al. [13] pointed out that $(1, q)$ -GFND (hence also $(1, q)$ -GCP) can be reduced to Cap- $(q + 1)$ -ECSS by setting the capacity of safe and unsafe edges to $q + 1$ and 1, respectively. For Cap- k -ECSS, the best-known approximation algorithms are $\mathcal{O}(\log n)$ -approximation by Chakrabarty et al. [16] and $\mathcal{O}(\log k)$ -approximation by Bansal et al. [9].

2 Preliminaries: Notation, Cut Formulation, Hardness

Graph notation. For an undirected graph $G = (V, E)$ and a vertex set $S \subset V$, we use $\delta_G(S)$ to denote the set of edges with exactly one endpoint in S . We write $\delta(S)$ if the underlying graph is clear from the context. An edge cut C is a subset of edges such that $G \setminus C$ has at least 2 connected components. If $|C| = 1$, we call $\{e\} = C$ a *bridge*. Further, if there is some terminal pair (s_i, t_i) such that s_i and t_i are in different connected components in $G \setminus C$, we say C is terminal-separating. Let $e = (u, v) \in E$. We use the notation of G/e to denote the graph obtained from G by contracting e , i.e., by deleting e and identifying u, v . Let $G' = (V', E')$ be some subgraph of G . We slightly abuse the notation of contraction and use G/G' to represent the graph obtained from G by contracting all edges in E' . Let $e \in E(G)$. We use $G - e$ to denote the graph $G \setminus \{e\}$. Similarly, we define $G + e$.

An equivalent cut formulation. Given an instance of (p, q) -Steiner-Connectivity Preservation, we define $\mathcal{S} = \{S \subset V \mid \exists i, |S \cap \{s_i, t_i\}| = 1, |\delta(S)| \leq p + q - 1\}$ as the set of *critical* (terminal-separating) cuts. Our problem can be equivalently formulated as the following cut-based integer program, in which we have to cover all critical cuts:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\ & x_e \in \{0, 1\} \quad \forall e \in E. \end{aligned} \tag{CutIP}$$

► **Proposition 5.** (CutIP) characterizes the feasible solutions of (p, q) -SCP. Moreover, a solution is feasible if and only if any critical cut contains at least p protected edges.

Proof. We show that an edge set X is a feasible solution if and only if for any vertex set $S \in \mathcal{S}$, $|X \cap \delta(S)| \geq p$. Consider a feasible solution X and suppose that there is some $S \in \mathcal{S}$ with $|\delta(S)| \leq p + q - 1$ and $|\delta(S) \cap X| < p$. Then, after removing no more than q edges from $\delta(S) \setminus X$, the remaining graph has a cut with less than p edges. Further, this cut separates some terminal pair. Thus, X is not a feasible solution.

Suppose for all $S \in \mathcal{S}$ we have $|\delta(S) \cap X| \geq p$. We show that after removing at most q unprotected edges, the remaining graph is still p -edge-connected between any terminal pair. For any cut $\delta(S)$ with $|\delta(S)| \geq p + q$, there are at least p remaining edges since we remove at most q edges. Fix any terminal pair s, t and any edge set $D \subseteq (E \setminus X)$ with $|D| \leq q$. We show that $|\delta(S) \setminus D| \geq p$ for any s - t -cut S , which implies p -edge-connectivity between s, t . If $|\delta(S)| \geq p + q$, then $|\delta(S) \setminus D| \geq p$. If $|\delta(S)| \leq p + q - 1$, then $|\delta(S) \cap X| \geq p$ by the constraint of (CutIP). Thus it also holds that $|\delta(S) \setminus D| \geq p$. ◀

Given a partial solution $X \subseteq E$, we call a cut *safe* (w.r.t. X) if it is not critical or it contains at least p edges in X . Otherwise, we call it *unsafe*.

NP-hardness. In addition to the aforementioned complexity observations, we now show that $(1, q)$ -GCP is NP-hard, even in the unweighted setting where we protect a minimum number of edges. Observe that any spanning tree of G is a feasible solution, which implies $\text{OPT} \leq |V| - 1$. However, we show that it is NP-complete to distinguish whether $\text{OPT} = |V| - 1$ or $\text{OPT} < |V| - 1$, by a reduction from the largest bond problem [21]. Therein, we are given an undirected graph $G = (V, E)$ and an integer $k \geq 1$. A bond is an edge set represented by $\delta(S)$ for some $S \subset V$ with both $G[S]$ and $G[V \setminus S]$ being connected. The task is to decide whether there is a bond of size at least k .

We outline the idea for the hardness proof as follows. Given an instance of the largest bond problem, we reduce it to an instance of $(1, q)$ -GCP using the same graph with $q := k - 1$. If there is a bond $\delta(S)$ of size at least $k = q + 1$, then protecting a spanning tree of $G[S]$ and $G[V \setminus S]$ is feasible, as the cut $\delta(S)$ is not critical, which implies $\text{OPT} < |V| - 1$. If $\text{OPT} < |V| - 1$, then the protected edges in the optimal solution induce multiple connected components. We can find a bond of size at least $q + 1$ by contracting the connected components induced by the protected edges and computing the minimum cut of the resulting graph.

► **Theorem 6.** *Unweighted $(1, q)$ -Global-Connectivity Preservation is NP-hard.*

Proof. Given an instance of the largest bond problem, we construct an instance of $(1, q)$ -Global-Connectivity Preservation using the same graph with $q = k - 1$.

If there is a bond $\delta(S)$ of size at least k , then the optimal solution value of the $(1, q)$ -Global-Connectivity Preservation instance is no more than $|V| - 2$, as we can simply protect a spanning tree of $G[S]$ and a spanning tree of $G[V \setminus S]$, which exist since $\delta(S)$ is a bond.

If there is no bond of size at least k , we claim that the optimal solution of the instance of $(1, q)$ -Global-Connectivity Preservation must be a spanning tree using $|V| - 1$ edges. Suppose the optimal solution is not a spanning tree, and it consists of connected components S_1, S_2, \dots, S_t , $t \geq 2$. After contracting S_1, S_2, \dots, S_t , the graph has to be k -edge-connected, by feasibility. Let G' be this graph. Note that in any graph there must be a minimum cut $Y \subseteq E$ such that $G \setminus Y$ consists of exactly 2 connected component. Hence, there is also such a minimum cut Y in G' and this cut has size at least k , as G' is k -edge-connected. But then Y corresponds to a bond of size at least k in the original graph, a contradiction. ◀

3 Exact Algorithms for small q

In this section we design three polynomial-time exact algorithms for different cases depending on p and q , i.e., we prove Theorems 7, 10, and 17, which together imply Theorem 1.

3.1 $(p, 1)$ -Steiner-Connectivity Preservation

To give some intuition, we first show a simple algorithm for $(p, 1)$ -Steiner-Connectivity Preservation. By Proposition 5, an instance is feasible if and only if there is no terminal-separating cut of size less than p . Hence, from now on we assume the instance is feasible.

The set of critical cuts is given by $\mathcal{S} = \{S \subset V \mid \exists i, |S \cap \{s_i, t_i\}| = 1, |\delta(S)| \leq p\}$. Hence, any feasible solution must contain *all* edges of any critical cut. Therefore, the only inclusion-wise minimal solution consists of all edges in any terminal-separating cut of size p and it remains to find all such edges. To this end, we assign different *capacities* to protected edges and unprotected edges such that any safe cut has a strictly larger capacity than that of any unsafe cut. The algorithm works as follows.

Algorithm 1. Let X be the current partial solution; initially $X = \emptyset$. In each iteration, we set the capacity of the edges to $\frac{p+1}{p}$ for all $e \in X$ and 1 otherwise. For every terminal pair s, t , we solve the Minimum s - t -Cut Problem using standard polynomial-time algorithms. If we find a terminal-separating cut of capacity less than $p + 1$, then this defines an unsafe critical cut $\delta(S)$ and we protect all edges in it, i.e., we add $\delta(S)$ to X and repeat. If each terminal-separating cut has capacity at least $p + 1$, output X .

► **Theorem 7.** *Algorithm 1 is a polynomial-time exact algorithm for $(p, 1)$ -Steiner-Connectivity Preservation.*

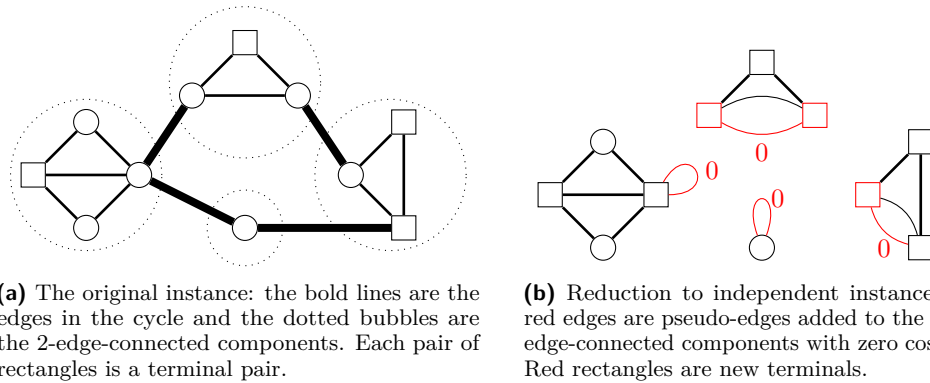
Proof. Algorithm 1 runs obviously in polynomial time. Note that we can decide the feasibility of the given instance by enumerating terminal pairs and checking whether there is a terminal-separating cut of size less than p . We now assume there is none and the instance is feasible.

Let $X \subseteq E$ be a partial solution. We claim that the capacity function in Algorithm 1 distinguishes safe and unsafe cuts with respect to X . Specifically, a cut is unsafe if and only if its capacity is strictly less than $p + 1$. By the feasibility of the instance, any terminal-separating cut has at least p edges. Let C be any terminal-separating cut. If $|C| > p$ or $C \subseteq X$, then the capacity of C is at least $p + 1$. If $|C| = p$ and $|C \cap X| < p$, then its capacity is smaller than $p + 1$. Thus we can find an unsafe terminal-separating cut by enumerating terminal pairs s, t and computing a minimum s - t cut with respect to the capacity function. By the preceding discussion, Algorithm 1 finds an optimum solution in polynomial time. ◀

3.2 $(1, 2)$ -Steiner-Connectivity Preservation

In this subsection we present a polynomial-time algorithm for $(1, 2)$ -Steiner-Connectivity Preservation. The set of critical cuts is $\mathcal{S} = \{S \subset V \mid \exists i, |S \cap \{s_i, t_i\}| = 1, |\delta(S)| \leq 2\}$. Hence, we distinguish between bridges and 2-edge-cuts. We first show that we can reduce to the case that the input graph G is 2-edge-connected.

Given any bridge e of G , if e separates some terminal pair, any feasible solution has to include e . In this case, we pay $c(e)$ and consider the new instance defined by G/e . If there is no such terminal pair, then any inclusion-wise minimal feasible solution should not include e , which implies that we can delete e and consider the two connected components of $G - e$ individually. As a result, we can assume that the input graph G is 2-edge-connected. Note that if the graph is 3-edge-connected, then there is no critical cut and we are done.



■ **Figure 1** Illustration of the decomposition (Lemma 8, Lemma 9).

Given a terminal-separating 2-edge-cut $\{e_1, e_2\}$ of G , at least one of e_1 and e_2 has to be contained in any feasible solution. However, deciding which edge to protect is non-trivial. We show how to further decompose our instance into smaller and independent instances according to the following structural lemma. See Figure 1a for an illustration.

► **Lemma 8.** *Given an undirected graph G which is 2-edge-connected but not 3-edge-connected, and a 2-edge-cut $\{e_1, e_2\}$ of G , there is a polynomial-time algorithm to decompose G into disjoint 2-edge-connected subgraphs G_1, \dots, G_k such that after contracting G_1, \dots, G_k the resulting graph $G/\bigcup_{i=1}^k G_i$ forms a cycle and e_1, e_2 belong to this cycle.*

Proof. Consider the graph $G' := G \setminus \{e_2\}$, which is connected but not 2-edge-connected. Let G'' arise from G' by contracting each 2-edge-connected component. Note that G'' is isomorphic to a tree. Since $G = G' \cup \{e_2\}$ is 2-edge-connected, G'' must be a path. Further, e_2 connects the two end-vertices of the path and e_1 is a path edge (e_1 is a bridge of G''). Let the nodes of the path G'' be v_1, \dots, v_k and for $1 \leq i \leq k$ let G_i be the 2-edge-connected component represented by v_i , respectively. We conclude that $G/\bigcup_{i=1}^k G_i$ forms a cycle and e_1, e_2 belong to this cycle. ◀

Given a decomposition as in Lemma 8, we claim that the problem reduces to solving certain subproblems defined by G_1, \dots, G_k (plus some additional pseudo-edge for each component) and the subproblem restricted to the cycle C . To do so, we view our problem as finding a minimum-cost edge set that intersects all 2-edge-cuts. Observe that any inclusion-wise minimal 2-edge-cut is either two edges on the cycle C , or two edges in G_i for some i . Hence, we can solve our problem by solving (i) the subproblem defined by 2-edge-cuts on the cycle C and (ii) the subproblems defined by 2-edge-cuts in each component G_i separately.

The subproblem (i) is a Steiner Forest problem on a cycle. This follows from the observation that any feasible solution must contain a path consisting of only protected edges between each terminal pair. We can solve the min-cost Steiner Forest problem on a cycle by enumerating which cycle-edge is not in the optimum solution and breaking the cycle into a path. On a path, the solution is the union of the unique paths between the terminal pairs. Then, we recursively solve the subproblems (ii) in each G_i . However, we cannot simply recurse on G_i since a 2-edge-cut of G_i may not be a 2-edge-cut of G . Instead, we recurse on a new graph obtained by adding a zero-cost edge e_i to G_i . This edge e_i connects the two vertices that are incident to the edges of C , which represents the connection in G_i between these vertices via the cycle C . We formalize this idea in the following lemma (See Figure 1b).

► **Lemma 9.** *Given a decomposition as in Lemma 8, an optimum solution to (1,2)-SCP can be obtained by combining optimum solutions of the following subproblems:*

- (i) *protect a minimum-cost edge set that intersects with any terminal-separating 2-edge-cut on the cycle, and*
- (ii) *for each G_i , let $u_i, v_i \in V(G_i)$ be the two vertices incident to the two edges in the cycle. Solve the problem on $G'_i = G_i \cup \{(u_i, v_i)\}$ with $c_{(u_i, v_i)} = 0$. Keep the terminal pairs with both terminals in G_i . For terminal pairs (s_i, t_i) with $s_i \in G_i, t_i \notin G_i$, replace it with $(s_i, u_i), (s_i, v_i)$.*

Proof. We first show that given any feasible solution X of G , the corresponding edges of X on each subproblem is a feasible solution for the subproblem. For the cycle subproblem (ii) this is trivial. For any subproblem $G'_i = G_i \cup \{(u_i, v_i)\}$, we show $X \cap G_i \cup \{(u_i, v_i)\}$ is a feasible solution. Observe that any 2-edge-cut C in G' cannot contain the edge $\{(u_i, v_i)\}$ since G_i is 2-edge-connected. Thus, C must also be a 2-edge-cut in G . If C separates some terminal pair in G'_i , so does it in G , which implies $C \cap X \neq \emptyset$. Therefore, each terminal-separating 2-edge-cut of G'_i is safe.

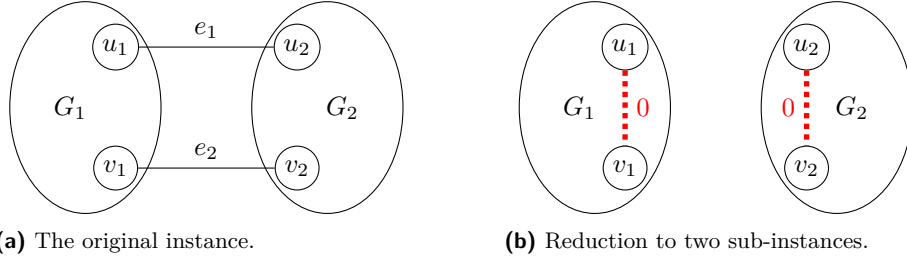
Given feasible solutions of the subproblems, we show how to obtain a feasible solution of G without increasing the cost. Let X be the edges protected in the subproblems except the new edges (u_i, v_i) . Thus, the cost of X is at most the sum of the cost of the solutions to the individual subproblems. It remains to argue that X is feasible for G . Let C be any terminal-separating 2-edge-cut of G . If C is on the cycle, by the feasibility of the subproblem on the cycle, $C \cap X \neq \emptyset$. If C is in G_i for some i , C must also be a terminal-separating 2-edge-cut of G'_i . Thus $C \cap X \neq \emptyset$. We conclude that X is feasible for G . ◀

Algorithm 2. We first protect terminal-separating bridges, contract them, and consider the 2-edge-connected components separated by non-terminal-separating bridges individually. This reduces to the case that G is 2-edge-connected. Then, as long as we find a terminal-separating 2-edge-cut (which is the only type of critical cut), we decompose the problem into a subproblem on a cycle and a collection of subproblems in smaller 2-edge-connected components. Then we recursively solve the individual subproblems. The decomposition stops either if G is 3-edge-connected (and hence we are done as there is no critical cut) or each component on the cycle consists of a single vertex, i.e., if G is a cycle. The cycle case is solved by enumerating which edge of the cycle is *not* contained in an optimum solution and then solving a Steiner Forest problem on a path where the optimal solution is trivial. Among all such solutions, we output the one with minimum cost.

► **Theorem 10.** *Algorithm 2 is a polynomial-time exact algorithm for (1,2)-Steiner-Connectivity Preservation.*

We remark that Bienstock and Diaz [12] studied a special case of (1, q)-SCP. They showed that it is NP-hard when $q = 8$ and they conjectured the NP-hardness for $q = 3$.

Interestingly, (1,2)-Global-Connectivity Preservation admits an easier algorithm. We view the problem as finding a minimum-cost edge set hitting all 2-edge-cuts, which reduces to a special case of Minimum Weighted Vertex Cover. Therein, each edge e of G corresponds to a vertex v_e in the Vertex Cover instance G' and there is an edge between two vertices v_e and $v_{e'}$ if and only if $\{e, e'\}$ forms a 2-edge-cut in G . The Vertex Cover instance G' has a special structure with each connected component being a complete graph. To see this observe that, if both $\{e_1, e_2\}$ and $\{e_1, e_3\}$ are 2-edge-cuts, then $\{e_2, e_3\}$ is also a 2-edge-cut ([35, Lemma 2.37]). The optimal vertex cover solution in a complete graph is trivial: select all vertices except the largest-weighted one. We conclude with the following result.



■ **Figure 2** Illustration of Lemma 12: Equivalence to solving two new instances on $G_1 \cup \{(u_1, v_1)\}$ where (u_1, v_1) is an edge with zero cost and $G_2 \cup \{(u_2, v_2)\}$ where $\{(u_2, v_2)\}$ has zero cost.

► **Lemma 11.** *The greedy algorithm that selects from each 2-edge-cut the cheaper edge solves (1, 2)-Global-Connectivity Preservation.*

3.3 (p, q) -Global-Connectivity Preservation when $p, q \leq 2$

We now present a polynomial-time algorithm for (2, 2)-GCP. Note that for all other $p, q \leq 2$, our previous results imply a polynomial-time algorithm for (p, q) -GCP. We outline our algorithm as follows. By Proposition 5, we can assume the input graph G to be 2-edge-connected, as otherwise, the instance is infeasible. Further, a feasible solution contains at least two edges in each 2-edge-cut and in each 3-edge-cut. We first show that if there is some 2-edge-cut, it is equivalent to solving two smaller independent instances (Lemma 12). Hence, we can assume that the input graph G is 3-edge-connected. Then we represent all the 3-edge-cuts using a standard tree representation [20, 30] and it remains to solve a weighted multi-commodity flow problem on the tree (introduced formally later, Lemma 14). Finally, we solve the weighted multi-commodity flow problem via dynamic programming (Lemma 16).

Suppose G is not 3-edge-connected and there is some 2-edge-cut $\{e_1, e_2\}$, i.e., $G \setminus \{e_1, e_2\}$ consists of 2 connected components G_1, G_2 . Let $e_1 = (u_1, u_2)$ with $u_1 \in G_1$ and $u_2 \in G_2$, $e_2 = (v_1, v_2)$ with $v_1 \in G_1$ and $v_2 \in G_2$ (see Figure 2). We create two new instances: I_1 on $G_1 \cup \{(u_1, v_1)\}$ where (u_1, v_1) is an edge with zero cost and I_2 on $G_2 \cup \{(u_2, v_2)\}$, where $\{(u_2, v_2)\}$ has zero cost. We show that it suffices to solve I_1, I_2 independently and combine their solutions to get a solution to the original instance.

► **Lemma 12.** $\text{OPT}(I) = c(e_1) + c(e_2) + \text{OPT}(I_1) + \text{OPT}(I_2)$.

Proof. Given a feasible solution X of I , we show that $X_1 = X \cap E(G_1) + (u_1, v_1)$ and $X_2 = X \cap E(G_2) + (u_2, v_2)$ are feasible solutions for I_1 and I_2 , respectively, implying $c(X) = c(e_1) + c(e_2) + c(X_1) + c(X_2)$ and $\text{OPT}(I) \geq c(e_1) + c(e_2) + \text{OPT}(I_1) + \text{OPT}(I_2)$. We show feasibility for X_1 ; the feasibility for X_2 is analogous. It suffices to show that for any critical cut in $G_1 + (u_1, v_1)$, at least 2 edges are protected. Consider any critical cut C of $G_1 + (u_1, v_1)$. If C does not contain the new edge (u_1, v_1) , then it is also a critical cut of G and therefore this cut is safe. Hence, we assume that C contains the new edge (u_1, v_1) and let $C = \{f_1, f_2, (u_1, v_1)\}$. Note that f_2 might not exist if $|C| = 2$. We show that $\{f_1, f_2, e_1\}$ is a critical cut in G . Consider $G'_1 = (G_1 \setminus \{f_1, f_2\}) + (u_1, v_1)$ and observe that (u_1, v_1) is a bridge in G'_1 , as C is a cut in $G_1 + (u_1, v_1)$. Hence, u_1 and v_1 are only connected in G'_1 via the edge (u_1, v_1) . This means that also in $G \setminus \{f_1, f_2\}$, any path connecting u_1 and v_1 must use e_1 . Hence, e_1 is a bridge in $G \setminus \{f_1, f_2\}$ and therefore $\{f_1, f_2, e_1\}$ is a critical cut in G . Thus $C \setminus (u_1, v_1) + e_1$ contains at least 2 protected edges, which implies $C \setminus (u_1, v_1)$ contains at least 1 protected edge. Since (u_1, v_1) is protected, C has at least 2 protected edges and is safe.

On the other hand, given solutions X_1 and X_2 of I_1 and I_2 , respectively, we show that $X = X_1 \cup X_2 + e_1 + e_2$ is feasible for I , implying $c(X) = c(e_1) + c(e_2) + c(X_1) + c(X_2)$ and $\text{OPT}(I) \leq c(e_1) + c(e_2) + \text{OPT}(I_1) + \text{OPT}(I_2)$. Let C be any critical cut of G . Without loss of generality, we assume that C is inclusion-wise minimal, i.e., it does not contain any smaller critical cut. We distinguish the following three cases. First, assume that $C \subseteq G_1$. Then, C must also be an edge cut of $G_1 \cup (u_1, u_2)$ and therefore C is safe. The case $C \subseteq G_2$ is analogous. For the second case, we assume that the first case does not apply and further assume that $C \cap \{e_1, e_2\} = \emptyset$. Hence, either $|C \cap G_1| = 1$ or $|C \cap G_2| = 1$. Without loss of generality assume $|C \cap G_1| = 1$. Observe that $(C \cap G_2) + (u_2, v_2)$ is a critical edge cut in $G_2 + (u_2, v_2)$ and $(C \cap G_1) + (u_1, v_1)$ is a critical edge cut in $G_1 + (u_1, v_1)$. Further, by feasibility of X_1 and X_2 , the only edge in $C \cap G_1$ and at least one of the two edges in $C \cap G_2$ must be protected, which implies C contains at least 2 protected edges and is safe. In the third and final case, we assume that none of the previous cases apply and further assume that C contains either e_1 or e_2 . Any cut containing both e_1 and e_2 is safe, as both are protected in X . Without loss of generality assume $e_1 \in C$. We claim that either $C - e_1 \subseteq E(G_1)$ or $C - e_1 \subseteq E(G_2)$. Otherwise, C contains one edge e_3 in G_1 and one edge e_4 in G_2 . Observe that $\{e_1, e_3\}$ is a 2-edge-cut of G , which contradicts the fact that C is inclusion-wise minimal. If $C - e_1 \subseteq E(G_1)$, then similar to the first part of this proof one can show that $C - e_1 + (u_1, v_1)$ is a cut in $G_1 + (u_1, v_1)$. Hence, $|X_1 \cap C| \geq 1$ and therefore, $|X \cap C| \geq 2$, as $e_1 \in X$. The case $C - e_1 \subseteq E(G_2)$ is analogous and hence this concludes the proof. \blacktriangleleft

By repeating the above process, we end up with a 3-edge-connected graph. The following tree representation gives us a clear structure about all the 3-edge-cuts and it can be computed in near-linear time [30].

► Definition 13 (Tree representation of min cuts [20]). *Let $G = (V, E)$ be an undirected graph and suppose the capacity of its minimum cut is an odd number k . There is a polynomial-time algorithm that constructs a rooted tree $T = (U, F)$ together with a (not necessarily surjective) mapping $\phi : V \rightarrow U$. Further, there is a one-to-one correspondence between any k -edge-cut of G and $f \in F$ as follows. For any $f \in F$, let T_f be the subtree of T beneath f and let $V(T_f) = \{v \in V \mid \phi(v) \in T_f\}$. Then for any tree edge $f \in F$, $\delta_G(V(T_f))$ defines a k -edge-cut of G . For any k -edge-cut C of G , there is some tree edge $f \in F$ such that $C = \delta_G(V(T_f))$.*

Given this tree representation, we show now that our problem (2, 2)-Global-Connectivity Preservation reduces to the weighted multi-commodity flow problem on a tree. This problem is defined as follows: given a tree T , a set of paths \mathcal{P} on the tree and a weight function $w : \mathcal{P} \rightarrow \mathbb{R}$, find a subset of pairwise edge-disjoint paths with maximum total weight.

► Lemma 14. *When the input graph G is 3-edge-connected, (2, 2)-Global-Connectivity Preservation reduces to the weighted multi-commodity flow problem on a tree.*

Proof. A solution $X \subseteq E$ is feasible if and only if it contains at least 2 edges in each 3-edge-cut. Equivalently, for each 3-edge-cut at most one edge is unprotected. We consider this complement problem in which we want to find a set of maximum-weight edges \bar{X} such that each 3-edge-cut contains at most one edge of \bar{X} . We use the standard tree representation [20] of all the 3-edge-cuts of G , in which each 3-edge-cut of the original graph is represented by an edge in the tree. Given a tree representation $T = (U, F)$ and $e = (u, v) \in E$, define P_e as the path on T between $\phi(u)$ and $\phi(v)$ and let the weight of P_e be the cost of e . Observe that every 3-edge-cut containing e corresponds to a tree edge on P_e and vice versa. Therefore, a solution $X \subseteq E$ is feasible if and only if the set of paths $\{P_e \mid e \in \bar{X} = E \setminus X\}$ are pairwise edge-disjoint. Hence finding the optimal X reduces to finding a set of edge-disjoint paths on T , maximizing the total weight, which is the weighted multi-commodity flow problem. \blacktriangleleft

► **Remark 15.** We can prove that Lemma 14 holds more generally for any even p : If G is $(p+1)$ -edge-connected, $(p, 2)$ -GCP reduces to the weighted multi-commodity flow problem on a tree for any even p . However, to solve $(p, 2)$ -GCP using this reduction, we need to reduce the problem to the case where G is $(p+1)$ -edge-connected, which remains unclear for $p \geq 4$.

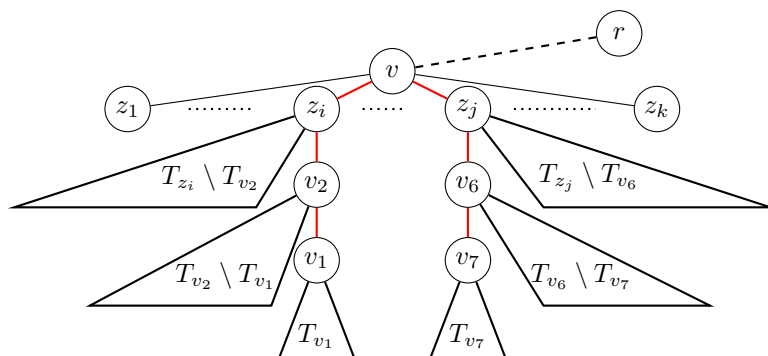
Garg et al. [28] considered an unweighted version of multi-commodity flow problem on a tree and obtained an exact polynomial-time greedy algorithm. However, their arguments do not extend to the weighted case. We design a dynamic program for the weighted version.

► **Lemma 16.** *The weighted multi-commodity flow problem on a tree can be solved in polynomial time.*

Proof. We root the tree at an arbitrary vertex r . Without loss of generality, we assume there is no path that consists of only one vertex since the selected paths have to be only edge-disjoint. For any vertex v , let T_v be the subtree rooted at vertex v . For any tree edge $e = (u, v)$ where u is closer to the root, we use T_e to represent the subtree $T_u \setminus T_v$ for short. Define the subproblem $\mathcal{I}(T_v)$ in the subtree T_v as follows: From the set of paths completely contained in T_v , select a maximum-weight subset of paths that are pairwise edge-disjoint. Let $f(T_v)$ be the optimal value of $\mathcal{I}(T_v)$. We define $\mathcal{I}(T_e)$ and $f(T_e)$ for each $e \in E(T)$ analogously. We only show how to compute $f(T_v)$; the computation of $f(T_e)$ is similar.

Fix some vertex v and consider the subproblem $\mathcal{I}(T_v)$. If v is a leaf, then $f(T_v) = 0$. Otherwise, let z_1, \dots, z_k be the children of v . Let \mathcal{P}_v be the set of paths intersecting v . We say a path P occupies the subtree T_{z_i} if it intersects T_{z_i} . Our first observation is that each subtree T_{z_i} can be occupied by at most one selected path, as otherwise the edge (v, z_i) is contained in multiple selected paths, which is infeasible. Since a path in \mathcal{P}_v can occupy either two subtrees T_{z_i}, T_{z_j} for some $1 \leq i < j \leq k$ or only one subtree T_{z_i} for some i , we reduce the problem $\mathcal{I}(T_v)$ to an instance $\mathcal{M}(T_v)$ of Maximum Weighted Matching. For $\mathcal{M}(T_v)$, we create an auxiliary graph $G(T_v)$ as follows. For each i with $1 \leq i \leq k$, we create a vertex u_i corresponding to the subtree T_{z_i} and a dummy vertex u'_i . For each path in \mathcal{P}_v , if it occupies T_{z_i} and T_{z_j} for some i, j , we create an edge between u_i and u_j . If it only occupies one subtree T_{z_i} , we create an edge between u_i and u'_i . We also create an extra edge between u_i and u'_i which represents the case where no selected path occupies T_{z_i} . It is not hard to see that there is a one-to-one correspondence between a feasible choice over \mathcal{P}_v in $\mathcal{I}(T_v)$ and a matching on the auxiliary graph $G(T_v)$. It remains to properly set the weights ω of the edges of $G(T_v)$ such that a maximum-weight matching in $G(T_v)$ corresponds to an optimum solution for $\mathcal{I}(T_v)$. To do so, we observe that for a given fixed feasible choice over \mathcal{P}_v , it remains to solve a collection of subproblems represented by $\mathcal{I}(T'_v)$ or $\mathcal{I}(T'_e)$ for some $v', e' \in T_v$ and combine their optimal solutions. Formally, let $P = (v_1, \dots, v_\ell)$ be a path in \mathcal{P}_v where $v = v_m$, $1 \leq m \leq \ell$. Assume P occupies two subtrees of v , say, $(v_1, \dots, v_{m-1}) \subseteq T_{z_i}$ and $(v_{m+1}, \dots, v_\ell) \subseteq T_{z_j}$. The case P only occupies one subtree of v follows analogously. Suppose we have selected P . Then it is still feasible to select paths completely contained in T_{z_i} and T_{z_j} , respectively, as long as they do not intersect P . This implies that the subproblems on $T_{z_i} \setminus E(P)$ and $T_{z_j} \setminus E(P)$ can be decomposed into $T_{v_1}, T_{(v_1, v_2)}, \dots, T_{(v_{m-2}, v_{m-1})}$ and $T_{v_\ell}, T_{(v_\ell, v_{\ell-1})}, \dots, T_{(v_{m+2}, v_{m+1})}$, respectively. See Figure 3 for an example. Hence, the gain of selecting P is the sum of the optimal values of these subproblems plus the weight of P itself. That is, we set $\omega((u_i, u_j)) := w(P) + f(T_{v_1}) + \sum_{k=1}^{m-2} f(T_{(v_k, v_{k+1})}) + f(T_{v_\ell}) + \sum_{k=m+1}^{\ell} f(T_{(v_k, v_{k+1})})$. For the extra edge between u_i and u'_i , which represents no selected path occupies T_{z_i} , we set its weight to $f(T_{z_i})$. It now easily follows that $f(T_v) = \text{OPT}(\mathcal{M}(T_v))$, which can be solved in polynomial time using algorithms for Maximum Weighted Matching [26]. ◀

Combining Lemmas 12, 14, and 16, we conclude with the theorem.



■ **Figure 3** Illustration of subproblems: consider the red path $(v_1, v_2, v_3 = z_i, v_4 = v, v_5 = z_j, v_6, v_7)$ through v . If we select the red path, the subtrees T_{z_i} and T_{z_j} break into $T_{v_1}, T_{(v_1, v_2)} = T_{v_2} \setminus T_{v_1}, T_{(z_i, v_2)} = T_{z_i} \setminus T_{v_2}$ and $T_{(z_j, v_6)} = T_{z_j} \setminus T_{v_6}, T_{(v_6, v_7)} = T_{v_6} \setminus T_{v_7}, T_{v_7}$, respectively. They define independent subproblems and their optimal solutions have been computed before we compute $f(T_v)$.

► **Theorem 17.** *There is a polynomial-time exact algorithm for (2, 2)-Global-Connectivity Preservation.*

4 Approximation Algorithms for large p or q

In this section we provide approximation algorithms and hardness results for large p and q .

4.1 Approximation for the cases with $p = 1$

We present a primal-dual algorithm for (1, q)-Steiner-Connectivity Preservation. Consider the corresponding linear programming relaxation of (CutIP) and its dual:

$$\begin{array}{ll}
 \min & \sum_{e \in E} c_e x_e \\
 \text{s.t.} & \sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S} \\
 & x_e \geq 0 \quad \forall e \in E
 \end{array}
 \qquad
 \begin{array}{ll}
 \max & \sum_{S \in \mathcal{S}} y_S \\
 \text{s.t.} & \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S \leq c_e \quad \forall e \in E \\
 & y_S \geq 0 \quad \forall S \in \mathcal{S}
 \end{array}$$

Algorithm 3. We start from a dual solution $\{y_S = 0 \mid S \in \mathcal{S}\}$ and maintain a partial solution $X \subseteq E$ which is the current protected edge set. At the beginning, $X := \emptyset$. We increase the dual variables iteratively and add edges to X whose corresponding dual constraints $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S \leq c_e$ become tight. In each iteration, we pick some $S \in \mathcal{S}$ with $\delta(S) \cap X = \emptyset$ and increase y_S . Such a vertex set S can be found by enumerating terminal pairs (s, t) and checking whether there is an s - t -cut of value less than $q + 1$ with respect to the following capacity function: set the capacity of e to $q + 1$ if $e \in X$ and to 1, otherwise. We increase y_S until for some edge $e \in \delta(S)$, the dual constraint $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S \leq c_e$ is tight. Then we add e to X and move to the next iteration until X is a feasible solution, which is the case if any terminal-separating cut has a capacity of at least $q + 1$ w.r.t. the above capacity function.

To bound the cost of X , we have

$$\sum_{e \in X} c_e = \sum_{e \in X} \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} y_S |\delta(S) \cap X| \leq \sum_{S \in \mathcal{S}} y_S |\delta(S)| \leq q \sum_{S \in \mathcal{S}} y_S \leq q \cdot \text{OPT}.$$

► **Theorem 18.** *Algorithm 3 is a polynomial-time q -approximation algorithm for $(1, q)$ -Steiner-Connectivity Preservation.*

The *global* connectivity variant $(1, q)$ -GCP has more symmetry since we do not need to distinguish whether an edge cut is terminal-separating. By exploiting the special structure of the family $\mathcal{S} = \{S \subset V \mid |\delta(S)| \leq p + q - 1\}$, Bansal et al. [8] obtained a primal-dual 16-approximation algorithm for the *Augmenting Small Cuts* problem, which generalizes $(1, q)$ -GCP. Recently, the factor has been reduced to 10 [37] and 5 [6] via refined analysis.

► **Theorem 19** (follows from [6, 8]). *There is a polynomial-time 5-approximation algorithm for $(1, q)$ -Global-Connectivity Preservation.*

Finally, we consider $(1, q)$ - s - t -Connectivity Preservation. We show that this problem is equivalent to the undirected *Minimum Shared Edge* problem: We are given a graph with edge weights and two specified vertices s, t . The task is to find k s - t paths with the minimum total weight of shared edges. Here, an edge is shared if it is contained in at least 2 paths.

► **Proposition 20.** *An edge set X is a feasible solution to $(1, q)$ -stCP if and only if there are $(q + 1)$ s - t -paths such that any edge shared by at least two paths belongs to X .*

Proof. To show necessity, we construct a graph $G = (V, E)$ with a capacity function u on the edges, where the capacity $u(e)$ of any edge e is $q + 1$ if $e \in X$, and 1 otherwise. Since X is a feasible solution, by Proposition 5 the capacity of any s - t cut is at least $q + 1$. Thus, there exist $q + 1$ s - t -paths such that edges shared by at least two paths belong to X .

As for the sufficiency, suppose we have $q + 1$ s - t -paths that only share edges in X . We claim that the shared edges form a feasible solution of $(1, q)$ -stCP. For each cut of size at most q , at least one edge must be shared by two paths and this edge is in X . Thus, every cut $\delta(S)$ with $|\delta(S)| \leq q$ satisfies $\delta(S) \cap X \neq \emptyset$. By Proposition 5, X is a feasible solution. ◀

► **Lemma 21.** *An edge set X is an inclusion-wise minimal solution to $(1, q)$ - s - t -Connectivity Preservation if and only if there are $(q + 1)$ s - t -paths such that the shared edges are exactly X .*

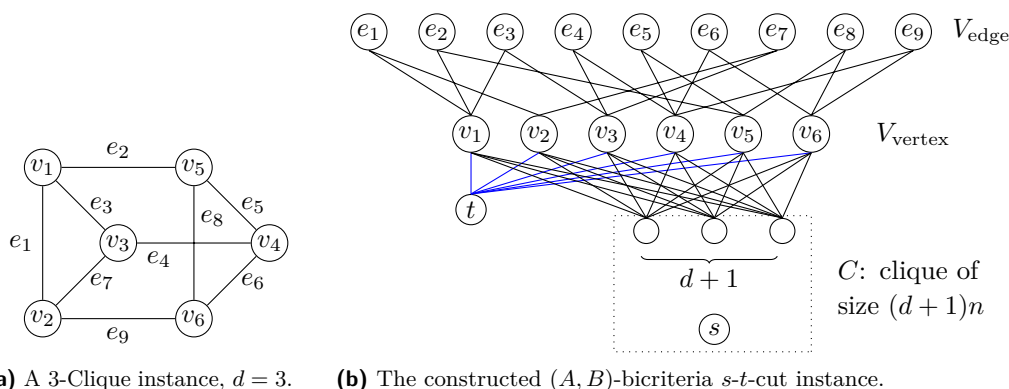
We conclude that $(1, q)$ -stCP is equivalent to the Minimum Shared Edge Problem. Hence, Lemma 21 and the results of [5, 23, 38] imply the following.

► **Theorem 22.** *When parameterized by q , $(1, q)$ -stCP admits an FPT algorithm for undirected graphs and an XP algorithm for directed graphs. Furthermore, $(1, q)$ -stCP on directed graphs admits no $\mathcal{O}(2^{\log^{1-\epsilon} \max\{q, n\}})$ -approximation, unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$.*

4.2 Extension for larger p

Before presenting algorithms for more general cases, we argue that (p, q) -SCP is quite hopeless when both p and q are part of the input. Indeed, if this is the case, there is no polynomial-time algorithm that verifies feasibility of any given solution unless $\text{P}=\text{NP}$.

By Proposition 5, a given protected edge set X is infeasible if and only if there is a terminal-separating cut $\delta(S)$ such that $|\delta(S)| \leq p + q - 1$ and $|\delta(S) \cap X| \leq p - 1$. We define and study the complexity of the following (A, B) -bicriteria s - t -cut problem: Given an undirected graph with two specified vertices s, t and a subset of edges protected, decide



■ **Figure 4** The reduction: the blue edges are protected edges and the black edges are unprotected.

whether there is an s - t -cut such that the number of protected edges in the cut is at most A ($= p - 1$) and the total number of edges in the cut is at most B ($= p + q - 1$) and. Recall that in the (p, q) - s - t -Flexible Network Design problem, verifying the feasibility of a solution can be formulated as follows. Given an undirected graph with safe and unsafe edges, decide whether there are p edge-disjoint paths between s and t after at most q failures of unsafe edges. Hence verifying the feasibility is equivalent to the (A, B) -bicriteria s - t -cut problem. We show that the (A, B) -bicriteria s - t -cut problem is NP-complete, which implies that there is no polynomial-time approximation algorithm for (p, q) - s - t -Connectivity Preservation or (p, q) - s - t -Flexible Network Design, unless $P = NP$.

► **Theorem 2.** *When both p and q are part of the input, verifying the feasibility of a solution to (p, q) - s - t -Connectivity Preservation or (p, q) - s - t -Flexible Network Design is NP-complete, even in perfect graphs. Hence, they do not admit an α -approximation for any α unless $P = NP$.*

Proof. We use a reduction similar to [24] from k -Clique on d -regular graphs, which is NP-complete [27]. See Figure 4 for an illustration. In k -Clique, we are given an undirected graph and we need to decide whether there is a clique of size k . Given an instance of k -clique with graph $G = (V, E)$, where G is d -regular, we construct an instance of the (A, B) -bicriteria s - t -cut problem (G', s, t, A, B) as follows. Let $n := |V|, m := |E|$. We create n vertices V_{vertex} corresponding to V and m vertices V_{edge} corresponding to E . In the following, when we say we connect two vertices, then they are connected by an *unprotected* edge by default. For each $e = (u, v) \in E$, we connect its corresponding vertex to the two vertices corresponding to u and v . Then we create a vertex t and connect it to each vertex in V_{vertex} with *protected* edges. Create an auxiliary clique C of size $n(d + 1)$ and fix an arbitrary vertex in the clique as s . Fix $d + 1$ vertices in the clique other than s and fully connect them to each vertex in V_{vertex} , which results in a complete bipartite subgraph $K_{n, d+1}$. Let $A := k, B := (d + 1)n - k(k - 1)$. We claim that G has a clique of size k if and only if the protected edges form an *infeasible* solution to the instance of the (A, B) -bicriteria s - t -cut problem.

For the first direction, suppose G has a clique $CL = (V_{CL}, E_{CL})$ of size k . Let S include all the vertices in V_{vertex} and V_{edge} corresponding to V_{CL}, E_{CL} , and the auxiliary clique C . We show $\delta(S)$ defines an (A, B) -bicriteria s - t -cut. In $\delta(S)$, the only protected edges are those edges between t and the vertices corresponding to V_{CL} . Hence there are exactly $k = A$ protected edges. As for $|\delta(S)|$, consider the edges between V_{vertex} and $t \cup C$. Each vertex in $V_{\text{vertex}} \setminus S$ contributes $d + 1$ and each vertex in $V_{\text{vertex}} \cap S$ contributes 1. Now consider the edges between V_{edge} and V_{vertex} . There are $d \cdot k$ edges incident to $V_{\text{vertex}} \cap S$, among which $k(k - 1)$ do not contribute to $|\delta(S)|$. Hence, $|\delta(S)| = (d + 1)(n - k) + k + dk - k(k - 1) = (d + 1)n - k(k - 1) = B$ and $\delta(S)$ is an (A, B) -bicriteria s - t -cut.

51:16 Protecting the Connectivity of a Graph Under Non-Uniform Edge Failures

For the other direction, suppose there is an (A, B) -bicriteria s - t -cut $\delta(S)$ with $|\delta(S)| \leq B$ such that $\delta(S)$ contains at most A protected edges. We show that $S \cap V_{\text{vertex}}$ corresponds to the vertices of a clique of size k in G and $S \cap V_{\text{edge}}$ contains the edges of this clique. Observe that $|S \cap V_{\text{vertex}}| \leq k$ since there are at most $A = k$ protected edges in $\delta(S)$. We will show that $|S \cap V_{\text{vertex}}| = k$ and $|S \cap V_{\text{edge}}| \geq k(k-1)/2$. Furthermore, these $k(k-1)/2$ vertices in $S \cap V_{\text{edge}}$ have both their neighbors in $S \cap V_{\text{vertex}}$. Hence $S \cap V_{\text{vertex}}$ defines a clique of size k in G .

Observe that S must include the whole auxiliary clique C , otherwise $|\delta(S)|$ would exceed B . Let $S' = S \setminus V_{\text{edge}}$ and note that $C \subseteq S'$. We prove that $|\delta(S')| = (d+1)n > B$ by considering the following process. Starting from $Y := C$, we add the vertices in $S' \setminus C$ one by one to Y . During the process, $|\delta(Y)|$ does not change since each vertex in $S' \setminus C$ is connected to exactly $d+1$ vertices in C , d vertices in V_{edge} and t . Hence $|\delta(S')| = |\delta(C)| = (d+1)n$. Now starting from $Y = S'$, we add the vertices in $S \setminus S'$ one by one to Y . During the process, the only case that adding a vertex decreases $|\delta(Y)|$ (by 2) is when both its neighbors are in $S \cap V_{\text{vertex}}$. Therefore, we have at least $k(k-1)/2$ vertices in $S \setminus S'$, each having both their neighbors in $S \cap V_{\text{vertex}}$, since $|\delta(S')| - |\delta(S)| \geq (d+1)n - B = k(k-1)$. Hence, $|S \cap V_{\text{vertex}}| \geq k$, and with the above inequality of $|S \cap V_{\text{vertex}}| \leq k$ we have $|S \cap V_{\text{vertex}}| = k$. Further, for any two vertices in $S \cap V_{\text{vertex}}$, there is some vertex in $S \cap V_{\text{edge}}$ connected to both of them, implying that $S \cap V_{\text{vertex}}$ corresponds to the vertices of a clique in G . Hence, G contains a clique of size k . ◀

On the positive side, if q is a constant, we can enumerate those edge sets F with $|F| \leq q$ such that some terminal pair in $(V, E \setminus F)$ is not p -edge-connected. For each of those sets, we need to protect at least one edge in the set, which reduces to the hitting set problem and admits a q -approximation where q is the largest size of the sets to be hit [10, 31].

In the following, we extend algorithm for $(1, q)$ -SCP to (p, q) -SCP with p being a constant. The idea is to start from an empty solution and augment the current solution by iteratively increasing the number of protected edges in each critical cut. Our algorithm consists of p phases. In phase i , we are given a partial solution X_{i-1} satisfying that each critical cut contains at least $i-1$ edges in X_{i-1} . We then (approximately) solve the following augmentation problem \mathcal{P}_i : Add to X_{i-1} a minimum-cost set of edges $Y_i \subseteq E \setminus X_{i-1}$ such that $X_i := X_{i-1} \cup Y_i$ includes at least i edges of each critical cut. That is, find a set Y_i that includes at least one edge from each critical cut with exactly $i-1$ protected edges in X_{i-1} . The augmentation problem is solved similarly to the primal-dual framework for $(1, q)$ -SCP.

Formally, let $\mathcal{S}_0 = \mathcal{S}, X_0 = \emptyset$. In phase i with $1 \leq i \leq p$, we define $\mathcal{S}_i = \{S \in \mathcal{S} \mid |\delta(S) \cap X_{i-1}| = i-1\}$, i.e., the critical cuts with exactly $i-1$ protected edges. Next, we solve the following problem \mathcal{P}_i : find a minimum-cost edge set $Y_i \subseteq E \setminus X_{i-1}$ such that $Y_i \cap \delta(S) \neq \emptyset$ for any $S \in \mathcal{S}_i$. Then we set $X_i := X_{i-1} \cup Y_i$ and go on to the next phase. To solve \mathcal{P}_i , we use a primal-dual algorithm based on the following LP to compute a $(p+q-1)$ -approximation solution to \mathcal{P}_i which is essentially the same as $(1, q)$ -Steiner-Connectivity Preservation. The approximation ratio is bounded by $p+q-1$ as the size of a critical cut is at most $p+q-1$.

$$\begin{array}{ll}
 \min & \sum_{e \in E \setminus X_{i-1}} c_e x_e \\
 \text{s.t.} & \sum_{e \in \delta(S) \setminus X_{i-1}} x_e \geq 1 \quad \forall S \in \mathcal{S}_i \\
 & x_e \geq 0 \quad \forall e \in E \setminus X_{i-1} \\
 \max & \sum_{S \in \mathcal{S}_i} y_S \\
 \text{s.t.} & \sum_{S: S \in \mathcal{S}_i, e \in \delta(S)} y_S \leq c_e \quad \forall e \in E \setminus X_{i-1} \\
 & y_S \geq 0 \quad \forall S \in \mathcal{S}_i
 \end{array}$$

The only difference is the process of finding a violating set $S \in \mathcal{S}_i$ with respect to some partial solution X . However, finding such a violating set is non-trivial. We are only aware of a solution when p is a constant, which we present in the following lemma.

► **Lemma 23.** *Given an edge set $X \supseteq X_{i-1}$, there is a polynomial-time algorithm that computes a set $S \in \mathcal{S}_i$ such that $\delta(S) \cap X = \emptyset$ when p is a constant.*

Proof. Since $X \supseteq X_{i-1}$, we have $|\delta(S) \cap X| \geq i - 1$ for any $S \in \mathcal{S}$. It suffices to find some $S \in \mathcal{S}$ with $|\delta(S) \cap X| = i - 1 \leq p$. To this end, we guess the edge set $X' = \delta(S) \cap X$. Note that $|X'| < p$. Further, for each edge $e = (u, v)$ in X' , we guess whether $u \in S, v \notin S$ or $u \notin S, v \in S$. Thus the number of possibilities is at most $\binom{m}{p} \cdot 2^p$, which is polynomial when p is constant. For the edges in X' , let the set of endpoints in S be A , and the other endpoints be B . It reduces to finding some $S \in \mathcal{S}$ with $A \subseteq S, B \not\subseteq S$ and $\delta(S) \cap X = X'$. This can be achieved by identifying the vertices in A and B by a new vertex v_A and v_B , respectively, contracting edges in $X \setminus X'$, and computing a minimum v_A - v_B cut in the resulting graph. If the cut has size less than $p + q$, then this cut belongs to \mathcal{S}_i and has $i - 1$ edges in X . ◀

To bound the total cost of the p phases of our algorithm, we use the following LP relaxation and its dual for the analysis. The constraints $x_e \leq 1$ cannot be omitted as we do for $p = 1$. Otherwise, an edge may be “protected” multiple times, which is not allowed.

$$\begin{array}{ll} \min & \sum_{e \in E} c_e x_e \\ \text{s.t.} & \sum_{e \in \delta(S)} x_e \geq p \quad \forall S \in \mathcal{S} \\ & 0 \leq x_e \leq 1 \quad \forall e \in E \end{array} \quad \begin{array}{ll} \max & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \\ \text{s.t.} & \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e \quad \forall e \in E \\ & y_S, z_e \geq 0 \quad \forall S \in \mathcal{S}, \forall e \in E \end{array}$$

In the following lemma, we compare the optimal cost of the augmentation problem \mathcal{P}_i to the optimal cost of (p, q) -Steiner-Connectivity Preservation.

► **Lemma 24.** *Given a feasible dual solution $y^{(i)}$ of \mathcal{P}_i , we can construct a feasible dual solution y of (p, q) -Steiner-Connectivity Preservation such that*

$$\sum_{S \in \mathcal{S}_i} y_S^{(i)} \leq \frac{1}{p - i + 1} \left(\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \right).$$

Proof. Let $y_S = y_S^{(i)}$ for $S \in \mathcal{S}_i$ and $y_S = 0$ for $S \in \mathcal{S} \setminus \mathcal{S}_i$. Let $z_e = 0$ for $e \in E \setminus X_{i-1}$ and $z_e = \sum_{S: S \in \mathcal{S}_i, e \in \delta(S)} y_S^{(i)}$ for $e \in X_{i-1}$. We claim that (y, z) forms a feasible dual solution. For any $e \in X_{i-1}$, $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e = 0$ by definition. For $e \in E \setminus X_{i-1}$, $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e = \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S^{(i)} \leq c_e$. Next, we compare the dual objective values of y and $y^{(i)}$. We have

$$\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e = \sum_{S \in \mathcal{S}_i} p \cdot y_S^{(i)} - \sum_{e \in X_{i-1}} \sum_{S: S \in \mathcal{S}_i, e \in \delta(S)} y_S^{(i)} = \sum_{S \in \mathcal{S}_i} y_S^{(i)} (p - |\delta(S) \cap X_{i-1}|).$$

By definition of \mathcal{S}_i , for any $S \in \mathcal{S}_i$, we have $|\delta(S) \cap X_{i-1}| = i - 1$. Thus, we conclude:

$$\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e = (p - i + 1) \sum_{S \in \mathcal{S}_i} y_S^{(i)}. \quad \blacktriangleleft$$

► **Theorem 3.** *There is a polynomial-time $\mathcal{O}((p + q) \log p)$ -approximation algorithm for (p, q) -Steiner-Connectivity Preservation when p is a constant.*

Proof. The algorithm consists of p phases. In phase i , we apply Lemma 23 and the primal-dual framework for $(1, q)$ -SCP to find a $(p+q-1)$ -approximation solution for the augmentation problem \mathcal{P}_i . By Lemma 24, the cost of Y_i in phase i is at most $(p+q-1) \cdot \text{OPT}(\mathcal{P}_i) \leq \frac{p+q-1}{p-i+1} \text{OPT}$. Thus the total cost is at most $\sum_{i=1}^p c(Y_i) \leq \sum_{i=1}^p \frac{p+q-1}{p-i+1} \text{OPT} \leq H_p \cdot (p+q-1) \cdot \text{OPT}$, where H_p is the p -th harmonic number. Using $H_p \leq \log(p) + 1$, we obtain the theorem. \blacktriangleleft

For (p, q) -Global-Connectivity Preservation, we can approximately solve the augmentation problem without requiring p to be a constant. Indeed, we reduce finding the critical cuts to finding certain 2-approximate minimum cuts in G , where each edge e has a capacity of $\frac{p+q}{i-1}$ if $e \in X_{i-1}$ and 1 otherwise. These cuts can be enumerated in polynomial time [33, 36].

Algorithm 4. In phase 1, we apply the 5-approximation algorithm from [6]. That is, we compute X_1 such that for any $S \in \mathcal{S}_0 = \mathcal{S}$, $X_1 \cap \delta(S) \neq \emptyset$. For phase i with $2 \leq i \leq p$, we approximately solve the augmentation problem \mathcal{P}_i by reducing it to Set Cover. Here, we view a set $S \in \mathcal{S}_i$ as an element in the Set Cover instance and view an edge $e \in E \setminus X_{i-1}$ as a set in the Set Cover instance. We use either the $\mathcal{O}(\log N)$ -approximation [19] where N is the number of elements to be covered, or the f -approximation [10, 31] where f is the maximum number of sets in which an element is contained. Note that applying Lemma 24 requires a dual feasible solution, which is fortunately a byproduct of these Set Cover algorithms.

► **Theorem 4.** *There is a polynomial-time $\mathcal{O}(\log p \cdot \min\{p+q, \log n\})$ -approximation algorithm for (p, q) -Global-Connectivity Preservation.*

Proof. The cost of phase 1 is no more than $5 \cdot \text{OPT}$. For phase i with $2 \leq i \leq p$, we apply Set Cover algorithms explicitly. We show that the number of elements to be covered is $|\mathcal{S}_i| = \mathcal{O}(|V|^4)$ and we can construct the Set Cover instance in polynomial time. To this end, we assign different capacities to edges in X_{i-1} and other edges such that for any $S \in \mathcal{S}_i$, $\delta(S)$ is a 2-approximate minimum cut with respect to the capacity function. By Karger's bound [33], the number of 2-approximate minimum cuts is $\mathcal{O}(|V|^4)$ and we can enumerate them in polynomial time [36]. Formally, let the capacity of edges in X_{i-1} be $\frac{p+q}{i-1}$ and the capacity of edges in $E \setminus X_{i-1}$ be 1. Given any cut C , the capacity of C is at least $p+q$ since it either contains at least $p+q$ edges or contains at least $i-1$ edges in X_{i-1} . For any $S \in \mathcal{S}_i$, the capacity of $\delta(S)$ is at most $(i-1)\frac{p+q}{i-1} + p+q-1 < 2(p+q)$. Thus $\delta(S)$ defines a 2-approximate minimum cut and we can find all the sets in \mathcal{S}_i in polynomial time.

Further, in the constructed Set Cover instance, an element is contained in at most $p+q-1$ sets since $|\delta(S)| \leq p+q-1$ for any $S \in \mathcal{S}_i$. Thus, we can compute an augmenting edge set $X_i \setminus X_{i-1}$ whose cost is $\mathcal{O}(\min\{\log n, p+q\} \cdot \sum_{S \in \mathcal{S}_i} y_S^{(i)})$ where $y^{(i)}$ is the dual feasible solution of \mathcal{P}_i . Combining it with Lemma 24, we conclude that the algorithm is an $\mathcal{O}(\log p \cdot \min\{\log n, p+q\})$ -approximation. \blacktriangleleft

5 Conclusion

We examine Connectivity Preservation from two perspectives. For small values of p and q , we focus on polynomial-time exact algorithms. For large values of p and q , we show hardness and devise approximation algorithms. Nonetheless, there remain some gaps between cases solvable in polynomial time and NP-hard ones. In particular, it remains open whether $(1, q)$ -GCP admits any polynomial-time exact algorithm for constant $q \geq 3$. Another interesting problem is $(1, q)$ -GCP with q being the capacity of the minimum cuts, i.e., finding a minimum-cost edge set that intersects with all the minimum cuts. Note that for the s - t -connectivity variant, this can be tackled via Min-cost Flow techniques.

References

- 1 Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Improving network connectivity using trusted nodes and edges. In *2017 American Control Conference (ACC)*, pages 328–333. IEEE, 2017. doi:10.23919/ACC.2017.7962974.
- 2 David Adjiashvili, Felix Hommelsheim, and Moritz Mühenthaler. Flexible graph connectivity. *Math. Program.*, 192(1):409–441, 2022. doi:10.1007/S10107-021-01664-9.
- 3 David Adjiashvili, Felix Hommelsheim, Moritz Mühenthaler, and Oliver Schaudt. Fault-tolerant edge-disjoint s-t paths - beyond uniform faults. In *SWAT*, volume 227 of *LIPICs*, pages 5:1–5:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.SWAT.2022.5.
- 4 David Adjiashvili, Sebastian Stiller, and Rico Zenklusen. Bulk-robust combinatorial optimization. *Math. Program.*, 149(1-2):361–390, 2015. doi:10.1007/S10107-014-0760-6.
- 5 Sepehr Assadi, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, and Hamid Zarrabi-Zadeh. The minimum vulnerability problem. *Algorithmica*, 70(4):718–731, 2014. doi:10.1007/S00453-014-9927-Z.
- 6 Ishan Bansal. A global analysis of the primal-dual method for pliable families, 2024. arXiv:2308.15714.
- 7 Ishan Bansal, Joe Cheriyan, Logan Grout, and Sharat Ibrahimpur. Algorithms for 2-connected network design and flexible steiner trees with a constant number of terminals. In *APPROX/RANDOM*, volume 275 of *LIPICs*, pages 14:1–14:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.APPROX/RANDOM.2023.14.
- 8 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions. *Algorithmica*, 86(8):2575–2604, 2024. doi:10.1007/S00453-024-01235-2.
- 9 Ishan Bansal, Joseph Cheriyan, Sanjeev Khanna, and Miles Simmons. Improved approximation algorithms for flexible graph connectivity and capacitated network design, 2024. doi:10.48550/arXiv.2411.18809.
- 10 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.
- 11 Marshall W. Bern and Paul E. Plassmann. The steiner problem with edge lengths 1 and 2. *Inf. Process. Lett.*, 32(4):171–176, 1989. doi:10.1016/0020-0190(89)90039-2.
- 12 Daniel Bienstock and Nicole Diaz. Blocking small cuts in a network, and related problems. *SIAM J. Comput.*, 22(3):482–499, 1993. doi:10.1137/0222034.
- 13 Sylvia C. Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Math. Program.*, 204(1):493–516, 2024. doi:10.1007/S10107-023-01961-5.
- 14 Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *STOC*, pages 370–383. ACM, 2021. doi:10.1145/3406325.3451086.
- 15 Ruoxu Cen, Jason Li, and Debmalya Panigrahi. Edge connectivity augmentation in near-linear time. In *STOC*, pages 137–150. ACM, 2022. doi:10.1145/3519935.3520038.
- 16 Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. *Algorithmica*, 72(2):493–514, 2015. doi:10.1007/S00453-013-9862-4.
- 17 Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 18 Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models, 2024. arXiv:2403.15547, doi:10.48550/arXiv.2403.15547.
- 19 Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

- 20 Efim A. Dinits, Alexander V. Karzanov, and Micael V. Lomonosov. On the structure of a family of minimum weighted cuts in a graph. *Studies in Discrete Optimization*, pages 209–306, 1976.
- 21 Gabriel L Duarte, Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Daniel Lokshтанov, Lehilton LC Pedrosa, Rafael CS Schouery, and Uéverton S Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83:1421–1458, 2021. doi:10.1007/S00453-020-00789-1.
- 22 Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976. doi:10.1137/0205044.
- 23 Till Fluschnik, Stefan Kratsch, Rolf Niedermeier, and Manuel Sorge. The parameterized complexity of the minimum shared edges problem. *J. Comput. Syst. Sci.*, 106:23–48, 2019. doi:10.1016/J.JCSS.2018.12.002.
- 24 Fedor V. Fomin, Petr A. Golovach, and Janne H. Korhonen. On the parameterized complexity of cutting a few vertices from a graph. In *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2013. doi:10.1007/978-3-642-40313-2_38.
- 25 Greg N. Frederickson and Joseph F. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981. doi:10.1137/0210019.
- 26 Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *SODA*, pages 434–443. SIAM, 1990. URL: <http://dl.acm.org/citation.cfm?id=320176.320229>.
- 27 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 28 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. doi:10.1007/BF02523685.
- 29 Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232. ACM/SIAM, 1994. URL: <http://dl.acm.org/citation.cfm?id=314464.314497>.
- 30 Zhongtian He, Shang-En Huang, and Thatchaphol Saranurak. Cactus representation of minimum cuts: Derandomize and speed up. In *SODA*, pages 1503–1541. SIAM, 2024. doi:10.1137/1.9781611977912.61.
- 31 Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982. doi:10.1137/0211045.
- 32 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Comb.*, 21(1):39–60, 2001. doi:10.1007/S004930170004.
- 33 David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30. ACM/SIAM, 1993. URL: <http://dl.acm.org/citation.cfm?id=313559.313605>.
- 34 Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004. doi:10.1137/S0097539702416736.
- 35 Hiroshi Nagamochi and Toshihide Ibaraki. *Algorithmic Aspects of Graph Connectivity*, volume 123 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2008. doi:10.1017/CB09780511721649.
- 36 Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing all small cuts in an undirected network. *SIAM J. Discret. Math.*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.
- 37 Zeev Nutov. Improved approximation ratio for covering pliable set families, 2024. arXiv:2404.00683, doi:10.48550/arXiv.2404.00683.
- 38 Masoud T. Omran, Jörg-Rüdiger Sack, and Hamid Zarrabi-Zadeh. Finding paths with minimum shared edges. *J. Comb. Optim.*, 26(4):709–722, 2013. doi:10.1007/S10878-012-9462-2.

- 39 David Pritchard. k -edge-connectivity: Approximation and LP relaxation. In *WAOA*, volume 6534 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 2010. doi:10.1007/978-3-642-18318-8_20.
- 40 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *FOCS*, pages 1–12. IEEE, 2021. doi:10.1109/FOCS52979.2021.00010.
- 41 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *SODA*, pages 3253–3272. SIAM, 2022. doi:10.1137/1.9781611977073.128.
- 42 Vera Traub and Rico Zenklusen. A $(1.5+\epsilon)$ -approximation algorithm for weighted connectivity augmentation. In *STOC*, pages 1820–1833. ACM, 2023. doi:10.1145/3564246.3585122.
- 43 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Comb.*, 15(3):435–454, 1995. doi:10.1007/BF01299747.
- 44 Jianan Zhang, Eytan Modiano, and David Hay. Enhancing network robustness via shielding. *IEEE/ACM Transactions on Networking*, 25(4):2209–2222, 2017. doi:10.1109/TNET.2017.2689019.

Polynomial Kernel and Incompressibility for Prison-Free Edge Deletion and Completion

Séhane Bel Houari-Durand  

ENS Lyon, France

Eduard Eiben  

Department of Computer Science, Royal Holloway University of London, UK

Magnus Wahlström  

Department of Computer Science, Royal Holloway University of London, UK

Abstract

Given a graph G and an integer k , the H -FREE EDGE DELETION problem asks whether there exists a set of at most k edges of G whose deletion makes G free of induced copies of H . Significant attention has been given to the *kernelizability* aspects of this problem – i.e., for which graphs H does the problem admit an “efficient preprocessing” procedure, known as a *polynomial kernelization*, where an instance I of the problem with parameter k is reduced to an equivalent instance I' whose size and parameter value are bounded polynomially in k ? Although such routines are known for many graphs H where the class of H -free graphs has significant restricted structure, it is also clear that for most graphs H the problem is *incompressible*, i.e., admits no polynomial kernelization parameterized by k unless the polynomial hierarchy collapses. These results led Marx and Sandeep to the conjecture that H -FREE EDGE DELETION is incompressible for any graph H with at least five vertices, unless H is complete or has at most one edge (JCSS 2022). This conjecture was reduced to the incompressibility of H -FREE EDGE DELETION for a finite list of graphs H . We consider one of these graphs, which we dub the *prison*, and show that PRISON-FREE EDGE DELETION has a polynomial kernel, refuting the conjecture. On the other hand, the same problem for the complement of the prison is incompressible.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Graph modification problems, parameterized complexity, polynomial kernelization

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.52

Related Version *Full Version*: <https://arxiv.org/pdf/2501.15952>

1 Introduction

Let H be a graph. A graph G is H -free if it does not contain H as an induced subgraph. More generally, let \mathcal{H} be a collection of graphs. A graph is \mathcal{H} -free if it is H -free for every $H \in \mathcal{H}$. In the H -FREE EDGE EDITING (respectively \mathcal{H} -FREE EDGE EDITING) problem, given a graph G and an integer k , the task is to add or delete at most k edges from G such that the result is H -free (respectively \mathcal{H} -free). The EDGE DELETION and EDGE COMPLETION variants are the variants where only deletions, respectively only adding edges is allowed. These are special cases of the much more general *graph modification* problem class, where a problem is defined by a graph class \mathcal{G} and the natural problem variants (\mathcal{G} EDGE EDITING/DELETION/COMPLETION respectively \mathcal{G} VERTEX DELETION) are where the input graph G is to be modified so that the result is a member of \mathcal{G} .

As Cai [2] noted, for every finite \mathcal{H} , the \mathcal{H} -free graph modification problems are FPT with a running time of $O^*(2^{O(k)})$ by a simple branching algorithm. However, the question of *kernelization* is much more subtle. A *polynomial kernelization* for a parameterized problem is a polynomial-time procedure that takes as input an instance of the problem, for example, $I = (G, k)$ in the case of a graph modification problem, and outputs an instance (G', k') of



© Séhane Bel Houari-Durand, Eduard Eiben, and Magnus Wahlström;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 52; pp. 52:1–52:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



the same problem such that $|V(G')|, k' \leq p(k)$ for some polynomially bounded function $p(k)$ of k , and such that (G', k') is a yes-instance if and only if (G, k) is a yes-instance. If so, we say that the problem has a *polynomial kernel*. This has been used as a way to capture the notion of efficient instance simplification and preprocessing, and deep and extensive work has been done on determining whether various parameterized problems have polynomial kernels or not (under standard complexity-theoretical assumptions). See the book by Fomin et al. [11].

For many graph modification problems, both those characterized by finite and infinite families \mathcal{H} , polynomial kernelization is known, but for many others, the question is wide open; see the survey of Crespelle et al. on parameterized edge modification problems [5]. For the structurally simpler case of H -FREE EDGE DELETION, if H is a clique then the problem reduces to d -HITTING SET for $d = |E(H)|$ and has a polynomial kernel by the sunflower lemma, and if $|E(H)| \leq 1$ then the problem is trivial. For the same reason, H -FREE VERTEX DELETION has a polynomial kernel for every fixed H . But in all other cases, the question is more intricate, since deleting an edge in one copy of H in G can cause another copy of H to occur, implying a dependency between modifications that is not present in the simpler cases. Beyond cliques and near-empty graphs, polynomial kernels are known when H is P_3 (i.e., H -free graphs are cluster graphs) [12], P_4 (i.e., H -free graphs are cographs) [13], the paw [7] and the diamond [4] (see Figure 1). Kernels are also known for several simple classes characterized by finite sets \mathcal{H} . But there are significant open cases; Crespelle et al. [5] highlight the classes of claw-free graphs and line graphs, although the case of line graphs has since been resolved [6]. Initially, progress led Fellows et al. [9] to ask (very speculatively) whether \mathcal{H} -free graph modification problems have polynomial kernels for all finite \mathcal{H} . This was refuted by Kratsch and Wahlström [14], and after a series of lower bounds, most importantly by Cai and Cai [3], the answer now appears to be the opposite – the \mathcal{H} -free graph modification problems have polynomial kernels only for particularly restrictive choices of \mathcal{H} . Furthermore, in all such cases the kernel depends intimately on the structural characterization of the graph class, such as structural decomposition results. However, it would appear unlikely that such a structural characterization of H -free graphs should exist for any arbitrary graph H , and correspondingly, we would expect H -free edge modification problems not to admit polynomial kernelization. For example, despite the above-mentioned positive results, H -FREE EDGE DELETION has no kernel for $H = P_\ell$ where $\ell \geq 5$, for $H = C_\ell$ for $\ell \geq 4$ or for any H such that H or its complement is 3-connected (excepting the trivial cases) [3]. Marx and Sandeep [15] pushed the pendulum in the other direction and conjectured that for graphs H on at least five vertices, only the above-mentioned immediate kernels exist, conjecturing the following.

► **Conjecture 1** (Conjecture 2 of [15]). *H -FREE EDGE DELETION does not have a polynomial kernel for any graph H on at least 5 vertices, unless H is complete or has at most one edge.*

They showed that this conjecture is equivalent to the statement that H -FREE EDGE DELETION admits no polynomial kernelization if H is one of nineteen specific graphs on five or six vertices. They also gave a corresponding conjecture for H -FREE EDGE EDITING (where $|E(H)| = 1$ is no longer a trivial case), and the case of H -FREE EDGE COMPLETION follows by dualization.

In this paper, we refute this conjecture. We study the graph H shown in Figure 1 (the complement of $P_3 + 2K_1$), which we dub a *prison* (given that it can be drawn as the 5-vertex “house” graph with additional crossbars added). This is the first graph in the set \mathcal{H} in [15]. We show that PRISON-FREE EDGE DELETION has a polynomial kernel. On the other hand, PRISON-FREE EDGE COMPLETION admits no polynomial kernelization unless the polynomial hierarchy collapses. We leave PRISON-FREE EDGE EDITING open for future work.

1.1 Our results

As expected, our result builds on a characterization of prison-free graphs. We then derive the kernelization and lower bound results working over this characterization.

Prison-free graphs. The start of our results is a structure theorem for prison-free graphs. To begin with, note that the 4-vertex induced subgraphs of a prison are K_4 , the diamond and the paw; see Figure 1. The structure of diamond-free and paw-free graphs are known: A graph is diamond-free if and only if it is *strictly clique irreducible*, i.e., every edge of the graph lies in a unique maximal clique [17], and a graph is paw-free if and only if every connected component is either triangle-free or complete multipartite [16]. The structure of prison-free graphs generalizes both.

A *complete multipartite graph* with classes P_1, \dots, P_m is a graph whose vertex set is the disjoint union $P_1 \cup \dots \cup P_m$ and with an edge uv for $u \in P_i$ and $v \in P_j$ if and only if $i \neq j$. Note that a clique is a complete multipartite graph where every part is a singleton. We show the following.

► **Theorem 2.** *A graph $G = (V, E)$ is prison-free if and only if the following holds: Let $F \subseteq V$ be an inclusion-wise maximal set such that $G[F]$ is complete multipartite with at least 4 parts, and let $v \in V \setminus F$. Then $N(v)$ intersects at most one part of F .*

Furthermore, let $\text{cmd}_p(G)$ for $p \in \mathbb{N}$ be the collection of all inclusion-wise maximal $F \subseteq V(G)$ such that $G[F]$ is complete multipartite with at least p classes. We show that $\text{cmd}_4(G)$ induces a form of partition of the cliques of G .

► **Corollary 3.** *Let $G = (V, E)$ be a prison-free graph. The following hold.*

1. *If $F, F' \in \text{cmd}_4(G)$ are distinct and $F \cap F' \neq \emptyset$, then $F \cap F'$ intersects only one class C of F and C' of F' , and there are no edges between $F \setminus C$ and $F' \setminus C$. In particular, $G[F \cap F']$ is edgeless.*
2. *If $F, F' \in \text{cmd}_4(G)$ with $F \cap F' = \emptyset$, then for every $v \in F'$, $N(v)$ intersects at most one class of F .*
3. *Let $e \in E(G)$ be an edge that occurs in at least one K_4 in G . Then there is a unique $F \in \text{cmd}_4(G)$ such that e occurs in $G[F]$.*

In particular, every K_p in G , $p \geq 4$ is contained in a unique $F \in \text{cmd}_4(G)$. It also follows that $\text{cmd}_4(G)$ can be enumerated in polynomial time in prison-free graphs.

Lower bound for prison-free completion. We show that PRISON-FREE EDGE COMPLETION is incompressible, i.e., admits no polynomial kernel parameterized by k unless the polynomial hierarchy collapses. Counterintuitively, this result exploits the property that minimum solutions for the problem can be *extremely expensive* – we can design a sparse graph G such that every prison-free supergraph of G contains $\Theta(n^2)$ edges (for example, by ensuring that the only possible cmd_4 -decomposition of a prison-free supergraph of G consists of a single component $F = V(G)$). More strongly, we use this to show an additive *gap hardness* version of the problem.

► **Theorem 4.** *For any $\varepsilon > 0$, it is NP-hard to approximate PRISON-FREE EDGE COMPLETION up to an additive gap of $g = \Theta(n^{2-\varepsilon})$, even if G has an edge e such that $G - e$ is K_4 -free.*

With this in place, we can proceed with the lower bound using standard methods, using the notion of cross-composition [1, 11]. We follow the method used in previous lower bounds against kernelization of H -free edge modification problems [14, 3]. Given a list I_1, \dots, I_t of

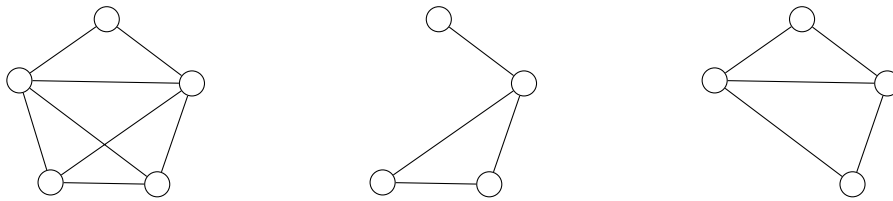
instances of the above gap-version of PRISON-FREE EDGE COMPLETION with parameter value k , our task is to produce an instance of PRISON-FREE EDGE COMPLETION with parameter $(k' + \log t)^{O(1)}$ which corresponds to the OR of the instances I_i . For this, we define a binary tree of height $O(\log t)$ and place the instances at the leaves of the tree. At the root of the tree, we place a single induced prison. For the internal nodes, we design *propagational gadgets* with the function that if the gadget at the node is edited, then one of the gadgets at the children of the node must be edited as well. Finally, for every instance $I_j = (G_j, k)$ with an edge e_j such that $G_j - e_j$ is prison-free, we connect e_j to the corresponding gadget at the leaf of the tree and remove e_j from G_j . This forces at least one edge e_j , $j \in [t]$ to be added to the resulting graph and the original instance $I_j = (G_j, k)$ must be solved.

The crux is that, unlike previous proofs (for example in Cai and Cai [3] when H is 3-connected) we cannot “control” the spread of the edge completion solution to be confined to G_j . On the contrary, the solution must spread all the way to the root and incorporate all vertices from gadgets on the root-leaf path of the binary tree into a single complete multipartite component of the resulting graph G' . Thus, we have no tight control over the number of edges added in the corresponding solution. However, by the strong lower bound on gap-hardness of Theorem 4 we do not need tight control – we can simply set the additive gap g large enough that the number of edges added outside of G_j in the resulting propagation is a lower-order term compared to g .

► **Theorem 5.** *PRISON-FREE EDGE COMPLETION does not have a polynomial kernel parameterized by k unless the polynomial hierarchy collapses.*

Kernel for prison-free deletion. The kernelization algorithm depends directly on the structural characterization of prison-free graphs. We start by using the sunflower lemma to obtain a small set \mathcal{P}' of prisons in the graph G such that any set of edges of size at most k that intersects all prisons in \mathcal{P}' has to intersect all prisons in G . We let S be the set of vertices of these prisons. Note that $|S| = \mathcal{O}(k^8)$, and outside of S we only need to be concerned by prisons that are created by deleting some edge from G . This lets us delete all edges not in $E(G[S])$ that do not belong to a strict supergraph of a prison. In addition, if a prison in G contains a single edge inside S , then such an edge has to be included in every solution of size at most k , so it can be deleted and k decreased. After exhaustive application of these two reduction rules, we show that the edges of $G[V(G) \setminus S]$ can be partitioned into maximal complete multipartite subgraphs of $G[V(G) \setminus S]$ (even those which do not occur in $\text{cmd}_4(G[V(G) \setminus S])$). For each of these maximal complete multipartite subgraphs, we can check whether it is in some larger complete multipartite subgraph F in G that is nicely separated from the rest of G , in the sense that every $x \in V(G) \setminus F$ neighbors vertices in at most one class of F . For any such F , no supergraph of a prison can contain edge both outside and inside of F and edges inside of F can be safely deleted from G . This allows us to bound the number of maximal complete multipartite subgraphs outside of S by $\mathcal{O}(|S|^3)$. In addition, we show that any supergraph of a prison in G that contains an edge fully outside of S is fully contained in $S \cup F$ for a single maximal multipartite subgraph F of $G[V(G) \setminus S]$. This allows us to treat these multipartite subgraphs separately. Moreover, using the fact that for any edge $e \in G[S]$, the graph $G[V(G) \setminus (S \setminus e)]$ is still prison-free, we can show that the interaction between S and a maximal multipartite subgraph F of $G[V(G) \setminus S]$ is very structured. This allows us to reduce the size of each of these subgraphs and we obtain the following theorem.

► **Theorem 6.** *PRISON-FREE EDGE DELETION admits a polynomial kernel.*



■ **Figure 1** Three graphs: The prison, the paw, and the diamond (as subgraphs of the prison).

Structure of the paper. In Section 2 we derive the basic facts about prison-free graphs. Section 3 contains the lower bound against PRISON-FREE EDGE COMPLETION and Section 4 contains the polynomial kernel for PRISON-FREE EDGE DELETION. We conclude in Section 5.

2 Structure of prison-free graphs

We begin our study by characterizing the structure of prison-free graphs. This generalizes two structures. The most closely related is for *paw-free graphs*; note that the paw is the subgraph of the prison produced by deleting an apex vertex. Olariu [16] showed that a graph is paw-free if and only if every connected component is either triangle-free or a complete multipartite graph. The paw-free graph modification problems have polynomial kernels due to Eiben et al. [7]. Second, less closely related but still relevant, the *diamond* $K_4 - e$ is a subgraph of the prison produced by deleting a vertex of degree 3. It is known that a graph is *diamond-free* if and only if every edge occurs in only one maximal clique [17]. The diamond-free graph modification problems have polynomial kernels due to Cao et al. [4]. In a sense, the structure of prison-free graphs generalizes both, as the edge-sets of cliques K_p , $p \geq 4$ in a prison-free graph G decomposes into complete multipartite induced subgraphs of G .

We first prove Theorem 2 from the introduction, then use it to derive the more informative Corollary 3.

► **Lemma 7** (\star^1). *Let G be a prison-free graph and let $K \subseteq V(G)$ induce a K_4 in G . Then there is a complete multipartite graph $G[F]$ in G with $K \subseteq F$. Furthermore, if F is maximal under this condition, then for any $v \in V(G) \setminus F$, $N(v)$ intersects at most one component of F .*

Inspired by this, for any graph G and $p \geq 2$, define $\text{cmd}_p(G)$ to consist of all maximal subsets $F \subseteq V$ such that $G[F]$ is complete multipartite with at least p parts. We refer to $\text{cmd}_p(G)$ as the *complete multipartite decomposition* of G (and we will note that it is indeed a decomposition for $p \leq 4$ if G is prison-free). The following theorem is now an extension of the previous lemma. Proofs of Theorem 2 and Corollary 3 are deferred to the full version.

► **Theorem 2.** *A graph $G = (V, E)$ is prison-free if and only if the following holds: Let $F \subseteq V$ be an inclusion-wise maximal set such that $G[F]$ is complete multipartite with at least 4 parts, and let $v \in V \setminus F$. Then $N(v)$ intersects at most one part of F .*

We will use this to derive a more useful description of prison-free graphs.

¹ Results marked with \star have their proofs deferred to the full version.

► **Corollary 3.** *Let $G = (V, E)$ be a prison-free graph. The following hold.*

1. *If $F, F' \in \text{cmd}_4(G)$ are distinct and $F \cap F' \neq \emptyset$, then $F \cap F'$ intersects only one class C of F and C' of F' , and there are no edges between $F \setminus C'$ and $F' \setminus C$. In particular, $G[F \cap F']$ is edgeless.*
2. *If $F, F' \in \text{cmd}_4(G)$ with $F \cap F' = \emptyset$, then for every $v \in F'$, $N(v)$ intersects at most one class of F .*
3. *Let $e \in E(G)$ be an edge that occurs in at least one K_4 in G . Then there is a unique $F \in \text{cmd}_4(G)$ such that e occurs in $G[F]$.*

In particular, the last item implies that every K_p , $p \geq 4$ is contained in a unique multipartite component $F \in \text{cmd}_4(G)$. Since every $F \in \text{cmd}_4(G)$ contains a K_4 , and each maximal component F can be found greedily, $\text{cmd}_4(G)$ can be computed efficiently.

3 Incompressibility of Prison-free Edge Completion

In this section, we show that PRISON-FREE EDGE COMPLETION admits no polynomial kernel unless the polynomial hierarchy collapses. The proof is in two parts. First we show a strong inapproximability result – it is NP-hard to approximate PRISON-FREE EDGE COMPLETION within an additive gap of $g = O(n^{2-\varepsilon})$ for every $\varepsilon > 0$, even for graphs with prison-free edge deletion number 1. We then use this to show a cross-composition (see below) for PRISON-FREE EDGE COMPLETION, thereby ruling out polynomial kernels under standard complexity conjectures. This latter part roughly follows the outline of previous proofs of incompressibility [14, 3].

3.1 Initial observations and support gadgets

We begin with some useful statements.

► **Proposition 8.** *For a complete multipartite graph K with parts of sizes a_1, a_2, \dots, a_m , the number of edges of K is $\frac{1}{2} \sum_{i \neq j} a_i a_j = \frac{1}{2} (|K|^2 - \sum_i a_i^2)$.*

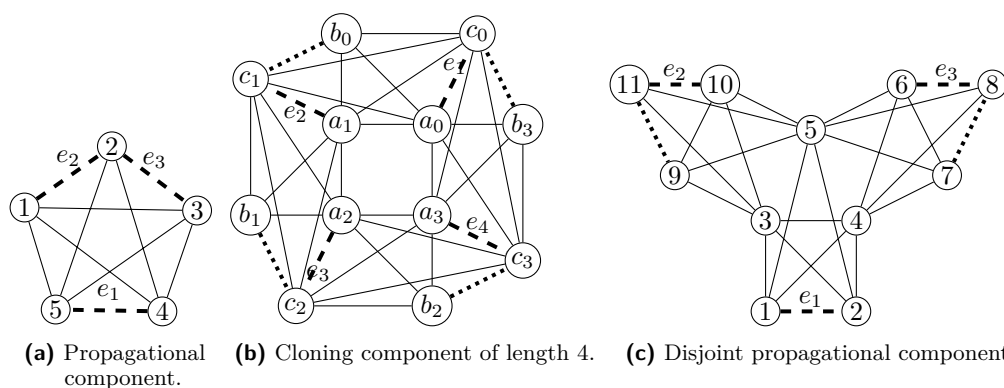
For a graph $G = (V, E)$ and a set of edges A over V , we let $G \cup A$ denote the graph $G' = (V, E \cup A)$. A *prison-free completion set* for G is an edge set A over $V(G)$ such that $G \cup A$ is prison-free. A *solution* to (G, k) is a prison-free completion set A for G with $|A| \leq k$. The following is essential in our lower bounds.

► **Lemma 9** (\star). *Let G be a graph with exactly one induced K_4 and let A be a minimal prison-free completion set for G . Then $\text{cmd}_4(G \cup A)$ has exactly 1 component and A lives within that component.*

We next show a way to enforce *forbidden edges*, i.e., non-edges uv in G such that no prison-free completion set for G of at most k edges contains uv .

► **Lemma 10** (\star). *Let G be a graph, $k \in \mathbb{N}$, and $u, v \in V(G)$ with $u \neq v$ such that $uv \notin E(G)$. There is a graph G' on vertex set $V(G') = V(G) \cup F$ such that $G = G' - F$ and the following holds: the minimal solutions to (G', k) are precisely the minimal solutions A to (G, k) such that $uv \notin A$. Furthermore, every solution A to (G', k) satisfies $uv \notin A$.*

We now proceed with gadget constructions. A *propagational component* is a graph containing 3 distinct non-edges e_1, e_2 and e_3 such that for any graph G and subset A of $V(G)^2$, if $G \cup A$ is prison-free and $e_1 \in A$, then $e_2 \in A$ or $e_3 \in A$. Figure 2a shows this component. In what follows, we will deduce gadgets from this propagation property, leaving the proof of their prison-freeness for later.



■ **Figure 2** Gadgets for Section 3. Dotted lines are forbidden edges; dashed lines are named “gadget-edges” with special semantics.

► **Definition 11.** Let $l \geq 4$. A cloning component of length l is a component over the vertices a_0, \dots, a_{l-1} , b_0, \dots, b_{l-1} , c_0, \dots, c_{l-1} with the edges such that for all $0 \leq i \leq l-1$, $a_{i+1}, c_{i+1}, b_i, c_i, a_i$ induces a propagational component with $e_1^i, e_2^i, e_3^i = a_i c_i$, $a_{i+1} c_{i+1}$, $c_{i+1} b_i$, and the edge $e_3^i = c_{i+1} b_i$ is forbidden. All arithmetic here is modulo l .

A cloning component is drawn in Figure 2b. Note that for all $0 \leq i \leq l-1$, if a solution A for an instance (G, k) contains e_1^i in a cloning component, then it contains e_2^i (since $e_2^i \in A$ or $e_3^i \in A$, but e_3^i is forbidden). We inductively obtain the next property.

► **Lemma 12** (*). Let $l \geq 4$, $k \geq 1$ and G be a graph containing an induced cloning component X of size l with vertices named a_i , b_i and c_i as above. Let A be a subset of non-edges such that $|A| \leq k$ and $G \cup A$ is prison-free. Then either $\{a_i c_i \mid 0 \leq i \leq l-1\} \cap A = \emptyset$ or $a_i c_i \in A$ for every $0 \leq i \leq l-1$. Furthermore, in the latter case all of X is contained in a complete multipartite component of $G \cup A$.

We define one final gadget, shown in Figure 2c. This does the same job as a propagational component – i.e., if $e_1 \in A$ then $e_2 \in A$ or $e_3 \in A$ – except that the edges e_1 , e_2 , e_3 are pairwise vertex-disjoint. This is the *propagational gadget* mentioned in the proof overview.

► **Definition 13.** A disjoint propagational component is a graph isomorphic to the graph shown in Figure 2c, i.e., a graph on a vertex set $V = \{v_1, \dots, v_{11}\}$ such that vertex sets $\{v_1, \dots, v_5\}$, $\{v_4, \dots, v_8\}$ and $\{v_3, v_5, v_9, v_{10}, v_{11}\}$ all induce propagational components, $v_1 v_2$, $v_3 v_5$, $v_4 v_5$, $v_6 v_8$ and $v_{10} v_{11}$ are standard non-edges and $v_7 v_8$ and $v_9 v_{11}$ are forbidden edges. The edge labelled $e_1 = v_1 v_2$ is referred to as input edge and the edges $e_2 = v_{10} v_{11}$ and $e_3 = v_6 v_8$ are output edges.

3.2 NP-hardness of Gap Prison-free Edge Completion

We now prove the first half of the incompressibility result for PRISON-FREE EDGE COMPLETION, namely that it remains NP-hard even in a strong additive gap version.

Let GAP PRISON-FREE EDGE COMPLETION be the variant of PRISON-FREE EDGE COMPLETION where the input is a triple (G, k, g) and the task is to distinguish between the following cases:

1. G has a prison-free completion set of at most k edges.
2. G has no prison-free completion set of fewer than $k + g$ edges.

For intermediate cases (where the size t of a minimum-cardinality prison-free editing set is $k < t < k + g$) the output may be arbitrary. The following is the more precise version of Theorem 4 from the introduction. We defer the construction to the full version of the paper.

► **Theorem 14** (★). *For any $\varepsilon > 0$, it is NP-hard to distinguish between yes-instances and no-instances (G, k, g) of GAP PRISON-FREE EDGE COMPLETION even if G contains an edge e such that $G - e$ is K_4 -free and the gap is $g = \Theta(n^{2-\varepsilon})$.*

Proof sketch. We can only give a brief sketch here and refer to the full version for details. The result is shown by a reduction from VERTEX COVER on cubic graphs. At its heart is the following principle. We create a graph G which has a single induced K_4 . Then by Lemma 9, for any minimal prison-free completion set A of G , $V(A)$ will be contained in a single complete multipartite component F of $G \cup A$. Using forbidden edges, we can have tight control over the partition of F , which lets us predict $|A|$ well. In particular, we can ensure that the number of edges $|A|$ added scales quadratically with $|V(A)|$.

Our reduction uses two gadgets: cloning components, of sufficiently large length ℓ , and disjoint propagational components. Let (G, t) where $G = (V, E)$ be an input instance of VERTEX COVER where G is a cubic graph. Using one initial cloning component X_0 with a single seeded active edge e , we can force the propagation of e into $m = |E|$ disjoint activation edges e_i , one for every edge of E . For every edge $ab \in E$, associated with an edge e_i of X_0 , we create a disjoint propagational component with input edge e_i and output edges $f_{a,i}$ and $f_{b,i}$ associated with the vertices a and b of G . Finally, for every vertex $v \in V$ we create a very large cloning component, which contains all edges $f_{v,j}$ associated with it, and whose length ℓ depends on the desired gap g . Thus, prison-free completion sets A of the resulting graph G' which activate the cloning components of s distinct vertices of G , lead to a prison-free supergraph $G' \cup A$ of G' where the unique complete multipartite component F of $\text{cmd}_4(G' \cup A)$ contains $O(\ell s)$ vertices, guaranteeing that $|A| = \Theta((s\ell)^2)$ for a corresponding minimum solution A . We can now achieve the desired gap by tuning ℓ to be sufficiently large. ◀

3.3 Compositionality of Prison-Free Edge Completion

We prove Theorem 5 by an or-composition over instances of GAP PRISON-FREE EDGE COMPLETION, using Theorem 4 to support the composition.

We recall some definitions [1]. A *polynomial equivalence relation* is an equivalence relation on Σ^* such that the following hold:

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ decides in time polynomial in $|x| + |y|$ whether x and y are equivalent.
2. For any finite set $S \subset \Sigma^*$, the number of equivalence classes that S is partitioned to is polynomially bounded in the size of the largest element of S .

Let $L \subseteq \Sigma^*$ be a language, \mathcal{R} a polynomial equivalence relation and $Q \subseteq \Sigma^* \times \mathbb{N}$ a parameterized language. An *OR-cross-composition of L into Q (with respect to \mathcal{R})* is an algorithm that given t instances $x_1, \dots, x_t \in \Sigma^*$ of L belonging to the same equivalence class of \mathcal{R} , uses time polynomial in $\sum_{i=1}^t |x_i|$ and outputs an instance (y, k) of Q such that the following hold:

1. The parameter value k is polynomially bounded in $\max_i |x_i| + \log t$.
 2. (y, k) is a yes-instance of Q if and only if at least one instance x_i is a yes-instance of L .
- If an NP-hard language L has an OR-cross-composition into a parameterized problem Q then Q admits no polynomial kernelization, unless the polynomial hierarchy collapses [1].

We proceed to show this for PRISON-FREE EDGE COMPLETION.

We present here a high-level sketch of the result, based directly on Theorem 4. In the full version, we present a more careful proof with explicit parameters. Thus for simplicity, let $(G_i, k_i, g_i)_{i=1}^t$ be a sequence of instances of GAP PRISON-FREE EDGE COMPLETION with a sufficiently high gap value $g_i = \Theta(|V(G_i)|^{2-\varepsilon})$, $\varepsilon > 0$, to be tuned later. Via a polynomial equivalence relation, we may assume that $|V(G_i)| = n_0$, $|E(G_i)| = m_0$, $k_i = k_0$ and $g_i = g$ holds for every input instance i . By Theorem 4, we assume that for every instance (G_i, k_i, g_i) there is a single edge $e_i \in E(G_i)$ such that $G_i - e_i$ is prison-free; refer to this as the *activation edge* of G_i and delete it from the graph.

For the composition, let $h \in \mathbb{N}$ be such that $2^{h-1} < t \leq 2^h$. Define a balanced binary tree of height h whose leaves are labelled L_i , $i = 1, \dots, t$. Place a disjoint propagational component for every internal node x of the tree, identifying the two output edges with the children of x and the input edge at x with the corresponding output edge of the parent of x . Initially, all input and output edges are absent except that the input edge at the root of the tree is present. Finally identify the output edge leading into the leaf L_i with the activation edge e_i of the graph G_i . If there are any unused output edges, for example if t is odd, make these edges forbidden. Let (G, k) be the resulting instance of PRISON-FREE EDGE COMPLETION where k is yet to be determined.

► **Lemma 15** (\star). *Let A be a prison-free completion set for G . Then there is some $i \in [t]$ such that A contains the activation edge e_i of the graph G_i . Furthermore, for every $i \in [t]$, any minimal prison-free edge completion set A_i for $G_i + e_i$ can be completed into a prison-free completion set A for G which contains every output edge in the path from the root to L_i but no other output edges from the tree.*

We are now ready to finish the proof.

► **Theorem 5.** *PRISON-FREE EDGE COMPLETION does not have a polynomial kernel parameterized by k unless the polynomial hierarchy collapses.*

Proof sketch. We show that the construction above is a cross-composition into PRISON-FREE EDGE COMPLETION with parameter k . Consider briefly the two cases. First, if for some $i \in [t]$ the input instance (G_i, k_0, g_0) is positive, let A_i be a prison-free completion set for G_i with $|A_i| \leq k$. By Lemma 15 there is a prison-free completion set $A \supseteq A_i$ for G that modifies edges along the root-leaf path to L_i of the binary tree, and does not contain any other output edge of the tree. By Lemma 9 and minimality of A and A_i we may now assume that A touches no vertices of the binary tree except on the root-leaf path to L_i , and contains no further edges inside G_i beyond A_i . Let n_t be the number of vertices in the gadgets of the tree incident with edges of A ; since the tree has height $h = O(\log t)$ and each node is constant size, we have $n_t = O(\log t)$. Thus

$$|A| \leq |A_i| + (n_h + n_0)n_h \leq k_0 + O((n_0 + \log t) \log t).$$

On the other hand, assume that for every $i \in [t]$ the minimum prison-free completion set A_i for G_i has $|A_i| \geq k_0 + g_0$. Let A be a prison-free completion set for G . By Lemma 15 there is some $i \in [t]$ such that the activation edge e_i of G_i is contained in A . Thus A contains a prison-free completion set for G_i and $|A| \geq k_0 + g_0$. By the construction of Theorem 4, we can tune the parameters so that these quantities separate, and choose a parameter k where

$$k_0 + O((n_0 + \log t) \log t) \leq k < k_0 + g_0$$

in which case the reduction is complete. ◀

4 Polynomial kernel for Prison-free Edge Deletion

In this section we will find a polynomial kernel for PRISON-FREE EDGE DELETION. Let G be a graph and $k \geq 1$. We will fix (G, k) throughout this section to be an instance of PRISON-FREE EDGE DELETION.

Throughout this section, for a graph G , and an edge $e = uv$ of G , we call common neighborhood of e the set $N_G(e) = N_G(u) \cap N_G(v)$. In addition, given a graph G and a set of vertices $S \subseteq V(G)$, we denote by \bar{S} the set $V(G) \setminus S$, when G is clear from the context.

4.1 Finding a Small Vertex Modulator

We start by finding a small subset of vertices S such that any edgeset $A \subseteq E(G)$ with at most k edges that intersects all prisons in $G[S]$ also intersects all prisons in G . While this is not sufficient for a kernel, as deleting an edge can create a prison, outside of this set, we only need to focus on prisons that are created by deleting an edge. To obtain this set, we will use well known Sunflower Lemma due to Erdős and Rado [8].

A *sunflower* in a set family \mathcal{F} is a subset $\mathcal{F}' \subseteq \mathcal{F}$ such that all pairs of elements in \mathcal{F}' have the same intersection called *core*.

► **Lemma 16** (Sunflower Lemma, [8, 10]). *Let \mathcal{F} be a family of subsets of a universe U , each of cardinality exactly b , and let $a \in \mathbb{N}$. If $|\mathcal{F}| \geq b!(a-1)^b$, then \mathcal{F} contains a sunflower \mathcal{F}' of cardinality at least a . Moreover, \mathcal{F}' can be computed in time polynomial in $|\mathcal{F}|$.*

► **Lemma 17** (\star). *We can in polynomial time either determine that (G, k) is no-instance of PRISON-FREE EDGE DELETION or compute a set $S \subseteq V(G)$ with $|S| \leq 5 \cdot 8! \cdot (k+1)^8$ such that for every prison P in G and every $A \subseteq E(G)$ with $|A| \leq k$, it holds that if $G[S] \Delta A$ is prison-free, then $A \cap E(G[P]) \neq \emptyset$.*

Proof Sketch. The proof is a straightforward application of Lemma 16. Let $\mathcal{S} = \{E(P) \mid P \text{ is a prison in } G\}$. Note that each set $X \in \mathcal{S}_0$ contains precisely 8 edges. We iteratively apply Lemma 16 on \mathcal{S} to find a sunflower of size $k+2$. If the core of the sunflower \mathcal{S}' is empty, then G contains $k+2$ edge-disjoint prisons and (G, k) is no instance. Else any solution of size at most k intersects the core and that is the case even if we require to hit only $k+1$ prisons of \mathcal{S}' , so we remove arbitrary prison from \mathcal{S} and repeat the procedure until no sunflower of size $k+2$ can be found. At that point \mathcal{S} contains at most $8! \cdot (k+1)^8$ many prisons and any $A \subseteq E(G)$ with $|A| \leq k$ that intersect all of the prisons that are left in \mathcal{S} intersects all prisons in G . We let S to be the set of all vertices in these prisons. ◀

For the rest of the section and of the proof, we let S be the set computed by Lemma 17. It follows, as long as we keep $G[S]$ as the subgraph of the reduced instance, we only need to be concerned about the prisons that are created by removing some edge from G , as all the prisons that were in G to start with are hit by a set A of at most k edges as long as $G[S] \Delta A$ is prison-free. Given the above, the following two reduction rules are straightforward.

► **Reduction Rule 1** (\star). *If an edge $e \in (E(G) \setminus E(G[S]))$ is not in a strict supergraph of a prison, delete it.*

► **Reduction Rule 2** (\star). *For every prison P in G , if $E(G[S]) \cap P = \{e\}$, then remove e and decrease k by one.*

Thanks to these Reduction Rules, we found a set S of vertices of size polynomial in k such that for any subset S' of vertices of S such that $G[S']$ has at most one edge, $G[\bar{S} \cup S']$ is prison-free. We note that we assume that all reduction rules are applied exhaustively;

that is whenever a reduction rule is applicable, we apply it and restart the process from the beginning. Hence, in all statements in the rest of the section, we implicitly assume that none of the reduction rules can be applied.

4.2 Consequences on the Structure of $G[\bar{S}]$

Now we are able to show the following properties that will be useful for the kernel. The following lemma gives us a stronger property than just the characterisation of prison-free graphs.

► **Lemma 18** (\star). *Let $F \in \text{cmd}_3(G[\bar{S}])$ be a maximal complete multipartite subgraph of $G[\bar{S}]$ such that F has exactly three classes. Then there exists $s \in S$ with $F \subseteq N(s)$. Moreover, there is a strict supergraph of a prison that contains s and an edge in F .*

Proof Sketch. Since F has three classes, there are u, v, w such that uvw is a triangle in $G[\bar{S}]$. Due to Reduction Rule 1, each edge of F is in a strict supergraph of a prison and hence in K_4 in G . Now given a K_4 (a, b, u, v) that contains uv , we conclude that w is adjacent to either a or b , else (a, b, u, v, w) induces a prison with at most one edge in S and Reduction Rule 2 applies. Hence $\{u, v, w\}$ is a subset of a K_4 , say (u, v, w, s) . As a consequence of Lemma 7 and since $G[\bar{S}]$ is prison-free, one can show that F has to be fully included in the K_4 -free part of $G[\bar{S}]$, and hence $s \in S$. Now, any vertex $x \in F \setminus \{u, v, w\}$ is adjacent to exactly two vertices in $\{u, v, w\}$ and hence if $sx \notin E(G)$, then (u, v, w, s, x) induces a prison without an edge in S , which is impossible due to the construction of S . ◀

► **Lemma 19** (\star). *For all $F \in \text{cmd}_3(G[\bar{S}])$, for all $x \in \bar{S} \setminus F$, $N_G(x)$ intersects at most one class of F .*

Proof Sketch. The Lemma follows for $F \in \text{cmd}_4(G[\bar{S}])$ by Lemma 7. Hence, we can assume $F \in \text{cmd}_3(G[\bar{S}]) \setminus \text{cmd}_4(G[\bar{S}])$. For the sake of contradiction, assume that for $x \in \bar{S} \setminus F$, we have that $N(x) \cap F$ contains an edge uv . Since F has three classes, for every w in the class C that does not contain u nor v , we have that uvw is a triangle. Moreover, similarly as in the proof of Lemma 18, one can show using Lemma 7 that (u, v, w, x) cannot be K_4 and so $wx \notin E(G)$. Since, F is inclusion maximal, there is $a \in F \setminus C$ such that $xa \notin E(G)$. By Lemma 18, there is $s \in S$ with $F \subseteq N(s)$. But then either (u, v, w, s, x) or (u, v, a, s, x) induces a prison with no edge in $G[S]$ (depending on whether $sx \in E(G)$ or not), which is impossible due to the construction of S . ◀

► **Lemma 20** (\star). *If a supergraph P of a prison of G has an edge in $F \in \text{cmd}_3(G[\bar{S}])$, then $P \subseteq S \cup F$.*

Proof Sketch. For the sake of contradiction, let P be a supergraph of a prison of G that has an edge uv in $F \in \text{cmd}_3(G[\bar{S}])$ and there is $x \in P \setminus (S \cup F)$. One can show that due to Lemma 19 and application of Reduction Rule 2, $P = (a, b, u, v, x)$ is a strict supergraph of a prison with $\{a, b\} \subseteq S$ and vx (or ux) being the only non-edge of P . Moreover, F contains triangle uvw , as it has at least three classes and $wx \notin E(G)$ by Lemma 19. Due to Reduction Rule 2, (u, v, w, a, b) cannot induce a prison, and either $wa \in E(G)$ or $wb \in E(G)$. If $wa \in E(G)$ (resp. $wb \in E(G)$), then (x, a, u, w, v) (resp. (x, b, u, w, v)) induces a prison in G with at most one edge in $G[S]$, contradicting application of Reduction Rule 2. ◀

Let's now focus on the edges of \bar{S} that are not in any triangle. We show that since Reduction Rules 1 and 2 has been exhaustively applied, even these edges can be partitioned to maximal complete bipartite subgraphs.

52:12 Kernelization for Prison-Free Edge Deletion and Completion

Let B be the set of edges of $G[\bar{S}]$ that are not in any triangle in $G[\bar{S}]$. For all $e = ab \in E(G[S])$, we note $B_e = B \cap \{uv : u, v \in N(a) \cap N(b)\}$. Note that every edge $f \in B$ belongs to a K_4 in G due to the application of Reduction Rule 1. Since, f is not in any triangle in $G[\bar{S}]$, it follows that it is in some B_e for $e \in E(G[S])$. On the other hand, we show that if $B_{e_1} \cap B_{e_2} \neq \emptyset$, then $B_{e_1} = B_{e_2}$, otherwise G contains a prison with at most one edge in $G[S]$, which gives us the following lemma.

► **Lemma 21** (\star). $\{B_e \mid e \in E(G[S])\}$ is a partition of B .

The following lemma is a straightforward consequence of Reduction Rule 2, since for every $e \in E(S)$, $G[\bar{S} \cup e]$ is prison-free and hence $N_G(e) \cap \bar{S}$ is \bar{P}_3 -free.

► **Lemma 22** (\star). For all $e \in E(G[S])$, $N_G(e) \cap \bar{S}$ is complete multipartite and if B_e is non empty, $N_G(e)$ is a complete bipartite graph.

► **Lemma 23** (\star). Let $e \in E(G[S])$. Assume that B_e is not empty. Then any induced supergraphs of a prison P that has an edge in B_e is in $B_e \cup S$.

Proof Sketch. Assume that there is an induced supergraphs of a prison $P = (a, b, u, v, w)$, where uv in B_e and $w \in \bar{S} \setminus B_e$. It follows from Reduction Rule 2 and the fact that uv is not in a triangle in $G[\bar{S}]$ that (1) P is a strict supergraph of a prison with only non-edge uw (resp. vw) and (2) $\{a, b\} \subseteq S$. By Lemma 21, $B_e = B_{ab}$. Since B_e is maximal complete bipartite subgraph of $G[\bar{S}]$, the class that contains v (resp. u), contains a vertex v' (resp. u') with $v'w \notin E(G)$. But then (a, b, v, v', w) (resp. (a, b, u, u', w)) induces a prison in G with only one edge inside S , which is a contradiction. ◀

It follows that we can partition the edges of $G[\bar{S}]$ in complete multipartite subgraphs, where one of these complete multipartite subgraphs can intersect another in at most one class. The following reduction rule lets us reduce the number of complete multipartite subgraphs in this partition to $|S|^3 + |S|^2 = \mathcal{O}(k^{24})$. Given this bound, we will reduce size of each of these components as well as number of isolated vertices in $G[\bar{S}]$ by a polynomial function in k as well.

► **Reduction Rule 3** (\star). Let $F' \in \text{cmd}_3(G[\bar{S}])$. If $F \subseteq V(G)$ is a maximal complete multipartite component such that $F \supseteq F'$ and for every vertex $v \notin F$, $N(v)$ intersects at most one class of F , remove all edges of F .

Proof Sketch. It is not difficult to show that there is no supergraph of a prison in G that contains an edge with both endpoints in F and at the same time an edge with at least one endpoint outside of F . Given this for every $A \subseteq E(G)$, every prison in $G \Delta A$ is either all edge in F or all edges in $G - E(F)$, since F is complete multipartite and hence prison-free, it suffices to hit all prisons in $G - E(F)$. ◀

Recall that we always assume that none of the previous reduction rules can be applied, in particular from now on we assume also that Reduction Rule 3 is not applicable.

From now on, we denote $\mathcal{F} = \text{cmd}_3(G[\bar{S}]) \cup \{B_e \mid e \in E(G[S])\}$.

► **Lemma 24** (\star). The edges of $G[\bar{S}]$ can be partitioned into at most $|S|^3 + |S|^2$ many maximal complete multipartite subgraphs of $G[\bar{S}]$.

Proof Sketch. Clearly every edge in $G[\bar{S}]$ is in a maximal complete multipartite subgraph of $G[\bar{S}]$. We only need to show that each edge is in precisely one such subgraph and that their number is at most $|S|^3 + |S|^2$. First note the edges in B are partitioned into at most

$|E(G[S])| \leq |S|^2$ many complete bipartite graphs by Lemma 21. Hence, we only need to consider the edges that are in some $F \in \text{cmd}_3(G[\bar{S}])$. It is a rather straightforward consequence of Lemma 19 that any edge e in F cannot be in any other maximal complete multipartite subgraph in $\text{cmd}_3(G[\bar{S}])$. It remains to show that $|\text{cmd}_3(G[\bar{S}])| \leq |S|^3$.

We will now define a function $f : \text{cmd}_3(G[\bar{S}]) \rightarrow \text{cmd}_4(G)$ such that for all $F \in \text{cmd}_3(G[\bar{S}])$, it holds that (1) $F \subseteq f(F)$ and (2) $f(F)$ contains at least one vertex in S . That is f is injective. We will show that every edge in $G[S]$ is in at most $|S|$ many complete multipartite subgraph in the image of f and every element of the image of f contains an edge of $G[S]$, bounding $\text{cmd}_3(G[\bar{S}])$ by $|E(G[S])| \leq |S|^3$.

If $F \in \text{cmd}_4(G[\bar{S}])$, then since Reduction Rule 3 has been exhaustively applied, there is $v \in S$ such that $N(v)$ intersects at least two classes of $G[F]$, and since $G[\bar{S} \cup \{v\}]$ is prison-free and F contains a K_4 , $G[F \cup \{v\}]$ is complete multipartite. We define $f(F)$ as any maximal complete multipartite component of G containing $F \cup \{v\}$. Else, if $F \in \text{cmd}_3(G[\bar{S}]) - \text{cmd}_4(G[\bar{S}])$, then by Lemma 18, there is a vertex $s \in S$ such $F \subseteq N(s)$ and $G[F \cup \{s\}]$ is a complete multipartite graphs with at least four parts. We define $f(F)$ as a maximal complete multipartite subgraph of G containing $F \cup \{s\}$.

▷ **Claim 25** (★). Let $e = uv$ be an edge of $G[S]$. Then at most $|S|$ elements of $\text{Im}(f)$ contains the vertices of e .

There are thus at most $|S|^3$ elements of $\text{Im}(f)$ that contains an edge in S .

▷ **Claim 26** (★). Each element of $\text{Im}(f)$ contains an edge in S .

So $|\text{Im}(f)| \leq |S|^3$. Since f is injective, $|\text{cmd}_3(G[\bar{S}])| \leq |S|^3$. Therefore, $|\mathcal{F}| = |\text{cmd}_3(G[\bar{S}])| + |\{B_e \mid e \in E(G[S])\}| \leq |S|^3 + |S|^2$. ◀

We will have a kernel once we have bounded the size of a maximal complete multipartite subgraphs and the number of isolated vertices in $G[\bar{S}]$. Before we show how to bound those, let us observe the following two simple lemmas.

▶ **Lemma 27** (★). Let $F \in \text{cmd}_3(G[\bar{S}])$ and $s \in S$. If $N(s)$ intersects more than one class of F , then either $N(s) \cap F = F \setminus C$, where C is a single class in F , or $N(s) \cap F = F$.

▶ **Lemma 28** (★). Let e such that B_e is non empty, and let $s \in S$. $N(s)$ either contains B_e or it intersects at most one class of $G[B_e]$.

4.3 Marking important vertices

We will now mark some important vertices that we will keep to preserve the solutions and afterwards argue that we can remove all the remaining vertices without changing the value of an optimal solution. To better understand why this marking procedure works, it is useful to recall Lemmas 20 and 23 that state that any (not necessarily strict) supergraph of a prison that has at least one edge in some $F \in \mathcal{F}$ is fully contained in $S \cup F$. That is only supergraphs of the prison that can contain vertices in more than one complete multipartite subgraph from \mathcal{F} are those with all edges either in S or between S and $N(S)$. Such supergraphs of a prison have always at most two vertices outside of S . Before we give the details of the marking procedure, let us introduce a concept of *neighborhood pattern*. Let $X \subseteq V(G)$ and $a \in V(G) \setminus X$, then the neighborhood pattern of a in X is $N(a) \cap X$. Moreover, for two vertices $a, b \in V(G) \setminus X$, we say a and b have the same *neighborhood pattern* in X if $N(a) \cap X = N(b) \cap X$.

52:14 Kernelization for Prison-Free Edge Deletion and Completion

Now, let $F \in \mathcal{F}$. By Lemmas 27 and 28, for every $s \in S$, if $N(s) \cap F \neq \emptyset$, then there are only three possibilities how $N(s)$ and F can interact. Namely,

1. $N(s) \cap F \subseteq C$ for a single part C of F ;
2. $N(s) \cap F = F \setminus C$ for a single part C of F ;
3. $N(s) \cap F = F$.

Given the above, we can split the classes C of F into two types.

Type 1. There is $s \in S$ such that $N(s) \cap F \subseteq C$ or $N(s) \cap F = F \setminus C$. We denote the set of classes of Type 1. as \mathcal{T}_F^1 ;

Type 2. The rest, which we denote \mathcal{T}_F^2 .

Note that $|\mathcal{T}_F^1| \leq |S|$ due to Lemmas 27 and 28 all classes C in F with $C \notin \mathcal{T}_F^1$ have exactly the same neighborhood in S .

We compute a set M_F of marked vertices for the component $F \in \mathcal{F}$ as follows. We note that whenever we say, we mark some number x of vertices with some property, we mean that if there are more than x many vertices with that property, we mark arbitrary x many of them, else we mark all of them. Let us start with marking vertices in a class C of Type 1. for every $C \in \mathcal{T}_F^1$. For each $S' \subseteq S$ with $|S'| = 4$, and for each neighborhood pattern ξ in S' , we add to M_F arbitrary $2k + 5$ vertices of C with the neighborhood pattern ξ in S' . Observe that for any S'' with $|S''| < 4$, there is $S' \supset S''$ with $|S'| = 4$ and so for any neighborhood pattern ξ'' in S'' , there is a neighborhood pattern in S' that is equal to ξ'' if restricted to S'' . Hence, using this marking, we marked at least $2k + 5$ vertices with any neighborhood pattern in any S' with $|S'| \leq 4$ as well.

In addition, we pick arbitrary $2k + 5$ classes of F that are in \mathcal{T}_F^2 and add arbitrary $2k + 5$ vertices from each picked class to M_F .

► **Lemma 29** (\star). *Given the above marking procedure, for all $F \in \mathcal{F}$, it holds that $|M_F| \leq |S|^5 \cdot (2k + 5) + (2k + 5)^2$.*

In addition to marking the set M_F for each $F \in \mathcal{F}$, we mark additional set of at most $2^4 \cdot \binom{|S|}{4} \cdot (2k + 5)$ vertices by going over all subsets S' of S of size 4 and for each neighborhood pattern ξ in S' , we mark at most $2k + 5$ additional vertices of $V(G) \setminus S$ that are not in any $F \in \mathcal{F}$ with the given neighborhood pattern ξ in S' . We denote this set of at most $|S|^4 \cdot (2k + 5)$ vertices as M_\emptyset . Additionally, observe that this way, we mark at least $2k + 5$ vertices for each neighborhood pattern in any subset of S of size at most four as well.

► **Lemma 30** (\star). *Let $G' = G[S \cup M_\emptyset \cup \bigcup_{F \in \mathcal{F}} M_F]$. Then (G, k) is yes-instance of PRISON-FREE EDGE DELETION if and only if (G', k) is. In addition $|V(G')| = \mathcal{O}(k^{65})$.*

Proof. The bound on the size of G' follows from the fact that that $|S| = \mathcal{O}(k^8)$, $|\mathcal{F}| \leq |S|^3 + |S|^2$, and Lemma 29. Now, G' is an induced subgraph of G and hence for every $A \subseteq E(G)$, if $G \Delta A$ is prison-free, then also $G' \Delta (A \cap E(G'))$ is. Therefore, if (G, k) is yes-instance, then so is (G', k) .

For the rest of the proof, let us assume that (G', k) is yes-instance and let $A \subseteq E(G')$ be such that $|A| \leq k$ and $G' \Delta A$ is prison-free. We show that $G \Delta A$ is also prison-free. For the sake of contradiction, let's assume that $P = (a, b, c, d, e)$ induces a prison in $G \Delta A$. Let us distinguish two cases depending on whether P contains an edge between two vertices outside of S or not.

Case 1. All edges of P have at least one endpoint in S . Note that in this case, there are at most two vertices of P outside of S .

If only one vertex of P , say a , is outside of S , then clearly a is the only vertex of P not in G' . Note that this also means that none of the edges incident with a is in A . Moreover,

G' contains at least $2k + 5$ vertices with the same neighborhood pattern in $\{b, c, d, e\}$, else a would have been either in M_\emptyset or in M_F for some $F \in \mathcal{F}$. Since $|A| \leq k$, it follows that at least one of these $2k + 5$ vertices, let's call it a' , is not incident to an edge in A and in particular to an edge between a' and a vertex in $\{b, c, d, e\}$. However, then $(G' \Delta A)[\{a', b, c, d, e\}]$ is isomorphic to P and induces a prison, which is a contradiction with $G' \Delta A$ being prison-free.

Now assume that two vertices a and b are outside of S , and there is no edge between a and b in $G \Delta A$. Note that the other non-edge of P share an endpoint with ab , and w.l.o.g., we can assume it is ac . As at least one of a and b is not in G' , $ab \notin E(G)$. Moreover, $ac \in E(G) \cap A$, else P is a prison in G and Lemma 17 implies that A intersects P . Hence, $b \notin V(G')$, by our marking procedure, there are at least $2k + 5$ vertices b' in $V(G') \setminus S$ with $\{c, d, e\} \subseteq N(b')$ and $bb' \notin E(G)$. One of these vertices is not incident on any edge in A . For such a vertex b' , either (a, b', c, d, e) or (a, b, b', d, e) is a prison in $G' \Delta A$ depending on whether $ab' \in E(G)$ or $ab' \notin E(G)$.

Case 2. P contains an edge in $G - S$. Then by Lemmas 20 and 23, it follows that there exists $F \in \mathcal{F}$ such that $P \subseteq S \cup F$. Let $S' = P \cap S$. Now, for $x \in P \setminus S$, if x belongs to $C \in \mathcal{T}_F^1$, then by construction of M_F and the fact that $|A| \leq k$, M_F either contains x or a vertex x' such that (1) $x' \in C$, (2) $N(x) \cap S' = N(x') \cap S'$, and (3) x' is not incident to any edge of A . On the other hand, if some of the vertices in $P \cap F$ are in the classes in \mathcal{T}_F^2 , then all such vertices have exactly same neighborhood in S , moreover, either all their respective classes contain vertices in M_F , or there are at least five classes of F that each has $2k + 5$ vertices in G' and no vertex in these classes is incident on an edge in A . Hence, for each vertex x in $P \cap F$ that belongs to a class $C \in \mathcal{T}_F^2$, we can find $C' \in \mathcal{T}_F^2$ and $x' \in C'$ such that (1) $x' \in M_F$, (2) x' is not incident on any edge in A , and (3) if x and y are in $P \cap F$, then x' and y' are in the same class of F if and only if x and y are. Let $P' = (a', b', c', d', e')$ be a subgraph of G' such that for all $x \in \{a, b, c, d, e\}$, if $x \in G'$, then $x' = x$ and else x' is computed as described above, depending on whether $x \in \mathcal{T}_F^1$ or $x \in \mathcal{T}_F^2$. It follows that for all $x, y \in \{a, b, c, d, e\}$, there is an edge $xy \in E(G)$ if and only if $x'y' \in E(G')$. Moreover, since $xy \in A$ implies that $\{x, y\} \subseteq V(G')$, it follows that $xy \in A$ if and only if $x'y' \in A$. Therefore, P' induces a prison in G' , which is a contradiction.

Hence, such prison P in $G \Delta A$ cannot exist and $G \Delta A$ is prison-free as well. Consequently, (G', k) is yes-instance of PRISON-FREE EDGE DELETION if and only if (G, k) is and the Lemma follows. \blacktriangleleft

The polynomial kernel for PRISON-FREE EDGE DELETION then follows from Lemma 30 by observing that all reduction rules as well as marking procedure can be applied in polynomial time.

► **Theorem 6.** *PRISON-FREE EDGE DELETION admits a polynomial kernel.*

5 Conclusions

We have showed that H -FREE EDGE DELETION has a polynomial kernel when H is the 5-vertex graph we call the “prison” (consisting of K_5 minus two adjacent edges). On the other hand, H -FREE EDGE COMPLETION for the same graph H does not have a polynomial kernel unless the polynomial hierarchy collapses. By edge complementation, this is equivalent to the statement that \overline{H} -FREE EDGE DELETION has no polynomial kernel, where \overline{H} is the edge complement of H . The positive result refutes a conjecture by Marx and Sandeep [15], who conjectured that H -FREE EDGE DELETION has no polynomial kernel for any graph H on at least five vertices except trivial cases.

In [15], the conjecture is reduced to the statement that H -FREE EDGE DELETION has no polynomial kernel for any graph H in a list \mathcal{H} of nineteen small graphs, via a sequence of problem reductions. In this naming scheme, the prison is the complement of H_1 . The exclusion of $\text{co-}H_1$ from this list introduces a sequence of new minimal graphs H' for which the kernelization problem is open, out of which the smallest are the prison plus a vertex v which is (respectively) an isolated vertex; attached to a degree-3 vertex of the prison; or attached to both a degree-3 and a degree-2 vertex of the prison (R. B. Sandeep, personal communication, using software published along with [15]). It is at the moment not known whether the new list \mathcal{H}' is finite using the methods of [15].

More broadly, the result suggests that the picture of kernelizability of H -free Edge Modification problems could be more complex than conjectured by Marx and Sandeep. If so, the question of precisely where the tractability line goes for polynomial kernelization seems highly challenging, as all kernelization results so far (including ours) rely on highly case-specific structural characterizations of H -free graphs. We leave these deeper investigations into the problem open for future work. We also leave open the question of a polynomial kernel for PRISON-FREE EDGE EDITING.

References

- 1 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discret. Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.
- 2 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
- 3 Leizhen Cai and Yufei Cai. Incompressibility of H -free edge modification problems. *Algorithmica*, 71(3):731–757, 2015. doi:10.1007/S00453-014-9937-X.
- 4 Yixin Cao, Ashutosh Rai, R. B. Sandeep, and Junjie Ye. A polynomial kernel for diamond-free editing. *Algorithmica*, 84(1):197–215, 2022. doi:10.1007/S00453-021-00891-Y.
- 5 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *Comput. Sci. Rev.*, 48:100556, 2023. doi:10.1016/J.COSREV.2023.100556.
- 6 Eduard Eiben and William Lochet. A polynomial kernel for line graph deletion. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 42:1–42:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.42.
- 7 Eduard Eiben, William Lochet, and Saket Saurabh. A polynomial kernel for paw-free editing. In *IPEC*, volume 180 of *LIPICs*, pages 10:1–10:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.IPEC.2020.10.
- 8 Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.
- 9 Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Peter Shaw. Efficient parameterized preprocessing for cluster editing. In Erzsébet Csuhaj-Varjú and Zoltán Ésik, editors, *Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Budapest, Hungary, August 27-30, 2007, Proceedings*, volume 4639 of *Lecture Notes in Computer Science*, pages 312–321. Springer, 2007. doi:10.1007/978-3-540-74240-1_27.
- 10 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, Berlin, 2006. doi:10.1007/3-540-29953-X.
- 11 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

- 12 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005. doi:10.1007/S00224-004-1178-Y.
- 13 Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. On the (non-)existence of polynomial kernels for P_t -free edge modification problems. *Algorithmica*, 65(4):900–926, 2013. doi:10.1007/S00453-012-9619-5.
- 14 Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. *Discret. Optim.*, 10(3):193–199, 2013. doi:10.1016/J.DISOPT.2013.02.001.
- 15 Dániel Marx and R. B. Sandeep. Incompressibility of H-free edge modification problems: Towards a dichotomy. *J. Comput. Syst. Sci.*, 125:25–58, 2022. doi:10.1016/J.JCSS.2021.11.001.
- 16 Stephan Olariu. Paw-fee graphs. *Inf. Process. Lett.*, 28(1):53–54, 1988. doi:10.1016/0020-0190(88)90143-3.
- 17 W. D. Wallis and G. H. Zhang. On maximal clique irreducible graphs. *J. Comb. Math. Comb. Comput*, 8:187–198, 1990.

On Read- k Projections of the Determinant

Pavel Hrubeš  

Institute of Mathematics of ASCR, Czech Republic

Pushkar S. Joglekar¹  

Vishwakarma Institute of Technology, Pune, India

Abstract

We consider read- k determinantal representations of polynomials and prove some non-expressibility results. A square matrix M whose entries are variables or field elements will be called *read- k* , if every variable occurs at most k times in M . It will be called a *determinantal representation* of a polynomial f if $f = \det(M)$. We show that

- the $n \times n$ permanent polynomial does not have a read- k determinantal representation for $k \in o(\sqrt{n}/\log n)$ (over a field of characteristic different from two).

We also obtain a quantitative strengthening of this result by giving a similar non-expressibility for $k \in o(\sqrt{n}/\log n)$ for an explicit n -variate multilinear polynomial (as opposed to the permanent which is n^2 -variate).

2012 ACM Subject Classification Theory of computation \rightarrow Algebraic complexity theory

Keywords and phrases determinant, permanent, projection of determinant, VNP completeness of permanent

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.53

Related Version *Full Version:* <https://ecc.weizmann.ac.il/report/2024/125/>

Funding *Pavel Hrubeš:* This work was supported by Czech Science Foundation GAČR grant 25-16311S.

1 Introduction

In algebraic complexity theory, two polynomials are of central interest: the determinant and the permanent of a square matrix. If X is an $n \times n$ matrix with indeterminates $x_{i,j}$ as entries, the polynomials are defined as

$$\det_n(X) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i=1}^n x_{i,\sigma(i)}, \quad \operatorname{perm}_n(X) = \sum_{\sigma} \prod_{i=1}^n x_{i,\sigma(i)},$$

where σ ranges over permutations of $\{1, \dots, n\}$ and $\operatorname{sgn}(\sigma) \in \{1, -1\}$ is the sign of σ . Motivated by similarity of these expressions, Pólya [9] asked whether there exists a simple expression of the permanent in terms of the determinant. This question, which may look like a mere mathematical curiosity, was placed into a deeper context by Valiant. In the seminal paper [10], he defined algebraic analogues of complexity classes P and NP, which we now call as VP and VNP. He showed that the permanent polynomial is complete for the class VNP (if the underlying field is of characteristic different from two). Since the determinant lies in VP, a “simple expression” of perm in terms of det would entail that the two complexity classes coincide.

¹ The author would like to thank Pavel Hrubeš for hosting him at the Institute of Mathematics of ASCR where a part of this work was done.

The precise version of Pólya's problem arising in this context is the following: if $m = m(n)$ is the smallest m so that we can express

$$\text{perm}_n(X) = \det_m(M), \quad (1)$$

where M is a matrix with variables or scalars as entries, can $m(n)$ be bounded by a polynomial in n or does it grow super-polynomially? This question is intimately related to the problem whether $\text{VP} = \text{VNP}$ and is one of the major open problems in algebraic complexity theory. It is generally believed that m grows exponentially with n . However, the strongest lower bound known today – due to Mignon and Ressayre² [7], see also [4] – is quadratic in n . More on the fascinating story of the determinant and permanent can be found in [3, 1].

The problem can be refined in many ways. In this paper, we consider read- k representations. A matrix M whose entries are variables or field elements is *read- k* , if every variable occurs at most k times in M . It will be called a determinantal representation of a polynomial f if $f = \det(M)$. Read- k determinantal representations (or read- k projections of determinant) were defined in [2] where it was shown that for sufficiently large n , perm_n does not have a read-once determinantal representation. Note that in this setting, the question is not about the size of M but merely about its existence. A more general model of rank- k projections was considered in [5]. There it was shown that perm_n cannot be expressed as the determinant of a matrix of the form $A + \sum_{i,j} B_{i,j}x_{i,j}$ with $B_{i,j}$ matrices of rank at most 1.

Continuing this line of research, we will prove that perm_n does not have a read- k determinantal representation for $k \in o(\sqrt{n}/\log n)$. In fact, we will show that any M satisfying (1) must have $\Omega(n^{2.5}/\log n)$ entries containing a variable.

This result is incomparable with the quadratic lower bound on the size $m(n)$ of a determinantal representation. Denoting $s(n)$ the smallest number of entries containing a variable in a determinantal representation of perm_n , the two quantities are related by

$$m(n)/2 \leq s(n) \leq m(n)^2$$

(the first inequality follows from Lemma 1 below, the latter is obvious). This does not allow to deduce our lower bound $s(n) \geq \Omega(n^{2.5}/\log n)$ from the bound $m(n) \geq \Omega(n^2)$ in [7], or vice versa. On the other hand, super-polynomial lower bounds on $s(n)$ and $m(n)$ are *equivalent*.

On a high level, our proof follows ideas of Nechiporuk [8] which were later adapted to the algebraic setting by Kalorkoti [6]. As a technical component, which may be of an independent interest, we identify a specific property differentiating the determinant and the permanent. We will show that every multilinear polynomial f in n variables can be expressed as the permanent of a *read-once* matrix (of an exponential size). This follows by inspecting Valiant's VNP-completeness proof in [10]. An analogous statement is false in the case of the determinant: there exists such an f which requires read- k determinantal representations with k exponential in n . This is proved by a non-constructive counting argument.

Notation and definitions

\mathbb{F} will denote an underlying field. Unless stated otherwise, the field is arbitrary. X will denote a set of variables.

Given a matrix M with entries from $\mathbb{F} \cup X$, its *variable size* is the number of entries containing a variable. It will be called *read- k* if every variable appears at most k times in M . For a polynomial $f \in \mathbb{F}[X]$, M will be called a *determinantal representation of f* if $f = \det(M)$.

As usual, $[n] = \{1, \dots, n\}$.

² In fact, the lower bound holds even if M is allowed to have affine functions as entries.

2 Some properties of determinantal representations

We first show that the size of a determinantal representation can be bounded in terms of its variable size.

► **Lemma 1.** *Let M be a square matrix with entries from $\mathbb{F} \cup X$ of variable size $s \geq 1$. Then there exists a $2s \times 2s$ matrix H of variable size s with entries from $\mathbb{F} \cup X$ such that $\det(H) = \det(M)$. Moreover, each variable occurs in H the same number of times as in M and the variables appear on the main diagonal of H only.*

Proof. The lemma is proved in two steps. In the first step, we transform M to a matrix M^* with the same determinant such that every row and column of M^* contains at most one variable. In the second step, we reduce the dimension of M^* .

Assume that M is an $m \times m$ matrix. For the first step, suppose that M contains a variable x in the (i, j) -th position. Let M' be the $(m+2) \times (m+2)$ matrix

$$M' := \begin{pmatrix} M_0 & e_i & & \\ & 1 & x & \\ e_j^t & 0 & 1 & \end{pmatrix},$$

where M_0 is obtained by setting the (i, j) -th entry to zero in M , e_i is the i -th unit column vector, e_j^t is the j -th unit row vector, and the unspecified entries are zero. Using cofactor expansion on the last column, we obtain $\det(M) = \det(M')$. The number of occurrences of variables has not changed while the displayed variable does not share a row or column with another variable. Repeating this construction s times for each variable in M , we indeed obtain an $(m+2s) \times (m+2s)$ matrix M^* with $f = \det(M^*)$ whose rows and columns contain at most one variable each.

We now proceed with the second step. Permuting rows and columns of M^* , we can write $\det(M) = \pm \det(N)$ with

$$N = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

where A, B, C, D are of dimensions $s \times s$, $s \times (s+m)$, $(s+m) \times s$, $(s+m) \times (s+m)$, respectively, and variables appear on the main diagonal of A only.

Since B has s rows, it has rank at most s . We can also assume that every column of B is a linear combination of its first s columns. Applying suitable column operations to the last $m+s$ columns of N , we can further convert N to the form

$$\begin{pmatrix} A & B_1 & 0 \\ C & D_1 & D_2 \end{pmatrix},$$

with B_1 being an $s \times s$ matrix. D_2 is of dimension $(m+s) \times m$ and hence has rank at most m . Assuming that every row of D_2 is a linear combination of its last m rows, we can apply row-operations to write

$$\det(M) = \pm \det \begin{pmatrix} A & B_1 & 0 \\ C_1 & D'_1 & 0 \\ C_2 & D''_1 & D'_2 \end{pmatrix} = \pm \det(D'_2) \det \begin{pmatrix} A & B_1 \\ C_1 & D'_1 \end{pmatrix},$$

where D'_2 is an $m \times m$ scalar matrix. The matrix $H = \begin{pmatrix} A & B_1 \\ C_1 & D'_1 \end{pmatrix}$ is a $2s \times 2s$ matrix consisting of scalars except for the s variables on the diagonal of A . Since the last column of H contains field elements only, the factor $\pm \det(D'_2)$ can be moved inside H by multiplying the last column. ◀

53:4 On Read- k Projections of the Determinant

This leads to the following non-constructive lower bound on variable size of determinantal representations.

► **Theorem 2.** *For every n , there exists a multilinear polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ such that every determinantal representation of f requires variable size $\Omega(2^{n/2})$.*

Proof. Let s_n be the smallest $s \geq 1$ such that every multilinear polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ has a determinantal representation of variable size s . Lemma 1 implies that every such f can be expressed as $f = \det(C + x_1 D_1 + \dots + x_n D_n)$ where C is a scalar matrix and D_1, \dots, D_n are diagonal matrices in $\mathbb{F}^{2s_n \times 2s_n}$. Viewing entries of C and diagonal entries of D_1, \dots, D_n as parameters, every coefficient of f is a polynomial function of these parameters. Since f has 2^n coefficients and there are $k = (2s_n)^2 + 2s_n n$ parameters, this gives a polynomial map $G: \mathbb{F}^k \rightarrow \mathbb{F}^{2^n}$ whose image contains all of \mathbb{F}^{2^n} . This implies $k \geq 2^n$.

To see this, assume first that \mathbb{F} is finite of size q . Then we must have $q^k \geq q^{2^n}$ and hence $k \geq 2^n$. If \mathbb{F} is infinite and $k < 2^n$, the components G_1, \dots, G_{2^n} of G are algebraically dependent over \mathbb{F} . Then there exists a non-trivial polynomial g with $g(G_1, \dots, G_{2^n}) = 0$ and g therefore vanishes on all of \mathbb{F}^{2^n} . But this is impossible: given a finite subset S of \mathbb{F} of size exceeding the degree of g , Schwartz-Zippel lemma implies that g does not vanish already on S^{2^n} .

Finally, $k \geq 2^n$ gives $s_n \geq (1 - \epsilon)2^{n/2-1}$ for every $\epsilon > 0$ and n sufficiently large. ◀

3 A property of the permanent

We now show that Lemma 1 and Theorem 2 fail when the determinant is replaced with the permanent polynomial. It follows from Valiant's completeness results that every multilinear polynomial in $\mathbb{F}[X]$ can be expressed both as $\text{perm}_m(M)$ and $\det_m(M')$ where M, M' are matrices over $\mathbb{F} \cup X$ of an exponential size. Hence, from the perspective of matrix size, the two polynomials are indistinguishable. However, we show that in the case of the permanent, the matrix M can be assumed to be *read-once*:

► **Theorem 3.** *Let \mathbb{F} be a field of characteristic different from two. Let $f \in \mathbb{F}[x_1, \dots, x_n]$ be a multilinear polynomial. Then there exists $m \leq O(2^n)$ and a matrix M with entries in $\mathbb{F} \cup \{x_1, \dots, x_n\}$ such that $f = \text{perm}_m(M)$ and each variable x_i appears in M exactly once. Moreover, every row and column of M contains at most one variable.*

We outline the proof of Theorem 3 in the rest of this section. It follows by inspection of Valiant's proof of VNP completeness of the permanent. We refer to [11], [3] for a detailed exposition of Valiant's work.

Recall that an *arithmetic formula* over a field \mathbb{F} is a rooted binary tree whose leaves are labelled with variables or field elements and other vertices are labelled with one of the operation $+$ or \times . As the *size* of a formula, we take the number of $+$, \times operations. Every vertex in a formula computes a polynomial in the obvious way.

The following two lemmas are paraphrased versions of Theorem 21.27 and 21.29 from [3].

► **Lemma 4** ([3]). *Let F be an arithmetic formula of size m computing a polynomial $f \in \mathbb{F}[X]$. Then there exists an $(2m+2) \times (2m+2)$ matrix M with entries from $\mathbb{F} \cup X$ such that $f = \text{perm}_{2m+2}(M)$ and every variable occurs in M the same number of times it occurs in F . Moreover, every column and row of M contains at most one variable.*

► **Lemma 5** ([3]). *Let \mathbb{F} be a field of characteristic different from two. Let M be an $m \times m$ matrix with entries from $\mathbb{F} \cup \{x_1, \dots, x_n, y_1, \dots, y_k\}$ having in each row and column at most one variable. Then there exists $m' \leq 10m$ and an $m' \times m'$ matrix M' with entries from*

$\mathbb{F} \cup \{x_1, \dots, x_n\}$ such that $\text{perm}_{m'}(M') = \sum_{y_1, \dots, y_k \in \{0,1\}} \text{perm}_m(M)$ and every variable x_i occurs in M' the same number of times it occurs in M . Moreover, every row and column of M' contains at most one variable.

We also need the following simple fact:

► **Lemma 6.** *Every n -variate multilinear polynomial can be computed by an arithmetic formula of size $O(2^n)$.*

Proof. If f is a multilinear polynomial with $n > 0$ variables, we can write it as

$$f(x_1, \dots, x_n) = x_n f_1(x_1, \dots, x_{n-1}) + f_0(x_1, \dots, x_{n-1}),$$

where f_1, f_0 are multilinear polynomials in $n - 1$ variables. Given formulas for f_0, f_1 of size at most s , we obtain a formula for f of size $\leq 2s + 2$. By induction, this gives a formula of size $O(2^n)$. ◀

Now we are ready to prove Theorem 3.

Proof of Theorem 3. Let X be the set of variables $\{x_1, \dots, x_n\}$. Let $f \in \mathbb{F}[X]$ be a multilinear polynomial

$$f = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i.$$

Introduce new variables $Y = \{y_1, \dots, y_n\}$. Let \hat{f} be the polynomial

$$\hat{f}(y_1, \dots, y_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} y_i \prod_{i \in [n] \setminus S} (1 - y_i).$$

This guarantees that for any boolean substitution $y_1, \dots, y_n \in \{0, 1\}$, $\hat{f}(y_1, \dots, y_n) = a_S$ where $S = \{i \mid y_i = 1\}$. We can therefore write

$$f = \sum_{y_1, \dots, y_n \in \{0,1\}} \hat{f}(y_1, \dots, y_n) \prod_{i=1}^n (x_i y_i + (1 - y_i)).$$

Note that in this expression, each x_i appears exactly once. Let g be the polynomial $\hat{f}(y_1, \dots, y_n) \prod_{i=1}^n (x_i y_i + (1 - y_i))$. As \hat{f} is multilinear, it has a formula of size $O(2^n)$ by Lemma 6. It follows that g has a formula of size $O(2^n)$ in which each variable from X appears exactly once. Lemma 4 gives an $m' \times m'$ matrix M' with $m' \leq O(2^n)$ with entries from $\mathbb{F} \cup X \cup Y$ such that $g = \text{perm}_{m'}(M')$, each variable from X appears exactly once in M' and every row or column of M' contains at most one variable. Since $f = \sum_{y_1, y_2, \dots, y_n \in \{0,1\}} g(X, Y)$ we can apply Lemma 5 to obtain the desired matrix M . ◀

4 Permanent versus determinant

We now prove our main result on variable size of determinantal representations of permanent.

► **Theorem 7.** *Over a field of characteristic different from two, every determinantal representation of perm_n requires variable size $\Omega(n^{5/2} / \log n)$.*

Proof. Let X be the set of variables $\{x_{i,j} \mid 1 \leq i, j \leq n\}$ and let \bar{X} be the $n \times n$ matrix with $x_{i,j}$ in (i, j) -th entry. Assume that $\text{perm}_n(\bar{X}) = \det(M)$ where M is a matrix with entries from $\mathbb{F} \cup X$.

Let $Z \subseteq X$ be a set of k variables $x_{i_1, j_1}, \dots, x_{i_k, j_k}$ where i_1, \dots, i_k are distinct and j_1, \dots, j_k are also distinct. If $k = \lfloor \log_2 n - c \rfloor$, where c is a suitable absolute constant, Theorem 3 implies³ the following:

for every multilinear polynomial $f \in \mathbb{F}[Z]$, there exists a matrix \bar{X}_f obtained by setting variables outside of Z to constants in \bar{X} such that $f = \text{perm}_n(\bar{X}_f)$.

Since $\text{perm}_n(\bar{X}) = \det(M)$, this means that also $f = \det(M_f)$ where M_f is obtained by setting variables outside of Z to constants in M . On the other hand, Theorem 2 shows that there exists a multilinear polynomial in $\mathbb{F}[Z]$ which requires determinantal representation of variable size $\Omega(2^{k/2})$. Hence M must contain $\Omega(2^{k/2})$ entries from Z . Inside X , we can find $t = n \lfloor \frac{n}{k} \rfloor$ such disjoint sets Z_1, \dots, Z_t . For every Z_i , M contains $\Omega(2^{k/2})$ entries from Z_i . Since the sets are disjoint, M contains $\Omega(t2^{k/2})$ entries from X altogether. This gives an $\Omega(n^{5/2}/\log n)$ lower bound on variable size of M . \blacktriangleleft

If perm_n has a read- k determinantal representation M then M has variable size at most n^2k . This implies:

► **Corollary 8.** *Over a field of characteristic different from two, every read- k determinantal representation of perm_n requires $k \geq \Omega(\sqrt{n}/\log n)$.*

5 A harder multilinear polynomial

We now present an explicit multilinear polynomial U_n for which we can prove a quantitatively stronger lower bound than the one presented in Theorem 7. Another improvement is that the lower bound holds over any field. Furthermore, U_n has a polynomial-size arithmetic formula and hence also a polynomial determinantal representation (whereas for perm_n this is not known).

For an integer $n \geq 2$, let $r := \lfloor \log_2 n \rfloor - 1$ and $\ell := \lfloor n/2^r \rfloor$. U_n has variables $x_{i,j}$, $i \in [\ell], j \in [r]$, and y_S , $S \subseteq [r]$, indexed by subsets of $[r]$. The number of variables is therefore $r\ell + 2^r \leq n$. U_n is defined as

$$U_n := y_\emptyset + \sum_{i \in [\ell]} \sum_{\emptyset \neq S \subseteq [r]} y_S \prod_{j \in S} x_{i,j}.$$

► **Theorem 9.** *Over any field, every determinantal representation of U_n requires variable size $\Omega(n^{1.5}/\log n)$.*

Proof sketch. U_n is defined to have the following property: given $i \in [\ell]$ and a multilinear polynomial $f \in \mathbb{F}[x_{i,1}, \dots, x_{i,r}]$ of the form $\sum_{S \subseteq [r]} a_S \prod_{j \in S} x_{i,j}$, we can set $x_{i',j}$ to zero for every $i' \neq i$ and y_S to a_S for every S to obtain the polynomial f from U_n . This is precisely the property we used in the proof of Theorem 7, except that U_n has fewer variables. The same argument as in Theorem 7 gives that every determinantal representation of U_n contains $\Omega(\ell 2^{r/2})$ variables which yields the bound $\Omega(n^{1.5}/\log n)$. \blacktriangleleft

³ Note that perm_n is invariant under permutations of rows and columns.

Let us make some comments:

- (i) Every read- k determinantal representation of U_n requires $k \geq \Omega(\sqrt{n}/\log n)$.
- (ii) On the other hand, U_n has a read- $O(n)$ determinantal representation of variable size $O(n^2)$.

(i) is an immediate consequence of Theorem 9. (ii) follows by, first, observing that U_n has an arithmetic formula in which every variable appears $O(n)$ times and, second, that Lemma 4 holds also when perm_{2m+2} is replaced with det_{2m+2} .

References

- 1 Manindra Agrawal. Determinant versus permanent. *Proceedings of the International Congress of Mathematicians*, 3:985–998, July 2008.
- 2 N. R. Aravind and P. S. Joglekar. On the expressive power of read-once determinants. In *Fundamentals of Computation Theory - 20th International Symposium*, volume 9210 of *Lecture Notes in Computer Science*, pages 95–105. Springer, 2015. doi:10.1007/978-3-319-22177-9_8.
- 3 P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.
- 4 J. Cai, X. Chen, and D. Li. A quadratic lower bound for the permanent and determinant problem over any characteristic $\neq 2$. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 491–498, 2008. doi:10.1145/1374376.1374446.
- 5 Ch. Ikenmeyer and J. M. Landsberg. On the complexity of the permanent in various computational models. *CoRR*, abs/1610.00159, 2016. arXiv:1610.00159.
- 6 K. Kalorkoti. A lower bound for the formula size of rational functions. *SIAM J. Comput.*, 14(3):678–687, 1985. doi:10.1137/0214050.
- 7 T. Mignon and N. Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, pages 4241–4253, 2004.
- 8 E. I. Nechiporuk. On a boolean function. *Soviet Math. Dokl.*, 7(4):999–1000, 1966.
- 9 G. Pólya. Aufgabe 424. *Arch. Math. Phys.*, 20(271), 1913.
- 10 Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 249–261. ACM, 1979. doi:10.1145/800135.804419.
- 11 Joachim von zur Gathen. Feasible arithmetic computations: Valiant’s hypothesis. *J. Symb. Comput.*, 4(2):137–172, 1987. doi:10.1016/S0747-7171(87)80063-9.

Multidimensional Quantum Walks, Recursion, and Quantum Divide & Conquer

Stacey Jeffery  

QuSoft, CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Galina Pass 

QuSoft, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

We introduce an object called a *subspace graph* that formalizes the technique of multidimensional quantum walks. Composing subspace graphs allows one to seamlessly combine quantum and classical reasoning, keeping a classical structure in mind, while abstracting quantum parts into subgraphs with simple boundaries as needed. As an example, we show how to combine a *switching network* with arbitrary quantum subroutines, to compute a composed function. As another application, we give a time-efficient implementation of quantum Divide & Conquer when the sub-problems are combined via a Boolean formula. We use this to quadratically speed up Savitch’s algorithm for directed *st*-connectivity.

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

Keywords and phrases Quantum Divide & Conquer, Time-Efficient, Subspace Graphs, Quantum Walks, Switching Networks, Directed *st*-Connectivity

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.54

Related Version *Full Version*: <https://arxiv.org/abs/2401.08355>

Funding *Stacey Jeffery*: Supported by ERC STG grant 101040624-ASC-Q, NWO Klein project number OCENW.Klein.061. SJ is a CIFAR Fellow in the Quantum Information Science Program. *Galina Pass*: Supported by the National Agenda for Quantum Technologies (NAQT), as part of the Quantum Delta NL programme.

1 Introduction

There are a number of graphical ways of reasoning about how the steps or subroutines of a classical algorithm fit together. For example, it is natural to think of a (randomized) classical algorithm as a (randomized) decision tree (or branching program), where different paths are chosen depending on the input, as well as random choices made by the algorithm. A deterministic algorithm gives rise to a computation *path*, a randomized algorithm to a computation *tree*. The edges of a path or tree, representing steps of computation, might be implemented by some subroutine that is also realized by a path (or tree) – we can abstract the subroutine’s details by viewing it as an edge, or zoom in and see those details, as convenient. More generally, we often think of a classical randomized algorithm as a random walk on a (possibly directed) graph, where there may be multiple parallel paths from point *a* to point *b*, with the cost of getting from *a* to *b* being derived from the expected length of these paths.

This picture appears to break down for quantum algorithms, at least in the standard circuit model. A quantum circuit can be thought of as a path, with edges representing its steps, but it is unclear how to augment this reasoning with subroutines. Consider calling subroutines with varying time complexities $\{T_i\}_i$ in superposition. Even if the subroutines



© Stacey Jeffery and Galina Pass;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 54; pp. 54:1–54:16



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



are all classical deterministic, in the standard quantum circuit model, we tend to incur a cost of $\max_i T_i$ if we call a superposition of subroutines, since we must wait for the slowest subroutine to finish before we can apply the next step of the computation. This problem was addressed in [13], where the technique of multidimensional quantum walks [15] was used to show how to get an average in place of a max in several settings where a quantum algorithm calls subroutines in superposition: a general setting, as well as the setting of quantum walks. The intuition behind [13] is that a quantum walk does keep the classical intuition of parallel paths representing a superposition of possible computations, and *any* quantum algorithm can be viewed as some sort of a quantum walk on a simple underlying graph (something like a path), but with some additional structure associated with it.

Multidimensional quantum walks, which we study more formally in this paper as an object than has been done previously, are valuable as a way of combining quantum and classical reasoning. A quantum algorithm can be abstracted as a graph with perhaps complicated internal structure, but a simple *boundary* with an “in” and an “out” terminal (called “*s*” and “*t*”), that can be seamlessly hooked into other graph-like structures, perhaps representing simple classical reasoning, such as a quantum random walk, or perhaps with their own complicated very quantum parts.

Subspace Graphs

While [15] and [13] use similar techniques, it is not formally defined what a multidimensional quantum walk is. We formally define an object called a *subspace graph* (Definition 2) that abstracts the structures in [15] and [13]. A subspace graph is simply a graph with some subspaces associated with each edge and vertex, where the structure of the graph constrains how the spaces can overlap. Defining what we mean, precisely, by a multidimensional quantum walk (i.e. subspace graph) is the first step to developing a general theory of recursive constructions of subspace graphs.

The recursive structure of subspace graphs is useful for composing quantum algorithms, but as a design tool, it is also convenient to be able to view a subspace graph in varying levels of abstraction. We can “zoom out” and view a complicated process as just a special “edge”, or zoom in on that edge and understand its structure as an involved graph with additional structure.

We cannot hope to be able to understand all quantum algorithms using purely classical ideas – quantum computing is *not* classical computing. But perhaps the next best thing is a way to seamlessly combine classical and quantum ideas, extending the classical intuition to its limits, and then employing quantum reasoning when needed, but with the possibility of abstracting out from it when needed as well, using a fully quantum form of abstraction.

In this work, we consider one specific kind of composition of subspace graphs called *switch composition* – another type is implicit in [13] – but we would like to emphasize the potential for more general types of recursion, which we leave for future work.

Time-Efficient Quantum Divide & Conquer

A particular type of recursive algorithm is *divide & conquer*, in which a problem is broken into multiple smaller sub-problems, whose solutions, obtained by recursive calls, are combined into a solution for the original problem. As a motivating example, consider the recursively defined *nand-tree function*. Let $f_{k,d} : \{0, 1\}^{d^k} \rightarrow \{0, 1\}$ be defined $f_{0,d}(x) = x$, and for $k \geq 1$,

$$f_{k,d}(x) = \text{NAND} \left(f_{k-1,d}(x^{(1)}), \dots, f_{k-1,d}(x^{(d)}) \right) := 1 - f_{k-1,d}(x^{(1)}) \dots f_{k-1,d}(x^{(d)}),$$

where each $x^{(j)} \in \{0, 1\}^{d^{k-1}}$, and $x = (x^{(1)}, \dots, x^{(d)})$. There is a natural way to break an instance x of $f_{k,d}$ into d sub-problems $x^{(1)}, \dots, x^{(d)}$ of $f_{k-1,d}$, and combine the solutions by taking the NAND (negated AND) of the d sub-problem solutions. Grover's algorithm computes this NAND in $O(\sqrt{d})$ queries, so we might hope for a speedup by recursive calls to this quantum algorithm. Unfortunately, since we recurse to depth k , the constant in front of \sqrt{d} is raised to the k -th power. This kills the quantum speedup completely when d is constant (for example, the most common setting of $d = 2$), and that is not even touching on the fact that we would seem to need to amplify the success probability of the subroutine, turning those constants into log factors. On the other hand, it is known [20] that $f_{k,d}$ can be evaluated in $O(\sqrt{d^k})$ quantum queries, even though our attempt to use classical divide-&-conquer reasoning combined with the basic Grover speedup failed.

More recently, [9] showed how to employ divide-&-conquer reasoning in the study of quantum query complexity, in which one only counts the number of queries to the input. They obtained their query upper bounds by composing *dual adversary solutions*. The key to their results is that dual adversary solutions exhibit *perfect* composition: no error, no log factors, not even constant overhead. However, their result were not constructive, as dual adversary solutions do not fully specify algorithms, and in particular, the time complexity analysis of their results was unknown. In this work, we use the framework of subspace graphs to give a time-complexity version of some of the query complexity results obtained in [9]. In particular, we show (see Theorem 16):

► **Theorem 1 (Informal).** *Let $\{f_{\ell,n} : D_{\ell,n} \rightarrow \{0, 1\}\}_{\ell,n}$ be a family of functions. Let φ be a symmetric Boolean formula on a variables, and suppose $f_{\ell,n} = \varphi(f_{\ell/b,n}, \dots, f_{\ell/b,n}) \vee f_{\text{aux},\ell,n}$, for some $b > 1$ and some auxiliary function $f_{\text{aux},\ell,n}$ with quantum time complexity $T_{\text{aux}}(\ell, n)$. Then the quantum time complexity of $f_{\ell,n}$ is $\tilde{O}(T(\ell, n))$ for $T(\ell, n)$ satisfying:*

$$T(\ell, n) \leq \sqrt{a}T(\ell/b, n) + T_{\text{aux}}(\ell, n).$$

Our framework also handles the case where $f_{\ell,n} = \varphi(f_{\ell/b,n}, \dots, f_{\ell/b,n}, f_{\text{aux},\ell,n})$ for any formula φ on $a + 1$ variables, but then some extra, somewhat complicated looking costs need to be accounted for, and there is also a scaling in the depth, although this is not an issue if the formula has been preprocessed to be balanced [8].

Comparing this with the analogous classical statement, which would have a instead of \sqrt{a} , we get an up to quadratic speedup over a large class of classical divide-&-conquer algorithms. As an application, we show a quadratic speedup of Savitch's divide-&-conquer algorithm for directed *st*-connectivity [22].

To achieve these results, it is essential that we compose subspace graphs, rather than algorithms. When we convert a subspace graph to a quantum algorithm, we get constant factors in the complexity, and these seem necessary without at least specifying what gateset we are working in. By first composing in the more abstract model of subspace graphs, and then only converting to a quantum algorithm at the end, we ensure these factors only come into the complexity once. This is similar to compositions done with other abstract models, such as span programs (of which dual adversary solutions are a special case) [19], and transducers [6].

Switching Networks

A switching network is a graph with Boolean variables associated with the edges that can switch the edges on or off. Originally used to model certain hardware systems, including automatic telephone exchanges, and industrial control equipment [23], a switching network

has an associated function f that is 1 if and only if two special vertices, s and t , are connected by a path of “on” edges. Shannon [23, 24] showed that series-parallel switching networks are equivalent to Boolean formulas, and Lee [16] showed that switching networks can model branching programs. These theoretical results have given this model a place in classical complexity theory, where they can be used to study classical space complexity and circuit depth (see [18]).

Quantum algorithms for evaluating switching networks¹ given query access to the edge variables were given in [14], using a span program construction based heavily on [7]. Let $\mathcal{R}_{s,t}(G(x))$ be the effective resistance in the subgraph of “on” edges whenever $f(x) = 1$, and whenever $f(x) = 0$, let $F_x \subseteq E$ be the minimum weight st -cut-set consisting of only edges that are “off”. Then there is a quantum algorithm evaluating the switching network using

$$\sqrt{\max_{x \in f^{-1}(1)} \mathcal{R}_{s,t}(G(x)) \max_{x \in f^{-1}(0)} \sum_{e \in F_x} w_e},$$

queries, with matching time complexity assuming we can implement a certain reflection related to the particular switching network in unit time. From this it follows that if we can query the variable associated with an edge e in time T_e , we can evaluate the switching network in time

$$\sqrt{\max_{x \in f^{-1}(1)} \mathcal{R}_{s,t}(G(x)) \max_{x \in f^{-1}(0)} \sum_{e \in F_x} w_e \cdot \max_{e \in E} T_e}.$$

Here we improve this to:

$$\sqrt{\max_{x \in f^{-1}(1)} \mathcal{R}_{s,t}(G(x)) \max_{x \in f^{-1}(0)} \sum_{e \in F_x} w_e T_e^2}.$$

This is analogous to results of [13], which showed a similar statement, but for quantum walks rather than switching networks². Because switching networks perfectly model Boolean formulas, without the constant overhead we get when we convert to a quantum algorithm, they can be used as a building block for our divide-&-conquer results.

Application to DSTCON

Quantum algorithms for st -connectivity on *undirected graphs*, which are closely related to evaluating switching networks, are well studied [10, 7, 14, 12, 3], including quantum algorithms that achieve optimal time- and space-complexity simultaneously, in both the edge-list access model, and the adjacency matrix access model.³ In contrast, quantum algorithms for *directed* st -connectivity (DSTCON) – the problem of deciding if there is a directed path from s to t in a directed graph – is less well understood. The algorithm of [10] also applies to directed graphs, deciding connectivity in $\tilde{O}(n)$ time and space⁴. This algorithm has optimal time complexity, whereas its space complexity is far from optimal.

¹ Switching networks have never been directly referred to in prior work on quantum algorithms, as far as we are aware, but the “ st -connectivity problems” referred to in [14] are, in fact, switching networks.

² Both our result, and the one for quantum walks in [13] include *variable-time quantum search* [2] as a special case.

³ We do not make a distinction between various access models, because they can simulate one another in $\text{poly}(n)$ time and $\log(n)$ space, so our result, which includes a $2^{O(\log n)}$ term, is the same in all models.

⁴ In the edge-list access model.

Directed st -connectivity, also called *reachability*, is a fundamental problem in classical space complexity. In particular, understanding if this problem can be solved in $\log(n)$ space by a quantum algorithm would resolve the relationship between quantum logspace complexity and NL, as DSTCON is NL-complete.

The best known classical (deterministic) space complexity of DSTCON is $O(\log^2(n))$, using Savitch's algorithm. We apply quantum divide & conquer (Theorem 1) to give a quadratic speedup to Savitch's algorithm, achieving $2^{\frac{1}{2}\log^2(n)+O(\log(n))}$ time, while still maintaining $O(\log^2(n))$ space (see Theorem 18).

Model of Computation

We work in the same model as [15], where we allow not only arbitrary quantum gates, but also assume subroutines are given via access to a unitary that applies the subroutine's t -th gate controlled on the value t in some time register. This is possible, for example, with quantum random access gates.

Related Work

While writing this manuscript, we became aware of an independent work that also achieves time-efficient quantum divide & conquer [1] when either (1) φ is an OR (equivalently, an AND) or (2) φ is a minimum or maximum. OR is a Boolean formula, while minimum/maximum is not. In that sense, our results are incomparable. Ref. [1] applies their framework to problems that are distinct from ours. The framework of [1] also differs from our work in that they explicitly treat the complexity of computing sub-instances (the cost of the “create” step), whereas we assume sub-instances are computable in unit cost. In one of the applications of [1], this cost is not negligible, and is even the dominating cost, so our framework, as stated, would not handle this application. This is not an inherent limitation of our techniques – it would be possible to take this cost into account in our framework as well.

We also mention that Ref. [9] analyzes the quantum query complexity of divide & conquer where φ is an arbitrary Boolean formula, as well as in settings where the function combining the sub-problems is more general. While our techniques do apply to composing quantum algorithms for arbitrary functions (already studied in [13]), an issue is a poor scaling in the error of subroutines. If we start with a bounded-error quantum algorithm for some function, we need to amplify the success probability, as it will be called many times, incurring logarithmic factors. This becomes a serious problem if the function is called recursively to depth more than constant. We get around this in the case of Boolean formulas by using a switching network construction (from which a quantum algorithm could be derived) rather than a bounded-error quantum algorithm for evaluating the formula. Our techniques would thus also readily apply to functions for which there is an efficient quantum algorithm derived from a switching network.

2 Technical Overview

Here we give a technical overview of our results, with full details available in the full version of the paper.

2.1 Subspace Graphs

Multidimensional quantum walks were introduced as such in [15], although they generalize various quantum algorithms that have appeared previously, including [25, 21, 4, 5, 11]. We wish to consider very general kinds of composition of multidimensional quantum walks, and in order to be clear about the precise types of objects we are composing, we give a more formal definition than has appeared previously.

► **Definition 2** (Subspace Graph). A subspace graph consists of a (undirected) graph $G = (V, E)$, a boundary $B \subseteq V$, and the following subspaces of a space $H = H_G$:

Edge and Boundary Spaces We assume H can be decomposed into a direct sum of spaces as follows: $H = \bigoplus_{e \in E} \Xi_e \oplus \bigoplus_{u \in B} \Xi_u$.

Edge and Boundary Subspaces For each $e \in E \cup B$, let Ξ_e^A and Ξ_e^B be subspaces of Ξ_e . These need not be orthogonal, and they may each be $\{0\}$, all of Ξ_e , or something in between.

Vertex Spaces and Boundary Space For each $u \in V$, let $\mathcal{V}_u \subseteq \bigoplus_{e \in E(u)} \Xi_e$ be pairwise orthogonal spaces. Let $\mathcal{V}_B \subseteq \bigoplus_{u \in B} \Xi_u$.

Then we define

$$\mathcal{A}_G = \bigoplus_{e \in E \cup B} \Xi_e^A \text{ and } \mathcal{B}_G = \bigoplus_{u \in V} \mathcal{V}_u + \mathcal{V}_B + \bigoplus_{e \in E \cup B} \Xi_e^B.$$

To motivate this perhaps complicated-looking definition, we mention some concrete examples of subspace graphs that already exist in the quantum algorithms literature:

1. A discrete-time quantum walk in Szegedy's framework [25], or the more general electric network framework [4] is a subspace graph.
2. The span programs for st -connectivity in [14], based on [7], are subspace graphs. These are actually precisely switching networks, which we will discuss more shortly (although the basis constructions for Boolean switching networks in this work are new).
3. In [13], certain subspaces are defined from any quantum algorithm, and these actually make up a subspace graph (see Section 2.2.3).

We will allow subspace graphs to implicitly depend on some input, and associate them with a computation. Specifically, if we fix some initial state $|\psi_0\rangle \in H_G$, then this, along with \mathcal{A}_G and \mathcal{B}_G , define a phase estimation algorithm that decides if $|\psi_0\rangle \in \mathcal{A}_G + \mathcal{B}_G$ by doing phase estimation of $(2\Pi_{\mathcal{A}_G} - I)(2\Pi_{\mathcal{B}_G} - I)$, where $\Pi_{\mathcal{A}_G}$ is the orthogonal projector onto \mathcal{A}_G , and similarly for \mathcal{B}_G . With this in mind, we say a subspace graph that depends on some input x , *computes a function* f (with respect to $|\psi_0\rangle$) if $f(x) = 0$ if and only if $|\psi_0\rangle \in \mathcal{A}_G + \mathcal{B}_G$. Let us give some intuition. The quantum algorithm corresponding to a subspace graph described above distinguishes between two cases:

Negative Case: $|\psi_0\rangle \in \mathcal{A}_G + \mathcal{B}_G$

Positive Case: $|\psi_0\rangle$ has a (large) component, called a *positive witness* in $(\mathcal{A}_G + \mathcal{B}_G)^\perp = \mathcal{A}_G^\perp \cap \mathcal{B}_G^\perp$

Then we can see all the vectors in \mathcal{A}_G and \mathcal{B}_G as linear constraints on the initial state – it must have a large component orthogonal to all of them in the positive case. \mathcal{A}_G and \mathcal{B}_G are each decomposed into sums of subspaces, representing subsets of constraints. The graph structure of G restricts how the different sets of constraints are allowed to overlap – the constraints associated with vertex v only overlap constraints associated with edges that are incident to v . This graph structure can then be used to help analyse the algorithm. For example, a positive witness is related to a *flow* on G .

Though it is possible to consider more general states $|\psi_0\rangle$, throughout this paper, we will assume G has a pair of special “terminal” vertices s and t , and let $|\psi_0\rangle = \frac{1}{\sqrt{2}}(|s\rangle - |t\rangle)$.

To implement the phase estimation algorithm, we need a pair of orthogonal bases for \mathcal{A}_G and \mathcal{B}_G that are easily generated, in order to implement their respective reflections. We therefore always assume we have such a basis pair associated with G , which we call the *working bases*.

To analyze a phase estimation algorithm, we need to exhibit positive and negative witnesses, which have a particular form in the case G has a property called *canonical st -boundary* (see Definition 8), as will always be the case in this paper. In particular, *st*-boundary implies that $\bigoplus_{u \in B} \Xi_u = \Xi_s \oplus \Xi_t$ has four degrees of freedom, which we denote $|s\rangle, |\leftarrow, s\rangle, |t\rangle, |\rightarrow, t\rangle$.

► **Definition 3** (Positive Witness for a Graph). *We say $|\hat{w}\rangle \in \Xi_E$ is a positive witness for G if*

$$|s\rangle - |\leftarrow, s\rangle + |\hat{w}\rangle + |\rightarrow, t\rangle - |t\rangle \in \mathcal{A}_G^\perp \cap \mathcal{B}_G^\perp.$$

We let $\hat{W}_+(G)$ be an upper bound (over some implicit input) on the minimum $\| |\hat{w}\rangle \|^2$ of any positive witness for G .

A negative witness for a phase estimation algorithm is a vector $|w_{\mathcal{A}}\rangle \in \mathcal{A}$ such that $|w_{\mathcal{B}}\rangle := |\psi_0\rangle - |w_{\mathcal{A}}\rangle \in \mathcal{B}$, which exists if and only if $|\psi_0\rangle \in \mathcal{A}_G + \mathcal{B}_G$. For a subspace graph, a negative witness is defined as follows.

► **Definition 4** (Negative Witness for a Graph). *We say $|\hat{w}_{\mathcal{A}}\rangle \in \mathcal{A}_G$ is a negative witness for G if $|\hat{w}_{\mathcal{A}}\rangle + |\leftarrow, s\rangle - |\rightarrow, t\rangle \in \mathcal{B}_G$. We let $\hat{W}_-(G)$ be an upper bound (over some implicit input) on the minimum $\| |\hat{w}_{\mathcal{A}}\rangle \|^2$ of any negative witness for G .*

This leads to the following theorem associating a quantum algorithm to a subspace graph.

► **Theorem 5.** *Let G be a subspace graph that computes $f : \{0, 1\}^n \rightarrow \{0, 1\}$, with working bases that can be generated in time T . Then there exists a quantum algorithm that decides f in time complexity $O\left(T\sqrt{\hat{W}_+(G)\hat{W}_-(G)}\right)$ and space $O(\log \dim H_G + \log(\hat{W}_+(G)\hat{W}_-(G)))$.*

This justifies defining $\hat{C}(G) := \sqrt{\hat{W}_+(G)\hat{W}_-(G)}$ as the *complexity* of a subspace graph.

Switches and Switching Networks

Next, we introduce switching networks. In the classical model of switching networks, a switching network is a graph with a literal (variable or negated variable) $\varphi(e)$ associated with each edge e , and two terminals $s, t \in V$. A switching network computes a function $f : \{0, 1\}^E \rightarrow \{0, 1\}$ if for any $x \in \{0, 1\}^E$, $f(x) = 1$ if and only if s and t are connected in $G(x)$, the subgraph consisting only of edges e such that $x_e = 1$ if $\varphi(e)$ is a positive literal, and $x_e = 0$ otherwise – that is, edges labelled by literals that are true when the variables are set according to x . Here, we let “switching network” refer to a special kind of subspace graph, but this subspace graph implements the switching network, in the sense that the algorithm referred to in Theorem 5 decides if s and t are connected in $G(x)$.

We first define what it means for an edge to be a switch. For a vertex $u \in V$, we let $E(u)$ denote the edges incident to u , which we divide into $E^\rightarrow(u)$ – edges “coming out of” u – and $E^\leftarrow(u)$ – edges “going into” u . As G is an undirected graph, these edge directions can be chosen arbitrarily.

► **Definition 6** (Switch Edge). *Fix a subspace graph G . We call an edge $e \in E$ a switch (or switch edge) if there is some value $\varphi(e) \in \{0, 1\}$ associated with that edge (implicitly depending on the input), such that*

$$\begin{aligned} \Xi_e &= \text{span}\{|\rightarrow, e\rangle, |\leftarrow, e\rangle\}, \\ \Xi_e^{\mathcal{A}} &= \text{span}\{|\rightarrow, e\rangle - (-1)^{\varphi(e)}|\leftarrow, e\rangle\} \quad \text{and} \quad \Xi_e^{\mathcal{B}} = \text{span}\{|\rightarrow, e\rangle + |\leftarrow, e\rangle\}, \end{aligned}$$

and moreover, if $e \in E^\rightarrow(u) \cap E^\leftarrow(v)$, (that is, $e = (u, v)$), then $\mathcal{V}_u \cap \Xi_e = \text{span}\{|\rightarrow, e\rangle\}$ and $\mathcal{V}_v \cap \Xi_e = \text{span}\{|\leftarrow, e\rangle\}$.

The idea behind a switch edge is that if $\varphi(e) = 0$, $\Xi_e^A + \Xi_e^B = \Xi_e$, and so $\Xi_e^\perp = \{0\}$. Recall that a *positive witness* is some $|\hat{w}\rangle$ such that $|w\rangle := |s\rangle - |\leftarrow, s\rangle + |\hat{w}\rangle + |\rightarrow, t\rangle - |t\rangle \in \mathcal{A}_G^\perp \cap \mathcal{B}_G^\perp$. If $\Xi_e^\perp = \{0\}$, then $|w\rangle$ can have no overlap with Ξ_e , so e is essentially blocked from use, so we say the edge is *switched off*. In switching networks, defined shortly, the positive witness is an *st-flow*, and it is restricted to edges in the subgraph $G(x)$ of edges that are switched on. Even in subspace graphs that are not switching networks, we can often think of $|w\rangle$ intuitively as a kind of flow.

We next define *simple vertices*, which are the type of vertices of switching networks, and also quantum walks.

► **Definition 7 (Simple Vertex).** Fix a subspace graph G with associated edge weights $\{w_e\}_{e \in E}$. For $u \in V$, define

$$|\psi_\star(u)\rangle := \sum_{e \in E^\rightarrow(u)} \sqrt{w_e} |\rightarrow, e\rangle + \sum_{e \in E^\leftarrow(u)} \sqrt{w_e} |\leftarrow, e\rangle.$$

A vertex $u \in V$ is *simple* if for all $e \in E^\rightarrow(u)$, $|\rightarrow, e\rangle \in \Xi_e$, for all $e \in E^\leftarrow(u)$, $|\leftarrow, e\rangle \in \Xi_e$, and if $u \in V \setminus B$ $\mathcal{V}_u = \text{span}\{|\psi_\star(u)\rangle\}$; and if $u \in B$, either $|\rightarrow, u\rangle \in \Xi_u$ and $\mathcal{V}_u = \text{span}\{|\rightarrow, u\rangle + |\psi_\star(u)\rangle\}$ or $|\leftarrow, u\rangle \in \Xi_u$ and $\mathcal{V}_u = \text{span}\{|\leftarrow, u\rangle + |\psi_\star(u)\rangle\}$.

Let us motivate this definition. In a random walk on a weighted graph, in order to take a step from a vertex $u \in V$ to a neighbour, the random walker chooses an edge $e \in E(u)$ to traverse, with probability $w_e / (\sum_{e' \in E(u)} w_{e'})$. This is precisely the distribution obtained by measuring $|\psi_\star(u)\rangle / \|\psi_\star(u)\|$. The states $|\psi_\star(u)\rangle$ correspond to *quantum walk states* – alternating a reflection around these with a reflection around the states $\{|\rightarrow, e\rangle + |\leftarrow, e\rangle\}_{e \in E}$ implements a discrete-time quantum walk, as in [25, 17, 4].⁵

Another important notion is that of *canonical st-boundary* for which we require the boundary to consist only of two vertices s and t with additional requirements on their subspaces.

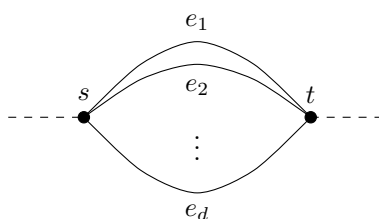
► **Definition 8 (Canonical st-boundary).** We say a subspace graph G has *canonical st-boundary* if $B = \{s, t\}$, where:

- $\Xi_s = \text{span}\{|s\rangle, |\leftarrow, s\rangle\}$, $\Xi_s^A = \text{span}\{|s\rangle + |\leftarrow, s\rangle\}$ and $\Xi_s^B = \{0\}$, where \mathcal{V}_s only overlaps $|\leftarrow, s\rangle$.
- $\Xi_t = \text{span}\{|\rightarrow, t\rangle, |t\rangle\}$, $\Xi_t^A = \text{span}\{|\rightarrow, t\rangle + |t\rangle\}$ and $\Xi_t^B = \{0\}$, where \mathcal{V}_t only overlaps $|\rightarrow, t\rangle$.
- $|\leftarrow, s\rangle + |\rightarrow, t\rangle \in \mathcal{B}_G$.

► **Definition 9 (Switching Network).** A switching network is a subspace graph with *canonical st-boundary* in which all edges are *switches* (Definition 6), and all vertices are *simple* (Definition 7).

Two more important definitions for the composition results are those of *composable bases* and *st-composable subspace graph* (Definition 10). The first one specifies conditions on working bases so that they can be smoothly composed with each other. We leave the rigorous definition to the full version of the paper since it is quite technical.

⁵ In some previous works, it is assumed that the graph is bipartite, and then the walk is implemented by alternating reflections around the states $|\psi_\star(u)\rangle$ of the two parts of the bipartition. We are actually doing the same thing here, we have just ensured the graph is bipartite by inserting a vertex in the middle of each edge.



■ **Figure 1** The graph G_{OR} . The dashed lines represent dangling boundary “edges”.

► **Definition 10.** We say a subspace graph G is st -composable if:

1. G has canonical st -boundary (Definition 8);
2. G is equipped with composable working bases;
3. for each $e \in E \setminus \bar{E}$, $\Xi_e^B = \{0\}$, where \bar{E} is the set of edges that are switches.

An st -composable subspace graph is a generalization of a switching network to allow for more complicated structures, such as arbitrary quantum algorithms. The main technical contribution of this paper (the composition theorem in Section 2.3) is to generalize a natural way that switching networks compose, as graphs, to any st -composable subspace graph.

2.2 Examples

We first give two examples of switching networks – one for computing the OR of d bits (Section 2.2.1) and one for the AND of d bits (Section 2.2.2) – which are also important building blocks for our later results. At a high level, these are quite simple to understand. The switching network for OR is just d parallel edges from s to t (see Figure 1) – s and t are connected if at least one of the edges is present. The switching network for AND is just a path of length d from s to t (see Figure 2) – s and t are connected if all of the edges are present. One can build up a graph that represents any formula by compositions where an edge is replaced by a graph whose terminals s and t are identified with the endpoints of the replaced edge (see also Figure 4). Our main composition theorem (see Section 2.3) generalizes this type of composition to apply to any st -composable subspace graph.

2.2.1 Switching network for OR

A switching network that computes the OR of d Boolean variables consists of two vertices connected by d parallel switch edges. The idea is that there is no flow if all the edges are blocked, that is all the variables take value 0, and there is a flow otherwise. Therefore, a negative witness corresponds to a cut in the graph, and a positive witness corresponds to a flow.

► **Lemma 11.** For any $d \geq 1$, and positive weights $\{w_i\}_{i \in [d]}$, there is a switching network $G_{\text{OR},d}$ that computes $\bigvee_{i=1}^d \varphi(e_i)$ with $\dim H_{G_{\text{OR},d}} = 2d + 4$ such that:

1. $G_{\text{OR},d}$ has st -composable working bases that can be generated in $O(\log d)$ time, assuming the state proportional to $\sum_{i=1}^d \sqrt{w_i} |i\rangle$ can be generated in time $O(\log d)$;
2. if $\varphi(e_i) = 1$ for some $i \in [d]$, $G_{\text{OR},d}$ has positive witness $|\hat{w}\rangle = \frac{1}{\sqrt{w_i}}(|\rightarrow, i\rangle - |\leftarrow, i\rangle)$; and
3. if $\varphi(e_i) = 0$ for all $i \in [d]$, $G_{\text{OR},d}$ has negative witness $|\hat{w}_A\rangle = \sum_{i=1}^d \sqrt{w_i}(|\rightarrow, i\rangle - |\leftarrow, i\rangle)$.

We remark that if $w_i = 1$ for all i , then Lemma 11 implies $\hat{W}_+(G_{\text{OR},d}) = 2$ and $\hat{W}_-(G_{\text{OR},d}) = 2d$, so by Theorem 5, there is a quantum algorithm for evaluating d -bit OR with time complexity $\tilde{O}(\sqrt{d})$, which is optimal. This is a good sanity check, but we will mostly be interested in this construction as a building block, rather than in its own right.

Let $G = G_{\text{OR},d}$ be defined, as shown in Figure 1 by:

$$V = \{s, t\} \text{ and } E = \{e_i : i \in [d]\}$$

where each e_i has endpoints s and t , so $E(s) = E^\rightarrow(s) = E$ and $E(t) = E^\leftarrow(t) = E$. Since G is a switching network, it has boundary $B = V = \{s, t\}$. We will let the graph be weighted, with weights $w_{e_i} = w_i$. The only reason to let these vary in i is for later when we replace an edge with a gadget by composition, but for the sake of intuition, the reader may wish to imagine $w_i = 1$ for all i . Since G is a switching network, every edge is a switch, which fixes the following spaces (we simplify notation by using i to label the edge e_i):

$$\begin{aligned} \forall i \in [d], \Xi_{e_i} &= \text{span}\{|\rightarrow, i\rangle, |\leftarrow, i\rangle\}, \\ \Xi_{e_i}^A &= \text{span}\{|\rightarrow, i\rangle - (-1)^{\varphi(e_i)}|\leftarrow, i\rangle\}, \text{ and } \Xi_{e_i}^B = \text{span}\{|\rightarrow, i\rangle + |\leftarrow, i\rangle\}. \end{aligned}$$

Furthermore, as a switching network has canonical st -boundary, the following spaces are fixed:

$$\begin{aligned} \Xi_s &= \text{span}\{|s\rangle, |\leftarrow, s\rangle\}, \quad \Xi_s^A = \text{span}\{|s\rangle + |\leftarrow, s\rangle\}, \text{ and } \Xi_s^B = \{0\} \\ \Xi_t &= \text{span}\{|\rightarrow, t\rangle, |t\rangle\}, \quad \Xi_t^A = \text{span}\{|\rightarrow, t\rangle + |t\rangle\}, \text{ and } \Xi_t^B = \{0\}. \end{aligned}$$

Finally, since all vertices are simple, the following spaces are fixed:

$$\mathcal{V}_s = \text{span}\left\{|\leftarrow, s\rangle + \underbrace{\sum_{i=1}^d \sqrt{w_i} |\rightarrow, i\rangle}_{|\psi_\star(s)\rangle}\right\} \text{ and } \mathcal{V}_t = \text{span}\left\{\underbrace{\sum_{i=1}^d \sqrt{w_i} |\leftarrow, i\rangle}_{|\psi_\star(t)\rangle} + |\rightarrow, t\rangle\right\}.$$

Then we have:

$$\begin{aligned} \mathcal{A}_G &= \bigoplus_{e \in E \cup B} \Xi_e^A = \text{span}\{|\rightarrow, i\rangle - (-1)^{\varphi(e_i)}|\leftarrow, i\rangle : i \in [d]\} \cup \{|s\rangle + |\leftarrow, s\rangle, |t\rangle + |\rightarrow, t\rangle\} \\ \text{and } \mathcal{B}_G &= \mathcal{V}_s \oplus \mathcal{V}_t + \bigoplus_{e \in E} \Xi_e^B = \mathcal{V}_s \oplus \mathcal{V}_t + \text{span}\{|\rightarrow, i\rangle + |\leftarrow, i\rangle : i \in [d]\}. \end{aligned} \quad (1)$$

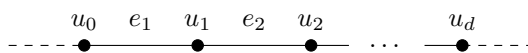
We prove the second and the third items of Lemma 11 in the full version of the paper. It remains to find working bases that can be generated efficiently. We show in the full version that there is an st -composable working basis for \mathcal{A}_G that can be generated in unit time and there is a st -composable working basis for \mathcal{B}_G that can be generated in $O(\log d)$ time.

2.2.2 Switching network for AND

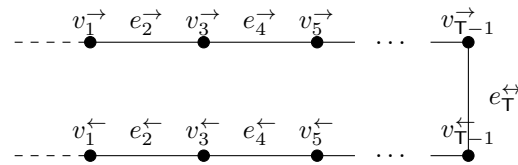
In this section, we describe a switching network that computes the AND of d Boolean variables. This switching network consists of two boundary vertices connected by a path of switch-edges of length d . If at least one edge is blocked, that is at least one of the variables takes value 0, then there is no flow, and there is a flow otherwise.

► **Lemma 12.** *For any $d \geq 1$, and positive weights $\{w_i\}_{i \in [d]}$, there is a switching network $G_{\text{AND},d}$ that computes $\bigwedge_{i=1}^d \varphi(e_i)$ with $\dim H_{G_{\text{AND},d}} = 2d + 4$ such that:*

1. $G_{\text{AND},d}$ has st -composable working bases that can be generated in $O(\log d)$ time, assuming the state proportional to $\sum_{i=1}^d \frac{1}{\sqrt{w_i}} |i\rangle$ can be generated in time $O(\log d)$;
2. if $\varphi(e_i) = 1$ for all $i \in [d]$, $G_{\text{AND},d}$ has positive witness $|\hat{w}\rangle = \sum_{i=1}^d \frac{1}{\sqrt{w_i}} (|\rightarrow, i\rangle - |\leftarrow, i\rangle)$; and



■ **Figure 2** The graph G_{AND} . The dashed lines represent dangling boundary “edges”.



■ **Figure 3** The graph representing a quantum computation with T steps. The top part should be thought of as computing a bit into the phase, and the bottom should be thought of as uncomputing. The dashed lines represent dangling boundary “edges”.

3. if $\varphi(e_i) = 0$ for some $i \in [d]$, $G_{\text{AND},d}$ has negative witness $|\hat{w}_{\mathcal{A}}\rangle = \sqrt{w_i}(|\rightarrow, i\rangle - |\leftarrow, i\rangle)$. As with the OR switching network, a corollary of this lemma is that there is a quantum algorithm for evaluating AND in optimal time $\tilde{O}(\sqrt{d})$.

Let $G = G_{\text{AND},d}$ be the graph in Figure 2, defined by

$$V = \{s = u_0, u_1, \dots, u_d = t\} \text{ and } E = \{e_i = (u_{i-1}, u_i) : i \in [d]\},$$

so $E(s) = E^{\rightarrow}(s) = \{e_1\}$, $E(t) = E^{\leftarrow}(t) = \{e_d\}$, and for all $i \in [d-1]$, $E^{\leftarrow}(u_i) = \{e_i\}$ and $E^{\rightarrow}(u_i) = \{e_{i+1}\}$. Since G is a switching network, it has boundary $B = \{s, t\}$. We will let the graph be weighted, with weights $w_{e_i} = w_i$. Since G is a switching network: every edge is a switch, which fixes Ξ_{e_i} , $\Xi_{e_i}^{\mathcal{A}}$ and $\Xi_{e_i}^{\mathcal{B}}$ for all $i \in [d]$; G has canonical st -boundary, which fixes Ξ_s , $\Xi_s^{\mathcal{A}}$, $\Xi_s^{\mathcal{B}}$, Ξ_t , $\Xi_t^{\mathcal{A}}$, and $\Xi_t^{\mathcal{B}}$; and every vertex is simple, which fixes the spaces \mathcal{V}_{u_i} for $i \in \{0, \dots, d\}$. In particular, letting $w_0 = w_{d+1} = 1$, $s = 0$, $t = d + 1$, and $i \in [d]$ label e_i , we must have:

$$\forall i \in \{0, \dots, d\}, \mathcal{V}_{u_i} = \text{span} \{ \sqrt{w_i} |\leftarrow, i\rangle + \sqrt{w_{i+1}} |\rightarrow, i+1\rangle \}.$$

This fully defines

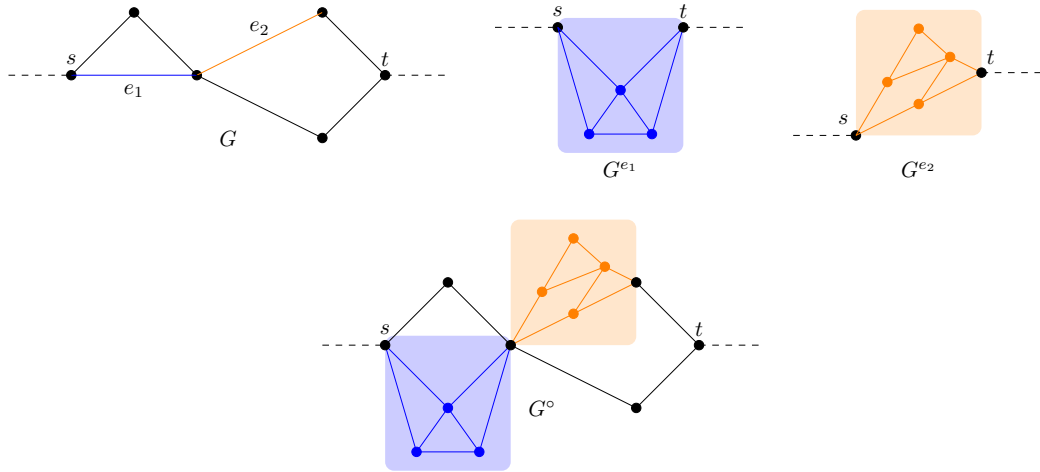
$$\mathcal{A}_G = \bigoplus_{e \in E \cup B} \Xi_e^{\mathcal{A}} \quad \text{and} \quad \mathcal{B}_G = \bigoplus_{i=0}^d \mathcal{V}_{u_i} + \bigoplus_{e \in E} \Xi_e^{\mathcal{B}}.$$

We prove the second and the third items of Lemma 12 in the full version of the paper. It remains to find working bases that can be generated efficiently. We show in the full version that there is an st -composable working basis for \mathcal{A}_G that can be generated in unit time and there is a st -composable working basis for \mathcal{B}_G that can be generated in $O(\log d)$ time.

2.2.3 Any Quantum Algorithm

Ref. [13] deals with composing arbitrary quantum algorithms, implicitly using subspace graphs for the task. It is shown how to define certain subspaces from an arbitrary quantum algorithm, where the overlap between these various spaces gives rise to a graph as in Figure 3. In the full version of the paper, we show that these subspaces are precisely a subspace graph, so that the following follows from [13].

► **Lemma 13.** *From any quantum algorithm computing some $f : \{0, 1\}^n \rightarrow \{0, 1\}$ in time T with S qubits, we can derive an st -composable subspace graph (Definition 10) G that computes f with $\dim H_G = O(2^S T)$, $\hat{W}_+(G) \leq 2T$ and $\hat{W}_-(G) \leq 2T$; with st -composable working bases that can be generated in time $O(\log(T))$.*



■ **Figure 4** An example of replacing edges e_1 and e_2 in G with graphs G^{e_1} and G^{e_2} to obtain G° . Boundary “edges” are represented by dashed lines. Switching networks already lend themselves to this type of recursion: we can replace an edge with a 2-terminal graph, and then the “edge” is traversable if and only if the terminals are connected. The difference with the more general recursion in Theorem 14 is that the subspace graphs used to replace edges (as well as other parts of G) might have more complicated structure than just their graph structure; for example, they might encode quantum algorithms like in Section 2.2.3.

2.3 Composition

Subspace graphs lend themselves well to very general kinds of recursion. We can compose subspace graphs by identifying some of the vertices on their boundaries. In full generality, this may result in a subspace graph that is difficult to analyze. For example, if we replace two parallel edges with subspace graphs derived from quantum algorithms, “flow” along these edges may have complex phases that interfere on the other side. Our nice classical intuition of flows breaks down in the fully general case, which is not surprising, since quantum algorithms are not classical. However, an important question is in which special cases, and to what extent, we can compose subspace graphs and keep enough classical intuition to analyze them.

One special case is implicit in [13], and here we present another special case, which we call *switch composition*. Switch composition generalizes a very simple kind of recursion that can be done in switching networks. Since an st -path (or more generally, flow) is allowed to use an edge if and only if $\varphi(e) = 1$, we can replace it with a switching network G^e for some function f_e , which will have an st -path from one endpoint to the other if and only if $f_e(x) = 1$. We can view this as removing e from the graph, and then adding its endpoints to the boundary, to then be identified with the boundary $\{s^e, t^e\}$ of G^e . By applying this type of composition, the OR and AND switching networks in Section 2.2.1 and Section 2.2.2 can be combined to make switching networks for any Boolean formula [14].

Here we investigate how to compose subspace graphs with specific properties that generalize switching networks – st -composable subspace graphs (Definition 10) – into the switches of a graph G . Specifically, if \overline{E} is the set of edges of G that are switches, we will replace $e \in \overline{E}$ with an st -composable graph G^e . We can assume without loss of generality that we replace every $e \in \overline{E}$ with some G^e , since letting G^e be the graph consisting of a single edge from s^e to t^e is just like not replacing e .

► **Theorem 14.** *Let G be an st -composable subspace graph with st -composable working bases that can be generated in time T . For each $e \in \overline{E}$, the set of switches of G , let G^e be an st -composable subspace graph with st -composable working bases that can be generated in time at most T' . Then there exists an st -composable subspace graph G° with $\dim H_{G^\circ} \leq \dim H_G - 2|\overline{E}| + \sum_{e \in \overline{E}} \dim H_{G^e}$ such that:*

- G° has st -composable working bases that can be generated in time $T + T' + O(1)$.
- If $|\hat{w}\rangle$ is a positive witness for G (Definition 3), and for each $e \in \overline{E}$ such that $\langle \rightarrow, e | \hat{w} \rangle \neq 0$, $|\hat{w}^e\rangle$ is a positive witness for G^e , then

$$|\hat{w}^\circ\rangle = \sum_{e \in \overline{E}} \langle \rightarrow, e | \hat{w} \rangle |\hat{w}^e\rangle + \Pi_{E \setminus \overline{E}} |\hat{w}\rangle$$

is a positive witness for G° .

- If $|\hat{w}_A\rangle$ is a negative witness for G (Definition 4), and for each $e \in \overline{E}$ such that $\langle \rightarrow, e | \hat{w}_A \rangle \neq 0$, $|\hat{w}_A^e\rangle$ is a negative witness for G^e , then

$$|\hat{w}_A^\circ\rangle = \sum_{e \in \overline{E}} \langle \rightarrow, e | \hat{w}_A \rangle |\hat{w}_A^e\rangle + \Pi_{E \setminus \overline{E}} |\hat{w}_A\rangle$$

is a negative witness for G° .

Informally, we obtain G° from G by, for each $e \in \overline{E}$, removing the edge e , and identifying its endpoints with the vertices $s = s^e$ and $t = t^e$ of the graph G^e . This is illustrated in Figure 4. The subspaces associated with G° are mostly inherited from G and G^e , except for those on the glued boundaries. There, intuitively, we replace $|\rightarrow, e\rangle$ with the edges incident to $s = s^e$ in G^e , and $|\leftarrow, e\rangle$ with the edges incident to $t = t^e$ in G^e .

Note that the witnesses in the composed subspace graph have a very logical form. We obtain a positive witness $|\hat{w}^\circ\rangle$ by taking a positive witness for G , and replacing the edge $|\rightarrow, e\rangle + |\leftarrow, e\rangle$ with a positive witness, and similarly for negative witnesses.

Next, we apply the composition construction of Theorem 14 to compose the switching networks for AND and OR in Section 2.2.2 and Section 2.2.1 with other st -composable subspace graphs computing some functions f_σ to get a subspace graph computing the composed function $\varphi \circ (f_\sigma)_{\sigma \in \Sigma}$ when φ is a Boolean formula on $\{0, 1\}^\Sigma$. This is the function obtained by computing the value of each variable x_σ of the formula φ as f_σ of some input. Subspace graphs for φ can be obtained simply by composing the switching networks for OR and AND (see [14]). Our composition theorem, Theorem 14, generalizes this simple switching network composition.

A formula is a rooted tree, where each internal node represents an AND or an OR gate, and each leaf represents a literal. We let $\overline{\Sigma} \setminus \Sigma$ denote the set of internal nodes, and for each $\sigma \in \overline{\Sigma} \setminus \Sigma$, d_σ is the number of children of σ .

A family of formulas is balanced if there is a constant c such that for every internal node σ , if its subtree has N leaves, then the sub-tree of each of its d_σ children has at most cN/d_σ leaves.

► **Lemma 15.** *Let φ be a balanced formula on $\{0, 1\}^\Sigma$. Let $\{f_\sigma\}_{\sigma \in \Sigma}$ be Boolean functions, and for each $\sigma \in \Sigma$, let G_σ be an st -composable subspace graph (Definition 10) computing f_σ , with working bases that can be generated in time at most T , and $\log \dim H_{G_\sigma} \leq S$.*

For each $\sigma \in \Sigma$, let C_σ be a known upper bound on $\hat{C}(G_\sigma)$. Then there is an st -composable subspace graph G° computing $\varphi \circ (f_\sigma)_{\sigma \in \Sigma}$ with $\log \dim H_{G^\circ} = S + O(\log |\Sigma|)$ and

$$\hat{C}(G^\circ)^2 \leq \sum_{\sigma \in \Sigma} C_\sigma^2,$$

as long as for each $\sigma \in \bar{\Sigma} \setminus \Sigma$, a superposition proportional to $\sum_{i \in [d_\sigma]} \sqrt{C_{\sigma,i}^\pm} |i\rangle$ can be generated in $O(\log d_\sigma)$ complexity, for certain values $C_{\sigma,i}^\pm$ related to the bounds $\{C_\sigma\}_{\sigma \in \Sigma}$. Furthermore, the working bases of G° can be generated in time $T + O(\log |\Sigma|)$.

2.4 Quantum Divide & Conquer

In this section, we state our time-efficient quantum divide & conquer results. The following is a time-efficient version of Strategy 1 in [9], restricted to symmetric formulas.

► **Theorem 16.** Fix a function family $f_{\ell,n} : D_{\ell,n} \rightarrow \{0,1\}$. Fix unit-time-computable functions $\lambda_1, \lambda_2 : \mathbb{N} \rightarrow \mathbb{N}$, and a formula φ on $\{0,1\}^{a+1}$ such that $\varphi = \varphi'(z_1, \dots, z_a) \vee z_{a+1}$ for some symmetric formula φ' . Suppose $\{\mathcal{P}_{\text{aux},\ell,n}\}_{\ell,n \in \mathbb{N}}$ is a family of quantum algorithms such that:

- $\mathcal{P}_{\text{aux},\ell,n}$ decides some $f_{\text{aux},\ell,n} : D_{\ell,n} \rightarrow \{0,1\}$ with time and space complexities $T_{\text{aux}}(\ell, n)$ and $S_{\text{aux}}(\ell, n)$;
- if $\ell \leq \ell_0$, $f_{\ell,n}(x) = f_{\text{aux},\ell,n}(x)$;
- if $\ell > \ell_0$, $f_{\ell,n}(x) = \varphi \circ (f_i)_{i \in [a+1]}$ where each f_i for $i \in [a]$ is such that $f_i(x) = f_{\lambda_1(\ell), \lambda_2(n)}(x^i)$ for some unit-time-computable instance x^i of $f_{\lambda_1(\ell), \lambda_2(n)}$, and $f_{a+1} = f_{\text{aux},\ell,n}$.

Then there is a bounded-error quantum algorithm that decides $f_{\ell,n}$ with time complexity $\tilde{O}(T(\ell, n))$, and space complexity $O(S_{\text{aux}}(\ell, n) + \log T(\ell, n))$, where for all $\ell > \ell_0$:

$$T(\ell, n) := \sqrt{aT(\lambda_1(\ell), \lambda_2(n))^2 + 4T_{\text{aux}}(\ell, n)^2} \leq \sqrt{a}T(\lambda_1(\ell), \lambda_2(n)) + 2T_{\text{aux}}(\ell, n)$$

and for $\ell \leq \ell_0$, $T(\ell, n) = 2T_{\text{aux}}(\ell, n)$.

To prove Theorem 16, we make a subspace graph that computes $f_{\ell,n} = \varphi \circ (f_i)_i$ using Lemma 15, which allows us to combine subspace graphs for $f_{\lambda_1(\ell), \lambda_2(n)}$ – built up inductively – and $f_{\text{aux},\ell,n}$ – built by turning the quantum algorithm $\mathcal{P}_{\text{aux},\ell,n}$ into a subspace graph using Lemma 13. Note that Lemma 15 applies to any balanced formula φ , but in the special case we consider in Theorem 16, the values $C_{\sigma,i}^\pm$ referred to in Lemma 15 are well behaved.

2.5 Application to DSTCON

In this section we consider the *directed st-connectivity* problem.

► **Problem 17 (DSTCON).** Given access to a directed graph $G = (V, E)$ via the oracle \mathcal{O}_G , and two vertices $s, t \in V$, decide whether there is a directed path from s to t in G .

There is a classical recursive algorithm for DSTCON that operates in the low-space regime due to Savitch [22] that decides DSTCON in time $O((2n)^{\log n} = 2^{\log^2 + O(\log n)})$ and space $O(\log^2 n)$. The algorithm recursively calls a subroutine that decides a function $f_{\ell,n}(G, u, v)$, which is 1 if and only if there is a path of length at most ℓ from u to v in G . We can express $f_{\ell,n}$ recursively, in terms of a symmetric formula φ' in $a = 2n$ variables:

$$f_{\ell,n}(G, u, v) = \bigvee_{w \in V} \underbrace{(f_{\ell/2,n}(G, u, w) \wedge f_{\ell/2,n}(G, w, v))}_{\varphi'}.$$

We show a quantum speedup for Savitch's algorithm via application of Theorem 16, which gives us the recursion

$$T(\ell, n) = \sqrt{2n}T(\ell/2, n) + O(1)$$

from which we get:

$$T(n, n) = \sqrt{2n}^{\log n} + O(\log n) = 2^{\frac{1}{2} \log^2(n) + O(\log n)},$$

which is the complexity of computing $f_{n,n}(G, s, t) = \text{DSTCON}(G)$. This yields the following.

► **Theorem 18.** *Let $G = (V, E)$ be a directed graph, $|V| = n$. Then there exists a recursive quantum algorithm that decides DSTCON on G with bounded error in time $\tilde{O}((\sqrt{2n})^{\log n}) = 2^{\frac{1}{2} \log^2 n + O(\log n)}$ and space $O(\log^2 n)$.*

References



- 1 Jonathan Allcock, Jinge Bao, Aleksandrs Belovs, Troy Lee, and Miklos Santha. On the quantum time complexity of divide and conquer. [arXiv:2311.16401](#), 2023.
- 2 Andris Ambainis. Quantum search with variable times. *Theory of Computing Systems*, 47:786–807, 2010. [arXiv:quant-ph/0609168](#) [doi:10.1007/s00224-009-9219-1](#).
- 3 Simon Apers, Stacey Jeffery, Galina Pass, and Michael Walter. (No) Quantum Space-Time Tradeoff for USTCON. In *31st Annual European Symposium on Algorithms (ESA 2023)*, pages 10:1–10:17, 2023. [doi:10.4230/LIPIcs.ESA.2023.10](#).
- 4 Aleksandrs Belovs. Quantum walks and electric networks. [arXiv:1302.3143](#), 2013.
- 5 Aleksandrs Belovs, Andrew M. Childs, Stacey Jeffery, Robin Kothari, and Frédéric Magniez. Time-efficient quantum walks for 3-distinctness. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 105–122, 2013. [doi:10.1007/978-3-642-39206-1_10](#).
- 6 Aleksandrs Belovs, Stacey Jeffery, and Duyal Yolcu. Taming quantum time complexity. [arXiv:2311.15873](#), 2023. [doi:10.48550/arXiv.2311.15873](#).
- 7 Aleksandrs Blovs and Ben W. Reichardt. Span programs and quantum algorithms for st-connectivity and claw detection. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 193–204, 2012. [doi:10.1007/978-3-642-33090-2_18](#).
- 8 M. L. Bonet and S. R. Buss. Size-depth tradeoffs for boolean formulae. *Information Processing Letters*, 49:151–155, 1994. [doi:10.1016/0020-0190\(94\)90093-0](#).
- 9 Andrew M. Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. Quantum divide and conquer. [arXiv:2210.06419](#), 2022.
- 10 Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. Earlier version in ICALP’04. [arXiv:quant-ph/0401091](#) [doi:10.1137/050644719](#).
- 11 Tsuyoshi Ito and Stacey Jeffery. Approximate span programs. *Algorithmica*, 79:2158–2195, 2019. [doi:10.1007/S00453-018-0527-1](#).
- 12 Michael Jarret, Stacey Jeffery, Shelby Kimmel, and Alvaro Piedrafita. Quantum algorithms for connectivity and related problems. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA)*, pages 49:1–49:13, 2018. [doi:10.4230/LIPIcs.ESA.2018.49](#).
- 13 Stacey Jeffery. Quantum subroutine composition. [arXiv:2209.14146](#), 2022.
- 14 Stacey Jeffery and Shelby Kimmel. Quantum algorithms for graph connectivity and formula evaluation. *Quantum*, 1(26), 2017.
- 15 Stacey Jeffery and Sebastian Zur. Multidimensional quantum walks and application to k -distinctness. In *Proceedings of the 55th ACM Symposium on the Theory of Computing (STOC)*, pages 1125–1130, 2023. [arXiv:2208.13492](#)
- 16 C. Y. Lee. Representation of switching functions by binary decision programs. *Bell Systems Technical Journal*, 38(4):985–999, 1959. [doi:10.1002/j.1538-7305.1959.tb01585.x](#).
- 17 Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011. Earlier version in STOC’07. [arXiv:quant-ph/0608026](#) [doi:10.1137/090745854](#).

- 18 Aaron H. Potechin. *Analyzing monotone space complexity via the switching network model*. PhD thesis, Massachusetts Institute of Technology, 2015.
- 19 Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 544–551, 2009. [arXiv:0904.2759](#) [doi:10.1109/FOCS.2009.55](#).
- 20 Ben W. Reichardt. Faster quantum algorithm for evaluating game trees. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 546–559, 2011. [doi:10.5555/2133036.2133079](#).
- 21 Ben W. Reichardt and Robert Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012. [doi:10.4086/toc.2012.v008a013](#).
- 22 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970. [doi:10.1016/S0022-0000\(70\)80006-X](#).
- 23 Claude E. Shannon. A symbolic analysis of relay and switching networks. *Transactions of the American Institute of Electrical Engineers*, 57(12):713–723, 1938. [doi:10.1109/T-AIEE.1938.5057767](#).
- 24 Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949. [doi:10.1002/j.1538-7305.1949.tb03624.x](#).
- 25 Mario Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 32–41, 2004. [arXiv:quant-ph/0401053](#) [doi:10.1109/FOCS.2004.53](#).

Modal Separation of Fixpoint Formulae

Jean Christoph Jung  

TU Dortmund University, Germany

Jędrzej Kołodziejcki  

TU Dortmund University, Germany

Abstract

Modal separability for modal fixpoint formulae is the problem to decide for two given modal fixpoint formulae φ, φ' whether there is a modal formula ψ that separates them, in the sense that $\varphi \models \psi$ and $\psi \models \neg\varphi'$. We study modal separability and its special case modal definability over various classes of models, such as arbitrary models, finite models, trees, and models of bounded outdegree. Our main results are that modal separability is PSPACE-complete over words, that is, models of outdegree ≤ 1 , EXPTIME-complete over unrestricted and over binary models, and 2-EXPTIME-complete over models of outdegree bounded by some $d \geq 3$. Interestingly, this latter case behaves fundamentally different from the other cases also in that modal logic does not enjoy the Craig interpolation property over this class. Motivated by this we study also the induced interpolant existence problem as a special case of modal separability, and show that it is CONEXPTIME-complete and thus harder than validity in the logic. Besides deciding separability, we also investigate the problem of efficient construction of separators. Finally, we consider in a case study the extension of modal fixpoint formulae by graded modalities and investigate separability by modal formulae and graded modal formulae.

2012 ACM Subject Classification Theory of computation \rightarrow Modal and temporal logics

Keywords and phrases Modal Logic, Fixpoint Logic, Separability, Interpolation

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.55

1 Introduction

For given logics $\mathcal{L}, \mathcal{L}^+$, the \mathcal{L} -separability problem for \mathcal{L}^+ is to decide given two \mathcal{L}^+ -formulae φ, φ' whether there is an \mathcal{L} -formula ψ that separates φ and φ' in the sense that $\varphi \models \psi$ and $\psi \models \neg\varphi'$. Obviously, a separator can only exist when φ and φ' are mutually exclusive, and the problem is only meaningful when \mathcal{L} is less expressive than \mathcal{L}^+ . Intuitively, a separator formulated in a “simpler” logic \mathcal{L} explains a given inconsistency in a “complicated” logic \mathcal{L}^+ . Note that, for logics \mathcal{L}^+ closed under negation, \mathcal{L} -separability generalizes the \mathcal{L} -definability problem for \mathcal{L}^+ : decide whether a given \mathcal{L}^+ -formula is equivalent to an \mathcal{L} -formula. Indeed, $\varphi \in \mathcal{L}^+$ is equivalent to an \mathcal{L} -formula iff φ and $\neg\varphi$ are \mathcal{L} -separable. Since separability is more general than definability, solving it requires an even better understanding of the logics under consideration. Both separability and definability are central problems with many applications in computer science. As seminal work let us only mention definability and separability of regular word languages by first-order logic [26, 29, 9].

In this paper we study definability and separability of formulae of the modal μ -calculus μ ML [27, 20] by formulae in propositional modal logic ML. μ ML is the extension of ML with fixpoints that encompasses virtually all specification languages such as PDL [12] and LTL and CTL [3]. Let us consider an example.

► **Example 1.** Consider the following properties P_1, P_2, P_3 of vertex-labelled trees:

- P_1 : there is an infinite path starting in the root on which each point satisfies a ;
- P_2 : on every path there are only finitely many points satisfying a ;
- P_3 : on every path at most two points satisfy a .

The properties are expressible in μ ML but not in ML, and both P_1, P_2 and P_1, P_3 are mutually exclusive. The properties P_1, P_3 are separated by the ML-formula $\psi = a \wedge \diamond(a \wedge \diamond a)$ which



© Jean Christoph Jung and Jędrzej Kołodziejcki;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 55; pp. 55:1–55:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Table 1** Overview of our results. All complexity results are completeness results.

	all models	words	binary trees	d -ary trees, $d \geq 3$
ML-definability	EXPTIME [24]	PSPACE	EXPTIME [24]	EXPTIME [24]
ML-separability	EXPTIME	PSPACE	EXPTIME	2-EXPTIME
separator construction	double exp.	single exp.	double exp.	triple exp.
ML interpolant existence	always	always	always	CONEXPTIME

expresses that there is a path starting with three points satisfying a . On the other hand, no ML-formula separates P_1, P_2 . The intuitive reason for this is that any ML-formula ψ only sees trees up to depth $|\psi|$, and one can find two trees with properties P_1, P_2 which nonetheless look the same up to depth $|\psi|$. ◀

We explore the definability and separability problems over several classes of models relevant for computer science: all models, words, trees of bounded or unbounded outdegree; as well as restrictions of all these classes to finite models. On top of analyzing the decision problems, we also address the problem of constructing efficient definitions and separators whenever they exist. The starting point for our research is the seminal paper of Otto [24], where he solves modal definability over models of bounded and unbounded outdegree. In this paper, we continue this line of research and establish a fairly complete and interesting picture. Table 1 summarizes our results. We now explain its content further.

The first line essentially repeats Otto’s results; we only add the observation that ML-definability over words is PSPACE-complete. Interestingly, separability is substantially more difficult. The case of words is the easiest one, both in terms of computational complexity and required arguments. Next come the cases of binary and of unrestricted trees. These two classes possess some nice structural properties which (although true for different reasons) enable a common algorithmic treatment. Finally, the cases of trees with outdegree bounded by a number $d \geq 3$ enter the stage. These trees lack the good properties essential for previous constructions which results in higher computational complexity. The hardness result for $d \geq 3$ is interesting for two reasons. First, as it is entirely standard to encode trees of higher outdegree into binary ones, one could expect the ternary (and higher) case to have the same complexity as the binary one. And second, even though there are known cases when separation is provably harder than definability (regularity of visibly pushdown languages is decidable [23, Theorem 19] but regular separability thereof is not [19, Theorem 2.4]), to the best of our knowledge our results are the only such case known in logic.

The complexity landscape for deciding separability is also reflected in the maximal sizes of the separators that we construct. Relying on the well-known connection of μ ML to automata, we provide effective constructions for the cases of all models, words, and binary trees. It is worth mentioning that equally effective constructions for definability over all models are given in [22], but they do not work for separability. The ternary case follows from a general argument. Our construction of separators over words is optimal. Under mild assumptions (there are at least two modalities) the constructions over binary and over unrestricted trees are optimal as well, but we leave it open whether these assumptions are needed for the lower bounds. In the case of ternary and higher outdegree trees we only conjecture optimality of the constructed separators.

Finally, we observe that ML lacks the Craig interpolation property over trees of outdegree bounded by $d \geq 3$. Recall that a *Craig interpolant* for $\varphi \models \varphi'$ in some logic \mathcal{L} is a formula $\psi \in \mathcal{L}$ only using the common symbols of φ and φ' and such that $\varphi \models \psi \models \varphi'$. A logic satisfies the *Craig interpolation property (CIP)* if a Craig interpolant of $\varphi \models \varphi'$ always exists.

It is known that ML enjoys CIP over all models and over words [15] and it follows from our techniques that this transfers to binary trees. In contrast and as mentioned above, over ternary and higher-arity trees ML lacks the CIP. It is worth mentioning that modal logic over frames of arity bounded by some d has been studied under the name $\mathbf{K} \oplus \mathit{alt}_d$ [4]. Our results imply that $\mathbf{K} \oplus \mathit{alt}_d$ enjoys CIP iff $d \leq 2$. Motivated by the lack of CIP over higher-arity trees, we study the induced interpolant existence problem – determining whether two given ML-formulae φ, φ' admit a Craig interpolant – as a special case of separability. We show it to be CONEXPTIME-complete over higher arity trees, and thus harder than validity. Interpolant existence has recently been studied for other logics without CIP [18, 1].

As an application of our results for d -ary trees with $d \geq 3$ we additionally present a case study: separability in the *graded* setting in which we allow counting modalities saying “there are at least k children such that [...]” [11]. Counting modalities are a standard extension of modal logic that is especially relevant in applications in knowledge representation for conceptual modeling [2]. We show that ML-separability of graded μ ML is 2-EXPTIME-complete, while it is EXPTIME-complete if we allow counting modalities also in the separator. The intuitive reason for the hardness in the former case is that trees of bounded arity are definable in graded μ ML. This former case is also related with a recent study about separating logics supporting counting quantifiers by logics without these [21].

It is worth to mention that ML-definability of μ ML-formulae generalizes the *boundedness problem* which asks whether a formula with a single fixpoint is equivalent to a modal formula. Boundedness has been studied for other logics such as monadic-second order logic [6], datalog [16], and the guarded fragment of first-order logic [5]. Our paper is an extension of the preliminary paper [17].

The paper is organized as follows. After this introduction 1, we set notation and recall basic facts in the preliminary Section 2. Next, we introduce some topic-specific terminology, discuss a relevant construction of Otto, and solve the case of all models in Section 3. In the following Sections 4 and 5 we deal with unary and binary trees, and in Section 6 we solve the most challenging case of trees of outdegree bounded by $d \geq 3$. Section 7 applies our results to the case with graded modalities. The last Section 8 contains conclusions and final remarks.

2 Preliminaries

We recall the main notions about modal logic ML and the modal μ -calculus μ ML. For the rest of this paper fix disjoint, countably infinite sets \mathbf{Prop} of *atomic propositions* and \mathbf{Var} of *variables*. The syntax of μ ML is given by the rule

$$\varphi ::= \tau \mid \neg\tau \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \diamond\varphi \mid \square\varphi \mid x \mid \mu x.\varphi \mid \nu x.\varphi$$

where $\tau \in \mathbf{Prop}$ and $x \in \mathbf{Var}$. We assume that formulae of μ ML are in a normal form such that every $x \in \mathbf{Var}$ appears at most once in a formula, and if it does appear then its appearance has a unique superformula ψ beginning with μx or νx . Modal logic ML is defined as the fragment of μ ML with no fixpoint operators μ and ν nor variables. Both in ML and μ ML, we use abbreviations like \top (for $a \vee \neg a$ for some $a \in \mathbf{Prop}$), $\diamond^n \varphi$ (for a formula $\diamond \dots \diamond \varphi$ with n leading \diamond 's), and $\neg\varphi$. We denote with $\mathit{sig}(\varphi)$ the set of propositions that occur in φ , and recall that the *modal depth* of an ML formula is the maximal nesting of \diamond, \square . With \mathbf{ML}^n we denote the class of all ML-formulae of modal depth at most n , and with \mathbf{ML}_σ^n we denote its subclass restricted to signature σ . The *size* $|\varphi|$ of a formula φ is the length of φ represented as a string. This choice of the simplest possible measure of size does not matter for most of our results. We will briefly discuss alternative notions of size in the concluding Section 8.

Both ML and μ ML are interpreted in pointed Kripke structures. More formally, a model \mathcal{M} is a quadruple $\mathcal{M} = (M, v_I, \rightarrow, \text{val})$ consisting of a set M called its *universe*, a distinguished point $v_I \in M$ called the *root*, an accessibility relation $\rightarrow \subseteq M \times M$, and a valuation $\text{val} : M \rightarrow \mathcal{P}(\text{Prop})$.

The semantics of μ ML can be defined in multiple equivalent ways. The one most convenient for us is through parity games (see [32] for an introduction). Given a model \mathcal{M} and a formula $\varphi \in \mu\text{ML}$ we define a *semantic game* $\mathcal{G}(\mathcal{M}, \varphi)$ played between players \exists ve and \forall dam. The positions are $M \times \text{SubFor}(\varphi)$. The moves depend on the topmost connective. From a position of the shape $(v, \psi \vee \psi')$ or $(v, \psi \wedge \psi')$ it is allowed to move to either (v, ψ) or (v, ψ') . From $(v, \diamond\psi)$ and $(v, \square\psi)$ the allowed moves lead to all (w, ψ) such that $v \rightarrow w$. In position (v, τ) or $(v, \neg\tau)$ the game stops and \exists ve wins iff v satisfies the formula component τ or $\neg\tau$, respectively. From $(v, \mu x.\psi)$ and $(v, \nu x.\psi)$ the game moves to (v, ψ) , and from (v, x) to (v, ψ) where ψ is the unique superformula of x beginning with μx or νx . \exists ve owns positions whose formula component has \vee or \diamond as the topmost connective and \forall dam owns all other positions. \exists ve wins an infinite play π if the outermost subformula seen infinitely often in π begins with ν . We say that \mathcal{M}, v *satisfies* φ and write $\mathcal{M}, v \models \varphi$ if \exists ve wins the game $\mathcal{G}(\mathcal{M}, \varphi)$ from position (v, φ) . Since \mathcal{M} is by definition pointed, we abbreviate $\mathcal{M}, v_I \models \varphi$ with $\mathcal{M} \models \varphi$.

The same symbol denotes entailment: $\varphi \models \psi$ means that every model of φ is a model of ψ . In the case only models from some fixed class \mathbf{C} are considered we talk about satisfiability and entailment *over* \mathbf{C} . Let \mathcal{L} be a subset of μML such as ML or ML_σ^n . If two models \mathcal{M} and \mathcal{N} satisfy the same formulae of \mathcal{L} then we call them \mathcal{L} -equivalent and write $\mathcal{M} \equiv_{\mathcal{L}} \mathcal{N}$.

In the paper we will study models of bounded and unbounded outdegree. The *outdegree* of a point $w \in M$ in a model $\mathcal{M} = (M, v_I, \rightarrow, \text{val})$ is the number of successors of w in the underlying directed graph $G_{\mathcal{M}} = (M, \rightarrow)$. We say that \mathcal{M} has *finite outdegree* if every point has finite outdegree and *bounded outdegree* if there is a finite uniform upper bound d on the outdegree of its points. In the latter case, we will call \mathcal{M} *d-ary*, and *binary* or *ternary* if $d = 2$ or $d = 3$. If $d = 1$, then we call \mathcal{M} a *word*. A *d-ary* model is *full* if each of its nodes is either a leaf (i.e. has no children) or has precisely d children. A model \mathcal{M} is a *tree* if $G_{\mathcal{M}}$ is a (directed) tree with root v_I . We denote with \mathbb{T}^d the class of all *d-ary* tree models. Both ML and μML are invariant under bisimulation, and every (*d-ary*) model is bisimilar to a (*d-ary*) tree. Hence, we do not lose generality by only looking at tree models.

A *prefix* of a tree is a subset of its universe closed under taking ancestors. When no confusion arises we identify a prefix $N \subseteq M$ with the induced subtree \mathcal{N} of \mathcal{M} that has N as its universe. The *depth* of a point is the distance from the root. The prefix of depth n (or just *n-prefix*) is the set of all points at depth at most n and is denoted by $M_{|_n}$ (and the corresponding subtree by $\mathcal{M}_{|_n}$).

Bisimulations

We define bisimulations and bisimilarity for trees, assuming for convenience that bisimulations only link points at the same depth. Let $\mathcal{M}, \mathcal{M}'$ be trees and $Z \subseteq M \times M'$ a relation between M and M' that relates only points of the same depth. Then, Z is a *bisimulation between \mathcal{M} and \mathcal{M}'* if it links the roots $v_I Z v'_I$, and for every $w Z w'$ the following conditions are satisfied:

(atom) $\text{val}(w) = \text{val}'(w')$,

(forth) for every $v \in M$ with $w \rightarrow v$ there is a $v' \in M'$ with $w' \rightarrow v'$ and $v Z v'$, and

(back) for every $v' \in M'$ with $w' \rightarrow v'$ there is a $v \in M$ with $w \rightarrow v$ and $v Z v'$.

A *functional bisimulation* (also known as *bounded morphism*) is a function whose graph is a bisimulation. If Z is a functional bisimulation from \mathcal{M} to \mathcal{M}' then we write $Z : \mathcal{M} \xrightarrow{\text{bis}} \mathcal{M}'$ and call \mathcal{M}' a *bisimulation quotient* of \mathcal{M} . The *bisimilarity quotient* of \mathcal{M} is a quotient \mathcal{M}' of \mathcal{M}

such that if $Z' : \mathcal{M}' \rightarrow \mathcal{M}''$ then $\mathcal{M}' = \mathcal{M}''$. It follows from analogous results for arbitrary models that every tree $\mathcal{M} \in \mathbb{T}^d$ has a unique (up to isomorphism) bisimilarity quotient $\mathcal{M}' \in \mathbb{T}^d$ and that two trees are bisimilar iff their bisimilarity quotients are isomorphic.

Further, for every $n \in \mathbb{N}$ and every subset $\sigma \subseteq \mathbf{Prop}$ of the signature we consider a restricted variant of bisimulations called (σ, n) -bisimulations. In a (σ, n) -bisimulation the atom condition is only checked with respect to σ and the back and forth conditions only for points at depth smaller than n . Formally, a relation $Z \subseteq M \times M'$ is a (σ, n) -bisimulation if it is a bisimulation between the n -prefixes of the σ -reducts of $\mathcal{M}, \mathcal{M}'$. We call a (σ, n) -bisimulation between $\mathcal{M}, \mathcal{M}'$ a (σ, n) -isomorphism if it is bijective on the n -prefixes of $\mathcal{M}, \mathcal{M}'$. We write $\mathcal{M} \simeq_{\sigma}^n \mathcal{M}'$ if there exists a (σ, n) -bisimulation between \mathcal{M} and \mathcal{M}' and $\mathcal{M} \cong_{\sigma}^n \mathcal{M}'$ if there is a (σ, n) -isomorphism between them. Crucially, over every class \mathbf{C} of models and for every finite σ the equivalences $\equiv_{\mathbf{ML}_{\sigma}^n}$ and \simeq_{σ}^n coincide, for every n .

Automata

We exploit the well-known connection of $\mu\mathbf{ML}$ and automata that read tree models. A *nondeterministic parity tree automaton (NPTA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, \text{rank})$ where Q is a finite set of states, $q_I \in Q$ is the initial state, $\Sigma = \mathcal{P}(\sigma)$ for some finite set $\sigma \subseteq \mathbf{Prop}$, rank assigns each state a priority, and δ is a transition function of type:

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q^{\leq d}),$$

where $Q^{\leq d}$ denotes the set of all tuples over Q of length at most d . A run of \mathcal{A} on a tree \mathcal{M} is an assignment $\rho : M \rightarrow Q$ sending the root of the tree to q_I and consistent with δ in the sense that $(\rho(v_1), \dots, \rho(v_k)) \in \delta(\rho(v), \text{val}(v) \cap \sigma)$ for every point v with children v_1, \dots, v_k . On occasion when considering trees of unbounded outdegree we will use automata with transition function of type $\delta : Q \times \Sigma \rightarrow \mathcal{P}(\mathcal{P}(Q))$. Then, consistency of ρ with δ means that $\{\rho(v') \mid v' \in V\} \in \delta(\rho(v), \text{val}(v) \cap \sigma)$ for every v with a set V of children. In either case, we call the run ρ *accepting* if for every infinite path $v_0, v_1 \dots$ in \mathcal{M} the sequence $\text{rank}(\rho(v_0)), \text{rank}(\rho(v_1)), \dots$ satisfies the parity condition. We write $\mathcal{M} \models \mathcal{A}$ in case \mathcal{A} has an accepting run on \mathcal{M} . An automaton that is identical to \mathcal{A} except that the original initial state is replaced with q is denoted $\mathcal{A}[q_I \leftarrow q]$. The *size* of an automaton \mathcal{A} is the number of its states and is denoted by $|\mathcal{A}|$.

An NPTA \mathcal{A} is *equivalent* to a formula $\varphi \in \mu\mathbf{ML}$ over a class \mathbf{C} of trees when $\mathcal{M} \models \varphi$ iff $\mathcal{M} \models \mathcal{A}$ for every tree $\mathcal{M} \in \mathbf{C}$. We rely on the following classical result (see for example the discussion in [31] and the well-presented Dealternation Theorem 5.7 in [7]):

► **Theorem 2.** *For every $\mu\mathbf{ML}$ -formula φ and class \mathbf{C} of trees, we can construct an NPTA with exponentially many states equivalent to φ over \mathbf{C} . The construction takes exponential time when $\mathbf{C} \subseteq \mathbb{T}^d$ for some d , and doubly exponential time in the unrestricted case.*

3 Foundations of Separability

We start with recalling the notion of separability and discuss some of its basic properties.

► **Definition 3.** *Assume a subset \mathcal{L} of all $\mu\mathbf{ML}$ formulae. Given $\varphi, \varphi' \in \mu\mathbf{ML}$, an \mathcal{L} -separator of φ, φ' is a formula $\psi \in \mathcal{L}$ with $\varphi \models \psi$ and $\psi \models \neg\varphi'$. If additionally $\text{sig}(\psi) \subseteq \sigma$ for some signature σ , ψ is called an \mathcal{L}_{σ} -separator.*

The \mathcal{L} -separability problem is to determine, given formulae $\varphi, \varphi' \in \mu\text{ML}$ and a signature σ , if they admit an \mathcal{L}_σ -separator ψ . \mathcal{L} -definability is the special case of \mathcal{L} -separability in which $\varphi' = \neg\varphi$, since an \mathcal{L} -separator of $\varphi, \neg\varphi$ is equivalent to φ . All notions can be relativized to a class \mathbf{C} of models by considering entailment over that class. We investigate ML-separability and ML-definability over different classes of models. The reader may have expected the problems to be defined without restrictions on σ , but in fact such versions of the problems are special instances of our problems with $\sigma = \text{sig}(\varphi) \cup \text{sig}(\varphi')$. Conversely, all lower bounds already hold for such special instances.

We start with observing that, by the tree model property and the finite model property of μML , ψ is an ML_σ -separator of φ, φ' (over all models) iff ψ is an ML_σ -separator of φ, φ' over trees iff ψ is an ML_σ -separator of φ, φ' over finite models. Thus, separability coincides over all these classes. Moreover, with the help of the μML -formula $\theta_\infty = \nu x. \Diamond x$ expressing the existence of an infinite path originating in the root, ML-separability over finite trees reduces to ML-separability over all models. More formally:

► **Lemma 4.** *Let $\varphi, \varphi' \in \mu\text{ML}$ and $\psi \in \text{ML}$. Then ψ is an ML_σ -separator of φ, φ' over finite trees iff ψ is an ML_σ -separator of $\varphi \wedge \neg\theta_\infty, \varphi' \wedge \neg\theta_\infty$. This is also true inside \mathbb{T}^d , for $d \in \mathbb{N}$.*

This lemma allows us to transfer all upper bounds obtained in the paper also to the restrictions of the classes to finite models. The lower bounds do not follow from this lemma, but analyzing the proofs yields that they actually work as well. Thus, in the rest of the paper we focus on the classes of all models and \mathbb{T}^d , for $d \in \mathbb{N}$.

The starting point for the technical developments in the paper are model-theoretic characterizations for separability. Similar to what has been done in the context of interpolation, see for example [28], they are given in terms of joint consistency, which we introduce next. Let R be a binary relation on some class of models, such as (σ, n) -isomorphism \cong_σ^n or ML_σ^n -equivalence $\equiv_{\text{ML}_\sigma^n}$. We call two formulae φ, φ' *joint consistent up to R* (in short *joint R -consistent*) if there are models $\mathcal{M} \models \varphi$ and $\mathcal{M}' \models \varphi'$ with $R(\mathcal{M}, \mathcal{M}')$. For technical reasons we will sometimes also talk about joint consistency of automata $\mathcal{A}, \mathcal{A}'$ in place of formulae φ, φ' . Joint R -consistency over a class \mathbf{C} of models is defined by only looking at models from \mathbf{C} . Clearly, if $R' \subseteq R$ and $\mathbf{C}' \subseteq \mathbf{C}$ then joint R' -consistency over \mathbf{C}' implies joint R -consistency over \mathbf{C} . We use the following standard equivalence:

$$\varphi, \varphi' \text{ are not } \text{ML}_\sigma^n\text{-separable over } \mathbf{C} \iff \varphi, \varphi' \text{ are joint } \simeq_\sigma^n\text{-consistent over } \mathbf{C}. \quad (\text{Base})$$

for every $\varphi, \varphi' \in \mu\text{ML}$, $n \in \mathbb{N}$, finite σ , and class \mathbf{C} . The implication from right to left is immediate. The opposite one follows from the observation that for every $n \in \mathbb{N}$ and finite σ there are only finitely many equivalence classes of \simeq_σ^n , and each such class is fully described with a single modal formula.

Let us illustrate how Equivalence (Base) is used to solve ML-separability. Let φ_1 and φ_2 be μML -formulae expressing the respective properties P_1 and P_2 from Example 1. Let \mathcal{M} be an infinite path in which every point satisfies a , and let \mathcal{M}_n be a finite path of length n in which every point satisfies a . Then, for each n the models $\mathcal{M}, \mathcal{M}_n$ witness joint \simeq^n -consistency of φ_1, φ_2 . By Equivalence (Base) this means that φ_1, φ_2 are not ML^n -separable for any n , and thus not ML-separable at all.

Definability is a special case of separability. Since the tools used for solving definability are a starting point for our work, we recall them now.

Modal Definability: A Recap

In his seminal paper [24] Otto showed that ML-definability of μML -formulae is EXPTIME -complete over all models and over \mathbb{T}^d for every $d \geq 2$.

► **Theorem 5** ([24, Main Theorem and Proposition 5]). *Over the class of all models, as well as over \mathbb{T}^d for every $d \geq 2$, ML-definability of μML -formulae is EXPTIME -complete.*

We start by recalling and rephrasing Otto’s construction and fixing a small mistake in the original proof. The lower bound follows by an immediate reduction from satisfiability of μML -formulae. We look at the upper bound. The first step is the following lemma, which is the heart of [24, Lemma 2].

► **Lemma 6.** *For every $\varphi \in \mu\text{ML}$ and $n, d \in \mathbb{N}$ the following are equivalent:*

1. $\varphi, \neg\varphi$ are joint \equiv_σ^n -consistent over \mathbb{T}^d .
2. $\varphi, \neg\varphi$ are joint \cong_σ^n -consistent over \mathbb{T}^d .

The lemma is true, but its proof in [24] is mistaken. The problem there is that the construction duplicates subtrees and hence may turn d -ary models into ones with outdegree greater than d . We present an easy alternative proof.

Proof. Only the implication $1 \Rightarrow 2$ is nontrivial. To prove it assume d -ary $\mathcal{M} \models \varphi, \mathcal{N} \models \neg\varphi$ with $\mathcal{M} \equiv_\sigma^n \mathcal{N}$ and assume towards contradiction that $\varphi, \neg\varphi$ are not \cong_σ^n -consistent over \mathbb{T}^d . We have $\mathcal{M} \cong_\sigma^n \mathcal{M}_{|n}^\sigma \equiv \mathcal{M}'$ where \mathcal{M}^σ is the σ -reduct of \mathcal{M} , and $\mathcal{M}' \in \mathbb{T}^d$ is the bisimilarity quotient of its n -prefix $\mathcal{M}_{|n}^\sigma$. By the assumption that $\varphi, \neg\varphi$ are not joint \cong_σ^n -consistent, $\mathcal{M} \models \varphi$ implies $\mathcal{M}_{|n}^\sigma \models \varphi$. By invariance of φ under \equiv , this in turn implies $\mathcal{M}' \models \varphi$. We construct $\mathcal{N}' \models \neg\varphi$ symmetrically. By definition, $\mathcal{M} \equiv_\sigma^n \mathcal{N}$ means that $\mathcal{M}_{|n}^\sigma$ and $\mathcal{N}_{|n}^\sigma$ are bisimilar, which is equivalent to saying that their bisimilarity quotients \mathcal{M}' and \mathcal{N}' are isomorphic, and hence (σ, n) -isomorphic. Thus, $\mathcal{M}', \mathcal{N}'$ witness joint \cong_σ^n -consistency of $\varphi, \neg\varphi$ over \mathbb{T}^d , a contradiction. ◀

Using automata-based techniques we to decide if Item 2 in Lemma 6 holds for all n .

► **Proposition 7.** *For every parity automata $\mathcal{A}, \mathcal{A}'$ and $d \in \mathbb{N}$: $\mathcal{A}, \mathcal{A}'$ are joint \cong_σ^n -consistent over \mathbb{T}^d for all $n \in \mathbb{N}$ iff $\mathcal{A}, \mathcal{A}'$ are joint \cong_σ^m -consistent over \mathbb{T}^d for $m = |\mathcal{A}| + |\mathcal{A}'| + 1$. The latter condition can be checked in time polynomial in $|\mathcal{A}| + |\mathcal{A}'|$.*

Proof (Sketch). Due to well-known relativization techniques we do not lose generality by only running $\mathcal{A}, \mathcal{A}'$ on full d -ary trees with no leaves. Let L be a language of finite full d -ary trees over σ such that $\mathcal{M} \in L$ iff \mathcal{M} is a prefix of a reduct of a model of \mathcal{A} . Let L' be an analogous language for \mathcal{A}' . The *tallness* of a finite tree is the minimal distance from the root to a leaf. Observe that $\mathcal{A}, \mathcal{A}'$ are \cong_σ^n -consistent over \mathbb{T}^d iff $L \cap L'$ contains a tree of tallness n . Thus, it suffices to check if $L \cap L'$ contains trees of arbitrarily high tallness. To that end construct an automaton \mathcal{B} recognizing $L \cap L'$ of size polynomial in $|\mathcal{A}| + |\mathcal{A}'|$. An easy pumping argument shows that the language $L \cap L'$ of \mathcal{B} contains trees of arbitrarily high tallness iff it contains a tree of tallness $m = |\mathcal{B}| + 1$. To test the latter condition it is enough to inductively compute a sequence $S_1 \supseteq S_2 \supseteq \dots \supseteq S_{|\mathcal{B}|+1}$ of subsets of states of \mathcal{B} , where S_i is the set of all states q such that $\mathcal{B}[q_I \leftarrow q]$ recognizes a tree of tallness at least i . ◀

We are ready to solve ML-definability over \mathbb{T}^d in exponential time. Assume μML -formula φ . For every n , we know by Equivalence (Base) that φ is equivalent over \mathbb{T}^d to some $\psi \in \text{ML}_\sigma^n$ iff $\varphi, \neg\varphi$ are not joint \equiv_σ^n -consistent over \mathbb{T}^d . By Lemma 6 this is equivalent to the lack of joint \cong_σ^n -consistency of $\varphi, \neg\varphi$ over \mathbb{T}^d . By Theorem 2 we can compute exponentially-sized

automata \mathcal{A} , \mathcal{A}' equivalent to φ and $\neg\varphi$ over \mathbb{T}^d . It follows that φ is not ML_σ -definable over \mathbb{T}^d iff $\mathcal{A}, \mathcal{A}'$ are joint \cong_σ^n -consistent over \mathbb{T}^d for every n . The last condition is decided using Proposition 7. The runtime of our algorithm is polynomial in $|\mathcal{A}| + |\mathcal{A}'|$, and thus exponential in $|\varphi|$. This proves the part of Theorem 5 about \mathbb{T}^d . The remaining part concerning unrestricted models is a special case of Theorem 9, which we will prove next.

Modal Separation: the Unrestricted Case

Over unrestricted models, separability turns out to be only slightly more complicated than definability. Lemma 6 becomes false if $\neg\varphi$ is replaced with arbitrary φ' (which would be the statement relevant for separability). We have the following lemma, however.

► **Lemma 8.** *For every $\varphi, \varphi' \in \mu\text{ML}$ and $n \in \mathbb{N}$ the following are equivalent:*

1. φ, φ' are joint \cong_σ^n -consistent over all models.
2. φ, φ' are joint \cong_σ^n -consistent over \mathbb{T}^d , where $d = |\varphi| + |\varphi'|$.

Proof. The implication (1) \Leftarrow (2) is immediate. To prove the other one (1) \Rightarrow (2) consider an intermediate property:

$$\varphi, \varphi' \text{ are joint } \cong_\sigma^n\text{-consistent over all models.} \quad (1.5)$$

The implication (1) \Rightarrow (1.5) can be read off from Otto's original proof. The remaining one (1.5) \Rightarrow (2) is a special case of a stronger claim which we prove later: the implication (3) \Rightarrow (4) of Lemma 27. ◀

Lemma 8 allows us to solve ML-separability in exponential time.

► **Theorem 9.** *Over all models, ML-separability of μML -formulae is EXPTIME-complete.*

Proof. The proof is almost the same as our proof of Theorem 5. The only difference is that we consider an arbitrary φ' in place of $\neg\varphi$, and hence use Lemma 8 in place of Lemma 6. ◀

Apart from deciding separability we also construct separators when they exist. Given a subset \mathcal{L} of μML formulae, $\varphi \in \mu\text{ML}$, and $\psi \in \mathcal{L}$, we call ψ an \mathcal{L} -uniform consequence of φ if $\psi \models \theta$ for every $\theta \in \mathcal{L}$ such that $\varphi \models \theta$. The notion relativizes to a fixed class \mathbf{C} of models by only considering entailment over that class. Observe that if φ, φ' are \mathcal{L} -separable and ψ is an \mathcal{L} -uniform consequence of φ then ψ is an \mathcal{L} -separator for φ, φ' . The same is true over any class \mathbf{C} .

Note that it follows from the proof of Theorem 9 that if φ, φ' are ML-separable then they admit a separator of modal depth n at most exponential in $|\varphi| + |\varphi'|$. It follows that constructing an ML_σ -separator for φ, φ' boils down to constructing an ML_σ^n -uniform consequence of φ . A naive construction which always works is to take the disjunction of all ML_σ^n -types consistent with φ over \mathbf{C} . Here, by an ML_σ^n -type we mean a maximal consistent subset of ML_σ^n . Since up to equivalence there are only finitely many formulae in ML_σ^n , each ML_σ^n -type can be represented as a single ML_σ^n -formula and the mentioned disjunction ψ is well-defined. This construction is non-elementary in n over all models and doubly exponential in n over models of bounded outdegree.

We present an efficient construction of ML_σ^n -uniform consequences. The construction works over unrestricted models, over \mathbb{T}^1 and over \mathbb{T}^2 but not over \mathbb{T}^d for $d \geq 3$. Since in the following Section 4 we will provide a more efficient construction for \mathbb{T}^1 , now we only look at the unrestricted and binary case. For convenience, we construct ML_σ^n -uniform consequences of automata instead of formulae, with definition adapted in an obvious way.

► **Proposition 10.** *Let \mathbf{C} be the class of all models or \mathbb{T}^2 . Assume an NPTA \mathcal{A} over \mathbf{C} , a signature σ and $n \in \mathbb{N}$. An ML_σ^n -uniform consequence of \mathcal{A} over \mathbf{C} can be constructed in time $|\mathcal{A}|^{O(n \cdot |\mathcal{A}|)}$ if \mathbf{C} is the class of all models and in time $2^{O(n \cdot |\mathcal{A}|)}$ if $\mathbf{C} = \mathbb{T}^2$.*

Proof. Let \mathcal{A} be an NPTA. Let $\mathcal{B} = (Q, \Sigma, q_I, \delta, \text{rank})$ be an automaton of the same size recognizing σ -reducts of models of \mathcal{A} . A formula ψ is an ML_σ^n -uniform consequence of \mathcal{A} over \mathbf{C} iff it is an ML^n -uniform consequence of \mathcal{B} over \mathbf{C} . Thus, it suffices to construct the latter.

We construct $\psi_{n,q}$ for every $q \in Q$ and $n \in \mathbb{N}$ by induction on $n \in \mathbb{N}$. For the base case we put:

$$\psi_{0,q} = \bigvee \{c \in \Sigma \mid \text{there is } \mathcal{N} \in \mathbf{C} \text{ with } \mathcal{N} \models \mathcal{B}[q_I \leftrightarrow q] \text{ and } \mathcal{N} \models c\}$$

For the induction step define:

$$\psi_{n+1,q} = \bigvee_{c \in \Sigma} \bigvee_{S \in \delta(q,c)} c \wedge \nabla \{\psi_{n,p} \mid p \in S\}$$

where $\nabla \Phi$ is an abbreviation for $\bigwedge_{\theta \in \Phi} \diamond \theta \wedge \bigvee_{\theta \in \Phi} \theta$. Assume \mathbf{C} is either the class of all models or \mathbb{T}^2 . The construction preserves the following invariant:

$$\mathcal{M} \models \psi_{n,q} \iff \text{there exists } \mathcal{N} \in \mathbf{C} \text{ with } \mathcal{N} \models \mathcal{B}[q_I \leftrightarrow q] \text{ and } \mathcal{M} \equiv^n \mathcal{N} \quad (1)$$

for every structure $\mathcal{M} \in \mathbf{C}$. Hence, ψ_{n,q_I} is an ML_σ^n -uniform consequence of \mathcal{A} over \mathbf{C} . It is routine to check that in either case the formula has the right size.

The proof of (1) proceeds by an easy induction, with slightly different details for the cases of binary and of unrestricted models. It is worth to point out, however, that the implication \Rightarrow from left to right would not be valid over \mathbb{T}^d with $d \geq 3$. ◀

Given the exponential construction of automata from Theorem 2 and the exponential upper bound on modal depth n of separators, Proposition 10 yields an efficient construction of separators.

► **Theorem 11.** *If φ, φ' are ML_σ -separable, then one can compute an ML_σ -separator in time doubly exponential in $|\varphi| + |\varphi'|$.*

It is not difficult to show that, in the presence of at least two accessibility relations \diamond_1, \diamond_2 , the construction is optimal: one can express in μML that the model embeds a full binary tree of depth 2^n and in which each inner node has both a \diamond_1 - and a \diamond_2 -successor. Using standard techniques, one can show that any modal formula expressing this property is of doubly exponential size [13]. Whether having two accessibility relations is necessary for this lower bound is an interesting question which we leave open.

It is interesting to note that the separators we compute are *not* the logically strongest separators and, in fact, strongest separators do not even have to exist.

► **Example 12.** Consider $\varphi = \theta_\infty$ from before and $\varphi = \Box \perp$. For every $n \in \mathbb{N}$, the modal formula $\diamond^n \top$ separates φ from φ' , and $\diamond^m \top \models \diamond^n \top$ whenever $m \geq n$.

The remaining open cases are the problems of ML -separability (and separator construction) over \mathbb{T}^d for $d \geq 1$. We investigate the cases of unary ($d = 1$), binary ($d = 2$), and higher maximal outdegree ($d \geq 3$) in turn. We emphasize that the outdegree d is not a part of the input but rather a property of the considered class of models.

4

 Unary Case

We first investigate ML-separability over \mathbb{T}^1 , that is, models that are essentially *words*. Note that satisfiability of μML over words is PSPACE-complete (an upper bound follows, e.g., via the translation to automata and the lower bound is inherited from LTL [30, Theorem 4.1]) which suggests that also definability and separability could be easier. Indeed, we show:

► **Theorem 13.** *ML-definability and ML-separability of μML -formulae is PSPACE-complete over \mathbb{T}^1 .*

Proof. The lower bound is by a reduction from satisfiability, and applies to definability.

Given formulae $\varphi, \varphi' \in \mu\text{ML}$ and a subset of the signature σ , consider the set of finite words $L = \{W \in \mathcal{P}(\sigma)^* \mid W \text{ is a } \sigma\text{-reduct of a prefix } V \text{ of some model } U \text{ of } \varphi\}$. Let L' be a similar language defined for φ' . Two unary models are bisimilar iff they are identical. Hence, by Equivalence (Base) the formulae $\varphi, \varphi' \in \mu\text{ML}$ are not ML_σ -separable over \mathbb{T}^1 iff $L \cap L'$ is infinite. It is standard to define a finite automaton \mathcal{A} recognizing $L \cap L'$ and check if its language is infinite (which is equivalent to checking if $L \cap L'$ contains input longer than $|\mathcal{A}|$). To do it in polynomial space, we nondeterministically guess the long input, letter by letter, and only remember the current state and a binary counter measuring the length of the input guessed so far. ◀

We conclude this section with proving that ML_σ -separators can be constructed in exponential time and are thus of at most exponential size. Note that this is optimal, since over \mathbb{T}^1 , μML is exponentially more succinct than ML. Indeed, it is standard to implement an exponential counter using a polynomially sized μML -formula.

► **Theorem 14.** *If $\varphi, \varphi' \in \mu\text{ML}$ are ML_σ -separable over \mathbb{T}^1 , then one can compute an ML_σ -separator in time exponential in $|\varphi| + |\varphi'|$.*

As argued in the previous section, it suffices to construct an ML^n -uniform consequence of the NPTA equivalent to φ , which we do next.

► **Proposition 15.** *Let \mathcal{A} be an NPTA over \mathbb{T}^1 with ℓ states, $n \in \mathbb{N}$, and σ a signature. An ML_σ^n -uniform consequence of \mathcal{A} over \mathbb{T}^1 can be constructed in time polynomial in n , σ , and ℓ .*

Proof. As argued in the previous section, it suffices to construct an ML^n -uniform consequence of the NPTA \mathcal{B} which recognizes precisely the σ -reducts of models of \mathcal{A} . Let \mathcal{B} have states Q . By construction of \mathcal{B} , we have $|Q| = \ell$. As an auxiliary step, we define for every $p, q \in Q$ and $m \leq n$ a formula $\psi_{p,q}^m \in \text{ML}_\sigma^n$ such that for every $\mathcal{M} \in \mathbb{T}^1$:

$$\mathcal{M} \models \psi_{p,q}^m \iff \text{there is a run of } \mathcal{B} \text{ from } p \text{ to } q \text{ over the } m\text{-prefix of } \mathcal{M}. \quad (2)$$

The ψ_{pq}^m are defined inductively with the base cases ($m \leq 1$) read off from \mathcal{B} , and using *divide and conquer* in the inductive step ($m > 1$), to keep the formulae small. More formally, we define ψ_{pq}^m for $m > 1$ and all $p, q \in Q$ by taking:

$$\psi_{pq}^m = \bigvee_{q' \in Q} (\psi_{pq'}^{\lfloor m/2 \rfloor} \wedge \diamond^{\lfloor m/2 \rfloor} \psi_{q'q}^{\lceil m/2 \rceil})$$

It is routine to verify that ψ_{pq}^m satisfies (2) and is of size $|\psi_{pq}^m| \in O(|Q| \cdot m^2)$. Based on the ψ_{pq}^m , one can define a formula ψ_n that describes all possible prefixes of length $\leq n$ of models of \mathcal{B} , and thus is the sought ML_σ -uniform consequence of \mathcal{B} . One can think of ψ_n as the disjunction of formulae $\psi_{q_0q}^n$ for q_0 the initial state of \mathcal{B} , but the full construction is slightly more involved since models accepted by \mathcal{B} might be also shorter than n . ◀

5 Binary Case

We next handle the binary case \mathbb{T}^2 . The key observation here is that, between full binary trees, bisimilarity entails isomorphism.

► **Proposition 16.** *Assume full binary trees $\mathcal{M}, \mathcal{M}' \in \mathbb{T}^2$. If \mathcal{M} and \mathcal{M}' are σ -bisimilar then they are σ -isomorphic.*

Proof. By definition a σ -bisimulation between two models is a bisimulation between their reducts to σ , and σ -isomorphism is such a bisimulation which is additionally bijective. It therefore suffices to show that if $\mathcal{M}, \mathcal{M}'$ are full binary trees and Z is a bisimulation between them then there is a bijective bisimulation $Z' \subseteq Z$. We pick such Z' inductively starting with the pair of roots (v_I, v'_I) . The key observation is that if v has children v_1, v_2 and w has children w_1, w_2 and vZw then either (i) v_1Zw_1 and v_2Zw_2 or (ii) v_1Zw_2 and v_2Zw_1 (the cases are not exclusive). ◀

Proposition 16 can be used to prove the Craig interpolation property of ML over \mathbb{T}^2 and implies the following separability-variant of Lemma 6 over \mathbb{T}^2 .

► **Lemma 17.** *For every $\varphi, \varphi' \in \mu\text{ML}$ and $n \in \mathbb{N}$ the following are equivalent:*

1. φ, φ' are joint $\stackrel{n}{\equiv}_\sigma$ -consistent over \mathbb{T}^2 .
2. φ, φ' are joint \cong_σ^n -consistent over \mathbb{T}^2 .

Proof. We show only the nontrivial implication $1 \Rightarrow 2$. Assume binary $\mathcal{M} \models \varphi$, $\mathcal{M}' \models \varphi'$ with $\mathcal{M} \stackrel{n}{\equiv}_\sigma \mathcal{M}'$. Let $\mathcal{N} \models \varphi$ and $\mathcal{N}' \models \varphi'$ be full binary trees obtained from \mathcal{M} and \mathcal{M}' by duplicating subtrees. By Proposition 16, $\mathcal{N} \cong_\sigma^n \mathcal{N}'$ which proves 2. ◀

Similarly to the definability case, Lemma 17 combined with Equivalence (Base) and Proposition 7 immediately give an exponential procedure for separability. Since the lower bound is inherited from definability, we get the following result.

► **Theorem 18.** *ML-separability and ML-definability of μML -formulae is EXP TIME -complete over \mathbb{T}^2 .*

With the same argument as for Theorem 11 we use Proposition 10 to conclude:

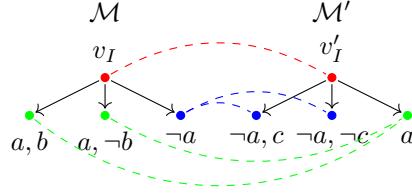
► **Theorem 19.** *If φ, φ' are ML_σ -separable over \mathbb{T}^2 , then one can compute an ML_σ -separator in time doubly exponential in $|\varphi| + |\varphi'|$.*

6 Ternary and Beyond

In this section we address the case of models with outdegree bounded by a number $d \geq 3$. We illustrate that this case behaves differently as it lacks the Craig interpolation property.

► **Example 20.** Consider ML-formulae $\varphi = \diamond(a \wedge b) \wedge \diamond(a \wedge \neg b)$ and $\varphi' = \diamond(\neg a \wedge c) \wedge \diamond(\neg a \wedge \neg c)$. Clearly, $\varphi \models \neg \varphi'$ over \mathbb{T}^3 . Observe that models $\mathcal{M}, \mathcal{M}'$ in Figure 1 witness that φ, φ' are joint $\equiv_{\{a\}}$ -consistent and thus joint $\equiv_{\{a\}}^n$ -consistent for every $n \in \mathbb{N}$. By Equivalence (Base) there is no $\text{ML}_{\{a\}}$ -separator, which is nothing else than a Craig interpolant. ◀

Motivated by the lack of the Craig interpolation property, we study the ML-*interpolant existence* problem: given $\varphi, \varphi' \in \text{ML}$ and signature σ , decide whether there is an ML_σ -separator of $\varphi, \neg \varphi'$, that is, $\psi \in \text{ML}_\sigma$ with $\varphi \models \psi \models \varphi'$. *Craig ML-*interpolant existence** is the special case in which $\sigma = \text{sig}(\varphi) \cap \text{sig}(\varphi')$. Observe that ML-*interpolant existence* is the



■ **Figure 1** Witness of joint consistency: dashed lines and colors indicate the $\{a\}$ -bisimulation.

special case of ML-separability of μ ML-formulae in which the input to the separability is restricted to ML-formulae. We show that already ML-interpolant existence over \mathbb{T}^3 is harder than ML-separability of μ ML-formulae over arbitrary models.

► **Theorem 21.** *For $d \geq 3$, ML-interpolant existence over \mathbb{T}^d is CONEXPTIME-complete. Hardness already applies to Craig ML-interpolant existence over \mathbb{T}^d .*

Proof. The upper bound is easy to establish based on the observation that $\varphi, \neg\varphi'$ of modal depth at most m do not admit an ML_σ -separator over \mathbb{T}^d iff they are joint \equiv_σ^m -consistent over \mathbb{T}^d . The witness $\mathcal{M}, \mathcal{M}'$ of joint \equiv_σ^m -consistency of $\varphi, \neg\varphi'$ can assumed to be of depth m . Such models are of exponential size (they have at most d^m points) and can thus be guessed by a non-deterministic exponential time bounded Turing machine.

The lower bound is more intriguing and relies on an extension of Example 20. Reconsidering the example it is important to note that in *every* witness $\mathcal{M}, \mathcal{M}'$ of joint $\equiv_{\{a\}}^n$ -consistency of φ, φ' , there are two successors of v_I that are bisimilar to the same successor of v'_I . We extend the idea and enforce exponentially many bisimilar points. More precisely, consider families $(\psi_i)_{i \in \mathbb{N}}, (\psi'_i)_{i \in \mathbb{N}}$ of modal formulae inductively defined as follows:

$$\begin{aligned} \psi_0 &= \psi'_0 = \top \\ \psi_{i+1} &= \diamond(a \wedge b_i) \wedge \diamond(a \wedge \neg b_i) \wedge \square(a \rightarrow (\psi_i \wedge (b_i \rightarrow \bigwedge_{j < i} \square^j b_i) \wedge (\neg b_i \rightarrow \bigwedge_{j < i} \square^j \neg b_i))) \\ \psi'_{i+1} &= \diamond(\neg a \wedge c) \wedge \diamond(\neg a \wedge \neg c) \wedge \diamond(a \wedge \psi'_i) \end{aligned}$$

Clearly, the size of ψ_i, ψ'_i is polynomial in i . Moreover, by induction on i , it is readily verified that for every $i \in \mathbb{N}$, for every $\mathcal{M}, \mathcal{M}' \in \mathbb{T}^3$ with $\mathcal{M} \models \psi_i, \mathcal{M}' \models \psi'_i$, and every $(\{a\}, i)$ -bisimulation S witnessing $\mathcal{M} \equiv_{\{a\}}^i \mathcal{M}'$, there are points w_0, \dots, w_{2^i-1} in depth i in \mathcal{M} and a point \hat{w} in depth i in \mathcal{M}' such that $(w_j, \hat{w}) \in S$ for all j and such that distinct w_j, w_k can be distinguished by some proposition in b_0, \dots, b_{i-1} . Intuitively, this means that ψ_i, ψ'_i enforce in joint $\equiv_{\{a\}}^i$ -consistent models $\mathcal{M}, \mathcal{M}'$ that \mathcal{M} contains 2^i points w_0, \dots, w_{2^i-1} which are all linked to the same point \hat{w} in \mathcal{M}' . We exploit this link to synchronize information between the w_j , following a strategy that has recently been used to show CONEXPTIME-hardness for interpolant existence in some description logics [1].

We reduce a NEXPTIME-complete tiling problem [14]: Given a set Δ of tile types and horizontal and vertical compatibility relations $H, V \subseteq \Delta \times \Delta$, and some $n \in \mathbb{N}$ in unary, decide whether one can tile the $2^n \times 2^n$ torus with tiles from Δ complying with H, V . Given Δ, H, V, n , we define formulae $\varphi_n = \psi_{2n} \wedge \square^{2n} \chi_n, \varphi'_n = \psi'_{2n} \wedge \square^{2n} \chi'_n$ of modal depth m and with common signature $\sigma = \text{sig}(\varphi_n) \cap \text{sig}(\varphi'_n)$ such that

$$\Delta, H, V, n \text{ has a solution} \quad \Leftrightarrow \quad \varphi_n, \varphi'_n \text{ are joint } \equiv_\sigma^m\text{-consistent.}$$

To explain the idea, let $\mathcal{M}, \mathcal{M}'$ witness joint \equiv_σ^m -consistency of φ_n, φ'_n . The gadget formulae ψ_{2n}, ψ'_{2n} enforce 2^{2n} points $w_0, \dots, w_{2^{2n}-1}$ in depth $2n$ in \mathcal{M} which are all linked via the bisimulation to a single point \hat{w} in \mathcal{M}' . These 2^{2n} points shall represent the $2^n \times 2^n$ cells

of the torus. The intended solution of the tiling problem is represented via propositions $p_d \in \sigma$, for each $d \in \Delta$. To synchronize them we proceed as follows. Using the $2n$ propositions b_0, \dots, b_{2n-1} (which are not in σ), we can associate coordinates $(x_i, y_i) \in \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^n - 1\}$ to each point w_i in the torus. To understand the purpose of χ_n, χ'_n , suppose for a moment that the outdegree of the points \hat{w} and the w_i is at most 2^{2n} (instead of 3). Then we could proceed by enforcing (via χ_n) below each w_i with coordinates (x_i, y_i) three successors v_i^1, v_i^2, v_i^3 such that

- v_i^1, v_i^2, v_i^3 have coordinates $(x_i, y_i), (x_i, y_i + 1)$, and $(x_i + 1, y_i)$, respectively;
- the coordinates of the v_i^j are made visible using propositions in σ ;
- v_i^1, v_i^2, v_i^3 satisfy $p_{d_1}, p_{d_2}, p_{d_3}$ for $d_1, d_2, d_3 \in \Delta$ such that $(d_1, d_2) \in V$ and $(d_1, d_3) \in H$.

These three successors stipulate bisimilar successors of \hat{w} . Since each point in the torus is stipulated three times as successor of some w_i and since the outdegree of \hat{w} is restricted to 2^{2n} , the three copies of the same point satisfy the same proposition p_d . By the last item above, the selected propositions comply with V, H and thus represent a solution to the tiling problem. Now, since the outdegree below \hat{w} is at most 3 (and not 2^{2n} as assumed), the χ_n, χ'_n have to be a bit more complicated, but the idea remains the same. ◀

We show next that the situation for the full separability problem is even worse.

► **Theorem 22.** *For every $d \geq 3$, ML-separability of μ ML-formulae over \mathbb{T}^d is 2-EXPTIME-complete.*

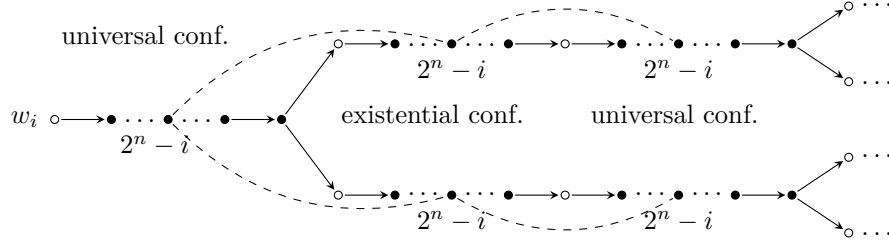
Thus, over \mathbb{T}^d for $d \geq 3$, ML-separability is provably harder than ML-definability, c.f. Theorem 5. Both the upper and the lower bound of Theorem 22 are non-trivial; we provide proof sketches in the following two subsections. Before doing that let us conclude this part with separator construction.

► **Theorem 23.** *If φ, φ' are ML_σ -separable over \mathbb{T}^d , $d \geq 3$, then one can compute an ML_σ -separator in time triply exponential in $|\varphi| + |\varphi'|$.*

Proof. (Sketch) It follows from the upper bound proof of Theorem 22 that, if φ, φ' admit an ML_σ -separator, then they admit one of modal depth bounded doubly exponentially in $|\varphi| + |\varphi'|$. Observe that over the signature of φ and φ' there are only triple exponentially many trees of fixed outdegree d and double exponential depth, and that each such tree is characterized by a modal formula of triply exponential size. The sought separator is then the disjunction of all such formulae consistent with φ . ◀

6.1 Lower Bound for Theorem 22

We reduce the word problem of exponentially space bounded alternating Turing machines (ATMs), which is known to be 2-EXPTIME-complete [8]. Informally, the states of such ATMs are partitioned into *universal states* Q_\forall and *existential states* Q_\exists . Configurations of ATMs are defined as usual, but computations are not sequences of configurations but trees of configurations such that an existential configuration has exactly one successor labeled with a universal configuration and a universal configuration has exactly two successors labeled with existential configurations. A computation tree for an input w is a tree whose root is labeled with the initial configuration and such that successor nodes contain successor configurations. w is accepted if there is a computation tree in which each path is infinite (this acceptance condition is slightly non-standard, but eases the proof).



■ **Figure 2** Computation tree of \mathfrak{A} below some w_i (drawn horizontally for space constraints).

The reduction relies on the same gadget formulae $(\psi_i)_{i \in \mathbb{N}}, (\psi'_i)_{i \in \mathbb{N}}$ as used in the proof of Theorem 21 and additionally uses ideas for showing 2-EXPTIME-hardness for recently studied interpolant existence problems for description logics [1]. For a given ATM \mathfrak{A} and input w of length n , we construct formulae $\varphi_n = \psi_n \wedge \Box^n \chi, \varphi'_n = \psi'_n \wedge \chi'$ such that

$$\varphi_n, \varphi'_n \text{ are joint } \Leftrightarrow_{\sigma}^m\text{-consistent for every } m \in \mathbb{N} \quad \text{iff} \quad \mathfrak{A} \text{ accepts } w.$$

This suffices by Equivalence (Base). The signature σ will consist of a, z , and propositions c_α for every possible cell content α of \mathfrak{A} , that is, $\alpha \in \Gamma \cup (Q \times \Gamma)$. Additionally, φ_n and φ'_n will use auxiliary propositions, e.g., to encode counters. The only purpose of χ' is to mention the propositions in σ ; the main work is done by ψ_n, ψ'_n, χ .

To explain the idea, let us consider witnesses $\mathcal{M}, \mathcal{M}'$ for joint $\Leftrightarrow_{\sigma}^m$ -consistency of φ_n, φ'_n for sufficiently large m . By the properties of ψ_n, ψ'_n , we find 2^n points w_0, \dots, w_{2^n-1} in depth n in \mathcal{M} which are bisimilar to a single point \hat{w} in depth n in \mathcal{M}' . Recall that in every w_i , we have access to its index i via a counter using propositions b_0, \dots, b_n . Now, χ is a μ ML-formula with the following properties, see also Figure 2 for illustration.

- χ enforces the “skeleton” of a computation tree for \mathfrak{A} , in which each configuration is modeled by a path of length 2^n (using an exponential counter), and in which universal and existential configurations alternate.
- χ also enforces that each point of the skeleton is labeled with some cell content via σ -propositions c_α , but without any synchronization except the initial configuration.
- χ makes sure that below w_i the positions $2^n - i$ of successor configurations are coordinated. The key point is that this enforces (due to bisimilarity) a computation tree below \hat{w} in which, due to the last item above, *all* positions of configurations are coordinated.

We remark that the hardness also holds when σ is not part of the input: one can reduce separability of φ, φ' by ML_{σ} -formulae to separability of φ, φ' by (arbitrary) ML-formulae.

6.2 Upper Bound for Theorem 22

We show that over models of outdegree at most d , ML-separability of fixpoint formulae can be solved in doubly exponential time. Let us start with establishing a technical but useful fact. For every language of d -ary trees $L \subseteq \mathbb{T}^d$ denote the language:

$$\text{bisQuot}(L) = \{\mathcal{M} \in \mathbb{T}^d \mid \text{there is } \mathcal{N} \in L \text{ and a functional bisimulation } Z : \mathcal{N} \xrightarrow{\text{bis}} \mathcal{M}\}$$

of bisimulation quotients of trees from L .

► **Proposition 24.** *For every NPTA \mathcal{A} , an NPTA \mathcal{B} recognizing $\text{bisQuot}(\mathcal{L}(\mathcal{A}))$ can be computed in time exponential in the size of \mathcal{A} .*

Proof. Fix an NPTA $\mathcal{A} = (Q, \Sigma, q_I, \delta, \text{rank})$. For every $\mathcal{M} \in \mathbb{T}^d$, we characterize existence of d -ary $\mathcal{N} \models \mathcal{A}$ with $\mathcal{N} \xrightarrow{\text{bis}} \mathcal{M}$ with the following parity game $\mathcal{G}_{\text{bisQuot}}(\mathcal{M}, \mathcal{A})$. The game has the set $M \times Q$ as positions. The pair (v_I, q_I) consisting of the root v_I of \mathcal{M} and q_I is the initial position. From a position (v, q) first \exists ve chooses $S \in \delta(q, \text{val}(c))$ and a surjective map $h : S \rightarrow \{v_1, \dots, v_k\}$ where $\{v_1, \dots, v_k\}$ is the set of children of v . Then \forall dam responds with a choice of $p \in S$ and the next round starts in position $(h(p), p)$. The game is a parity game: the ranks are inherited from \mathcal{A} in the sense that the rank of (v, q) equals $\text{rank}(q)$. It is easy to show that:

$$\exists \text{ve wins } \mathcal{G}_{\text{bisQuot}}(\mathcal{M}, \mathcal{A}) \iff \mathcal{M} \in \text{bisQuot}(\mathcal{L}(\mathcal{A})) \quad (3)$$

for every $\mathcal{M} \in \mathbb{T}^d$. Using (3) we prove Proposition 24. It suffices to construct an automaton \mathcal{B} which accepts \mathcal{M} iff \exists ve wins $\mathcal{G}_{\text{bisQuot}}(\mathcal{M}, \mathcal{A})$. To that end, using standard techniques we encode \exists ve's positional strategies for $\mathcal{G}_{\text{bisQuot}}(\mathcal{M}, \mathcal{A})$ as colorings of \mathcal{M} with $\mathcal{P}(Q \times Q)$ and construct, in time exponential in $|Q|$, an automaton \mathcal{B}^+ recognizing models labelled with such winning positional strategies. We then obtain \mathcal{B} recognizing $\text{bisQuot}(\mathcal{L}(\mathcal{A}))$ by projecting out the additional colors $\mathcal{P}(Q \times Q)$ from \mathcal{B}^+ . \blacktriangleleft

With the help of Proposition 24 we prove Theorem 22. Fix d , μML -formulae φ and φ' and signature σ . By Equivalence (Base), it suffices to check if φ and φ' are jointly \simeq_σ^n -consistent over \mathbb{T}^d for every n . However, unlike with definability or in the binary case, we cannot conclude joint \cong_σ^n -consistency from joint \simeq_σ^n -consistency. Instead, we use Proposition 24 to directly decide joint \simeq_σ^n -consistency for all n . For a language $L \subseteq \mathbb{T}^d$, define the language:

$$\text{QPL}(L) = \{\mathcal{N} \in \mathbb{T}^d \mid \text{there is } \mathcal{M} \in L, \text{ finite prefix } \mathcal{M}_0 \text{ of } \mathcal{M} \text{ and } Z : \mathcal{M}_0 \xrightarrow{\text{bis}} \mathcal{N}\}$$

of finite d -ary trees which are bisimulation quotients of finite prefixes of models from L . By Proposition 24 and the closure properties of parity automata, for every \mathcal{A} one can construct in exponential time an automaton \mathcal{B} recognizing $\text{QPL}(\mathcal{L}(\mathcal{A}))$.

We prove the upper bound from Theorem 22. Using Theorem 2 compute automata $\mathcal{A}, \mathcal{A}'$ accepting σ -reducts of models of φ, φ' . Compute $\mathcal{B}, \mathcal{B}'$ recognizing $\text{QPL}(\mathcal{L}(\mathcal{A}))$ and $\text{QPL}(\mathcal{L}(\mathcal{A}'))$. Recall that any two trees are bisimilar iff they have isomorphic bisimulation quotients. It follows that φ, φ' admit a ML_σ^n -separator over \mathbb{T}^d iff $\mathcal{A}, \mathcal{A}'$ are joint \simeq_σ^n -consistent iff $\mathcal{B}, \mathcal{B}'$ are joint \cong_σ^n consistent. By Proposition 7, the latter condition holds for all $n \in \mathbb{N}$ iff it holds for $n = |\mathcal{B}| + |\mathcal{B}'| + 1$ and this can be tested in time polynomial in $|\mathcal{B}| + |\mathcal{B}'|$. Since $\mathcal{A}, \mathcal{A}'$ are exponential, and $\mathcal{B}, \mathcal{B}'$ are doubly exponential in the size of φ, φ' , this gives the upper bound from Theorem 22.

7 Case Study: Graded Modalities

In this section we apply our techniques and results to the case with *graded modal operators*. Formally, we extend μML with formulae of the shape $\diamond_{\sim g} \psi$ and $\square_{\sim g} \psi$, where $\sim \in \{\leq, \geq\}$ and the *grade* $g \in \mathbb{N}$ is a natural number. Intuitively, $\diamond_{\geq g} \psi$ is true in a point w if w has at least g successors satisfying ψ and dually, $\square_{\leq g} \psi$ is true in w if all but at most g successors satisfy ψ [11, 25]. We denote with grML and $\text{gr}\mu\text{ML}$ the extension of ML and μML , respectively, with such graded modalities. Clearly, for any $d \in \mathbb{N}$, \mathbb{T}^d is $\text{gr}\mu\text{ML}$ -definable by the formula $\theta_d = \nu x. (\diamond_{\leq d} \top \wedge \square x)$, which is an additional motivation to study grML and $\text{gr}\mu\text{ML}$.

Indeed, using the results and techniques from the previous section one can easily prove that ML -separability of $\text{gr}\mu\text{ML}$ -formulae (defined as expected) is 2-EXPTime-complete.

► **Theorem 25.** *ML-separability of gr μ ML-formulae is 2-EXPTIME-complete.*

Proof. For the lower bound, we reduce ML-separability of μ ML-formulae over \mathbb{T}^3 in spirit similar to Lemma 4. Since the former problem is 2-EXPTIME-hard by Theorem 22, the latter is as well. Recall the formula θ_3 defining \mathbb{T}^3 . Then, for any μ ML-formulae φ, φ' and $\psi \in \text{ML}$, we have that ψ is an ML_σ -separator of φ, φ' over \mathbb{T}^3 iff ψ is an ML_σ -separator of $\varphi \wedge \theta_d, \varphi' \wedge \theta_d$.

Towards the upper bound, suppose $\varphi, \varphi' \in \text{gr}\mu\text{ML}$. Using standard arguments, one can show that φ, φ' are ML-separable over all models iff they are ML-separable over \mathbb{T}^d , where $d = g \times (|\varphi| + |\varphi'|)$ and g is the greatest grade occurring in φ, φ' . We then construct NPTA $\mathcal{A}, \mathcal{A}'$ equivalent to φ, φ' over d -ary trees via (an analogue for gr μ ML of) Theorem 2 and proceed with $\mathcal{A}, \mathcal{A}'$ as described in the upper bound proof of Theorem 22. ◀

Interestingly, the problem becomes easier if we allow grades in the separating formula.

► **Theorem 26.** *grML-separability of gr μ ML-formulae is EXPTIME-complete.*

The lower bound follows by the usual reduction from satisfiability. We thus focus on the upper bound. Similarly to the non-graded case, we establish first a model-theoretic characterization, based on the appropriate notion of bisimilarity that characterizes the expressive power of grML [10]. A relation Z between models is a *graded bisimulation* if it satisfies (atom) and *graded* variants of the (back) and (forth) conditions of bisimulations. The graded (forth) condition says that if vZw then for every $k \in \mathbb{N}$ and pairwise different children v_1, \dots, v_k of v , there are pairwise different children w_1, \dots, w_k of w satisfying $v_i Z w_i$ for all $i \leq k$. The graded (back) condition is symmetric. It is a *g-graded bisimulation* if the graded (forth) and (back) conditions need to be satisfied only for $k \leq g$. We denote with $\mathcal{M} \stackrel{\text{grd}}{\equiv} \mathcal{M}'$ (resp., $\mathcal{M} \stackrel{g}{\equiv} \mathcal{M}'$) the fact that there is a graded bisimulation (resp., a g -graded bisimulation) between \mathcal{M} and \mathcal{M}' that relates their roots. Variants with bounded depth n and/or given signature σ are defined and denoted as expected.

► **Lemma 27.** *For every $\varphi, \varphi' \in \text{gr}\mu\text{ML}$ with maximal grade g_{\max} , signature σ , and $n \in \mathbb{N}$, the following are equivalent:*

1. φ, φ' are not grML_σ^n -separable (over all models).
2. φ, φ' are joint $\stackrel{\text{grd}, \sigma}{\equiv}^n$ -consistent (over all models).
3. φ, φ' are joint \cong_σ^n -consistent (over all models).
4. φ, φ' are joint \cong_σ^n -consistent over \mathbb{T}^d for $d = g_{\max} \times (|\varphi| + |\varphi'|)$.

Using Lemma 27, one can solve grML-separability of gr μ ML formulae in exponential time, following the approach described in Section 3. More precisely, given φ, φ' , we construct NPTA $\mathcal{A}, \mathcal{A}'$ equivalent to φ, φ' over d -ary trees, d as in Lemma 27, and decide whether $\mathcal{A}, \mathcal{A}'$ are joint \cong_σ^n -consistent over \mathbb{T}^d for all n via Proposition 7.

Let us provide some details on the proof of the central Lemma 27.

Proof. We show the implications $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 1$ in turn. The implication $4 \Rightarrow 1$ is immediate.

For $1 \Rightarrow 2$, suppose φ, φ' are not grML_σ^n -separable. Hence, for every $g \in \mathbb{N}$ there is a pair of models $\mathcal{M}_g \models \varphi$ and $\mathcal{M}'_g \models \varphi'$ with $\mathcal{M}_g \not\stackrel{g, \sigma}{\equiv} \mathcal{M}'_g$. One can encode with an FO-sentence θ that two models \mathcal{M} and \mathcal{M}' are depth- n trees, \mathcal{M} is a prefix of some $\mathcal{M}_+ \models \varphi$ and \mathcal{M}' of some $\mathcal{M}'_+ \models \varphi'$. If Z is a fresh binary symbol, then it is also possible to encode with an (infinite) set T of FO-sentences that Z is a graded bisimulation between \mathcal{M} and \mathcal{M}' . Every finite fragment of $\{\theta\} \cup T$ only mentions finitely many grades and hence by assumption is satisfiable. Thus, by compactness of FO, the entire $\{\theta\} \cup T$ is satisfiable. This gives us $\mathcal{M} \stackrel{\text{grd}, \sigma}{\equiv}^n \mathcal{M}'$ with extensions $\mathcal{M}_+ \models \varphi$ and $\mathcal{M}'_+ \models \varphi'$.

For $2 \Rightarrow 3$, fix witnesses $\mathcal{M}, \mathcal{M}'$ of joint $\equiv_{\text{grd}, \sigma}^n$ -consistency, that is, $\mathcal{M} \equiv_{\text{grd}, \sigma}^n \mathcal{M}'$ and there are extensions $\mathcal{M}_+, \mathcal{M}'_+$ of $\mathcal{M}, \mathcal{M}'$ with $\mathcal{M}_+ \models \varphi$ and $\mathcal{M}'_+ \models \varphi'$. By the Löwenheim-Skolem property of FO we may assume that both models are at most countable. It remains to apply the known fact that countable trees \mathcal{N} and \mathcal{N}' satisfy $\mathcal{N} \equiv_{\text{grd}} \mathcal{N}'$ iff \mathcal{N} and \mathcal{N}' are isomorphic. For the sake of completeness, we add a brief justification of this latter statement. Assume $w \in \mathcal{N}$ and $w' \in \mathcal{N}'$ with respective children $w_1, w_2, \dots = \bar{w}$ and $w'_1, w'_2, \dots = \bar{w}'$ such that $w \equiv_{\text{grd}} w'$. For every \equiv_{grd} -equivalence class X of \bar{w} the corresponding equivalence class $\{w'_i \mid \exists_{j \leq k}. w_j \equiv_{\text{grd}} w'_i\} = X'$ has the same cardinality as X . This is immediate for finite X , and for infinite X it follows because in countable models every two infinite subsets have the same cardinality. This allows us to inductively pick a *bijective* subrelation Z of \equiv_{grd} between \mathcal{N} and \mathcal{N}' which is still a graded bisimulation.

For $3 \Rightarrow 4$, fix witnesses $\mathcal{M}, \mathcal{M}'$ of joint \cong_{σ}^n -consistency, that is, $\mathcal{M} \cong_{\sigma}^n \mathcal{M}'$ and there are extensions $\mathcal{M}_+, \mathcal{M}'_+$ of $\mathcal{M}, \mathcal{M}'$ with $\mathcal{M}_+ \models \varphi$ and $\mathcal{M}'_+ \models \varphi'$. We trim \mathcal{M}_+ and \mathcal{M}'_+ so that the outdegree becomes at most d . Without losing generality we assume that the prefixes of \mathcal{M}_+ and \mathcal{M}'_+ are not only isomorphic but identical. The semantics of every $\psi \in \mu\text{ML}$ in a model \mathcal{N} is captured by a parity game whose positions are $N \times \text{SubFor}(\psi)$. We extend the definition of the game to $\mu\text{ML}_{\text{grd}}$. The set of positions $N \times \text{SubFor}(\psi)$ and the winning condition are defined as in the classical case, and so are the moves for all the positions with topmost connective other than the graded modalities. In the classical game, from $(v, \diamond\theta)$ Eve chooses a child v' of v and the next position is (v', θ) . In $(v, \diamond_{\geq k}\theta)$, first Eve chooses a subset v_1, \dots, v_k of size k of children of v , then $\forall\text{dam}$ chooses one of these children v_i and the next round starts at (v_i, θ) . Dually, in $(v, \square_{\leq k}\theta)$ first Eve picks a subset v_1, \dots, v_k of at most k v 's children, then $\forall\text{dam}$ responds with a choice of some v' not in v_1, \dots, v_k and the next position is (v', θ) . It is tedious but straightforward to check that Eve wins the game from v, ψ iff ψ is true at v , as in the classical case. Note that if we take a submodel \mathcal{N}_0 of \mathcal{N} which contains at least the root and all \diamond -witnesses (that is, points chosen by a winning strategy ζ in for positions of shape $(v, \diamond_{\geq k}\theta)$) then (the restriction of) ζ to \mathcal{N}_0 is a winning strategy for $\mathcal{G}(\mathcal{N}_0, \psi)$.

Let ζ and ζ' be positional winning strategies for Eve in the semantic games $\mathcal{G}(\mathcal{M}_+, \varphi)$ and $\mathcal{G}(\mathcal{M}'_+, \varphi')$. We take submodel $\mathcal{M}_0 \models \varphi$ of \mathcal{M}_+ as follows. In the n -prefix we take the root and all \diamond -witnesses for both ζ and ζ' . In the rest of the model we only take \diamond -witnesses for ζ . A submodel \mathcal{M}'_0 of \mathcal{M}'_+ is defined symmetrically. It follows that $\mathcal{M}_0 \models \varphi$ and $\mathcal{M}'_0 \models \varphi'$.

Recall that g is the maximal grade appearing in φ and φ' . Since the respective sets of positions of $\mathcal{G}(\mathcal{M}_+, \varphi)$ and $\mathcal{G}(\mathcal{M}'_+, \varphi')$ are $M_+ \times \text{SubFor}(\varphi)$ and $M'_+ \times \text{SubFor}(\varphi')$, for every point v there are at most $g \times |\varphi|$ \diamond -witnesses chosen by ζ from a position which has v on the first coordinate. Consequently, the outdegree of \mathcal{M}_0 and \mathcal{M}'_0 is not greater than $d = g \times (|\varphi| + |\varphi'|)$. This proves Lemma 27. \blacktriangleleft

8 Conclusion

We have presented an in-depth study of modal separation of μML -formulae over different classes of structures. For us, the most interesting results are the differences that are obtained over classes of bounded outdegree for different bounds $d = 1, d = 2, d \geq 3$. Without much effort our results on trees of bounded outdegrees can be transferred to infinite words and to *ranked* trees, via reductions similar to Lemma 4.

Throughout the paper we used the simplest possible measure of formula size: the length of a formula written as a string. Alternative more succinct measures, such as the number of non-isomorphic subformulae (*DAG-size*), are also interesting. Thus, a natural question is

to what extent our results depend on the choice of size measure. In principle, using a more succinct measure makes the problems of definability and separability harder. However, all our decision procedures, with an exception of Theorem 21, are automata-based. Consequently, these procedures carry over with unchanged complexity to any size measure for which the translation from logic to nondeterministic automata has the same complexity as in Theorem 2. In the remaining case of Theorem 21 a weaker assumption suffices: the modal depth of a formula is at most polynomial in its size. Both the mentioned assumptions are arguably modest.

A place where the choice of size measure matters a little more is the *construction* of modal definitions and separators. In the cases of unrestricted, unary (\mathbb{T}^1), and high outdegree models (\mathbb{T}^d for $d \geq 3$) the constructed formulae have DAG-size essentially the same as size: doubly, singly, and triply exponential, respectively. Interestingly, however, in the binary case \mathbb{T}^2 our formulae have only singly exponential DAG-size, which is easily seen to be optimal and contrasts with their doubly exponential size. This demonstrates that the lower bounds for size of modal definitions over \mathbb{T}^2 cannot work for DAG-size. The same lower bound construction fails for DAG-size over unrestricted models, although there the exact DAG-size complexity of optimal separators remains unknown.

We mention some interesting open problems. First, the relative succinctness of μML over ML is to the best of our knowledge open in the setting with only one accessibility relation. Second, as we have mentioned in Section 3, the separators we compute are not necessarily the logically strongest ones. The logically strongest separators of φ, φ' are precisely the ML -uniform consequences of φ (if they exist) and are a natural object of study. Clearly, modal definability of φ is a sufficient condition, but not a necessary one. Let $\varphi = \psi \wedge \neg\theta_\infty$ and $\varphi' = \psi$ for some $\psi \in \text{ML}$. Then φ is not equivalent to a modal formula, but ψ is a strongest separator. In the context of $\text{gr}\mu\text{ML}$, open questions are ML -definability (and separability) and μML -definability (and separability) of $\text{gr}\mu\text{ML}$ -formulae. We conjecture them to be easier than 2-EXPTIME. Finally, let us mention that definability of μML -formulae by *safety formulae* has been studied in [22]. It would be natural to investigate separability there as well.

References

- 1 Alessandro Artale, Jean Christoph Jung, Andrea Mazzullo, Ana Ozaki, and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in description and modal logics with nominals and role inclusions. *ACM Trans. Comput. Log.*, 24(4):34:1–34:51, 2023. doi:10.1145/3597301.
- 2 Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. URL: <http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-description-logic?format=PB#17zVGeWD2TZUeu6s.97>.
- 3 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
- 4 Fabio Bellissima. On the lattice of extensions of the modal logics KAlt_n . *Arch. Math. Log.*, 27(2):107–114, 1988. doi:10.1007/BF01620760.
- 5 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 293–304. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.36.
- 6 Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Log. Methods Comput. Sci.*, 10(3), 2014. doi:10.2168/LMCS-10(3:2)2014.

- 7 Mikołaj Bojańczyk and Wojciech Czerwiński. *Automata Toolbox*. University of Warsaw, 2018. URL: <https://www.mimuw.edu.pl/~bojan/papers/toolbox.pdf>.
- 8 Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.
- 9 Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is PSpace-complete. *Theor. Comput. Sci.*, 88(1):99–116, 1991. doi:10.1016/0304-3975(91)90075-D.
- 10 Maarten de Rijke. A note on graded modal logic. *Stud Logica*, 64(2):271–283, 2000. doi:10.1023/A:1005245900406.
- 11 Kit Fine. In so many possible worlds. *Notre Dame J. Formal Log.*, 13(4):516–520, 1972. doi:10.1305/NDJFL/1093890715.
- 12 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. doi:10.1016/0022-0000(79)90046-1.
- 13 Tim French, Wiebe van der Hoek, Petar Iliev, and Barteld P. Kooi. On the succinctness of some modal logics. *Artif. Intell.*, 197:56–85, 2013. doi:10.1016/j.artint.2013.02.003.
- 14 Martin Fürer. The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In *Logic and Machines: Decision Problems and Complexity, Proceedings of the Symposium "Rekursive Kombinatorik"*, volume 171 of *Lecture Notes in Computer Science*, pages 312–319. Springer, 1983. doi:10.1007/3-540-13331-3_48.
- 15 Dov M. Gabbay. Craig’s interpolation theorem for modal logics. In *Conference in Mathematical Logic — London ’70*, pages 111–127, Berlin, Heidelberg, 1972. Springer Berlin Heidelberg.
- 16 Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Undecidable boundedness problems for datalog programs. *J. Log. Program.*, 25(2):163–190, 1995. doi:10.1016/0743-1066(95)00051-K.
- 17 Jean Christoph Jung and Jędrzej Kołodziejcki. Modal separability of fixpoint formulae. In *Proceedings of the 37th International Workshop on Description Logics (DL 2024)*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-3739/paper-5.pdf>.
- 18 Jean Christoph Jung and Frank Wolter. Living without Beth and Craig: Definitions and interpolants in the guarded and two-variable fragments. In *Proceedings of Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–14. IEEE, 2021. doi:10.1109/LICS52264.2021.9470585.
- 19 Eryk Kopczynski. Invisible pushdown languages. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 867–872. ACM, 2016. doi:10.1145/2933575.2933579.
- 20 Dexter Kozen. Results on the propositional μ -calculus. *Theor. Comput. Sci.*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.
- 21 Louwe Kuijjer, Tony Tan, Frank Wolter, and Michael Zakharyashev. Separating counting from non-counting in fragments of two-variable first-order logic (extended abstract). In *Proc. of DL 2024*, 2024.
- 22 Karoliina Lehtinen and Sandra Quickert. Deciding the first levels of the modal μ alternation hierarchy by formula construction. In *Proceedings of Annual Conference on Computer Science Logic CSL*, volume 41 of *LIPICs*, pages 457–471. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.CSL.2015.457.
- 23 Christof Löding and Christopher Spinrath. Decision problems for subclasses of rational relations over finite and infinite words. *Discrete Mathematics & Theoretical Computer Science*, Vol. 21 no. 3, January 2019. doi:10.23638/DMTCS-21-3-4.
- 24 Martin Otto. Eliminating recursion in the μ -calculus. In *Proceedings of 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1563 of *Lecture Notes in Computer Science*, pages 531–540. Springer, 1999. doi:10.1007/3-540-49116-3_50.
- 25 Martin Otto. Graded modal logic and counting bisimulation. *CoRR*, abs/1910.00039, 2019. arXiv:1910.00039.

- 26 Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Log. Methods Comput. Sci.*, 12(1), 2016. doi:10.2168/LMCS-12(1:5)2016.
- 27 Vaughan R. Pratt. A decidable mu-calculus: Preliminary report. In *Proceedings of 22nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 421–427. IEEE Computer Society, 1981. doi:10.1109/SFCS.1981.4.
- 28 Abraham Robinson. A result on consistency and its application to the theory of definition. *Journal of Symbolic Logic*, 25(2):174–174, 1960. doi:10.2307/2964240.
- 29 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Inf. Control.*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.
- 30 A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985. doi:10.1145/3828.3837.
- 31 Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of International Colloquium Automata, Languages and Programming (ICALP)*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998. doi:10.1007/BFB0055090.
- 32 Yde Venema. Lectures on the modal μ -calculus, 2020.

Transforming Stacks into Queues: Mixed and Separated Layouts of Graphs

Julia Katheder  

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany

Michael Kaufmann  

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany

Sergey Pupyrev  

Menlo Park, CA, USA

Torsten Ueckerdt  

Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Germany

Abstract

Some of the most important open problems for linear layouts of graphs ask for the relation between a graph's queue number and its stack number or mixed number. In such, we seek a vertex order and edge partition of G into parts with pairwise non-crossing edges (a stack) or with pairwise non-nesting edges (a queue). Allowing only stacks, only queues, or both, the minimum number of required parts is the graph's stack number $\text{sn}(G)$, queue number $\text{qn}(G)$, and mixed number $\text{mn}(G)$, respectively.

Already in 1992, Heath and Rosenberg asked whether $\text{qn}(G)$ is bounded in terms of $\text{sn}(G)$, that is, whether stacks “can be transformed into” queues. This is equivalent to bipartite 3-stack graphs having bounded queue number (Dujmović and Wood, 2005). Recently, Alam et al. asked whether $\text{qn}(G)$ is bounded in terms of $\text{mn}(G)$, which we show to also be equivalent to the previous questions.

We approach the problem by considering *separated* linear layouts of bipartite graphs. In this natural setting all vertices of one part must precede all vertices of the other part. Separated stack and queue numbers coincide, and for fixed vertex orders, graphs with bounded separated stack/queue number can be characterized and efficiently recognized, whereas the separated mixed layouts are more challenging. In this work, we thoroughly investigate the relationship between separated and non-separated, mixed and pure linear layouts.

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

Keywords and phrases Separated linear Layouts, Stack Number, Queue Number, mixed Number, bipartite Graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.56

Related Version *Full Version:* <https://doi.org/10.48550/arXiv.2409.17776> [21]

Funding *Julia Katheder:* The work of J. Katheder is supported by DFG grant Ka 812-18/2.

Michael Kaufmann: The work of M. Kaufmann is supported by DFG grant Ka 812-18/2.

Torsten Ueckerdt: The work of T. Ueckerdt is supported by DFG - 520723789.

1 Introduction

In this paper we study linear layouts of graphs¹. That is, for a graph $G = (V, E)$, we consider a total order σ of its vertex set V , while σ defines the relative position of its edges. In particular, we investigate for a graph G its well-known parameters stack number, $\text{sn}(G)$, and its queue number, $\text{qn}(G)$, as well as their common generalization called its mixed number, $\text{mn}(G)$. Their precise definitions go as follows.

¹ Throughout, all graphs in this paper are simple, undirected, finite, and have at least one edge.



Stack, Queue, and Mixed Numbers. Given a vertex order σ , two edges (u, v) and (x, y) in E are said to *cross* in σ if any order of their endvertices symmetric to $u <_{\sigma} x <_{\sigma} v <_{\sigma} y$ is prescribed by σ . Roughly speaking, for the stack number of G , we want a vertex order σ in which not many edges pairwise cross. Formally, for an integer $s \geq 1$, an *s-stack layout* of $G = (V, E)$ is a total order σ of V together with a partition of E into subsets E_1, \dots, E_s , called *stacks*, such that no two edges in the same stack cross. The minimum number s of stacks needed for a stack layout of graph G is called its *stack number* and denoted by $\text{sn}(G)$. Stack numbers were first investigated by Bernhart and Kainen [6] in 1979, building on Kainen and Ollman [20, 24]².

As a concept “dual” to *s-stack layouts*, for an integer $q \geq 1$, a *q-queue layout* of $G = (V, E)$ is a total order σ of V together with a partition of E into subsets E_1, \dots, E_q , called *queues*, such that no two edges in the same queue *nest*; that is, there are no edges (u, v) and (x, y) in a queue with $u <_{\sigma} x <_{\sigma} y <_{\sigma} v$ (or any symmetric order). The minimum number q of queues needed for a queue layout of graph G is called its *queue number* and denoted by $\text{qn}(G)$. Queue numbers were introduced by Heath and Rosenberg [18] in 1992.

Stack and queue layouts are generalized through the notion of a *mixed layout*. For integers $s, q \geq 1$, an *s-stack q-queue layout* of $G = (V, E)$ is a total order σ of V together with a partition of E into s stacks and q queues. The minimum value of $s + q$ needed for an *s-stack q-queue layout* of graph G is called its *mixed number* and denoted by $\text{mn}(G)$. Mixed layouts were already considered by Heath and Rosenberg [18] in 1992, while a thorough study started only recently [16, 22, 26]. In contrast to mixed layouts, we call a layout *pure* if only stacks or only queues are being used.

Comparing Stack, Queue, and Mixed Numbers. Besides their similar definitions, there are more similarities between *k-stack graphs*, *k-queue graphs*, and *k-mixed graphs*, where a *k*-(stack, queue, mixed) graph is defined as a graph admitting a *k*-(stack, queue, mixed) layout. For example, in each case we have sparse graphs with only $\mathcal{O}(kn)$ edges for n vertices [6, 18, 25]. Moreover, stack, queue, and mixed numbers are all bounded for planar graphs [5, 8, 14, 30] and for bounded treewidth graphs [1, 9, 15, 28]. On the other hand, neither stack nor queue number is bounded for 3-regular graphs [3, 29]. Already in 1992, Heath, Leighton, and Rosenberg [17] investigated the relationship between stack and queue layouts; in particular whether stack layouts and queue layouts can be transformed into each other. They introduced the following fundamental question, which remains unanswered despite a wealth of studies on linear graph layouts.

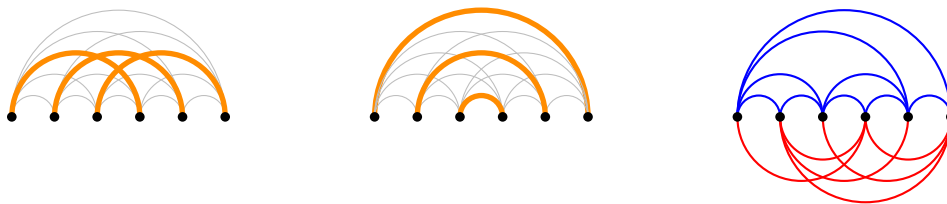
► **Open Problem 1** (Heath, Leighton, and Rosenberg [17], see also [7, 11, 13]).

Do graphs with bounded stack number have bounded queue number?

We emphasize that a companion question (“do graphs with bounded queue number have bounded stack number?”) has been recently resolved in the negative by Dujmović et al. [7]. One attempt to resolve Open Problem 1 was made by Dujmović and Wood [11] who showed that the problem is equivalent to the question of whether bipartite 3-stack graphs have bounded queue number. Note that 1-stack graphs and 2-stack graphs are planar, and hence, they have a bounded queue number [8], but the question remains open for 3-stack graphs.

Turning to mixed layouts, it follows from the definition of the mixed number that for every graph G we have $\text{mn}(G) \leq \text{sn}(G)$ and $\text{mn}(G) \leq \text{qn}(G)$. However for some graphs G , their mixed number $\text{mn}(G)$ is strictly less than $\text{sn}(G)$ and $\text{qn}(G)$; for example, for the

² We remark that *s-stack layouts* are also known as *s-page book embeddings*, where stacks are then called pages. Similarly, the stack number $\text{sn}(G)$ is sometimes called the book thickness or page number of G .



■ **Figure 1** Linear layouts of K_6 verifying that $\text{sn}(K_6) \geq 3$ due to a 3-twist (left), $\text{qn}(K_6) \geq 3$ due to a 3-rainbow (middle), and $\text{mn}(K_6) \leq 2$ due to a 1-stack 1-queue layout (right).

complete graph K_6 it holds that $\text{sn}(K_6) = \text{qn}(K_6) = 3$ but $\text{mn}(K_6) = 2$, as illustrated in Figure 1. In particular, graphs with bounded mixed number potentially form a strictly larger class than those of bounded stack number. Thus, the following problem of Alam et al. [22] asks for something (potentially) stronger than Open Problem 1.

► **Open Problem 2** (Alam et al. [22]).

Do graphs with bounded mixed number have bounded queue number?

As one of our results (cf. Theorem 4), we shall show that Open Problems 1 and 2 are in fact equivalent. That is, either both questions have a YES-answer or both have a NO-answer. In that sense, it is easier to prove a NO-answer by finding graphs of unbounded queue number but bounded mixed number (instead of bounded stack number). As a second result, also in Theorem 4, we shall prove that it indeed suffices to look for graphs of mixed number 2; specifically, graphs that admit 1-stack 1-queue layouts. That is, Open Problems 1 and 2 have a YES-answer if and only if all 1-stack 1-queue graphs have bounded queue number.

Separated Linear Layouts. Since Open Problems 1 and 2 are likely very challenging in their full generality, we investigate a special case for *separated mixed* layouts of bipartite graphs. A vertex order σ of a bipartite graph G with bipartition³ (A, B) is *separated* if all vertices in A precede all vertices in B (or vice versa). This naturally gives rise to separated k -stack, separated k -queue, and separated s -stack q -queue layouts of bipartite graphs G , as well as the corresponding separated stack number $\overline{\text{sn}}(G)$, separated queue number $\overline{\text{qn}}(G)$, and separated mixed number $\overline{\text{mn}}(G)$.

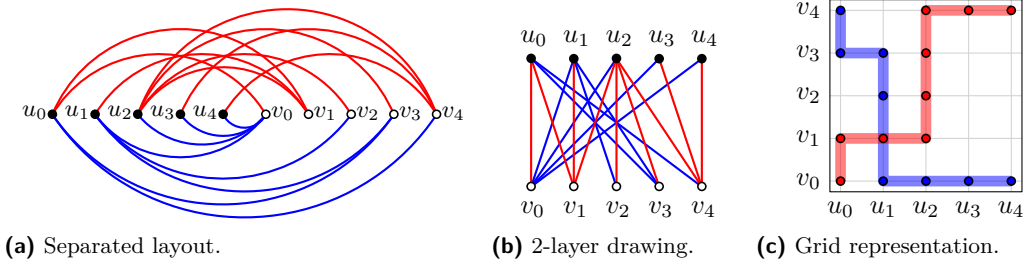
Observe that a separated s -stack q -queue layout can be transformed into a separated q -stack s -queue layout by reversing the vertex order of one of the parts. It follows that $\overline{\text{sn}}(G) = \overline{\text{qn}}(G)$ for every bipartite graph G . Furthermore, every separated vertex order σ of $G = (V, E)$ with bipartition (A, B) induces an injective mapping of the edges of G to points of the $|A| \times |B|$ integer grid; see Figure 2. This grid is sometimes referred to as a *reduced adjacency matrix* of G . One can easily verify that a subset $S \subseteq E$ of edges forms a queue (resp. a stack) if and only if the corresponding points form a monotonically increasing (resp. decreasing) subset in the $|A| \times |B|$ grid. The stack-queue transformation mentioned above then corresponds to mirroring the grid along the x -axis or along the y -axis.

This separated setting has been widely studied (sometimes, implicitly) under different names, such as *2-track layouts* [10, 27] or *2-layer drawings* [2, 12, 23]. For the present paper, let us introduce the following special variant of Open Problem 2.

► **Open Problem 3.**

Do bipartite graphs with bounded separated mixed number have bounded queue number?

³ (A, B) is a bipartition if $A \cap B = \emptyset$, $A \cup B = V$, and both induced subgraph $G[A], G[B]$ are edgeless.



■ **Figure 2** Various representations of a separated 1-stack 1-queue bipartite graph.

Our Contributions. We make progress towards the resolution of Open Problems 1 and 2, as well as our new Open Problem 3. Each of these problems asks whether graphs with low stack number (Open Problem 1), low mixed number (Open Problem 2), or low separated mixed number (Open Problem 3) have also low queue number. Formally, we say that the *queue number is bounded by stack number* (similarly, *bounded by mixed number*, and *bounded by separated mixed number*) if for every graph G with stack number $\text{sn}(G)$ and queue number $\text{qn}(G)$, it holds that $\text{qn}(G) \leq f(\text{sn}(G))$ for some function f (independent of G). The question of whether such functions, f , exist are Open Problems 1–3.

Clearly, a function f that bounds the queue number in terms of the stack number exists if and only if for all $k \in \mathbb{N}$ there is a constant $C = C_k$ such that every k -stack graph has queue number at most C . Surprisingly, it is enough to consider just one particular value for k . In fact, Dujmović and Wood [11] show that Open Problem 1 is equivalent to the existence of a constant C such that every 3-stack graph has queue number at most C . (It is known that 1-stack and 2-stack graphs have queue number at most 2 [17] and at most 42 [4], respectively.)

Here, we extend the list to four equivalent statements, showing in particular that Open Problems 1 and 2 are equivalent, and that it is enough to consider 1-stack 1-queue graphs.

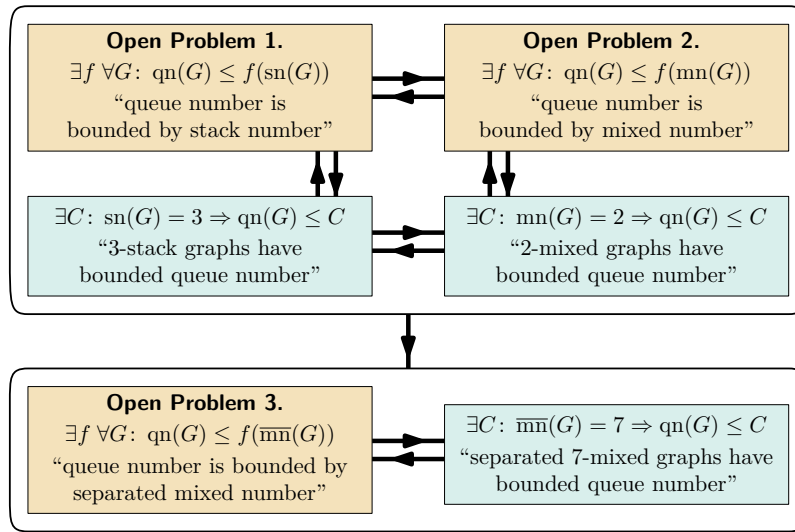
► **Theorem 4.** *The following are equivalent:*

- (1) every s -stack graph has queue number at most $f(s)$ for some function f (“queue number is bounded by stack number”);
- (2) every 3-stack graph has queue number at most C for some constant C ;
- (3) every 1-stack 1-queue graph has queue number at most C for some constant C ;
- (4) every s -stack q -queue graph has queue number at most $f(s, q)$ for some function f (“queue number is bounded by mixed number”).

Turning to separated layouts of (necessarily bipartite) graphs, our main contribution is also a list of four equivalent statements, one of which, namely (1), is Open Problem 3. Statements (3) and (4) show that it is enough to consider 1-stack q -queue graphs, respectively even only 1-stack 6-queue graphs, for solving Open Problem 3. Additionally, we discuss a natural approach for Open Problem 3 where we start with the grid representation of a separated s -stack q -queue layout, and then permute the rows and columns so as to obtain a separated $f(s, q)$ -queue layout. Another surprising contribution is that it is always enough to apply the same permutation to the rows and the columns, which is statement (2).

► **Theorem 5.** *The following are equivalent:*

- (1) every separated s -stack q -queue graph G has queue number at most $f(s, q)$ for some function f (“queue number is bounded by mixed number”);



■ **Figure 3** An overview of the new and known relationships between different linear layouts.

- (2) every separated s -stack q -queue layout of a graph $G = (A \cup B, E)$ with $|A| = |B|$ can be transformed into a separated $f(s, q)$ -queue layout for some function f by applying the same permutation to A and B ;
- (3) every separated 1-stack q -queue graph has queue number at most $f(q)$ for some function f ;
- (4) every separated 1-stack 6-queue graph has queue number at most C for some constant C .

Finally, let us discuss the connections between non-separated and separated layouts. Standard techniques for queue layouts (which we present in Section 2) easily give that Open Problem 2 implies Open Problem 3. In fact, Corollary 7 states that for every bipartite graph G we have $\overline{qn}(G) \leq 2 \text{qn}(G)$. However, it remains open whether Open Problem 3 implies Open Problem 2 (or equivalently Open Problem 1). The situation is summarized in Figure 3.

2 Preliminaries

In this section we collect several facts about manipulating vertex orders in linear layouts, which keep the stack and/or queue numbers within a constant factor from each other.

► **Lemma 6 (Riffle-Lemma).** *Let $G = (V, E)$ be a graph with a given q -queue layout using σ as the vertex order and V_1, \dots, V_k be a partition of V . Then, for any vertex order σ' where for vertices $u, v \in V_i$ it holds that $u <_{\sigma'} v$ whenever $u <_{\sigma} v$:*

- (1) G admits a $k^2 \cdot q$ -queue layout.
- (2) G admits a $2 \cdot \ell \cdot (k - \ell) \cdot q$ -queue layout if G is bipartite, $\bigcup_{i=1}^{\ell} V_i = A$ and $\bigcup_{j=\ell+1}^k V_j = B$.

Proof. We show that every queue $Q \subseteq E$ for σ can be split into k^2 queues for σ' ; see Figure 4a. Overall, this results in the desired $k^2 \cdot q$ -queue layout. For $i, j \in [k]$ let $E_{i,j} \subseteq Q$ contain all edges $(u, v) \in Q$ such that $u <_{\sigma} v$ and $u \in V_i$ and $v \in V_j$. This partitions Q into k^2 many edge sets $E_{i,j}$, and we claim that each $E_{i,j}$ is a queue for σ' . For $i = j$, the relative order of any $u, v \in V_i$ is unchanged in σ' , and hence $E_{i,i}$ is a queue for σ' . For $i \neq j$, let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two edges in $E_{i,j}$ with $u_1, u_2 \in V_i$ and $v_1, v_2 \in V_j$. Without loss of generality assume that $u_1 <_{\sigma'} u_2$ (hence $u_1 <_{\sigma} u_2$) and assume for a contradiction that e_1 nests e_2 in σ' , that is, $u_1 <_{\sigma'} u_2 <_{\sigma'} v_2 <_{\sigma'} v_1$ or $v_2 <_{\sigma'} v_1 <_{\sigma'} u_1 <_{\sigma'} u_2$. In either case,

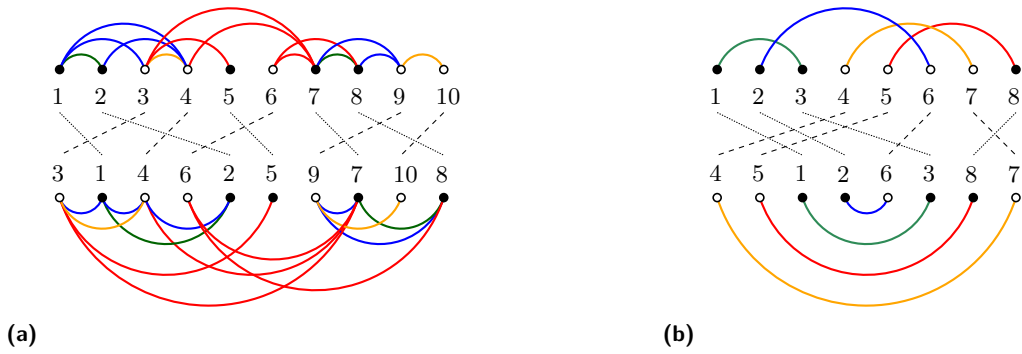


Figure 4 (a) Illustrating Lemma 6 (1) and its tightness for $k = 2$ (b). Edge colors illustrate the partition of the edges with respect to V_1 (black) and V_2 (white). The black lines signify the orders within V_1 (dotted) and V_2 (dashed).

it follows that $v_2 <_{\sigma'} v_1$ and hence $v_2 <_{\sigma} v_1$. Together with $u_2 <_{\sigma} v_2$ (as $(u_2, v_2) \in E_{i,j}$) this yields $u_1 <_{\sigma} u_2 <_{\sigma} v_2 <_{\sigma} v_1$ and e_1 nests e_2 in σ – a contradiction to Q being a queue. It follows that each $E_{i,j}$ requires at most one queue in the layout with vertex order σ' , which concludes the first case of the proof.

In the bipartite case, for each queue Q in the original layout, let $e = (u, v)$ be an edge in Q with $u \in V_i$ with $i \leq \ell$ and $v \in V_j$ with $\ell < j \leq k$. As there are $\ell \cdot (k - \ell)$ combinations of i and j , and we require two queues, namely $E_{i,j}$ and $E_{j,i}$, for each combination, the resulting layout with vertex order σ' requires $2 \cdot \ell \cdot (k - \ell) \cdot q$ -queues in total. ◀

Applying the lemma to bipartite graphs with $\ell = 1$ and $k = 2$ such that $V_1 = A$ and $V_2 = B$ yields the following (well-known) fact; see [25] for an alternative proof.

► **Corollary 7.** *Every q -queue graph with bipartition (A, B) has a separated $2q$ -queue layout.*

The next two results concern graph subdivisions. A *subdivision* of a graph G is a graph obtained from G by replacing each edge (u, v) in G by a path with one or more edges. Internal vertices on such a path are called *division vertices*.

► **Lemma 8** (Lemma 2 in [11]). *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be a subdivision of G with at most one division vertex per edge. If G admits a q -queue layout with vertex order σ , then G' with bipartition V and $V' \setminus V$ admits a separated $(q + 1)$ -queue layout in which V is ordered as in σ .*

A converse operation to a subdivision is a (path-)contraction. We use a more general transformation here. An r -shallow minor is a restricted form of a graph minor in which each (connected) subgraph that is contracted to form the minor has radius at most r , where the radius is defined as the minimum of the eccentricities of its vertices. For an s -stack q -queue graph G , we will combine the following result with finding an s' -stack q' -queue subdivision H where $s' \leq s$ and $q' \leq q$, which has G as an r -shallow minor in order to prove equivalence statements in Theorem 4 and Theorem 5.

► **Lemma 9** (Lemma 12 in [19]). *For every graph G and H , where G is a r -shallow minor of H , it holds that $\text{qn}(G) \leq (2r + 1)(2 \text{qn}(H))^{2r+1}$.*

3 Non-Separated Layouts

In this section we explore Open Problem 2. Dujmović and Wood [11] studied graph subdivisions of stack and queue layouts. They proved that every s -stack graph has

- (i) a 3-stack subdivision with $2\lceil \log_2 s \rceil - 2$ division vertices per edge, and
- (ii) a 1-stack 1-queue subdivision with $4\lceil \log_2 s \rceil$ division vertices per edge.

Similarly, every q -queue graph has

- (i) a 3-stack subdivision with $2\lceil \log_2 q \rceil + 1$ division vertices per edge, and
- (ii) a 2-queue subdivision with $2\lceil \log_2 q \rceil + 1$ division vertices per edge, and
- (iii) a 1-stack 1-queue subdivision with $4\lceil \log_2 q \rceil + 2$ division vertices per edge.

Notice some asymmetry in the statements of the two results: It is not always possible to subdivide an s -stack graph with $f(s)$ division vertices per edge (for an arbitrary function f) such that the result admits an $\mathcal{O}(1)$ -queue layout. Otherwise, such a subdivision combined with Lemma 9 would imply that every s -stack graph admits an $\mathcal{O}(1)$ -queue layout, contradicting the main result of Dujmović et al. [7].

The crux of the contributions of Dujmović and Wood [11] is a technical construction, which we formalize next. Let T be a rooted tree with nodes $V(T)$ and edges $E(T)$. Given a graph $G = (V, E)$, a T -partition of G is a pair $(T, \{T_x : x \in V(T)\})$ consisting of a tree T and a partition of V into sets $\{T_x : x \in V(T)\}$, such that for every edge $(u, v) \in E$ one of the following holds: (i) $u, v \in T_x$ for some $x \in V(T)$, or (ii) there is an edge (x, y) of T with $u \in T_x$ and $v \in T_y$. The sets $T_x, x \in V(T)$ are called the *bags* of the T -partition. A T -layout of a graph G is a T -partition of G in which the bags are ordered, that is, $<_x$ is a total order of vertices in T_x ; see Figure 5a for an example. The vertex orders are described in terms of two functions, $\mathcal{S} : V(T) \rightarrow \mathbb{N}_0$ and $\mathcal{Q} : V(T) \rightarrow \mathbb{N}_0$, defined for the nodes of T . We prescribe each bag to contain either only stacks or only queues formed by intra-bag edges. Here $\mathcal{S}(x)$ (respectively, $\mathcal{Q}(x)$) denotes the stack (queue) number of the intra-bag edges on T_x under vertex order $<_x$ such that if T_x is prescribed to contain stacks, i.e., if $\mathcal{S}(x) > 0$ then $\mathcal{Q}(x) = 0$, and if T_x is a bag containing queues, then $\mathcal{Q}(x) > 0$ and $\mathcal{S}(x) = 0$. Similarly, for the edges of T , $\mathcal{K}(x, y) : E(T) \rightarrow \mathbb{N}_0$ is the minimum number of edge sets $E_1, \dots, E_{\mathcal{K}(x, y)}$ needed to partition the inter-bag edges of G between T_x and T_y such that they are pairwise non-crossing. Under the concatenation of the orders $<_x$ and $<_y$, each $E_i, 1 \leq i \leq \mathcal{K}(x, y)$ will form a queue, while each E_i forms a stack when concatenating $<_x$ and $>_y$. The leaf nodes of a tree T are denoted $\tilde{V}(T)$.

In this paper, we work with *simple* T -layouts, where

- for every non-leaf node $x \in V(T) \setminus \tilde{V}(T)$, bag T_x forms an independent set in G , that is, $\mathcal{S}(x) = \mathcal{Q}(x) = 0$;
- for every leaf node $x \in \tilde{V}(T)$, bag T_x admits a 1-stack or a 1-queue layout under $<_x$, that is, $\mathcal{S}(x) = 1, \mathcal{Q}(x) = 0$ or $\mathcal{S}(x) = 0, \mathcal{Q}(x) = 1$;
- for every edge $(x, y) \in E(T)$, it holds that $\mathcal{K}(x, y) = 1$.

In all our constructions of tree-partitions, we utilize a *binary tree*, that is, a rooted tree in which every node has at most two children. The following result provides a subdivision and a tree-layout for a given graph.

► **Lemma 10** (a special case of Lemma 21 in [11]). *Let G be a graph that admits a k -stack (respectively, k -queue) layout with vertex order σ , and T be a binary tree of height h with k or more leaves. Then G has a subdivision, D , with an even number of division vertices per edge such that D has a simple T -layout in which $\mathcal{S}(x) = 1$ (respectively, $\mathcal{Q}(x) = 1$) for every leaf node $x \in \tilde{V}(T)$. The number of division vertices per edge is at most $2h$, or exactly $2h$*

if all the leaves of T are at depth h . Moreover, the vertices of G correspond to vertices in the root bag of T and their order in the T -layout is σ , whereas all other bags contain only division vertices.

In what follows we consider a (not necessarily proper) 2-coloring of edges of a tree using colors blue and red, i.e., we allow edges with the same color at a common endvertex. Throughout, we associate blue colors with stacks, and red colors with queues. The set of blue edges is $E^s(T) \subseteq E(T)$ and the set of red edges is $E^q(T) \subseteq E(T)$.

► **Lemma 11** (a special case of Lemma 22 in [11]). *Let G be a graph with a T -layout for some tree T . Suppose that edges of T are 2-colored in red and blue, and its nodes, $V(T)$, are ordered by σ such that the blue edges, E^s , form a stack and the red edges, E^q , form a queue. Let*

$$\lambda_s = \max_{x \in V(T)} \left(\mathcal{S}(x) + \sum_{(y,x) \in E^s: y <_\sigma x} \mathcal{K}(y,x) + \sum_{(x,y) \in E^s: x <_\sigma y} \mathcal{K}(x,y) \right), \text{ and}$$

$$\lambda_q = \max_{x \in V(T)} \left(\mathcal{Q}(x) + \max_{y \in V(T): y \leq_\sigma x} \sum_{(y,z) \in E^q: x \leq_\sigma z} \mathcal{K}(y,z) \right).$$

Then G admits a mixed λ_s -stack λ_q -queue layout. Furthermore, the order of vertices in the root bag of the T -layout is preserved in the mixed layout.

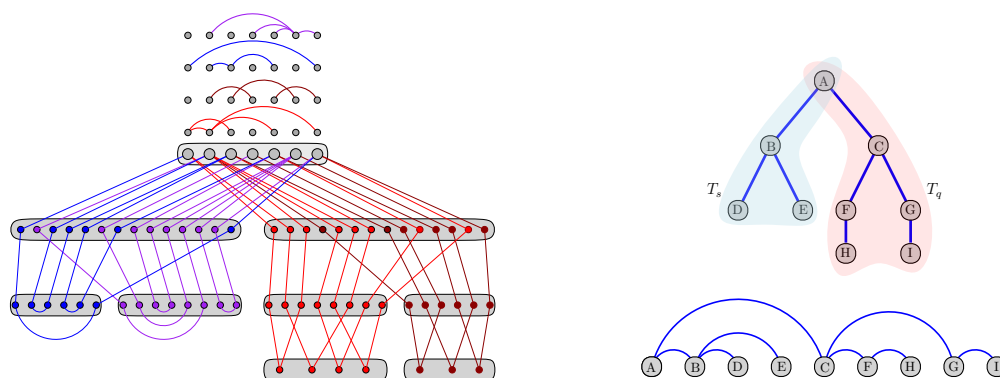
Now we can state the new results of the section.

► **Lemma 12.** *Let G be an s -stack q -queue graph. Then G has a 3-stack subdivision with at most $2\lceil \log_2(\max(s,q)) \rceil + 3$ division vertices per edge.*

Proof. First we show how to obtain a tree-layout for an s -stack q -queue graph $G = (V, E)$ with the vertex order σ . Denote $h = \lceil \log_2(\max(s,q)) \rceil$ so that $\max(s,q) \leq 2^h$. Assume that $E = S_1 \cup \dots \cup S_s \cup Q_1 \cup \dots \cup Q_q$, where $S_i, 1 \leq i \leq s$ are stacks and $Q_i, 1 \leq i \leq q$ are queues.

Consider the subgraph of G induced by all stack edges, $G^s = (V, S_1 \cup \dots \cup S_s)$. Let T_s be a binary tree of height $h+1$ in which the root node has a single child, each internal non-root node has exactly two children, and all leaves are at the same depth. By Lemma 10, there exists a subdivision of G^s , denoted D^s , with exactly $2(h+1)$ division vertices per edge and a simple T_s -layout of D^s . It holds that $\mathcal{S}(x) = 1, \mathcal{Q}(x) = 0$ for $x \in \tilde{V}(T_s)$, $\mathcal{S}(x) = \mathcal{Q}(x) = 0$ for $x \in V(T_s) \setminus \tilde{V}(T_s)$, and $\mathcal{K}(x,y) = 1$ for $(x,y) \in E(T_s)$; see Figure 5.

Next consider the subgraph of G induced by the queue edges, $G^q = (V, Q_1 \cup \dots \cup Q_q)$. Start with a binary tree of height $h+1$, denoted T'_q , which is a copy of T_s (that is, the root node has one child and each internal non-root node has two children). There exists a subdivision of G^q with exactly $2h+2$ subdivision vertices and its simple T'_q -layout by Lemma 10. We modify the subdivision and the tree-layout as follows. Consider a leaf node $x' \in \tilde{V}(T'_q)$ and let $G_{x'}$ be the graph induced by vertices in $T_{x'}$ and the corresponding intra-bag edges. Observe that $\mathcal{Q}(x') = 1$ in the T'_q -layout, and hence, $\text{qn}(G_{x'}) = 1$ with $<_{x'}$ as the vertex order. Now, we apply Lemma 8 by subdividing every edge of $G_{x'}$ once and let G_x be the resulting graph. This results in a 2-queue layout of the subdivision G_x in which the vertices of $G_{x'}$ are separated from the new division vertices and are ordered as in the T'_q -layout. Thus, we can build a new tree, denoted T_q , by appending a child, x , to every node $x' \in \tilde{V}(T'_q)$. This results in a (non-simple) T_q -layout for G_q , where the leaf nodes of T_q contain the division vertices in the order given by Lemma 8. Since the queue number of G_x is at most 2, the inter-bag edges between $G_{x'}$ and G_x can be partitioned into two sets of pairwise non-crossing edges, i.e., we have that $\mathcal{K}(x',x) = 2$ for all $(x',x) \in E(T_q)$, where $x \in \tilde{V}(T_q)$; see Figure 5a for an illustration.



(a) A tree-layout of the subdivision of a 2-stack 2-queue graph. Edge colors in the tree-layout correspond to the original queues and stacks of G shown above the root node.

(b) A tree-partition and a 1-stack layout of the corresponding binary tree, T_{sq} , where the blue edge color corresponds to Lemma 11.

■ **Figure 5** An illustration for Lemma 12: Subdividing a 2-stack 2-queue graph G with at most $2\lceil \log_2(\max(s, q)) \rceil + 3 = 5$ division vertices per edge to obtain a 3-stack graph.

By Lemma 10, the root bags of the T_s -layout and the T_q -layout contain the same vertices, V , both ordered by σ . Thus, we can merge the two tree-layouts into a joint one, called T_{sq} -layout; see Figure 5b. The corresponding subdivision of G , denoted D^{sq} , has stack edges subdivided $2h + 2$ times and queue edges subdivided $2h + 3$ times. To apply Lemma 11 for the T_{sq} -layout, we color all the edges of T_{sq} blue and find a 1-stack layout of the tree via a depth-first search traversal such that every node precedes its children in the order. Let us argue that the result of applying Lemma 11 is a 3-stack layout, that is, $\lambda_s = 3$ and $\lambda_q = 0$.

Note that $\mathcal{Q}(x) = 0$ for all nodes of T_{sq} and the tree contains no red edges; thus, $\lambda_q = 0$. For the stack number, consider four disjoint groups of nodes of T_{sq} :

- for $x \in \tilde{V}(T_s)$, it holds that $\mathcal{S}(x) = 1$ and $\mathcal{K}(y, x) = 1$ for $(y, x) \in E^s$;
- for $x \in V(T_s) \setminus \tilde{V}(T_s)$ and $x \in V(T'_q) \setminus \tilde{V}(T'_q)$, it holds that $\mathcal{S}(x) = 0$, there exists one edge $(y, x) \in E^s$ with $\mathcal{K}(y, x) = 1$ and two edges $(x, y) \in E^s$ with $\mathcal{K}(x, y) = 1$;
- for $x \in \tilde{V}(T'_q)$, it holds that $\mathcal{S}(x) = 0$, $\mathcal{K}(y, x) = 1$ for a single edge $(y, x) \in E^s$, and $\mathcal{K}(x, y) = 2$ for a single edge $(x, y) \in E^s$;
- for $x \in \tilde{V}(T_q)$, it holds that $\mathcal{S}(x) = 0$, and $\mathcal{K}(y, x) = 2$ for a single edge $(y, x) \in E^s$.

Combining everything together, we get

$$\lambda_s = \max_{x \in V(T_{sq})} \left(\mathcal{S}(x) + \sum_{(y,x) \in E^s} \mathcal{K}(y,x) + \sum_{(x,y) \in E^s} \mathcal{K}(x,y) \right) \leq 3.$$

Therefore, subdivision D^{sq} of G admits a 3-stack layout by Lemma 11. ◀

With a similar technique we can find a 1-stack 1-queue subdivision (Lemma 13) instead of a 3-stack subdivision (Lemma 12). The proof of the next result is given in [21].

▶ **Lemma 13.** *Let G be an s -stack q -queue graph. Then G has a 1-stack 1-queue subdivision with at most $4\lceil \log_2(\max(s, q)) \rceil + 6$ division vertices per edge.*

We now prove Theorem 4, in particular that Open Problems 1 and 2 are equivalent.

► **Theorem 4.** *The following are equivalent:*

- (1) *every s -stack graph has queue number at most $f(s)$ for some function f (“queue number is bounded by stack number”);*
- (2) *every 3-stack graph has queue number at most C for some constant C ;*
- (3) *every 1-stack 1-queue graph has queue number at most C for some constant C ;*
- (4) *every s -stack q -queue graph has queue number at most $f(s, q)$ for some function f (“queue number is bounded by mixed number”).*

Proof. It is immediate that (4) implies (1), (2), and (3). That (1) \iff (2) is proven by Dujmović and Wood [11].

Now we show that (2) \implies (4). Assume that every 3-stack graph has a bounded queue number, $C \in \mathcal{O}(1)$. Let G be an s -stack q -queue graph. By Lemma 12, G admits a 3-stack subdivision, D , with at most $k = 2\lceil \log_2(\max(s, q)) \rceil + 3$ division vertices per edge. By the assumption, $\text{qn}(D) \leq C$. Note that G is an r -shallow minor of D for $r = (k + 1)/2 = \lceil \log_2(\max(s, q)) \rceil + 2$. By Lemma 9, we get

$$\text{qn}(G) \leq (2r + 1)(2\text{qn}(D))^{2r+1} = (2\lceil \log_2(\max(s, q)) \rceil + 5)(2C)^{2\lceil \log_2(\max(s, q)) \rceil + 5},$$

which proves that G has a bounded queue number, as long as s, q and C are constants.

Similarly we show that (3) \implies (4). Suppose that every 1-stack 1-queue graph has queue number at most $C \in \mathcal{O}(1)$. Let G be an s -stack q -queue graph. By Lemma 13, G admits a 1-stack 1-queue subdivision, D , with $k = 4\lceil \log_2(\max(s, q)) \rceil + 6$ division vertices per edge. By the assumption, $\text{qn}(D) \leq C$, and G is an r -shallow minor of D for $r = \lceil (k + 1)/2 \rceil$. Again by Lemma 9, we get

$$\text{qn}(G) \leq (2r + 1)(2\text{qn}(D))^{2r+1} = (4\lceil \log_2(\max(s, q)) \rceil + 8)(2C)^{4\lceil \log_2(\max(s, q)) \rceil + 8},$$

which shows that G has a bounded queue number, as long as s, q and C are constants. ◀

4 Separated Layouts

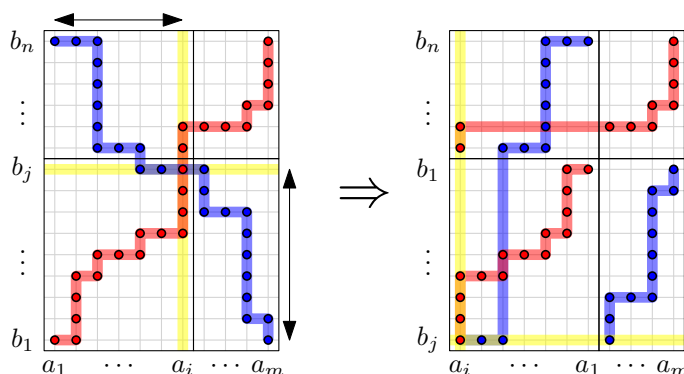
In this section we investigate separated mixed layouts of bipartite graphs and Open Problem 3. Recall that a vertex order σ of a bipartite graph $G = (V, E)$ with bipartition (A, B) is *separated* if all vertices of A precede all vertices of B in σ (or vice versa).

► **Observation 14.** *Every separated s -stack q -queue layout of a bipartite graph $G = (V, E)$ with bipartition (A, B) can be transformed to a separated q -stack s -queue layout by reversing the order of all vertices in A (or alternatively all vertices in B).*

Reversing the order of some consecutive vertices (as in Observation 14) is the simplest modification we could do to a given layout. It turns out that this is already enough to show that graphs with separated 1-stack 1-queue layouts have a constant queue number.

► **Theorem 15.** *Every separated 1-stack 1-queue graph admits a separated 4-queue layout.*

Proof. Let $G = (V, E)$ be a bipartite graph with bipartition (A, B) admitting a separated 1-stack 1-queue layout with vertex order σ . Consider the reduced adjacency matrix M with columns $A = (a_1, \dots, a_m)$ and rows $B = (b_1, \dots, b_n)$ ordered according to σ . The edges $S \subset E$ in the stack form a monotonically weakly decreasing subset of $|A| \times |B|$. The edges $Q \subset E$ in the queue form a monotonically weakly increasing subset of $|A| \times |B|$. Without loss of generality we can assume (by adding edges to the graph) that S and Q correspond to inclusion-maximal such subsets; see Figure 6.



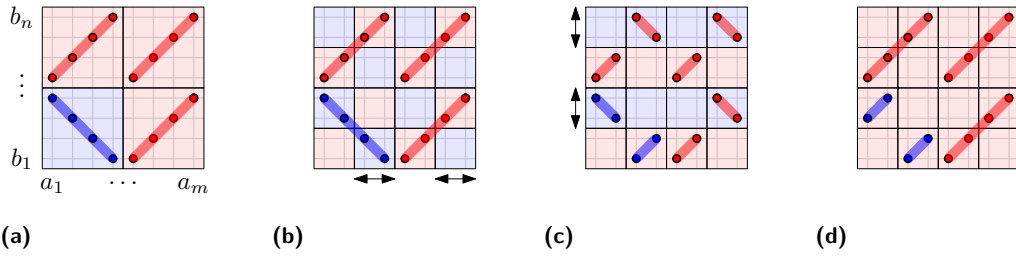
■ **Figure 6** Obtaining a separated 4-queue layout from a separated 1-stack 1-queue layout by reversing column and row orders. The blue edge color of the transformed stack edges is preserved for clarity.

Let $(a_i, b_j) \in A \times B$ be a point/edge that is contained in both, S and Q . Now consider the vertex order $a_i, \dots, a_1, a_{i+1}, \dots, a_m$ of the columns obtained by reversing the order of a_1, \dots, a_i . Similarly, consider the vertex order $b_j, \dots, b_1, b_{j+1}, \dots, b_n$ of the rows obtained by reversing the order of b_1, \dots, b_j . Let σ' denote the resulting separated vertex order of G . Now the edges in Q incident to a_1, \dots, a_i form a queue Q_1 in σ' . Also the remaining edges in Q , incident to a_{i+1}, \dots, a_m form a queue Q_2 in σ' . Similarly, the edges in S incident to b_1, \dots, b_j form a queue Q_3 in σ' . Also the remaining edges in S , incident to b_{j+1}, \dots, b_m form a queue Q_4 in σ' ; see again Figure 6. This is a separated 4-queue layout of G . ◀

Note that by applying Observation 14, separated 1-stack 1-queue graphs also admit a separated 4-stack layout. However, we do not know whether the bound of 4 in Theorem 15 is best-possible and remark that $K_{3,3}$ admits a separated 1-stack 1-queue layout but its separated stack/queue number is 3.

The simple operation of reversing the order of some consecutive vertices can be employed in a “checkerboard” fashion. Consider a given separated mixed layout (e.g., with s stacks and q queues). Assume we have partitioned the reduced adjacency matrix by means of a superimposed grid with a checkerboard odd-even pattern, in such a way that odd grid cells contain no stack edges and even grid cells contain no queue edges. Then, the vertex order of every second row and column can be reversed to derive a separated pure queue layout. Finding such a grid refinement is always possible, that is, down to individual rows and columns, however the derived \overline{sn} and \overline{qn} are dependent on the grid granularity and thus may not be bounded by \overline{mn} . An example for specific separated s -stack q -queue layouts with bounded \overline{qn} number are grids where each cell contains either an increasing or decreasing diagonal. In this case, grid columns and rows can be halved in order to apply the checkerboard approach; see Figure 7.

However there exist graphs, where such operations do not suffice and more involved row and column permutations are necessary. In [21] we provide as a challenging example a subcubic bipartite graph that admits a separated 1-stack 2-queue layout and a fairly simple 4-queue pure layout. However, the grid granularity has to be super-constant in order to transform the mixed into the pure layout. We conclude that even for low-degree graphs with $\overline{mn}(G) = 3$, it is not obvious how to transform a separated mixed layout into a separated pure layout, say with few queues.



■ **Figure 7** Transforming a separated mixed layout of a grid with diagonals (a) into a separated pure layout (d) by halving columns and rows (b) and reversing every second row and column (c). The blue edge color of the transformed stack edges is preserved for clarity.

While we do not know how to generalize Theorem 15 even just to separated 1-stack 2-queue layouts, we can narrow down the difficult case of Open Problem 3. If Open Problem 3 is answered positively, we can find an $f(s, q)$ -queue layout for any bipartite graph admitting a separated s -stack q -queue layout. In Theorem 5 we show three equivalent formulations of this problem. In the challenging example presented in our preprint [21], the transformation from the separated mixed to the separated pure layout applies the same column and row permutation. Now, Theorem 5 states that we may always assume that rows and columns are identically permuted when transforming a separated mixed layout into a separated pure layout.

Further, we show in Lemma 17 that every separated s -stack q -queue layout can be transformed into a separated 1-stack $f(s, q)$ -queue layout, while this transformation does not change the separated queue number by too much. This implies that one can restrict the attention to separated 1-stack q -queue layouts, for solving Open Problem 3.

► **Theorem 5.** *The following are equivalent:*

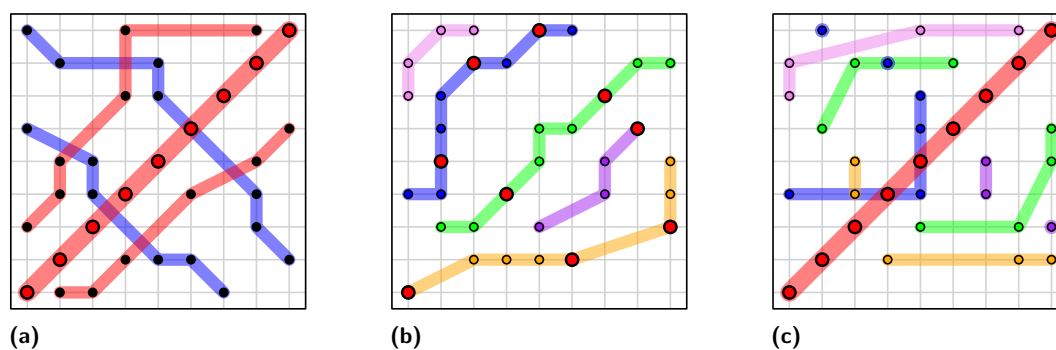
- (1) every separated s -stack q -queue graph G has queue number at most $f(s, q)$ for some function f (“queue number is bounded by mixed number”);
- (2) every separated s -stack q -queue layout of a graph $G = (A \cup B, E)$ with $|A| = |B|$ can be transformed into a separated $f(s, q)$ -queue layout for some function f by applying the same permutation to A and B ;
- (3) every separated 1-stack q -queue graph has queue number at most $f(q)$ for some function f ;
- (4) every separated 1-stack 6-queue graph has queue number at most C for some constant C .

Proof. Observe that (1) immediately implies (3), which implies (4). That (2) implies (1) is also immediate, when adding isolated vertices to guarantee $|A| = |B|$. The proofs of the other directions are deferred, in particular see Lemma 16 for (1) \Rightarrow (2) and Lemma 17 for (3) \Rightarrow (1). For (4), we show equivalence by proving that (4) implies (3) in Lemma 18. ◀

► **Lemma 16.** *If there is a function f such that every separated s -stack q -queue graph admits a separated $f(s, q)$ -queue layout, then every separated s -stack q -queue layout of a bipartite graph $G = (A \cup B, E)$ with $|A| = |B|$ can be transformed into a separated $2f(s, q + 1)^2$ -queue layout by applying the same permutation to A and B .*

In other words, Lemma 16 proves that (1) implies (2) in Theorem 5.

Proof. Let u_1, \dots, u_n and v_1, \dots, v_n be the order of A and B in the given separated s -stack q -queue layout of G . We add the “identity” queue Q , where for each i , $1 \leq i \leq n$, there is the edge (u_i, v_i) (if not already present in G). Let $G' = (A \cup B, E \cup Q)$ be the resulting bipartite



■ **Figure 8** Illustration of the proof of Lemma 16: Adding the identity queue (a), obtaining a separated queue layout (b), and restoring the identity queue (c).

graph. By assumption, there exists a separated $f(s, q + 1)$ -queue layout of G' with vertex order σ_0 . We are looking for another vertex order σ such that, compared to the initial order, A and B are permuted the same, that is, the columns and rows in the reduced adjacency matrix are identically permuted. Since we added the identity queue Q to obtain G' in the beginning, this is equivalent to restoring Q to the diagonal in the matrix; see Figure 8.

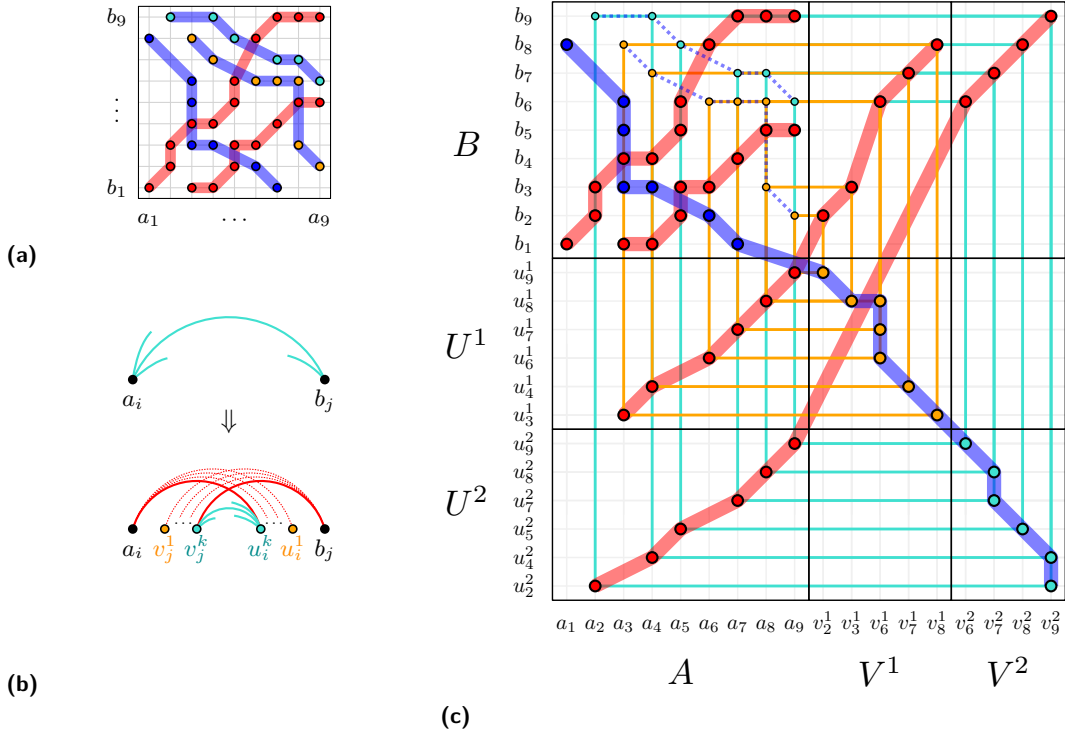
To that end, we shall apply Lemma 6 to change only the order of vertices in A , which corresponds to permuting the columns in the matrix. Let $k = f(s, q + 1)$ and E_1, \dots, E_k be the queues in the $f(s, q + 1)$ -queue layout of G' and $A_i = \{u \in A \mid (u, v) \in Q \cap E_i\}$, $1 \leq i \leq k$. We want to apply the Riffle-Lemma (Lemma 6-(2)) with the partition of V into A_1, \dots, A_k and B . To this end, we keep the order of B (the rows), and restore Q as a diagonal by simply sorting the columns according to their row in Q . Since each E_i is a queue, the relative order within each A_i is kept intact, as required by Lemma 6. Hence, by Lemma 6-(2), the resulting layout has at most $2 \cdot f(s, q + 1) \cdot f(s, q + 1)$ queues in total. ◀

► **Lemma 17.** *Let G be a bipartite graph with a separated s -stack q -queue layout. Then there exists another bipartite graph H admitting a separated 1-stack $(s + q - 1)$ -queue layout, where G is a 1-shallow minor of H , such that $\text{qn}(G) \in \mathcal{O}(\text{qn}(H)^3)$.*

If (3) of Theorem 5 holds, then the separated 1-stack $(s + q - 1)$ -queue layout of H has bounded queue number, i.e. $\text{qn}(G) \in \mathcal{O}(\text{qn}(H)^3) \leq f(s + q - 1)$ for some function f . Hence $\text{qn}(G) \leq f'(s, q) = f(s + q - 1)$, implying (1) of Theorem 5.

Proof. Assume we have a bipartite graph G , that admits a separated s -stack q -queue layout. We will build a new bipartite graph H , such that H has a separated 1-stack $(s + q - 1)$ -queue layout. We will then show that for the separated queue numbers of G and H , it holds that $\text{qn}(G) \in \mathcal{O}(\text{qn}(H)^3)$. That is, a small queue number of H implies a small queue number of G . Next we make the argument formal.

Let $G = (A \cup B, E)$, $|A| = n$, $|B| = m$, S_1, \dots, S_s be the stacks and Q_1, \dots, Q_q be the queues in the separated s -stack q -queue layout. We build H as follows. The vertices of H are $A \cup (\bigcup_k U^k) \cup B \cup (\bigcup_k V^k)$, where $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_m\}$ are copied from G , while $U^k \subseteq \{u_1^k, \dots, u_n^k\}$ and $V^k \subseteq \{v_1^k, \dots, v_m^k\}$ for any $1 \leq k \leq s - 1$ are new. U^k contains a vertex u_i^k for each vertex a_i incident to an edge $(a_i, b_j) \in S_k$. Likewise, V^k contains a vertex v_j^k for each such b_j . Vertices in A and B that are not incident to an edge in S_k are omitted in U^k and V^k . (Note that for each stack S_k , the new vertices in U^k and V^k are added to construct parts of the graph H starting from G , while the last stack S_s is left as is.) In the reduced adjacency matrix of H , the new vertices are represented by additional



■ **Figure 9** Transforming a separated 3-stack 2-queue layout into a separated 1-stack 4-queue layout: The initial graph G (a), modifying a stack edge (a_i, b_j) (b), the constructed graph H (c).

blocks of n_k consecutive rows for each U^k and m_k consecutive columns for each V^k , where n_k (m_k) is the number of vertices of A (B) having an edge in S_k . Row blocks of U^k and column blocks of V_k are subsequent for $k = 1, \dots, s - 1$. Intuitively, we move the stack S_k from $A \times B$ to $U^k \times V^k$, $k = 1, \dots, s - 1$. We leave the last stack S_s with all q initial queues in $A \times B$. Finally, we put one new queue each in $A \times U^k$ and $V^k \times B$, $k = 1, \dots, s - 1$. Figure 9 shows the construction and in particular the order of the rows U^1, \dots, U^{s-1} and columns V^1, \dots, V^{s-1} .

More formally, for every edge $(a_i, b_j) \in Q_1 \cup \dots \cup Q_q$, there is an edge (a_i, b_j) in H . For every stack edge, (a_i, b_j) of stack S_k , $1 \leq k \leq s - 1$, there are three edges (a_i, u_i^k) , (u_i^k, v_j^k) , (v_j^k, b_j) in H . (In case of multiple edges between a pair of vertices, we keep only one instance.) It is easy to verify that H admits a separated 1-stack $(s + q - 1)$ -queue layout, as for every original stack S_k , $1 \leq k \leq s - 1$ the edges (a_i, u_i^k) and (v_j^k, b_j) for every $(a_i, b_j) \in S_k$ form a queue respectively, due to the increasing column and row indices of each block U^k and V^k starting from the bottom left of the matrix. In fact, edges between vertices in A and U^k and edges between vertices in B and V^k fit in a single queue, since the former lie below and to the left of the latter in the matrix. The original queues remain, yielding $q + s - 1$ queues in total. For a stack S_k , the edges $(a_i, b_j) \in S_k$ are a decreasing subset in $A \times B$ and due to the increasing ordering of indices in every column and row block of the matrix, the edges (u_i^k, v_j^k) in $U^k \times V^k$ have the same property. The ordering of the row blocks U^k and column blocks V^k , where $A \times B$ and $U^k \times V^k$, $1 \leq k \leq s - 1$ are positioned diagonally from the top left to the bottom right of the matrix, then allows to combine the s stacks of G into one single stack in H .

Finally, we show that $\text{qn}(G) \in \mathcal{O}(\text{qn}(H)^3)$. To this end, consider a $\text{qn}(H)$ -queue layout of H . Now, contract for each $i = 1, \dots, n$ the vertex a_i with its neighbors of the form u_i^k , $1 \leq k \leq s - 1$. Similarly, contract for each $j = 1, \dots, m$ the vertex b_j with its neighbors of the form v_j^k , $1 \leq k \leq s - 1$. The result is a 1-shallow minor of H , and most crucially, the result is exactly the initial graph G . Hence, Lemma 9 with radius $r = 1$ gives that $\text{qn}(G) \leq (2r + 1)(2 \text{qn}(H))^{2r+1} = 24 \cdot \text{qn}(H)^3$. ◀

Using the techniques from Section 3, we can provide the final missing piece of Theorem 5.

► **Lemma 18.** *Let G be a bipartite graph with a separated 1-stack q -queue layout. Then G has a subdivision D with at most $2 \lceil \log_2 q \rceil$ division vertices per edge that admits a separated 1-stack 6-queue layout.*

In other words, Lemma 18 proves that (4) implies (3) in Theorem 5, as applying Lemma 9 for the r -shallow minor G of D with $k = 2 \lceil \log_2 q \rceil$ and $r = \lceil (k + 1)/2 \rceil$ implies bounded queue number of G for a function $f(q)$.

$$\text{qn}(G) \leq (2r + 1)(2 \text{qn}(D))^{2r+1} = (2 \lceil \log_2 q \rceil + 2)(12)^{2 \lceil \log_2 q \rceil + 2}$$

Proof. Let $G = (V, E_s \cup E_1 \cup \dots \cup E_q)$ be a bipartite graph that admits a separated 1-stack q -queue layout such that E_s is a stack and E_i , $1 \leq i \leq q$ are queues. Denote $h = \lceil \log_2 q \rceil$ so that $q \leq 2^h$. We consider the subgraph of G induced by the queue edges, $G^q = (V, E_1 \cup \dots \cup E_q)$. Let T be a complete binary tree of height h . By Lemma 10, there exists a subdivision of G^q , denoted D^q , and a simple T -layout of D^q in which $\mathcal{S}(x) = 0$, $\mathcal{Q}(x) = 1$ for all $x \in \tilde{V}(T)$, $\mathcal{S}(x) = \mathcal{Q}(x) = 0$ for all non-leaf nodes $x \in V(T) \setminus \tilde{V}(T)$, and $\mathcal{K}(x, y) = 1$ for all $(x, y) \in E(T)$. Note that D^q contains exactly $2h$ division vertices per edge and it is bipartite; see Figure 10.

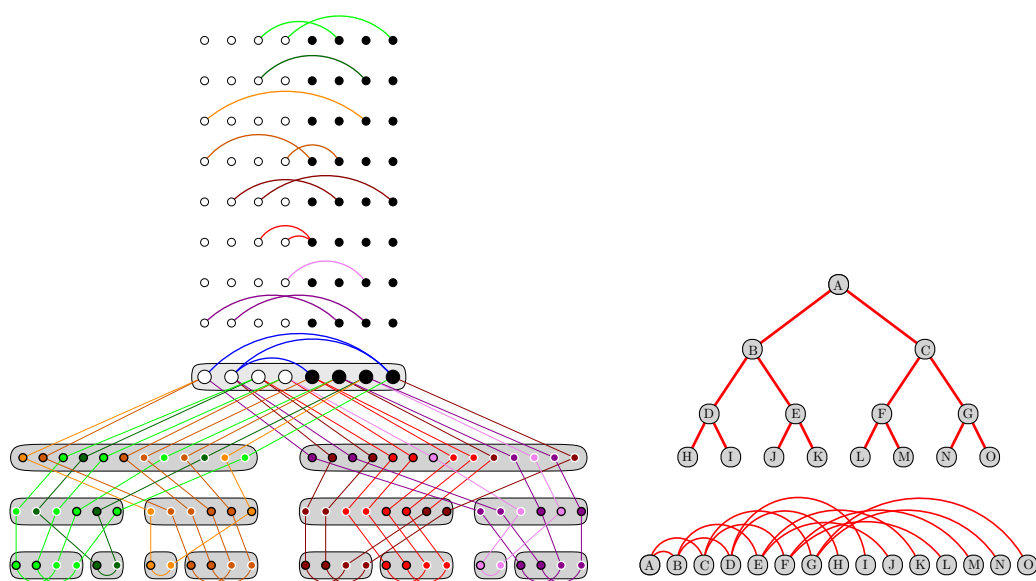
Now, color all edges of T red and find a 1-queue layout of T via a breadth-first search traversal such that every node precedes its children in the order. By Lemma 11 applied to graph G^q and its simple T -layout, we obtain a linear layout of D^q . Let us argue that this result is in fact a 3-queue layout of D^q , i.e., that $\lambda_s = 0$ and $\lambda_q \leq 3$ as defined in Lemma 11. On the one hand, $\mathcal{S}(x) = 0$ for all the nodes of T and there are no blue edges in T . Thus, we have $\lambda_s = 0$. On the other hand, every node $x \in V(T)$ has at most two outgoing edges in the layout of T , i.e., the set $\{(y, z) \in E^q \mid y \leq_\sigma x \leq_\sigma z\}$ contains at most two edges. Hence,

$$\begin{aligned} \lambda_q &= \max_{x \in V(T)} \left(\mathcal{Q}(x) + \max_{y \in V(T): y \leq_\sigma x} \sum_{(y,z) \in E^q: x \leq_\sigma z} \mathcal{K}(y, z) \right) \\ &\leq \max_{x \in V(T)} \left(1 + \max_{y \in V(T): y \leq_\sigma x} 2 \right) \leq 3. \end{aligned}$$

The final step is to convert the 3-queue layout of D^q into a separated layout. This is accomplished by Corollary 7, which in worst case doubles the number of queues. The transformations of Lemma 11 and Corollary 7 keep the relative order of the original vertices V unchanged. Thus, the resulting layout of D^q can be joined with the stack edges E_s to finally yield a subdivision D of G (in which the stack edges are not subdivided) with a separated 1-stack 6-queue layout. ◀

5 Conclusions and Open Questions

We have explored the relation between mixed and pure stack or queue layouts, in the separated as well as in the non-separated setting. An interesting (and somewhat surprising) result is the equivalence of Open Problems 1 and 2, which we believe sheds some light on the famous open problem, whether bounded stack number always implies bounded queue number. Let us highlight a few intriguing questions and possible directions in the area.



(a) A tree-layout of the subdivision of a separated 1-stack 8-queue graph G . Colors of edges and division vertices in the tree-layout correspond to the original queues and stack of G shown above the root node, while vertex outline colors in non-root nodes correspond to the vertex partitions in G .

(b) A tree-partition and a layout of the corresponding binary tree T , where the red edge color corresponds to Lemma 11.

■ **Figure 10** An illustration for Lemma 18: Subdividing a separated 1-stack 8-queue graph G with at most $2\lceil \log_2 q \rceil = 6$ division vertices per edge to obtain a mixed 1-stack 6-queue graph.

1. Do graphs with (non-separated) 1-stack 1-queue layouts have a constant queue number? This might be hard in general, but one could for example start with subcubic graphs.
2. Do graphs with separated 1-stack 2-queue layouts have a constant queue number? Lemmas 16 and 18 combined show that separated 1-stack 6-queue graphs are as hard as the general case, and we feel that the same holds for separated 1-stack 2-queue graphs. On the other hand, Theorem 15 solves the separated 1-stack 1-queue case.
3. Is there a constant C such that every bipartite 1-stack 1-queue graph admits a separated C -stack C -queue layout? A positive answer would imply that Open Problems 1–3 are all equivalent.

References

- 1 Jawaherul Md Alam, Michael A Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. *Algorithmica*, pages 1–22, 2020. doi:10.1007/s00453-020-00697-4.
- 2 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, April 2023. doi:10.1093/comjnl/bxad038.
- 3 János Barát, Jirí Matousek, and David R. Wood. Bounded-degree graphs have arbitrarily large geometric thickness. *The Electronic Journal of Combinatorics*, 13(1), 2006. doi:10.37236/1029.
- 4 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. On the queue number of planar graphs. In *International Symposium on Graph Drawing and Network Visualization*, pages 271–284, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-92931-2_20.

- 5 Michael A. Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi N. Raftopoulou, and Torsten Ueckerdt. Four pages are indeed necessary for planar graphs. *Journal of Computational Geometry*, 11(1):332–353, 2020. doi:10.20382/jocg.v11i1a12.
- 6 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.
- 7 Vida Dujmović, David Eppstein, Robert Hickingbotham, Pat Morin, and David R Wood. Stack-number is not bounded by queue-number. *Combinatorica*, pages 1–14, 2021. doi:10.1007/s00493-021-4585-7.
- 8 Vida Dujmović, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R Wood. Planar graphs have bounded queue-number. *Journal of the ACM*, 67(4):1–38, 2020. doi:10.1145/3385731.
- 9 Vida Dujmović, Pat Morin, and David R Wood. Layout of graphs with bounded tree-width. *SIAM Journal on Computing*, 34(3):553–579, 2005. doi:10.1137/S0097539702416141.
- 10 Vida Dujmović, Attila Pór, and David R Wood. Track layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):497–522, 2004. doi:10.46298/DMTCS.315.
- 11 Vida Dujmović and David R Wood. Stacks, queues and tracks: Layouts of graph subdivisions. *Discrete Mathematics and Theoretical Computer Science*, 7:155–202, 2005. doi:10.46298/dmtcs.346.
- 12 Peter Eades and Nicholas C Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.
- 13 David Eppstein, Robert Hickingbotham, Laura Merker, Sergey Norin, Michał T Seweryn, and David R Wood. Three-dimensional graph products with unbounded stack-number. *Discrete & Computational Geometry*, pages 1–28, 2023. doi:10.1007/s00454-022-00478-6.
- 14 Henry Förster, Michael Kaufmann, Laura Merker, Sergey Pupyrev, and Chrysanthi N. Raftopoulou. Linear layouts of bipartite planar graphs. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2023. doi:10.1007/978-3-031-38906-1_29.
- 15 Joseph L. Ganley and Lenwood S. Heath. The pagenumber of k-trees is $O(k)$. *Discrete Applied Mathematics*, 109(3):215–221, 2001. doi:10.1016/S0166-218X(00)00178-5.
- 16 Deborah Haun, Laura Merker, and Sergey Pupyrev. Forbidden patterns in mixed linear layouts. In *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, 2024. doi:10.48550/arXiv.2412.12786.
- 17 Lenwood S Heath, Frank Thomson Leighton, and Arnold L Rosenberg. Comparing queues and stacks as machines for laying out graphs. *SIAM Journal on Discrete Mathematics*, 5(3):398–412, 1992. doi:10.1137/0405031.
- 18 Lenwood S Heath and Arnold L Rosenberg. Laying out graphs using queues. *SIAM Journal on Computing*, 21(5):927–958, 1992. doi:10.1137/0221055.
- 19 Robert Hickingbotham and David R. Wood. Shallow minors, graph products, and beyond-planar graphs. *SIAM Journal on Discrete Mathematics*, 38(1):1057–1089, 2024. doi:10.1137/22M1540296.
- 20 Paul C. Kainen. Some recent results in topological graph theory. In Ruth A. Bari and Frank Harary, editors, *Graphs and Combinatorics*, pages 76–108, Berlin, Heidelberg, 1974. Springer Berlin Heidelberg. doi:10.1007/BFb0066436.
- 21 Julia Katheder, Michael Kaufmann, Sergey Pupyrev, and Torsten Ueckerdt. Transforming stacks into queues: Mixed and separated layouts of graphs. *CoRR*, abs/2409.17776, 2024. doi:10.48550/arXiv.2409.17776.
- 22 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. The mixed page number of graphs. *Theoretical Computer Science*, 931:131–141, 2022. doi:10.1016/j.tcs.2022.07.036.

56:18 Transforming Stacks into Queues: Mixed and Separated Layouts of Graphs

- 23 Hiroshi Nagamochi. An improved bound on the one-sided minimum crossing number in two-layered drawings. *Discrete & Computational Geometry*, 33(4):569–591, 2005. doi:10.1007/S00454-005-1168-0.
- 24 L Taylor Ollmann. On the book thicknesses of various graphs. In *Proc. 4th Southeastern Conference on Combinatorics, Graph Theory and Computing*, volume 8, page 459. Utilitas Math., 1973.
- 25 Sriram Venkata Pemmaraju. *Exploring the powers of stacks and queues via graph layouts*. PhD thesis, Virginia Polytechnic Institute and State University, 1992.
- 26 Sergey Pupyrev. Mixed linear layouts of planar graphs. In *International Symposium on Graph Drawing and Network Visualization*, pages 197–209. Springer, 2017. doi:10.1007/978-3-319-73915-1_17.
- 27 Sergey Pupyrev. Improved bounds for track numbers of planar graphs. *Journal of Graph Algorithms and Applications*, 24(3):323–341, 2020. doi:10.7155/JGAA.00536.
- 28 Veit Wiechert. On the queue-number of graphs with bounded tree-width. *The Electronic Journal of Combinatorics*, 24(1):P1.65, 2017. doi:10.37236/6429.
- 29 David R. Wood. Bounded-degree graphs have arbitrarily large queue-number. *Discrete Mathematics and Theoretical Computer Science*, 10(1), 2008. doi:10.46298/DMTCS.434.
- 30 Mihalis Yannakakis. Embedding planar graphs in four pages. *Journal of Computer and System Sciences*, 38(1):36–67, 1989. doi:10.1016/0022-0000(89)90032-9.

Approximate Minimum Tree Cover in All Symmetric Monotone Norms Simultaneously

Matthias Kaul  

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany
University of Bonn, Germany

Kelin Luo  

University of Bonn, Germany
University at Buffalo, NY, USA

Matthias Mnich  

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Heiko Röglin  

Universität Bonn, Germany

Abstract

We study the problem of partitioning a set of n objects in a metric space into k clusters V_1, \dots, V_k . The quality of the clustering is measured by considering the vector of cluster costs and then minimizing some monotone symmetric norm of that vector (in particular, this includes the ℓ_p -norms). For the costs of the clusters we take the weight of a minimum-weight spanning tree on the objects in V_i , which may serve as a proxy for the cost of traversing all objects in the cluster, for example in the context of Multirobot Coverage as studied by Zheng, Koenig, Kempe, Jain (IROS 2005), but also as a shape-invariant measure of cluster density similar to Single-Linkage Clustering.

This problem has been studied by Even, Garg, Könemann, Ravi, Sinha (*Oper. Res. Lett.*, 2004) for the setting of minimizing the weight of the largest cluster (i.e., using ℓ_∞) as MIN-MAX TREE COVER, for which they gave a constant-factor approximation algorithm. We provide a careful adaptation of their algorithm to compute solutions which are approximately optimal with respect to *all* monotone symmetric norms *simultaneously*, and show how to find them in polynomial time. In fact, our algorithm is purely combinatorial and can process metric spaces with 10,000 points in less than a second.

As an extension, we also consider the case where instead of a target number of clusters we are provided with a set of *depots* in the space such that every cluster should contain at least one such depot. One can consider these as the fixed starting points of some agents that will traverse all points of a cluster. For this setting also we are able to give a polynomial-time algorithm computing a constant-factor approximation with respect to all monotone symmetric norms simultaneously.

To show that the algorithmic results are tight up to the precise constant of approximation attainable, we also prove that such clustering problems are already APX-hard when considering only one single ℓ_p norm for the objective.

2012 ACM Subject Classification Theory of computation \rightarrow Approximation algorithms analysis

Keywords and phrases Clustering, spanning trees, all-norm approximation

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.57

Related Version *Full Version:* <https://arxiv.org/abs/2501.05048> [18]

Funding *Matthias Kaul:* Most work done while at the Hamburg University of Technology.

Kelin Luo: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 390685813.

Heiko Röglin: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459420781 and by the Lamarr Institute for Machine Learning and Artificial Intelligence lamarr-institute.org.



© Matthias Kaul, Kelin Luo, Matthias Mnich, and Heiko Röglin;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 57; pp. 57:1–57:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

A typical clustering problem takes as input a set of n objects in a metric space (V, d) , and seeks a partition of these objects into k clusters V_1, \dots, V_k , so as to optimize some objective function. For example, we might try to place k facilities onto the nodes of an edge-weighted graph with node set V , and then assign each remaining node to some facility. To model the cost of serving these nodes from their facility, one can use the cost of a minimum-weight spanning tree T_i on the subgraph induced by them. For $i = 1, \dots, k$, let $w(T_i) = \sum_{e \in E(T_i)} w(e)$ be the weight of tree T_i . Historically, these kinds of problems have been studied for two different objectives. On the one hand, in the MIN-SUM TREE COVER problem one might want to find k trees T_1, \dots, T_k to cover all nodes in V , while minimizing the total length of the service network, i.e., the sum $\sum_{i \in [k]} w(T_i)$ of the weights of the spanning trees. On the other hand, it is desirable that each facility does not serve too large of a network, so we can instead minimize the weight $\max_{i \in [k]} w(T_i)$ of the heaviest tree, which leads to the MIN-MAX TREE COVER problem [10, 19].

Many fundamental clustering problems were studied under this min-sum objective and min-max objective. These objectives can equivalently be considered as that of minimizing the 1-norm, or the ∞ -norm, of the clustering. Examples include the k -median problem and the minimum-load k -facility location problem. These two problems are siblings in that they both deal with the assignment of points (or clients) to k centers (or facilities), with the costs of connecting those points to respective centers. Each point v is then assigned to one of these open centers, denoted as $c(v)$. The cost cost_c for each center c , which also reflects the cost associated with each facility, is calculated as the sum of distances between the center and all the points allocated to it, i.e., $\text{cost}_c = \sum_{v \in V: c(v)=c} d(v, c)$. In the k -median problem [14, 7, 21, 2], the objective is to minimize the ℓ_1 -norm of $\{\text{cost}_c\}_c$, i.e., $\sum_c \text{cost}_c$, which represents the total cost of all clusters; whereas the minimum-load k -facility location problem [1] seeks to minimize the maximum load of an open facility, symbolised by the ℓ_∞ -norm function of $\{\text{cost}_c\}_c$, i.e., $\max_c \text{cost}_c$.

There is a diverse array of cost functions that could be applicable to a variety of problem domains, and often, efficient algorithms are crafted to suit each specific objective. However, it is crucial to note that an optimal solution for one objective may not perform well for another. For instance, a solution for an instance of the TREE COVER that minimizes the ℓ_1 -norm might be particularly inefficient when it comes to minimizing the ℓ_∞ -norm, and vice versa (see examples in Figure 1a and Figure 1b). Therefore, one may wonder what the “generally optimal” solution would be for a given problem. Finding such a generally optimal solution is the task of the following problem:

► **Definition 1** (All-norm clustering problem). *An all-norm clustering problem takes as input a metric space (V, d) , a cost function $w : \mathcal{P}(V) \rightarrow \mathbb{R}_{\geq 0}$, and a positive integer k . The goal is to partition V into clusters V_1, \dots, V_k which minimize*

$$\alpha = \max_{p \in \mathbb{R}_{\geq 1} \cup \{\infty\}} \frac{(\sum_i (w(V_i))^p)^{1/p}}{OPT_p},$$

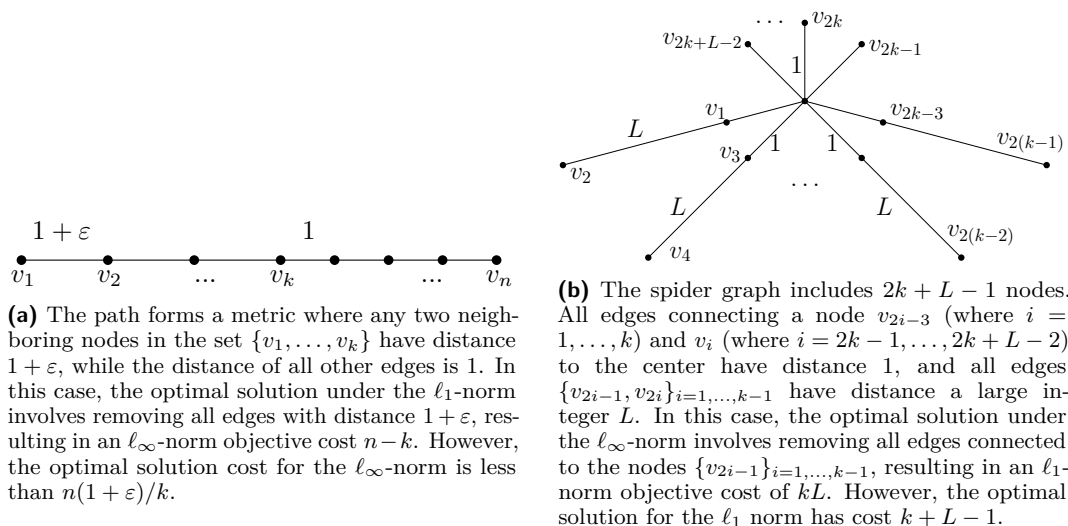
where OPT_p denotes the value of an optimal solution for the k -clustering problem under the ℓ_p -norm objective. Here, α is referred to as the all-norm approximation factor.

Our focus in this paper is the all-norm clustering problem where the cost $w(V_i)$ of each cluster is the cost of a minimum-weight spanning tree on V_i with respect to the metric d . We call this problem the ALL-NORM TREE COVER problem in line with the naming convention

in the literature, and denote its instances by (V, d, w, k) . (Note that, for simplicity, we only consider ℓ_p -norms here. However, the proofs turn out to work for any norm which is monotone and symmetric, cf. Ibrahimpur and Swamy [16] for a detailed introduction to such norms.) Observe that the number k of clusters is part of the input, i.e., it is *not* fixed.

The choice of the cost of a minimum-weight spanning tree might appear to be somewhat arbitrary here, but spanning trees are the key connectivity primitive in network design problems. Any constant-factor approximation for ALL-NORM TREE COVER will, for example, transfer to a constant-factor approximation for the all-norm version of the MULTIPLE TRAVELING SALESPERSON PROBLEM [5] where we ask to cover a metric space by k cycles instead of trees. This use of spanning trees as a proxy for the traversal times of the clusters was a key ingredient for by Zheng et al. [24] to partition a floorplan into similar-size areas to be served by different robots.

Let us observe that solving the ALL-NORM TREE COVER problem is non-trivial, as a solution which is good for one objective (i.e., for one particular norm ℓ_p) may be bad for another objective (i.e., for another norm $\ell_{p'}$), and vice versa. For examples of this phenomenon, see Figure 1.



■ **Figure 1** Two instances of the ALL-NORM TREE COVER problem. Figure 1a is an instance where an optimal ℓ_1 -norm solution does not return a good approximation in the ℓ_∞ -norm; Figure 1b is an instance where an optimal ℓ_∞ -norm solution does not return a good approximation in the ℓ_1 -norm.

The ALL-NORM TREE COVER problem, alongside the related path cover and cycle cover problems, involves covering a specified set of nodes of a(n edge-weighted) graph with a limited number of subgraphs. These problems have attracted significant attention from the operations research and computer science communities due to their practical relevance in fields like logistics, network design, and data clustering. For instance, these problems are naturally applicable in scenarios such as vehicle routing, where the task involves designing optimal routes to service a set of customers with a finite number of vehicles. Different optimization objectives could be considered depending on the specific requirements. One could aim to minimize the maximum waiting time for any customer, an objective that is equivalent to the ℓ_∞ -norm of the cost function associated with each vehicle. This ensures fairness, as it attempts to prevent any single customer from waiting excessively long. Alternatively, one could aim to minimize the total travel time or cost, which corresponds to the ℓ_1 -norm of

the cost function across all vehicles [6, 4]. This objective seeks overall efficiency, making it beneficial from an operational perspective as it reduces fuel consumption and allows for more customers to be serviced within the same time frame [11, 4, 10, 19]. Understanding these problems in an all-norm setting thus enables the development of routing strategies that are adaptable to different priorities, including customer satisfaction, operational efficiency, or a balance between the two. Given that minimum spanning trees provide constant-factor approximations to traveling salesperson tours [8], we explore the possibility of covering the nodes of a graph with k trees.

In a scenario where each vehicle already has an assigned station, and the task is limited to the assignment of nodes to the vehicles, we are confronted with a variation of the ALL-NORM TREE COVER problem known as the ALL-NORM TREE COVER WITH DEPOTS problem (also commonly referred to as the ROOTED ALL-NORM TREE COVER problem), denoted by (V, d, w, D) where $D \subseteq V$ is a set of depots [10]. These problems under all norms will be formally defined and addressed in the subsequent sections of this paper.

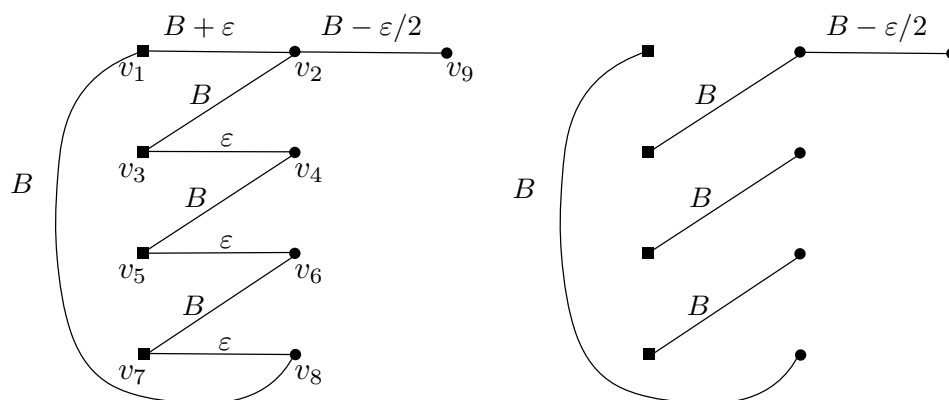
1.1 Our contributions

Computing optimal tree covers is NP-hard even for the *single* ℓ_∞ -norm; for this norm, it admits a constant-factor approximation in polynomial time. Our goal in this paper is thus to design constant-factor approximations for ALL-NORM TREE COVER, that is, for *all* monotone symmetric norms *simultaneously*. Building upon the example in Figure 1, it becomes clear that a constant-factor approximation for one norm objective does not necessarily guarantee a constant approximation for another norm. Meanwhile, achieving an all-norm approximation factor better than $3/2$ within polynomial time is infeasible, as a lower bound of $3/2$ on the approximability of the TREE COVER problem under the ℓ_∞ -norm has been demonstrated by Xu and Wen [23] (assuming $P \neq NP$). In previous work, Even et al. [10] proposed a 4-approximation algorithm for the MIN-MAX TREE COVER problem, which is representative of the ℓ_∞ -norm. That algorithm was subsequently refined by Khani and Salavatipour [19], who devised a 3-approximation. However, these existing algorithms fail to guarantee a constant approximation factor for optimal solutions under other norms, proving to be unbounded specifically, this holds even for the ℓ_1 -norm (for an example, see Figure 3 in Section 3).

Our first main contribution is a polynomial-time constant-factor approximation algorithm for the ALL-NORM TREE COVER problem. Our algorithm amalgamates the strategies of suitable algorithms for both the ℓ_1 -norm and the ℓ_∞ -norm. It is well-understood that an optimal solution for the ℓ_1 -norm objective involves successively eliminating the heaviest edges until precisely k components remain. Conversely, the algorithm for ℓ_∞ -norm objective concludes when the number of trees generated by the edge-decomposition with respect to a guessed optimal value R is about to exceed k [10]. Our proposed method continually approximates the decomposed trees towards an ideal state. So in that sense the algorithm proposed by Even et al. [10] is effectively refined to solve other norm problems. Notably, our algorithm produces a feasible solution that is at most double the optimal solution for the ℓ_1 -norm and four times the optimal solution for the ℓ_∞ -norm simultaneously.

► **Theorem 2.** *There exists a polynomial-time $O(1)$ -approximation algorithm for the ALL-NORM TREE COVER problem.*

We next consider the more involved ALL-NORM TREE COVER problem with depots. For ALL-NORM TREE COVER with depots, each tree in a tree cover solution is rooted at a specific depot. The special problem case of the single ℓ_∞ -norm is the MIN-MAX TREE



■ **Figure 2** Example instance where the algorithm of Even et al. [10] does not return a good approximation in the ℓ_1 -objective. The left figure is an example of the graph metric when $k = 4$ and the right figure is the final solution obtained by the rooted-tree-cover algorithm by Even et al. [10].

COVER problem with depots, for which Even et al. [10] devised a 4-approximation algorithm. Subsequently, Nagamochi [22] enhanced this framework by offering a $(3 - \frac{2}{k+1})$ -approximation algorithm under the restriction that all depots are located at the same node. However, neither of those algorithms can assure any constant approximation factor for the ℓ_1 -norm objective (refer to the example in Figure 2).

Our key contribution extends the $O(1)$ -approximation algorithm for ALL-NORM TREE COVER from Theorem 2 to the problem with depots:

► **Theorem 3.** *There exists a polynomial-time $O(1)$ -approximation algorithm for the ALL-NORM TREE COVER WITH DEPOTS problem.*

Our methodology comprises three stages:

1. Initially, we partition the nodes according to their distances to the depots, where partition class i contains all nodes at distance between 2^{i-1} and 2^i .
2. We then apply a version of the algorithm of Even et al. [10] in each class separately to find a good “pre-clustering” of the nodes. This is simplified by the previous partitioning since it allows us to assume that all nodes are at the same distance to the depots, up to a factor of 2.
3. Finally, we assign the clusters from the second step to the depots by iteratively computing matchings of clusters to depots, considering ever larger clusters, and allowing them to be matched to ever more distant depots. In this way, we essentially maintain a running estimate of the necessary tree weights, updating it when the matching process does not succeed in assigning all nodes to some depot. If it does succeed, we can show that (up to constant factors) no cluster is larger than the estimate, and that the estimate is correct, i.e., every solution has trees at least as large as predicted by the estimate.

Note that both the depot and non-depot algorithms will require only very simple algorithmic primitives such as minimum spanning trees and bipartite matchings. Thus, our algorithms can be implemented to run very quickly using purely combinatorial techniques, in particular without resorting to linear programming which can be impractically slow (see also Davies et al. [9] for recent work on combinatorial clustering algorithms). In fact, an implementation of our algorithm can process metric spaces with 10,000 points in less than a second, see Section 6 for details.

Our final contribution complements the algorithmic result from Theorem 3 by a complexity-theoretic lower bound: we show that one cannot expect polynomial-time approximation schemes to exist, even in the presumably easier setting ℓ_p -TREE COVER WITH DEPOTS where we only want to approximate the optimum with respect to *one specific* ℓ_p -norm:

► **Theorem 4.** *For every $p \in (1, \infty]$ there exists a constant c such that ℓ_p -TREE COVER WITH DEPOTS is NP-hard to approximate within a factor c . The NP-hardness holds under randomized reductions.*

Specifically, we show that ℓ_p -TREE COVER is NP-hard to approximate to a factor

$$\left[\frac{(106 - \frac{1}{4} + \frac{\varepsilon}{2})3^p + (\frac{1}{8} - \frac{\varepsilon}{4})(2^p + 4^p)}{(106 - 2\varepsilon)3^p + \varepsilon(2^p + 4^p)} \right]^{1/p} > 1,$$

for any choice of $\varepsilon > 0$.

2 Preliminaries

We set up some formal definitions. We will identify metric spaces and metrically edge-weighted graphs to allow for an easier treatment of connectedness, trees, and similar graph-theoretic objects. For any such graph G , we always keep in mind the underlying metric space V induced by the shortest-path metric on G . We also assume that the metrics used are integral, for ease of presentation. Our results extend to rational metrics by rescaling the metric.

We here concern ourselves with tree covers of graphs, which are to be defined as follows:

► **Definition 5** (Tree cover). *For a graph G , a tree cover is a collection $\{T_1, \dots, T_k\}$ of trees for which $\bigcup_i V(T_i) = V$. We call k the size of such a tree cover.*

► **Definition 6** (Tree cover with depots). *For a graph G and a depot set $D \subseteq V$, a tree cover with depots is a tree cover $\{T_1, \dots, T_{|D|}\}$ of G , where each T_i contains a unique depot from D .*

Notice in particular that there is no expectation of disjointness for the trees. If disjointness is required, we may assign every node to exactly one of the trees currently containing it and recompute minimum-weight spanning trees for each of the resulting clusters. This increases the cluster weights by at most a factor of 2 due to the gap between Steiner trees and minimum-weight spanning trees (see Lemma 9). For any connected subgraph H of a graph G , we use $w(H)$ to denote the weight of a minimum-weight spanning tree in the subgraph H . It is important to note that this weight can differ from the sum of the weight of the edges of H . We then use the p -norm ($p \geq 1$) as a measure of the quality of a tree cover. Specifically, the cost of a tree cover $\{T_1, \dots, T_k\}$ is defined as the p -norm of the corresponding tree weights, that is, $w_p = \left(\sum_{i \in [k]} (w(T_i))^p \right)^{1/p}$.

There are two natural optimization questions for tree covers (and for tree covers with depots) which have been considered in the literature: The ℓ_1 -TREE COVER problem, where the aim is to minimize the sum of the weights of the k trees, and the ℓ_∞ -TREE COVER problem in which the objective is to minimize the weight of the heaviest tree within the cover. The ℓ_1 -TREE COVER problem admits a polynomial-time algorithm, regardless of the presence of depots; in contrast, the ℓ_∞ -TREE COVER problem is APX-hard, again regardless of whether depots are present or not.

We consider the interpolation between these two problems by allowing arbitrary ℓ_p norms:

► **Definition 7** (ℓ_p -TREE COVER (resp. ℓ_p -TREE COVER WITH DEPOTS)). Given a graph $G = (V, d)$ and some integer k (resp. depots $D \subseteq V$) and a $p \in [1, \infty)$, find a tree cover $\{T_1, \dots, T_k\}$ (resp. with depots) which minimizes the expression $OPT_{p,k} := \left(\sum_{i=1}^k (w(T_i))^p\right)^{1/p}$.

We will usually omit k if it is clear from context. All of the ℓ_p -variants (except $p = 1$) are NP-hard, as the proof of hardness for the ℓ_∞ -case works for all values of $p > 1$ [23].

► **Definition 8** (ALL-NORM TREE COVER PROBLEM (resp. ALL-NORM TREE COVER PROBLEM WITH DEPOTS)). Given a graph $G = (V, d)$ and some integer k (resp. depots $D \subseteq V$), find a tree cover $\{T_1, \dots, T_k\}$ (resp. with depots) which minimizes

$$\max_{p \in [1, \infty)} \frac{\left(\sum_{i=1}^k (w(T_i))^p\right)^{1/p}}{OPT_p}.$$

To distinguish the problem versions for a *fixed* norm p from those for *all* p simultaneously, we denote by (ℓ_p, k) (or (ℓ_p, D) for problems involving depots) the tree cover problem for a specific value of p . Meanwhile, we use $(\ell_{p \in [1, \infty)}, k)$ (or $(\ell_{p \in [1, \infty)}, D)$ when depots are involved) to represent the ALL-NORM TREE COVER problem that encompasses all values of $p \in [1, \infty)$. We omit naming the underlying metric space (V, d) , unless explicitly stated otherwise.

We will state a result about the relationship between the cost of minimum-weight Steiner trees and minimum-weight spanning trees, as we will harness this argument repeatedly:

► **Lemma 9** (Kou et al. [20]). Let (V, d) be a metric space with some terminal set $F \subseteq V$, let \hat{T} be a minimum-weight (Steiner) tree containing all nodes from F , and let T be a minimum-weight spanning tree of $(F, d|_F)$. Then $w(T) \leq (2 - 2/|F|) \cdot w(\hat{T})$.

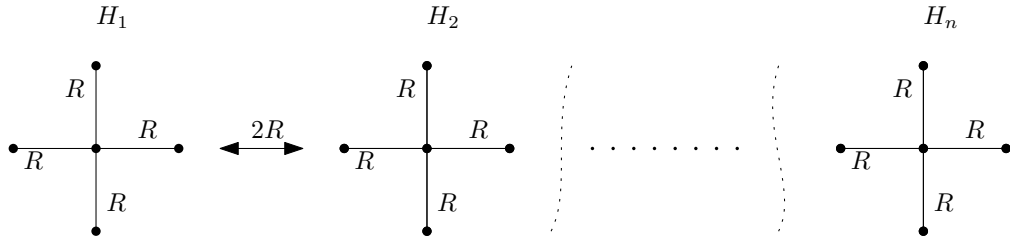
Proof. An easy way to obtain the result is to take \hat{T} and use the tree-doubling heuristic to compute a traveling salesperson tour \mathcal{T} of F in (V, d) , costing at most $2 \cdot w(\hat{T})$. Removing the heaviest edge of \mathcal{T} yields a (perhaps not yet minimal) spanning tree T' in $(F, d|_F)$ of weight at most $(2 - 2/|F|)w(\hat{T})$, allowing us to conclude that $w(T) \leq w(T') \leq (2 - 2/|F|)w(\hat{T})$. ◀

Lemma 9 will prove useful as we occasionally want to forget about some nodes of our instance. This may actually increase overall costs, since certain Steiner trees become unavailable. However, the lemma ensures that the increase in cost is bounded.

3 Simultaneous Approximations for the All-Norm Tree Cover Problem

First, to provide some good intuition of the key techniques for the general case, we show in detail that the TREE COVER algorithm of Even et al. [10] gives a constant-factor approximation to the ℓ_p -tree cover problem without depots provided that it returns k trees. Indeed, we show that the argument will work for all monotone symmetric norms. The original algorithm works as follows, for a given graph $G = (V, d)$:

1. Guess an upper bound R on the optimum value of the ℓ_∞ -norm tree cover problem for instance G , where initially $R = 1$.
2. Remove all edges with weight larger than R from G , and compute a minimum-weight spanning tree T_i for each connected component of the resulting graph.
3. Check that $\sum \lfloor \frac{w(T_i) + 2R}{2R} \rfloor = k$; otherwise, reject R and go to step 1 with $R := 2R$.
4. Call a spanning tree T_i *small* if $w(T_i) < 2R$.



■ **Figure 3** Example instance where the algorithm of Even et al. does not return a good approximation in the ℓ_1 -objective. The instance consists of n identical copies H_i of the 4-star where all edges have length R and the copies are pairwise at distance $2R$. For $k = 5n - 1$, the instance has $OPT_\infty = OPT_1 = R$, but the algorithm will return $2n$ trees, each of weight $2R$. Observe that the partition requirement in Step 5 would also be fulfilled if the trees are not further partitioned and kept at size $4R$, however the algorithm of Even et al. produces the former solution. Both solutions do not achieve a good approximation in the ℓ_1 objective.

5. Call a spanning tree T_i *large* if $w(T_i) \geq 2R$. Decompose each such tree into edge-disjoint subtrees \tilde{T}_j such that $2R \leq w(\tilde{T}_j) \leq 4R$ for all but one residual \tilde{T}_j in each component¹.
6. Output the family of all small spanning trees, as well as all subtrees \tilde{T}_j created in Step 5.

Even et al. show that this procedure will output *at most* k trees if $OPT_\infty \leq R$, although they relax the condition in Step 3 to $\sum \lfloor \frac{w(T_i) + 2R}{2R} \rfloor \leq k$. For technical reasons, however, we need the equality in this spot. Since every tree has weight most $4R$, the algorithm evidently gives a 4-approximation in the ℓ_∞ -norm if the correct value of $R = OPT_{\infty,k}$ is guessed. Otherwise, one can use binary search to find, in polynomial time, the smallest R for which one gets at most k trees.

In general, however, we notice that this algorithm will sometimes compute fewer than k trees, causing it to not return a good approximation for the other ℓ_p -norms, not even for the ℓ_1 -norm (see Figure 3).

For this reason, we need the strengthened property in Step 3. Then, however, notice that such a value R does not necessarily exist; for example, for the instance in Figure 3 this is the case. We will describe later how to avoid this issue of non-existence; for now, we assume that such a value R exists and can be computed in polynomial time.

So suppose that the algorithm has returned k trees T_1, \dots, T_k , where we simply remove some edges should the algorithm return fewer than k trees. This removal only improves the objective value, so we will assume – without loss of generality – that the algorithm already returned k trees. As a warm-up, we will start by considering only the case that $p = 1$, i.e., we show that this algorithm computes a solution that is a good approximation for (ℓ_1, k) tree cover.

► **Lemma 10.** *Let $\{T_1, \dots, T_k\}$ be the set of trees returned by the algorithm for some fixed value of R . Then we have $\sum w(T_i) \leq 2OPT_{1,k}$.*

Proof. We observe first that an optimal solution for (ℓ_1, k) tree cover can be computed by removing iteratively the heaviest edge as long as this does not cause the graph to decompose into more than k components, as this procedure is equivalent to running Kruskal’s algorithm. As the optimal solution contains no edges of weight greater than R , we consider the graph $G' := G - \{e \mid d(e) > R\}$. Let k_s be the number of small spanning trees S_i computed by

¹ This can be done with a simple greedy procedure, cutting of subtrees of this weight. For the details of this algorithm, we refer to Even et al. [10].

the algorithm, and let k_ℓ be the number of large spanning trees L_i . Then we certainly have $\sum w(T_i) \leq \sum w(S_i) + \sum w(L_i)$, since the trees computed by the algorithm are edge-disjoint subtrees of the initial spanning trees. Further, we have

$$OPT_{1,k} \geq \sum w(S_i) + \sum w(L_i) - (k - k_s - k_\ell)R,$$

because the optimal solution will remove $k - k_s - k_\ell$ edges from within the spanning trees computed by the algorithm, each of weight at most R . But notice that we can use Step 3 to obtain

$$k = \sum \left\lfloor \frac{w(S_i)}{2R} + 1 \right\rfloor + \sum \left\lfloor \frac{w(L_i)}{2R} + 1 \right\rfloor \leq k_s + k_l + \sum \frac{w(L_i)}{2R},$$

which implies that $(k - k_s - k_\ell)2R \leq \sum w(L_i)$. This inequality allows us to conclude that

$$\begin{aligned} OPT_{1,k} &\geq \sum w(S_i) + \sum w(L_i) - \frac{1}{2} \sum w(L_i) \\ &\geq \frac{1}{2} (\sum w(S_i) + \sum w(L_i)) \\ &\geq \frac{1}{2} \sum w(T_i) . \end{aligned}$$

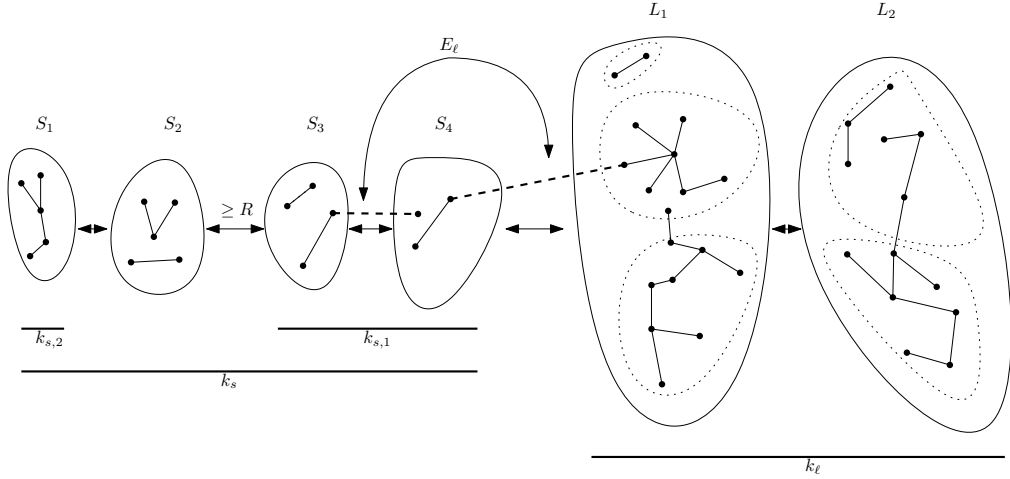
Notice that the strengthened version of Step 3 is indeed necessary here.

We can use a similar technique to show that the solution computed by the algorithm is a constant-factor approximation for (ℓ_p, k) tree cover for any choice of p , and with a constant of approximation independent of p . The key argument will be to show that an optimal solution to (ℓ_p, k) tree cover must, as in Lemma 10, have a total weight comparable to that of the solution computed by the algorithm. In a second step, one can then show that the algorithm distributes the total weight of its trees fairly evenly, because the large trees all have weight between $2R$ and $4R$.

Meanwhile, for the small trees we can demonstrate that choosing them differently from the algorithm will incur some cost of at least R . Thus, either the algorithms' solution has correctly chosen the partition on these small trees, or the optimal solution actually has a heavy tree not in the algorithms' solution, which we can use to pay for one of the large trees that the algorithm has, but the optimal solution does not. Notice that, if the algorithm achieves an equal distribution of some total weight that is comparable to the total weight of an optimal solution under the ℓ_p -norm, it also achieves a good approximation of OPT_p due to convexity of the ℓ_p -norms.

To start with, we will only give a rough analysis of the quality of the solution returned by the algorithm, although it already shows a constant factor of approximation. The full version on arXiv gives a more detailed proof that achieves a better constant [18].

Let us fix some $p \in [1, \infty)$ and any (ℓ_p, k) tree cover solution $\{\hat{T}_1, \dots, \hat{T}_k\}$ (you may imagine that it is optimal). Then let k_s be the number of connected components computed by the algorithm that had a small spanning tree, and call those components S_1, \dots, S_{k_s} . Similarly, for the large spanning trees, let there be k_ℓ components L_1, \dots, L_{k_ℓ} . Now we denote by E_ℓ the set of edges that are both contained in some \hat{T}_i and in the cut induced by some S_j or L_j . In particular, all these edges have weight at least R . We count separately the small T_i incident to some edge from E_ℓ , say there are $k_{s,1}$ of them. We will also denote by $k_{s,2}$ the number of S_i for which one of the \hat{T}_j is a minimum-weight spanning tree, and denote the set of these \hat{T}_j as T^- . Note that any small component not counted by $k_{s,1}$ or $k_{s,2}$ is split into at least two components by the \hat{T}_i . For an illustration of this setting, we refer to Figure 4.



■ **Figure 4** Illustration of how the algorithm's solution and some optimal solution can align against each other. The algorithm's partition of the graph into connected component is indicated by solid zones, the further subdivision of the large components by dotted zones. The trees of the optimal solution are drawn, and all edges crossing the boundary of a connected component are dashed.

We can now start to measure the total sum of edge weights in the \hat{T}_j , i.e., $\sum_j w(\hat{T}_j)$. We begin by relating it to the small trees of the algorithm's solution that do not agree with the optimal solution.

► **Observation 11.** *It holds that $\sum_j w(\hat{T}_j) - \sum_{\hat{T}_i \in T^=} w(\hat{T}_i) \geq k_{s,1} \frac{R}{2}$.*

Proof. We have $k_{s,1}$ pairwise disjoint node sets in G , each incident to at least one edge of weight at least R present in E_ℓ , so we get $|E_\ell| \geq k_{s,1}/2$ from the handshake lemma, and thus

$$\sum_j w(\hat{T}_j) - \sum_{\hat{T}_i \in T^=} w(\hat{T}_i) \geq |E_\ell| R \geq \frac{k_{s,1}}{2} R,$$

noting that the trees in $T^=$ do not contain any of the edges from E_ℓ . ◀

To compare the total weight against the number of large trees, we can use a similar argument as in Lemma 10. We again note that the initial spanning trees have weight at least $(k - k_\ell - k_s)2R$, and any tree cover cannot remove too many edges from them. The second part of this argument is no longer true though, since it is now possible for a solution to have many edges between the components of $G - \{e \mid d(e) \geq R\}$, allowing it to remove many edges within the components. However, a careful analysis will show that this case does not pose a problem, since such inter-component edges are themselves heavy.

► **Observation 12.** *We have $\sum_j w(\hat{T}_j) \geq (k - k_\ell - k_{s,1} - k_{s,2})R + \sum_{\hat{T}_i \in T^=} w(\hat{T}_i)$.*

Proof. Suppose we delete from the \hat{T}_j all edges from E_ℓ . This will yield $k + |E_\ell|$ trees. We will count only those trees that lie in a large component L_i , of which there are at most $k + |E_\ell| - 2k_s + k_{s,1} + k_{s,2}$. This is because every small component contains at least 2 of the \hat{T}_j , except for $k_{s,1} + k_{s,2}$ many. Now observe that to get a $(k + |E_\ell| - 2k_s + k_{s,1} + k_{s,2})$ -component spanning forest of *minimum* weight for the L_i , we start with the initial minimum spanning trees in each component (which have weight at least $(k - k_s - k_\ell)2R$) and remove at most $(k + |E_\ell| - k_\ell - 2k_s + k_{s,1} + k_{s,2})$ edges, each of weight at most R . The total remaining weight is $(k - k_s - k_\ell)2R - (k + |E_\ell| - k_\ell - 2k_s + k_{s,1} + k_{s,2})R = (k - k_\ell - k_{s,1} - k_{s,2} - |E_\ell|)R$.

Thus, we can measure the total weight of edges which are part of some \hat{T}_i and lie in a large component as being at least the total weight the spanning trees of the L_i , minus the weight of the edges that may have been removed, and obtain

$$\begin{aligned} \sum_j w(\hat{T}_j) - \sum_{\hat{T}_i \in T=} w(\hat{T}_i) - |E_\ell|R &\geq (k - k_\ell - k_{s,1} - k_{s,2} - |E_\ell|)R. \\ \implies \sum_j w(\hat{T}_j) - \sum_{\hat{T}_i \in T=} w(\hat{T}_i) &\geq (k - k_\ell - k_{s,1} - k_{s,2})R. \quad \blacktriangleleft \end{aligned}$$

Notice that with these two observations, we can almost show some result along the lines of $OPT_{1,k} \geq c \cdot k \cdot R$ for some $c \in \mathbb{R}$, since either there are many large trees, in which case Observation 12 gives us the desired result, or there are many small trees in which case we can use Observation 11, unless $k_{s,2}$ is large. However, the case that $k_{s,2}$ is large, i.e., we have computed many of the trees that are present in the optimal solution, should also be beneficial, so we maintain a separate record of $k_{s,2}$ in the analysis to obtain:

► **Lemma 13.** *It holds that $\sum_j w(\hat{T}_j) \geq \frac{R}{6}(k - k_{s,2}) + \sum_{\hat{T}_i \in T=} w(\hat{T}_i)$.*

Proof. We combine Observation 12 and Observation 11 convexly with coefficients $1/3, 2/3$ to obtain

$$\begin{aligned} \sum_j w(\hat{T}_j) &\geq \frac{1}{3} [(k - k_\ell - k_{s,1} - k_{s,2})R] + \frac{2}{3} \left[k_{s,1} \frac{R}{2} \right] + \sum_{\hat{T}_i \in T=} w(\hat{T}_i) \\ \iff \sum_j w(\hat{T}_j) &\geq \frac{R}{3} (k - k_\ell - k_{s,2}) + \sum_{\hat{T}_i \in T=} w(\hat{T}_i) \\ \implies \sum_j w(\hat{T}_j) &\geq \frac{R}{6} (k - k_{s,2}) + \sum_{\hat{T}_i \in T=} w(\hat{T}_i), \end{aligned}$$

where the final inequality follows from the following reasoning:

$$k = \sum \left\lfloor \frac{w(S_i) + 2R}{2R} \right\rfloor + \sum \left\lfloor \frac{w(L_i) + 2R}{2R} \right\rfloor \geq k_s + 2k_\ell,$$

and thus $k - k_\ell - k_{s,2} \geq k - \frac{k - k_s}{2} - k_{s,2} = \frac{k}{2} + \frac{k_s}{2} - k_{s,2} \geq \frac{k - k_{s,2}}{2}$. ◀

The upshot of Lemma 13 is then that any tree cover using k trees with some $k_{s,2}$ trees in common with the algorithm's solution will have the property that the other $k - k_{s,2}$ trees have an average weight of $\Omega(R)$.

► **Theorem 14.** *If the algorithm of Even et al. returns k trees T_1, \dots, T_k , then we have*

$$\left[\sum_{i=1}^k (w(T_i)^p) \right]^{1/p} \leq 24 \left[\sum_{i=1}^k (w(\hat{T}_i)^p) \right]^{1/p}$$

for any p and for any tree cover $\{\hat{T}_1, \dots, \hat{T}_k\}$ of G with k trees.

Proof. From Lemma 13 we obtain

$$\sum_{i=1}^k (w(\hat{T}_i)^p) \geq (k - k_{s,2}) \left(\frac{R}{6} \right)^p + \sum_{\hat{T}_i \in T=} w(\hat{T}_i)^p,$$

using convexity of $|x|^p$ for any $p \geq 1$. At the same time, from the algorithm it is clear that

$$\sum_{i=1}^k (w(T_i)^p) \leq (k - k_{s,2})(4R)^p + \sum_{\hat{T}_i \in T=} w(\hat{T}_i)^p,$$

57:12 Approximate All-Norm Tree Cover in All Symmetric Monotone Norms

because every tree is either identical to one of the \hat{T}_j , or has weight at most $4R$. Putting these inequalities together yields the statement of the theorem:

$$\frac{\sum_{i=1}^k (w(T_i))^p}{\sum_{i=1}^k (w(\hat{T}_i))^p} \leq \frac{(k - k_{s,2})(4R)^p + \sum_{\hat{T}_i \in T^=} w(\hat{T}_i)^p}{(k - k_{s,2}) \left(\frac{R}{6}\right)^p + \sum_{\hat{T}_i \in T^=} w(\hat{T}_i)^p} \leq \frac{(k - k_{s,2})(4R)^p}{(k - k_{s,2}) \left(\frac{R}{6}\right)^p} \leq 24^p . \quad \blacktriangleleft$$

The whole proof, in particular the result of Lemma 13 can be reinterpreted to give an even stronger result: namely, that the tree cover returned by the algorithm is approximately “strongly optimal”, a concept introduced by Alon et al. [3] for analysing all-norm scheduling algorithms. The point is to show a strong version of lexicographic minimality of the constructed solution. Formally, let $\{T_1, \dots, T_k\}$ be the solution computed by the algorithm for a given instance $(\ell_{p \in [1, \infty)}, k)$ and $\hat{T}_1, \dots, \hat{T}_k$ any other tree cover for $(\ell_{p \in [1, \infty)}, k)$. Further, let the trees be sorted non-increasingly by weight as $w(T_1) \geq w(T_2) \geq \dots$ and $w(\hat{T}_1) \geq w(\hat{T}_2) \geq \dots$. Then $\{T_1, \dots, T_k\}$ is *strongly optimal* if for any j we have

$$\sum_{i=1}^j w(T_i) \leq \sum_{i=1}^j w(\hat{T}_i) .$$

Similarly, one can speak of *c-approximate strongly optimality* if for some $c \in \mathbb{R}_{\geq 1}$ we have

$$\sum_{i=1}^j w(T_i) \leq c \sum_{i=1}^j w(\hat{T}_i) .$$

This property was also considered as “global c -balance” by Goel and Meyerson [12].

Approximate strong optimality suffices for our purposes, since a solution that is c -approximate strongly optimal is also a c -approximation with respect to any p -norm; in fact, we obtain a stronger result here, since c -approximate strongly optimality implies a c -approximation with respect to *any* convex symmetric function, including all monotone symmetric norms. This fact is the backbone of many all-norm approximation algorithms, for example those by Golovin, Gupta, Kumar and Tangwongsan [13] for set cover variants. Thus, we will obtain a 24-approximation from this analysis not only for all p -norms, but for all monotone symmetric norms.

► **Lemma 15.** *Let $\{T_1, \dots, T_k\}$ be the solution computed by the algorithm for a given instance $(\ell_{p \in [1, \infty)}, k)$, and let $\{\hat{T}_1, \dots, \hat{T}_k\}$ be any other tree cover for $(\ell_{p \in [1, \infty)}, k)$, where $w(T_1) \geq w(T_2) \geq \dots$ and $w(\hat{T}_1) \geq w(\hat{T}_2) \geq \dots$. Then $\sum_{i=1}^j w(T_i) \leq 24 \sum_{i=1}^j w(\hat{T}_i)$ for $j = 1, \dots, k$.*

Proof. We obtain an upper bound for the values $\sum_{i=1}^j w(T_i)$ by assuming that all trees which are not accounted for by the $k_{s,2}$ small component trees shared between the T_i and the \hat{T}_i have weight exactly $4R$. This will ensure that the shared trees are $T_{k-k_{s,2}+1}, \dots, T_k$. Similarly, we obtain a lower bound $\sum_{i=1}^j w(\hat{T}_i)$ by allowing a non-descending permutation of the \hat{T}_i . That is, we reorder the \hat{T}_i such that the first $k - k_{s,2}$ trees are not the shared small component trees with the T_i .

It then follows directly that

$$\sum_{i=1}^j w(T_i) \leq j \cdot 4R = 24(j \cdot \frac{R}{6}) \leq 24 \sum_{i=1}^j w(\hat{T}_i) \text{ for } j \leq k - k_{s,2},$$

as well as

$$\begin{aligned} \sum_{i=1}^{k-k_{s,2}} w(T_i) + \sum_{i=1+k-k_{s,2}}^j w(T_i) &\leq 24 \sum_{i=1}^{k-k_{s,2}} w(\hat{T}_i) + \sum_{i=1+k-k_{s,2}}^j w(\hat{T}_i) \\ \implies \sum_{i=1}^j w(T_i) &\leq 24 \sum_{i=1}^j w(\hat{T}_i) \text{ for } j > k - k_{s,2} . \end{aligned} \quad \blacktriangleleft$$

3.1 Ensuring k trees

Recall that the previous analysis of the approximation factor relies on the algorithm being able to find some R such that Step 3 holds, which is not generally true as per Figure 3. We now demonstrate how to modify the algorithm in such a way that this is avoided. Consider a list e_1, \dots, e_m of the edges of G , such that $d(e_i) \leq d(e_j)$ if $i \leq j$, i.e., they are sorted by length with ties broken arbitrarily. Then we consider separately the graphs $G_j := (V, \{e_i \mid i \leq j\})$ for all $j = 0, \dots, m$ and try to find for each G_j an $R \in [d(e_{j-1}), d(e_j)]$ that is accepted by the algorithm, where we set $d(e_0) := 0$, and allow the larger interval $[d(e_m), \sum_i d(e_i)]$ for G_m . Note that the intervals can potentially only contain a single node, but they are not empty.

Now observe that for G_0 with $R = 0$, the algorithm would only accept this choice of R if $k = |V|$. Similarly, for G_m and $R = \sum_i d(e_i)$, the algorithm would require $k = 1$. We can then show that the k changes by at most one as we change R or keep R and move from G_j to G_{j+1} . Let $k(G_j, R)$ denote the value of k that would be accepted by the algorithm. Thus

► **Observation 16.** *It holds that $k(G_{j-1}, d(e_j)) - k(G_j, d(e_j)) \leq 1$.*

Proof. Between G_{j-1} and G_j , only the presence of e_j changes. If this does not change the connected components, we have $k(G_{j-1}, d(e_j)) = k(G_j, d(e_j))$. Otherwise, there is exactly one connected component C in G_j that is split into two parts C_1, C_2 by removing e_j . We then see that

$$\begin{aligned} \left\lfloor \frac{w(C_1) + 2d(e_j)}{2d(e_j)} \right\rfloor + \left\lfloor \frac{w(C_2) + 2d(e_j)}{2d(e_j)} \right\rfloor &\leq 2 + \left\lfloor \frac{w(C_1)}{2d(e_j)} + \frac{w(C_2)}{2d(e_j)} \right\rfloor \\ &\leq 1 + \left\lfloor \frac{w(C) + 2d(e_j)}{2d(e_j)} \right\rfloor . \end{aligned} \quad \blacktriangleleft$$

To see that changing R by a sufficiently small amount also only changes $k(G_j, R)$ by at most one, consider that there are only finitely many critical nodes where $k(G_j, R)$ changes at all, and they can be computed in polynomial time. They are all of the form $R = w(C_i)/2\ell$ for some connected component C_i of G_j and $\ell \in 1, \dots, n$. At these nodes, $w(C_i)/2R$ will be an integer, so $\lfloor w(C_i)/2R \rfloor = 1 + \lfloor (w(C_i) - \varepsilon)/2R \rfloor$. If all critical nodes are pairwise different, we can just iterate over them to find some G_j and R with $k(G_j, R) = k$.

If on the other hand a node is critical for multiple different components, assign to each component a distinct weight which can be taken to be arbitrarily small, ensuring that all critical nodes are now different. In effect, this is equivalent to taking all components where $\frac{w(C_i)+2R}{2R}$ is an integer and allowing the expression $\lfloor \frac{w(T_i)+2R}{2R} \rfloor$ to also take the value $\frac{w(T_i)}{2R}$. One may check that this does not impact the analysis, since in the analysis we only need that each component has a minimum spanning tree of weight at least $\lfloor \frac{w(T_i)}{2R} \rfloor \cdot 2R$. However, for legibility reasons we will suppress this and generally assume that some R can be found with $k(G_j, R) = k$. Combining with Theorem 14, this completes the proof of Theorem 2.

4 All-norm tree cover problem with depots

The algorithm for the ALL-NORM TREE COVER problem *with* depots is considerably more involved, in particular because the simple lower bound for the depot-less algorithm of assuming an even distribution of the total weight can no longer work: it might be necessary to have unbalanced cluster sizes, for instance if some depots are extremely far from all nodes.

Instead our algorithm constructs a c -approximately (here, c is a constant) strongly optimal solution by also maintaining some (implicit) evidence that the optimum solution contains large trees if it decides to create a large tree itself. More concretely we do the following:

1. First, we partition the node set V into layers L_i such that all nodes in L_i have distance between 2^{i-1} and 2^i to the depots.
2. Then we consider separately the nodes in the odd and even layers; this implies that nodes in different layers have a large distance to each other. Computing separate solutions for these two subinstances loses at most a factor 2 in the approximation factor due to Lemma 9.
3. Next, we partition the nodes in each layer L_i using the non-depot algorithm with $R = 2^i$. This yields a collection of subtrees in L_i such that the cost of connecting such a subtree to its nearest depot is in $\Theta(2^i)$. This allows us to treat them basically identically, losing only the constant in the Θ . Indeed, we can show explicitly that this prepartitioning into subtrees can be assumed to be present in an optimal solution, up to a constant factor increase in the weight of each tree. For the full reasoning refer to the full version on arXiv [18].
4. To assign these trees to the depots, we iteratively maintain an estimate of the largest tree necessary in any solution as 2^i . We then collect all trees of weight $\Theta(2^i)$ and compute a maximum matching between them and the depots at distance $\Theta(2^i)$ to them. All unmatched trees are then combined to form trees of weight $\Theta(2^{i+1})$, and we update our estimate to 2^{i+1} .

To analyse the output of this algorithm, it will suffice to show that the estimate was correct up to a constant factor. That is, if in round i some trees (suppose k_i trees) of weight 2^i were assigned, we will be able to prove that any tree-cover solution must also have some family of at most k_i trees with total weight at least $k_i \cdot 2^i$.

From this we are then able to conclude c -approximate strong optimality for some value of $c < 10^6$. This is a rather large upper bound on the approximation factor, however, we conjecture that the actual constant achieved by the algorithm is much smaller. For the purposes of a clean presentation, we did not try to optimize the constant. The complete proof can be found in the full version on arXiv [18].

5 Computational Hardness

To complement our algorithmic results, we establish hardness results for all-norm tree cover problems and discuss on what kind of algorithmic improvements (to our algorithms) are potentially possible in light of these complexity results. Specifically, we show:

1. For any $p \in (1, \infty]$, problem ℓ_p -TREE COVER WITH DEPOTS is weakly NP-hard, even with only 2 trees.
2. For any $p \in (1, \infty]$, problem ℓ_p -TREE COVER WITH DEPOTS with k trees is (strongly) W[1]-hard parameterized by depot size $|D|$.
3. For any $p \in (1, \infty]$ there exists some $\varepsilon > 0$ such that ℓ_p -TREE COVER WITH DEPOTS is NP-hard to approximate within a factor $< 1 + \varepsilon$ under randomized reductions.

Results 1 and 2 were already essentially presented by Even et al. [10], but we restate them for completeness. Note that, given these complexity results, numerous otherwise desirable algorithmic outcomes become unattainable. For instance, there cannot be polynomial-time approximation schemes for ℓ_p -TREE COVER WITH DEPOTS, nor can we expect fixed-parameter algorithms (parameterized by the number of depots) finding optimal solutions, unless established hardness hypotheses ($P \neq NP$, $FPT \neq W[1]$) fail. Further, the reduction of Item 1 yields bipartite graphs of tree-depth 3, so parameterization by the structure of the graph supporting the input metric also appears out of reach.

The result in Item 3 follows by a direct gadget reduction from MAX-SAT for 3-ary linear equations modulo 2 (MAX-E3LIN2), which was shown to be APX-hard by Håstad [15]. We show that one can transform such equations into an instance of ℓ_p -TREE COVER WITH DEPOTS where every unsatisfied equation will correspond roughly to a tree of above average weight. This allows us to recover approximately the maximum number of simultaneously satisfiable constraints of such systems of equations.

Formally, we reduce from $3R\{2,3\}L2$, a modification of MAX-E3LIN2 where every variable occurs in exactly 3 equations, and for which hardness of approximation was shown by Karpinski et al. [17]. From an instance of $3R\{2,3\}L2$ we construct an instance of ℓ_p -TREE COVER WITH DEPOTS by introducing gadgets for the variables and clauses:

- For every variable x , introduce three nodes \hat{x}, x_0 , and x_1 , as well as edges \hat{x}, x_i for $i = 0, 1$ with weight 3. Add the x_i 's as depots.
- For every ternary clause C , introduce nodes $\hat{C}, C_{000}, C_{110}, C_{101}$, and C_{011} with edges $\{\hat{C}, C_i\}$ of weight 3. Add the C_i 's as depots.
- For every binary clause $C = x \oplus y = 0$, introduce nodes \hat{C}, C_{00} , and C_{11} with edges $\{\hat{C}, C_i\}$ of weight 2. Add the C_i 's as depots. Further add nodes \hat{C}_{00} and \hat{C}_{11} where \hat{C}_i is connected only to C_i by an edge of weight 1.
- For every binary clause $C = x \oplus y = 1$, introduce nodes \hat{C}, C_{01} , and C_{10} with edges $\{\hat{C}, C_i\}$ of weight 2. Add the C_i 's as depots. Further add nodes \hat{C}_{01} and \hat{C}_{10} where \hat{C}_i is connected only to C_i by an edge of weight 1.

We connect the gadgets as follows:

- For every clause $C = x \oplus y \oplus z = 0$ we connect $C_{b_1 b_2 b_3}$ to x_{b_1}, y_{b_2} , and z_{b_3} with a path of length 2 where both edges have weight 1.
- For every clause $C = x \oplus y = b$ we connect $C_{b_1 b_2}$ to x_{b_1}, y_{b_2} with a path of length 2 where both edges have weight 1.

Intuitively, there is a depot for every way a clause could be satisfied, and the depots for a clause have a joint neighbour that is expensive to connect. The depot absorbing this neighbour should correspond to the way in which the clause is satisfied. Similarly there are two depots for each variable representing the two possible assignments to the variable. In this case the depot that does not get assigned the joint neighbour corresponds to the chosen assignment.

There are then additional vertices between the clause and vertex depots which can be assigned to the clause depots, unless the adjacent clause depot corresponds to the satisfying assignment of that clause. In that case they need to be assigned to an adjacent variable depot, which is to say the variables of the clause will have to be assigned in such a way that the clause is satisfied.

One can quickly check that a satisfying assignment to the clauses will allow us to compute a tree cover where every tree has size 3. Meanwhile every non-satisfied clause will push at least one unit of excess weight to a tree of size at least 3. Quantifying this relationship then permits us to compute approximately the maximum number of satisfiable clauses in such a system of equations.

6 Computational Experiments

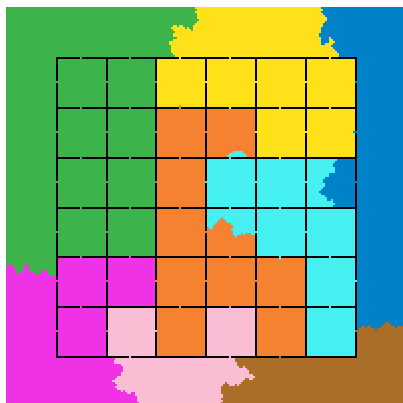
To illustrate the practical performance achievable by our clustering algorithms, we implemented the algorithm from Section 3 for the setting without depots in C++ and tested it on instances proposed by Zheng et al. [24]. Those instances model real-world terrain, which is to be partitioned evenly so that a fixed number robots can jointly traverse it. They consist of a grid where either random cells are set to be obstacles, i.e., inaccessible, or a grid-like arrangement of rooms is superimposed with doors closed at random. A metric is induced on the accessible cells of the grid by setting the distance of neighbouring cells to be 1.

For all instances on grids of size 200 by 200 (ca. 40,000 nodes), our implementation was able to compute a clustering in less than 200ms on an Intel i5-10600K under Windows with 48GB of available memory (although actual memory usage was negligible). The resulting partitions are illustrated in Figure 5. Note that we make two small heuristic changes to the original algorithm, which are, however, not amenable to be analyzed formally:

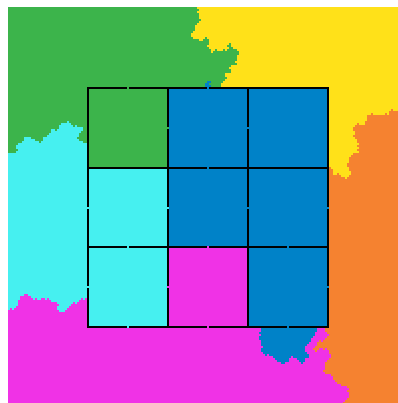
First, we adapt the partitioning of the large trees in Step 5 to try to cut the trees into subtrees of weight at least $2R$ rather than $4R$. Note that the choice of $4R$ captures a worst-case scenario where the instance contains edges of size almost R that are to be included in a solution. If the heaviest edges are considerably smaller than R , the necessary cutoff approaches $2R$ rather than $4R$. Thus, we run the partitioning algorithm with $2R$ rather than $4R$, and resort to the higher cutoff only in the case that this fails. Yet, for the considered instances this was not necessary.

Second, we post-process the computed solution to ensure that it has exactly k components. It is in principle possible that the algorithm computes a solution with fewer than k trees; in this case, we iteratively select the largest tree and split it into two parts of as similar a size as possible until we obtain exactly k components.

The clusterings obtained in this way in Figure 5 are all at least 3-approximately strongly optimal when compared to a hypothetical solution that distributes the total weight perfectly evenly on the k clusters.



(a) Output of the algorithm on a 200×200 grid with walls partitioned into 8 clusters. The cluster sizes are 2433, 5516, 9528, 4550, 5271, 3985, 2482 and 4240; consequently the solution is at least 2.01-approximately strongly optimal.



(b) Output of the algorithm on a 200×200 grid with walls partitioned into 6 clusters. The cluster sizes are 7993, 4774, 6390, 8004, 5267, and 6638; consequently, the solution is at least 1.61-approximately strongly optimal.

■ **Figure 5** Visualizations of the results of our implementation of the non-depot tree cover algorithm. Inaccessible sections of the grid are marked in black; other colors represent the computed clusters.

References

- 1 Sara Ahmadian, Babak Behsaz, Zachary Friggstad, Amin Jorati, Mohammad R Salavatipour, and Chaitanya Swamy. Approximation algorithms for minimum-load k -facility location. *ACM Trans. Algorithms*, 14(2):1–29, 2018. doi:10.1145/3173047.
- 2 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k -means and Euclidean k -median by primal-dual algorithms. *SIAM J. Comput.*, 49(4):FOCS17–97, 2019. doi:10.1137/18M1171321.
- 3 Noga Alon, Yossi Azar, Gerhard J. Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *J. Sched.*, 1(1):55–66, 1998. doi:10.1002/(SICI)1099-1425(199806)1:1<55::AID-JOS2>3.0.CO;2-J.
- 4 Cristina Bazgan, Refael Hassin, and Jérôme Monnot. Approximation algorithms for some vehicle routing problems. *Discrete Appl. Math.*, 146(1):27–42, 2005. doi:10.1016/J.DAM.2004.07.003.
- 5 Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006. doi:10.1016/j.omega.2004.10.004.
- 6 Mandell Bellmore and Saman Hong. Transformation of multisalesman problem to the standard traveling salesman problem. *J. ACM*, 21(3):500–504, 1974. doi:10.1145/321832.321847.
- 7 Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k -median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):1–31, 2017. doi:10.1145/2981561.
- 8 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.
- 9 Sami Davies, Benjamin Moseley, and Heather Newman. Fast combinatorial algorithms for min max correlation clustering. In *Proc. ICML 2023*, pages 7205–7230, 2023.
- 10 Guy Even, Naveen Garg, Jochen Könemann, Ramamoorthi Ravi, and Amitabh Sinha. Min-max tree covers of graphs. *Oper. Res. Lett.*, 32(4):309–315, 2004. doi:10.1016/J.ORL.2003.11.010.
- 11 Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. In *Proc. SFCS 1976*, pages 216–227, 1976. doi:10.1109/SFCS.1976.6.
- 12 Ashish Goel and Adam Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica*, 44:301–323, 2006. doi:10.1007/S00453-005-1177-7.
- 13 Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-norms and all- ℓ_p -norms approximation algorithms. In *Proc. FSTTCS 2008*, volume 2 of *Leibniz Int. Proc. Informatics*, pages 199–210, 2008. doi:10.4230/LIPICS.FSTTCS.2008.1753.
- 14 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999. doi:10.1006/JAGM.1998.0993.
- 15 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.
- 16 Sharat Irahimpur and Chaitanya Swamy. Approximation algorithms for stochastic minimum-norm combinatorial optimization. In *Proc. FOCS 2020*, pages 966–977, 2020. doi:10.1109/FOCS46700.2020.00094.
- 17 Marek Karpinski, Michael Lampis, and Richard Schmieid. New inapproximability bounds for TSP. *J. Comput. Syst. Sci.*, 81(8):1665–1677, 2015. doi:10.1016/J.JCSS.2015.06.003.
- 18 Matthias Kaul, Kelin Luo, Matthias Mnich, and Heiko Röglin. Approximate minimum tree cover in all symmetric monotone norms simultaneously, 2025. doi:10.48550/arXiv.2501.05048.
- 19 M Reza Khani and Mohammad R Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69(2):443–460, 2014. doi:10.1007/S00453-012-9740-5.
- 20 L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15(2):141–145, 1981. doi:10.1007/BF00288961.

57:18 Approximate All-Norm Tree Cover in All Symmetric Monotone Norms

- 21 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013. doi:10.1016/J.IC.2012.01.007.
- 22 Hiroshi Nagamochi. Approximating the minmax rooted-subtree cover problem. *IEICE Trans. Fund. Electr., Comm. Comp. Sci.*, 88(5):1335–1338, 2005. doi:10.1093/IETFEC/E88-A.5.1335.
- 23 Zhou Xu and Qi Wen. Approximation hardness of min–max tree covers. *Oper. Res. Lett.*, 38(3):169–173, 2010. doi:10.1016/J.ORL.2010.02.004.
- 24 Xiaoming Zheng and Sven Koenig. Robot coverage of terrain with non-uniform traversability. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.2007*, pages 3757–3764, 2007. doi:10.1109/IR0S.2007.4399423.

Violating Constant Degree Hypothesis Requires Breaking Symmetry

Piotr Kawalek  

TU Wien, Austria

Jagiellonian University in Kraków, Poland

Armin Weiß  

University of Stuttgart, Germany

Abstract

The Constant Degree Hypothesis was introduced by Barrington et. al. [5] to study some extensions of q -groups by nilpotent groups and the power of these groups in a computation model called NuDFA (non-uniform DFA). In its simplest formulation, it establishes exponential lower bounds for $\text{MOD}_q \circ \text{MOD}_m \circ \text{AND}_d$ circuits computing AND of unbounded arity n (for constant integers d, m and a prime q). While it has been proved in some special cases (including $d = 1$), it remains wide open in its general form for over 30 years.

In this paper we prove that the hypothesis holds when we restrict our attention to symmetric circuits with m being a prime. While we build upon techniques by Grolmusz and Tardos [23], we have to prove a new symmetric version of their *Degree Decreasing Lemma* and use it to simplify circuits in a symmetry-preserving way. Moreover, to establish the result, we perform a careful analysis of automorphism groups of $\text{MOD}_m \circ \text{AND}_d$ subcircuits and study the periodic behaviour of the computed functions. Our methods also yield lower bounds when d is treated as a function of n .

Finally, we present a construction of symmetric $\text{MOD}_q \circ \text{MOD}_m \circ \text{AND}_d$ circuits that almost matches our lower bound and conclude that a symmetric function f can be computed by symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits of quasipolynomial size if and only if f has periods of polylogarithmic length of the form $p^k q^\ell$.

2012 ACM Subject Classification Theory of computation \rightarrow Circuit complexity; Theory of computation \rightarrow Complexity classes

Keywords and phrases Circuit lower bounds, constant degree hypothesis, permutation groups, CC^0 -circuits

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.58

Related Version *Full Version*: <https://arxiv.org/abs/2311.17440> [32]

Funding *Piotr Kawalek*: This research was funded in whole or in part by National Science Centre, Poland #2021/41/N/ST6/03907. 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. Funded by the European Union (ERC, POCOCOP, 101071674). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Armin Weiß: Partially funded by German DFG Grant WE 6835/1-2.

1 Introduction

Establishing strong lower bounds for general Boolean circuits represents one of the paramount and yet unattained objectives in the field of Computational Complexity Theory. Whenever such lower bounds can be obtained, it is usually in some very restricted setting. One of the standard limitations imposed on circuits in this context is the restriction of their depth. Some strong results were obtained when the circuits have depth bounded by a constant h



© Piotr Kawalek and Armin Weiß;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 58; pp. 58:1–58:21



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



and are built of unbounded fan-in Boolean AND/OR gates and unary \neg gates (so-called AC^0 circuits). By a classical result of Furst, Saxe and Sipser [19], proved independently by Ajtai [1], polynomial-size AC^0 circuits cannot compute the PARITY function (i.e., the sum of the input bits modulo 2). In fact, a followup paper by Yao [38] strengthens the lower bound for n -ary PARITY to be of the form $2^{\Omega(n^c)}$, with a final result of Håstad [26] finding a precise $c = \frac{1}{h-1}$. Interestingly, extending the AC^0 lower bounds from PARITY to MOD_m (i.e. the characteristic function of addition modulo an arbitrary integer m) can be achieved by the very same proof as in [26]. For a precise formulation and even more general results in this direction, see the subsequent work by Smolensky [36].

Here, a natural dual question arises: can modulo counting gates represent the n -ary Boolean AND_n function in the bounded-depth setting? To be more precise, a $CC_h[m]$ circuit is a circuit of depth h using only (unbounded fan-in) MOD_m^R gates. Each such gate sums the inputs modulo m and outputs 1 if the sum belongs to the set R (we allow different $R \subseteq [m]$ for different gates), otherwise it outputs 0. Thus, the question is, after fixing h and m , what size does a $CC_h[m]$ circuit require to compute AND_n ? Is there a polynomial-size construction for AND_n , making the class ACC^0 collapse to CC^0 (where $CC^0 = \bigcup_{h,m} CC_h[m]$)? The first question has a trivial answer when m is a prime power, as then $CC_h[m]$ circuits can express only bounded-arity AND (see [5] or [29] for more details). Surprisingly, for m having multiple prime divisors, only slightly super-linear lower bounds are known [12] and only for the number of wires – even more: to the best of our knowledge it is consistent with the current understanding that $NP \subseteq CC_2$ [6]. At the same time, the current best construction for AND_n has size $2^{O(n^c)}$ [11, 29] for some constant c depending on h and m .

This huge gap between lower and upper bounds suggests that the problem of establishing lower bounds in this context is very difficult. Hence, one can consider simpler computational models before answering the above more general questions. Interestingly, group theory outlines in-between steps which can be considered in this context. Barrington, Straubing and Thérien [5] studied a model of non-uniform DFA (NuDFA) over finite groups (or, more generally, monoids), which they used to recognize Boolean languages. They discovered that, if a group is an extension of a p -group by an abelian group, then its corresponding NuDFA can recognize all languages (however, most of them in exponential size). Nevertheless, such NuDFAs cannot compute AND_n unless they have size at least $2^{\Omega(n)}$ [5]. Later this result was restated in a circuit language, saying that, if m is an integer and q is a prime, then any 2-level $MOD_q \circ MOD_m$ circuit computing AND_n requires size $2^{\Omega(n)}$ [23, 22, 37] (here, as usual, the circuits have to be read that the MOD_q gate is the output gate – other than e.g. in [29]). The equivalence of the two statements is due to the fact that these (solvable) groups have an internal structure based on modulo counting. The authors of [5] conjectured that this $2^{\Omega(n)}$ lower bound generalizes to NuDFAs over extensions of nilpotent groups by p -groups. This again can be reformulated to a $2^{\Omega(n)}$ lower bound for $MOD_q \circ MOD_m \circ AND_d$ circuits computing AND_n (see [23]). This conjecture is known as Constant Degree Hypothesis (CDH for short), whose name corresponds to adding a layer of constant-arity AND_d gates on the input level to a $MOD_q \circ MOD_m$ circuit. Interestingly enough, recently in [30] it was proven that all the other groups (which do not correspond to CDH) do not admit this lower bound, i.e. one can construct AND_n of size $2^{O(n^c)}$ for some $c < 1$ using NuDFAs (or corresponding circuits) over these groups. In particular, it follows from [30] as well as the related work [3, 29] that the only $MOD_{m'} \circ MOD_m \circ AND_d$ circuits (where m, m' are arbitrary integers) for which subexponential constructions of AND_n are *not* known are either the ones described by CDH (i.e., m, m' prime) or such circuits with $m = p^\alpha, m' = p^\alpha q^\beta$, where p^α, q^β are powers of different primes. However, in the latter case, replacing $m' = p^\alpha q^\beta$ with just q^β does not

meaningfully change the expressive power of the related circuits (based on [29]). As a result, $\text{MOD}_q \circ \text{MOD}_m \circ \text{AND}_d$ circuits are really the only (algebraically) natural subclass of CC^0 circuits for which these strong $2^{\Omega(n)}$ lower bounds remain to be proven (or disproven).

Low-level CC^0 circuits have many surprising connections. For instance, the techniques used in the construction of relatively small CC^0 circuits for the OR_n function (equivalently, AND_n) found in [3] are useful in constructing small explicit Ramsey-type graphs [21, 20]. These constructions are also used to produce better locally-decodable error-correcting codes [17, 15], private information retrieval schemes [16], and secret sharing schemes [33]. The lower bounds for codes considered in [17] imply lower bounds for certain CC^0 circuits. On the contrary, good lower bounds for low-level CC^0 circuits imply faster algorithms for solving equations in solvable groups [30], faster algorithms for certain algebraic versions of circuit satisfiability problems [28] and also faster algorithms for some variants of the Constraint Satisfaction Problem with Global Constraints [8].

These diverse interconnections encourage to put even more effort to find the correct sizes for optimal modulo-counting circuits computing AND_n . In this pursue, proving (or disproving) CDH plays a central role. The hypothesis is already proven in several special cases: in particular, the case $d = 1$ was confirmed in the very same paper the hypothesis was defined. Moreover, if there is a bound on the number of AND_d gates that are wired to each MOD_m gate, the desired lower bound is also true [23]. More precisely, the number of such connections is required to be $o(\frac{n^2}{\log n})$. The technique used in this case is based on the so-called *Degree Decreasing Lemma*, whose name corresponds to gradually decreasing the degree d , which eventually leads to the $d = 1$ case. The Degree Decreasing Lemma can also be used when the polynomials over \mathbb{Z}_m corresponding to the $\text{MOD}_m \circ \text{AND}_d$ part of the circuit can be written using a sublinear number of binary multiplications [21].

In many studies of different circuit complexity classes, *symmetry* seems to play an important role. In this context both symmetric circuits, as well as symmetric functions were considered. Here, symmetry for a circuit/function means that permuting its inputs/variables does not change the considered circuit/function. Let us here mention the recent results on lower bounds for symmetric arithmetic circuits for the permanent and also a construction of short symmetric circuits for the determinant [13] as well as the lower bound from [27] for computing a certain entry in a product of matrices (here, symmetry means invariance under permuting rows and columns of matrices).

Symmetry seems to play also a special role for $\text{CC}_h[m]$ circuits. The remarkable construction of relatively small circuits for AND_n in [3] uses symmetric polynomials as an intermediate object before translating them to circuits. This translation, when done carefully, leads also to symmetric circuits. Similarly, some of the newer, more optimal constructions of two level $\text{CC}_2[m]$ circuits for AND_n can be performed fully symmetrically [11, 29]. Additionally, [23, 22, 37] analyze the periodic behaviour of the symmetric functions that can be represented by small (not necessarily symmetric) $\text{MOD}_q \circ \text{MOD}_m$ circuits. A value of a symmetric Boolean function $f(x_1, \dots, x_n)$ is determined by the number of ones among x_1, \dots, x_n . Hence, for an integer $0 \leq m \leq n$, we can naturally define $f(m)$ as $f(1^m 0^{n-m})$ and say that an integer r is a period of f whenever $f(m+r) = f(m)$ for all $0 \leq m \leq n-r$. It follows from [23, 37] that the only symmetric functions that have representations as $\text{MOD}_q \circ \text{MOD}_m$ circuits of subexponential size must have periods of the form $m \cdot q^k$ with $m \cdot q^k \leq n$. In particular, AND_n must have exponential-size circuits.

The dual question, namely the behaviour of symmetric functions computed by small AC^0 circuits, has been studied quite a lot: In [14], polynomial-size symmetric AC^0 circuits of arity n are shown to represent only functions that are constant on the interval $\{n^\varepsilon, \dots, n - n^\varepsilon\}$ (for large enough n).

The same result has been obtained in [18] also showing that, if a symmetric function f is constant on the interval $\{\log^k n, \dots, n - \log^k n\}$ (for some k and for large enough n), then it is in AC^0 . Soon after, [9] showed that the latter condition is actually an if and only if.

These results were extended to $\text{AC}^0[p]$ circuits of quasipolynomial size by Lu [34]: $f = (f_n)_{n \in \mathbb{N}}$ is symmetric with $f \in \text{qAC}^0[p]$ if and only if f_n has period $p^{t(n)} = \log^{O(1)} n$ except at both ends of length $\log^{O(1)} n$. Here, for a function $f : \{0, 1\}^* \rightarrow \{0, 1\}$, we write $f = (f_n)_{n \in \mathbb{N}}$ where $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$ is the restriction of f to $\{0, 1\}^n$. As usual we say that f is computed by a family of circuits if for each n there is a circuit computing f_n .

For further results in this direction allowing threshold or majority gates see [4, 24, 39]. Recently, a new technique called torus polynomials were introduced [6] as a possible method to separating TC^0 from ACC^0 and was shown that MAJORITY cannot be approximated by small-degree *symmetric* torus polynomials.

Contribution. In this paper we prove that symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits computing AND_n have exponential size. A key to the proof is to analyze the periodic behaviour of the functions computed by such circuits. Our techniques work also when d is unbounded and is considered as a function of n . The following theorem characterizes the periodic behaviour of such functions.

► **Theorem 1.** *Let p and q be primes and $n \geq 13$ and let $1 \leq d \leq n$. Then any function computed by an n -input symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit of size $s < 2^{n/9}$ has a period $p^{k_p} q^{k_q}$ given that $p^{k_p} > d$ and $q^{k_q} > \log s + 1$.*

To fully understand the periodic behaviour of $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits, we would also like to construct relatively small circuits given a function f with period of the form $p^{k_p} q^{k_q}$. We present such a construction below in Proposition 19. The most interesting consequence of this construction is that we get a tight characterization of the periodic behaviour of functions computed by quasipolynomial-size $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits (recall that a quasipolynomial is a function of the form $2^{\log^k n}$ for some constant k).

► **Corollary 2.** *Let $p \neq q$ be primes and $d : \mathbb{N} \rightarrow \mathbb{N}$ with $d(n) \leq n/2$ for all n . A function $f = (f_n)_{n \in \mathbb{N}}$ (with $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$) can be computed by symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_{d(n)}$ circuits of quasipolynomial size if and only if, for each n , f_n has a period $p^{k_p(n)} q^{k_q(n)} \in \log^{O(1)}(n)$ for some functions $k_p, k_q : \mathbb{N} \rightarrow \mathbb{N}$.*

Next let us consider the case of the AND_n function more carefully. The following theorem is a careful application of Theorem 1.

► **Theorem 3.** *Let p and q be primes, let n be a large enough integer, and let $d \leq \sqrt{n}$. Then every symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit computing the AND_n function has size at least $2^{n/(2dpq)}$.*

Note that the restriction $d \leq \sqrt{n}$ still includes the most interesting case. Indeed, for $\sqrt{n} \leq d \leq n - \sqrt{n}$ we get an almost trivial lower bound of $2^{\sqrt{n}}$ (see Theorem 21). Moreover, Theorem 3 suggests an interesting trade-off between the degree and the size at $d \approx \sqrt{n}$. Then we can reach a lower bound for the size of the form $2^{\Omega(\sqrt{n})}$.

As a direct consequence of Theorem 3 we get the desired result for AND_n .

► **Corollary 4.** *For constant d , and primes p , and q every symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit for AND_n has size at least $2^{\Omega(n)}$. Thus, CDH holds for symmetric circuits with p being prime.*

Before we go to the more technical part, let us briefly mention an opposite perspective on the results of this paper. Although current evidence seems to support CDH and lower bounds for AND_n for general $\text{CC}_h[m]$ circuits, it is known that $\text{CC}_h[m]$ circuits using $O(\log n)$ random bits are able to compute AND_n in polynomial size [25]. This was even improved in [31], by showing that $\text{MOD}_q \circ \text{MOD}_p$ circuits can also be used for representing AND_n in this probabilistic model. This might be interpreted as an argument against lower bounds, because now it is enough to derandomize the construction for $\text{MOD}_q \circ \text{MOD}_p$ circuits. We already understand these 2-level circuits relatively well (to the point that we can prove strong lower bounds for them for the AND_n function itself). Our Corollary 4 implies that one cannot construct AND_n with polynomial-size symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits. Hence, to make short deterministic constructions one needs to either go beyond the symmetric setting or consider larger depths.

Outline. The paper is organized as follows: in Section 2 we fix our notation on circuits as well as hypergraphs and group actions. These notions are essential in the later study of the symmetric structure of circuits. In Section 3, we describe how to rewrite a circuit into a nicer form that we use throughout the paper. Then in Section 4 we present our key lemmas including proofs or short proof sketches and show how to derive our main results. The missing proofs can be found in the full version on arXiv [32]. In Section 5 we add some discussion of our results.

2 Preliminaries

Hypergraphs. For $d \in \mathbb{N}$ we write $[d]$ for the set of integers $\{1, \dots, d\}$. For a set X we denote its power set by $\mathcal{P}(X)$. A *hypergraph* on a set of vertices V is a pair (V, E) with $E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$. A \mathbb{F}_p -*labeled hypergraph* is a pair $G = (V, \lambda)$ where $\lambda: (\mathcal{P}(V) \setminus \{\emptyset\}) \rightarrow \mathbb{F}_p$. We obtain an (unlabeled) hypergraph by setting $E = \{e \subseteq V \mid \lambda(e) \neq 0\}$ and call each e with $\lambda(e) \neq 0$ an *edge* of G . Thus, an \mathbb{F}_p -labeled hypergraph is indeed a hypergraph where we assign to each edge a number from $\mathbb{F}_p \setminus \{0\}$. Moreover, (V, λ) is called an \mathbb{F}_p -*labeled d -hypergraph* if for all $e \in \mathcal{P}(V)$ with $|e| > d$ we have $\lambda(e) = 0$. We write $\mathcal{H}_p^d(V)$ for the set of \mathbb{F}_p -labeled d -hypergraphs on V . For $C \subseteq V$ we write $\overline{C} = V \setminus C$ for the complement of C .

If $G = (V, \lambda)$ and $H = (V, \zeta)$ are \mathbb{F}_p -labeled hypergraphs on the same set of vertices V , we define $G + H$ (resp. $G - H$) as $(V, \lambda + \zeta)$ (resp. $(V, \lambda - \zeta)$) where $\lambda + \zeta$ denotes the point-wise addition. We interpret any subset $E \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$ as a hypergraph by setting $\lambda(e) = 1$ if $e \in E$ and $\lambda(e) = 0$ otherwise (be aware of the slight ambiguity as V is not uniquely defined by E – but it always will be clear from the context). Thus, we have defined the addition $G + E$ (resp. $G - E$). We extend this to $G + e = G + \{e\}$.

Permutation groups. For any set V we denote the group of permutations on V by $\text{Sym}(V)$ (i.e., the symmetric group). For an integer n we write $\text{Sym}(n)$ or S_n for the abstract symmetric group acting on any n -element set. Any subgroup $\Gamma \leq \text{Sym}(V)$ acts faithfully on V and is called a permutation group. A subset $U \subseteq V$ is called an *orbit* of the action of Γ on V if $U = G \cdot x$ for some $x \in V$. If there is only one orbit, the action of Γ on V is called transitive. Clearly, the orbits form a partition of V ; moreover, if $U_1, \dots, U_k \subseteq V$ are the orbits of the action of $\Gamma \leq \text{Sym}(V)$ on V , then $\Gamma \leq \text{Sym}(U_1) \times \dots \times \text{Sym}(U_k)$ where \times denotes the direct product of groups.

Finally, let $\Gamma' \leq \Gamma$ be a subgroup. A *left-transversal* (in the following simply *transversal*) of Γ' in Γ is a subset $R \subseteq \Gamma$ such that R is a system of representatives of Γ/Γ' – in other words, if $R\Gamma' = \Gamma$ and $r\Gamma' \cap s\Gamma' = \emptyset$ for $r, s \in R$ with $r \neq s$. For further details on permutation groups, we refer to [10].

Actions on hypergraphs. Given an action of $\text{Sym}(V)$ on V , it induces an action on $\mathcal{P}(V)$. Moreover, this extends to an action on $\mathcal{H}_p^d(V)$ where a permutation $\pi \in \text{Sym}(V)$ maps (V, λ) to $\pi((V, \lambda)) = (V, \pi\lambda)$ for $\pi\lambda$ defined by $(\pi\lambda)(e) = \lambda(\pi^{-1}(e))$. Be aware that the $^{-1}$ is not by accident but rather guarantees that if some $e \in \mathcal{P}(V)$ has label $\gamma = \lambda(e)$, then $\pi(e)$ has label $\pi\lambda(\pi(e)) = \lambda(e)$. Note that two labeled hypergraphs with vertices V are isomorphic if and only if they are in the same orbit under $\text{Sym}(V)$. A permutation $\pi \in \text{Sym}(V)$ is called an *automorphism* of $G = (V, \lambda)$ if $\pi(G) = G$ – with other words, if $\lambda(\pi(e)) = \lambda(e)$ for all $e \in \mathcal{P}(V)$. For a labeled hypergraph G , we denote its group of automorphisms by $\text{Aut}(G)$.

Circuits. A circuit is usually defined as a directed acyclic graph with labels on its vertices that inform what kind of operation (like for instance $\wedge, \vee, \neg, \text{MOD}_p^R$) a given vertex (gate) computes. We allow multiple edges between any pair of gates. A depth- d circuit of arity n is a circuit that consists of n input gates x_1, \dots, x_n and d layers (or levels) G_1, \dots, G_d of inner gates (we do not count the input gate as a level). Between neighbour layers G_{i-1} and G_i there is a layer of wires W_i which contains directed edges between $g \in G_{i-1}$ and $h \in G_i$ (where $G_0 = \{x_0, \dots, x_n\}$). We allow for multiple (directed) edges between the same pair of gates. Moreover, gates are labeled with necessary information which allows to compute a function they represent. In our case we use MOD_p^R gates where p is a prime and $R \subseteq \mathbb{F}_p$. A MOD_p^R with inputs y_1, \dots, y_k outputs 1 if and only if the sum of its inputs modulo p is contained in R . A circuit is called an *expression* if it is a tree when removing the input layer. A subexpression of an expression is a subgraph containing for every gate also all its predecessors (towards the input gates). For circuits C, D with n inputs we write $C \equiv D$ if for all inputs $\bar{b} \in \{0, 1\}^n$ they evaluate to the same value. We define the *size* of a circuit as its number of non-input gates.

In this article we consider $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits: such a circuit consist of 3-levels. On level 1 there are AND_d gates each of which receives inputs from at most d input gates. The second level G_2 consists of MOD_p^R gates – each of them is labeled with an accepting set $R \subseteq \{0, \dots, p-1\}$. The output layer G_3 contains only one MOD_q^R gate, which sums all the wires from W_3 modulo q .

We say that a circuit C is symmetric if no permutation of the input wires changes the circuit. Note that here the word *symmetric* refers to a *syntactic* structure of a circuit, rather than a semantic property of the function computed by it. More formally, a circuit C on inputs x_1, \dots, x_n is called *symmetric* if for any $\pi \in \text{Sym}(\{x_1, \dots, x_n\})$ there is a permutation π' on the set of gates extending π (meaning that $\pi(x_i) = \pi'(x_i)$ for all $i \in [n]$) such that there are k wires connecting gate i to gate j if and only if there are k wires connecting gates $\pi'(i)$ to gate $\pi'(j)$.

3 Preparation: Circuits, Expressions and Hypergraphs

For a simpler notation of expressions, let us denote MOD_p^R with inputs y_1, \dots, y_k instead by $\mathbf{b}(\sum_{i=1}^k y_i; R)$ for $R \subseteq \mathbb{F}_p$, where \mathbf{b} computes the function

$$\mathbf{b}(y; R) = \begin{cases} 1 & \text{if } y \in R \\ 0 & \text{if } y \notin R. \end{cases}$$

Be aware that we use \mathbf{b} for different domains, i.e. as a function $\mathbb{F}_p \rightarrow \{0, 1\}$ and $\mathbb{F}_q \rightarrow \{0, 1\}$. The domain will be clear from the context.

From circuits to expressions. Any 2-level $\text{MOD}_p \circ \text{AND}_d$ circuit corresponds to polynomial over the field \mathbb{F}_p . Indeed, the AND_d gates act like a multiplications on the two element domain $\{0, 1\} \subseteq \mathbb{F}_p$ and the MOD_p^R gate sums the results and checks whether the sum belong to the accepting set R . Because our circuits are of constant-depth, we can unfold the circuits to obtain expressions. This means, if a gate g has an outgoing wire to several other gates, we create multiple copies of g , so that each gate has only a single output wire. Note that this might lead to a polynomial blow-up in size (more precisely, a circuit with size s and depth bounded by h is converted to an expression of size at most s^{h-1} – thus, in our case s^2). Moreover, note that unfolding the circuit does not destroy the property of being symmetric. Hence, every symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit yields also a symmetric expression

$$\mathbf{b}\left(\sum_{i=0}^l \alpha_i \mathbf{b}(\mathbf{p}_i(\bar{x}); R_i); R\right) \tag{1}$$

for suitable $\alpha_i \in \mathbb{F}_q$, $R_i \subseteq \mathbb{F}_p$ and polynomials \mathbf{p}_i of degree bounded by d for $i \in \{1, \dots, l\}$ and $R \subseteq \mathbb{F}_q$ which computes the same function. Here, l is the number of MOD_p gates used in the $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit, while α_i tells us how many times a given MOD_p gate is wired to the MOD_q gate. Let us take a closer look at what being symmetric means for an expression of the form (1): for each $\pi \in \text{Sym}(n)$ there exist $\pi' \in S_l$ such that for all $i \in \{1, \dots, l\}$ we have $\alpha_i = \alpha_{\pi'(i)}$, $R_i = R_{\pi'(i)}$, and $\mathbf{p}_i(x_1, \dots, x_n) = \mathbf{p}_{\pi'(i)}(x_{\pi(1)}, \dots, x_{\pi(n)})$ (here = refers to equality in the polynomial ring $\mathbb{F}_p[x_1, \dots, x_n]$).

Next, observe that if we omit the outer \mathbf{b} of the expression (1), the function computed by the resulting expression certainly does not have any new (smaller) periods than the one of the complete expression. Therefore, as we are aiming for an upper bound on the periods of the considered symmetric circuits, we will now concentrate on the symmetric expressions of the form

$$\mathbf{f} = \sum_{i=0}^l \alpha_i \mathbf{b}(\mathbf{p}_i(\bar{x}); R_i) \tag{2}$$

with $R_i \subseteq \mathbb{F}_p$, $\alpha_i \in \mathbb{F}_q$ and polynomials \mathbf{p}_i of degree bounded by d . Indeed, every period of \mathbf{f} is a period of $\mathbf{b}(\mathbf{f})$, so for proving lower bounds, it is enough to consider the periods of \mathbf{f} . An expression of the form (2) is called a $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression and each $\mathbf{b}(\mathbf{p}_i(\bar{x}); R_i)$ is an *elementary subexpression* of \mathbf{f} .

In the following, let us write $\mathbf{b}(\mathbf{p}(\bar{x}); r)$ for $\mathbf{b}(\mathbf{p}(\bar{x}); \{r\})$. Using this notation we have $\mathbf{b}(\mathbf{p}(\bar{x}); R) = \sum_{r \in R} \mathbf{b}(\mathbf{p}(\bar{x}); r)$. Moreover, we always assume that for $i \neq j$ we have $(\mathbf{p}_i, r_i) \neq (\mathbf{p}_j, r_j)$ as otherwise we can replace $\alpha_i \mathbf{b}(\mathbf{p}_i(\bar{x}); r_i) + \alpha_j \mathbf{b}(\mathbf{p}_j(\bar{x}); r_j)$ by $\alpha_{ij} \mathbf{b}(\mathbf{p}_i(\bar{x}); r_i)$ where $\alpha_{ij} = \alpha_i + \alpha_j$. Thus, using $\text{pol}(n, d)$ to denote the set of multilinear polynomials in $\mathbb{F}_p[x_1, \dots, x_n]$ with degree bounded by d , we rewrite \mathbf{f} in (2) as

$$\mathbf{f} = \sum_{\mathbf{p} \in \text{pol}(n, d)} \sum_{r \in \mathbb{F}_p} \alpha_{\mathbf{p}, r} \mathbf{b}(\mathbf{p}(\bar{x}); r). \tag{3}$$

Note that to compute the size of \mathbf{f} we only need to count the non-zero $\alpha_{\mathbf{p}, r}$ (plus the number of AND gates computing the polynomials \mathbf{p}).

Polynomials and hypergraphs. Let us take a closer look at the $\text{MOD}_p \circ \text{AND}_d$ part of a $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit or expression. As any such expression is represented by a polynomial of degree d , we will need to deal with these polynomials and their symmetries. Notice that without loss of generality, we can assume that the polynomial corresponding to a $\text{MOD}_p \circ \text{AND}_d$ circuit is multi-linear since, because the values of variables are restricted to $\{0, 1\}$ each occurrence of a higher power x^k of a variable x can be simply replaced by x .

In order to deal better with the combinatorics and symmetries of polynomials, we think of polynomials as hypergraphs. A multilinear polynomial $\mathbf{p} \in \mathbb{F}_p[x_1, \dots, x_n]$ with the degree bounded by d can be naturally identified with an \mathbb{F}_p -labeled d -hypergraph $G = (V, \lambda)$ as follows:

1. Treat each variable x_i in $\mathbf{p}(x_1, \dots, x_n)$ as a vertex in the graph $G_{\mathbf{p}}$. Thus, $V = \{x_1, \dots, x_n\}$, which we also identify with the set $[n]$.
2. Each monomial $\gamma \cdot x_1 \cdot \dots \cdot x_d$ is represented by a hyperedge with a label γ , i.e., we have $\lambda(\{x_1, \dots, x_d\}) = \gamma$.

Thus, we get a one-to-one correspondence between multilinear polynomials over \mathbb{F}_p of degree at most d and \mathbb{F}_p -labeled d -hypergraphs. This means that we can also do the reverse – for each labeled graph G we can create its corresponding polynomial \mathbf{p}_G . Moreover, note that also the arithmetic operations we defined on hypergraphs as well as the group actions agree with those on polynomials. Therefore, in the following, we use polynomials and hypergraphs interchangeably.

Now, we can use our graph notation for polynomials in a more general setting and denote each expression $\mathbf{b}(\mathbf{p}(\bar{x}); r)$ by $\mathbf{b}(G_{\mathbf{p}}; r)$ or simply $\mathbf{b}(G; r)$ (when we start with a hypergraph representing a given polynomial). Thus, we can reformulate any expression of the form (3) as $\sum_{\mathbf{p} \in \text{pol}(n, d)} \sum_{r \in \mathbb{F}_p} \alpha_{\mathbf{p}, r} \mathbf{b}(G_{\mathbf{p}}; r) = \sum_{G \in \mathcal{H}_p^d(V)} \sum_{r \in \mathbb{F}_p} \alpha_{G, r} \mathbf{b}(G; r)$.

Symmetric expressions induced by hypergraphs. Now we define several notions, useful in analysing symmetric expressions. For $G = (V, \lambda)$ and $\pi \in \text{Sym}(V)$, let us write $\mathbf{b}^\pi(G; R)$ for $\mathbf{b}(\pi G; R)$. The action of $\text{Sym}(V)$ on V now extends naturally to an action on expressions of the form $\mathbf{f} = \sum_{G \in \mathcal{H}_p^d(V)} \sum_{r \in \mathbb{F}_p} \alpha_{G, r} \mathbf{b}(G; r)$ by setting

$$\pi(\mathbf{f}) = \sum_{G \in \mathcal{H}_p^d(V)} \sum_{r \in \mathbb{F}_p} \alpha_{G, r} \mathbf{b}^\pi(G; r).$$

Now, \mathbf{f} being symmetric can be simply expressed as the fact that for each $\pi \in \text{Sym}(V)$ we have $\pi(\mathbf{f}) = \mathbf{f}$.

► **Definition 5.** Let $G = (V, \lambda)$ be a labeled d -hypergraph. Let $\text{Aut}(G)$ be its group of automorphisms and let π_1, \dots, π_k be a transversal of $\text{Sym}(V)/\text{Aut}(G)$. For a given $r \in \mathbb{F}_p$, define $\mathbf{s}(G; r)$ to be the following $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression

$$\mathbf{s}(G; r) = \sum_{i=0}^k \mathbf{b}^{\pi_i}(G, r). \quad (4)$$

One needs to check that the above definition does not depend on the choice of the transversal, as there is a choice in picking the specific traversal π_1, \dots, π_k which we use to create $\mathbf{s}(G; r)$. However, as G is invariant under its automorphisms, no matter how we choose the specific π_1, \dots, π_k , we get the same expression in the end. In fact, every symmetric $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression containing $\mathbf{b}(G; r)$ as subexpression, must also contain $\mathbf{s}(G; r)$ as subexpression. So $\mathbf{s}(G; r)$ is a symmetric closure of the basic expression $\mathbf{b}(G; r)$. Let us summarize this as follows:

► **Remark 6.** For every labeled d -hypergraph G and every $r \in \mathbb{F}_p$, the expression $\mathbf{s}(G; r)$ is symmetric. Moreover, it is the smallest symmetric expression that contains $\mathbf{b}(G; r)$ as an elementary subexpression.

► **Fact 7.** Every symmetric $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression \mathbf{f} can be written as a sum

$$\mathbf{f}(\bar{x}) = \sum_{G \in \mathcal{H}_p^d(V)} \sum_{r \in \mathbb{F}_p} \beta_{G,r} \cdot \mathbf{s}(G; r)$$

for $\beta_{G,r} \in \mathbb{F}_q$ (recall that $\mathcal{H}_p^d(V)$ denotes the set of labeled d -hypergraphs on V).

Proof. If a symmetric \mathbf{f} has some $\beta \cdot \mathbf{b}(G, r)$ as an elementary subexpression, it must also have $\beta \cdot \mathbf{s}(G; r)$ as a subexpression (see Remark 6). But now $\mathbf{f} - \beta \cdot \mathbf{s}(G; r)$ is a symmetric expression which is shorter than \mathbf{f} , and hence we can use induction to prove the desired decomposition for $\mathbf{f}(\bar{x})$, by adding $\beta \cdot \mathbf{s}(G; r)$ to the decomposition of $\mathbf{f} - \beta \cdot \mathbf{s}(G; r)$. ◀

4 Description of the Proof

We now start with an expression as in Fact 7 and prove our main theorems. For this, we need several definitions and intermediate results. For some of these intermediate results, the full proofs are deferred to the full version [32]; instead, we present short proof sketches, give some high-level ideas how the respective results are used, and then, in Section 4.4, show how our main results follow from the intermediate results. As every symmetric expression \mathbf{f} is decomposed into an appropriate sum of elements of the form $\alpha \cdot \mathbf{s}(G; r)$, we need a deeper understanding of each $\mathbf{s}(G; r)$. We investigate these expressions $\mathbf{s}(G; r)$ in three main steps:

1. we analyze the symmetries of G to find a large so-called *fully symmetric set* (see Lemma 10),
2. we process the hypergraph G further to make it *symmetry purified* (see Definition 13 and Lemma 14) applying two versions of the Degree Decreasing Lemma (Lemma 11 and Lemma 12),
3. we analyze the periods of the resulting expressions $\mathbf{s}(G; r)$ (see Theorem 16).

4.1 Symmetries of Hypergraphs

Recall that one of our goals is to prove exponential lower bounds on the size of symmetric circuits/expressions computing AND_n . In Lemma 10 we are going to show that, if in an expression \mathbf{f} we find a very asymmetric graph G , we know that the size of \mathbf{f} must be relatively large. This is because the automorphism group of G is small and, hence, the length of the expression of the form (4) induced by G , i.e. $\mathbf{s}(G; R)$, must be large (more precisely, k as defined above is large). On the other hand, for highly symmetric graphs G , we can find a big, very regular substructure of G , which we will call a *pseudo-clique*.

► **Definition 8.** Let G be an \mathbb{F}_p -labeled hypergraph $G = (V, \lambda)$ (i.e. $\lambda : \mathcal{P}(V) \setminus \{\emptyset\} \rightarrow \mathbb{F}_p$). We say that a subset $C \subseteq V$ is *fully symmetric*, if for each pair of subsets $e_1, e_2 \subseteq V$ with $|e_1| = |e_2|$ and $e_1 \cap \bar{C} = e_2 \cap \bar{C}$ we have $\lambda(e_1) = \lambda(e_2)$.

Moreover, an \mathbb{F}_p -labeled hypergraph $G = (V, \lambda)$ is called a *pseudo-clique* if $\text{Aut}(G) = \text{Sym}(V)$ – or, equivalently, if for each $d \in [n]$ there is some λ_d such that $\lambda(e) = \lambda_d$ all $e \subseteq V$ with $|e| = d$.

Note that an induced subgraph on a fully symmetric subset of vertices is a pseudo-clique. We obtain the following easy observations.

► **Fact 9.** Let G be an \mathbb{F}_p -labeled hypergraph $G = (V, \lambda)$.

- A subset $C \subseteq V$ is fully symmetric if and only if $\text{Sym}(C) \leq \text{Aut}(G)$.
- If $C, D \subseteq V$ are fully symmetric sets with $C \cap D \neq \emptyset$, then so is $C \cup D$.
- If $C \subseteq V$ is a maximal fully symmetric set with $|C| > |V|/2$, then $\text{Aut}(G) = \text{Sym}(C) \times \Gamma$ for some $\Gamma \leq \text{Sym}(\bar{C})$.

58:10 Violating Constant Degree Hypothesis Requires Breaking Symmetry

Now, we are ready to present a key lemma, which allows us to restrict our attention only to very symmetric hypergraphs.

► **Lemma 10.** *Let $0 < \varepsilon < 1/8$. Every \mathbb{F}_p -labeled hypergraph $G = (V, \lambda)$ with $n = |V| \geq 13$ has either*

- *a fully symmetric subset on at least $n - \lfloor \varepsilon n \rfloor$ vertices, or*
- *its automorphism group satisfies $|\text{Sym}(V)/\text{Aut}(G)| > 2^{\lfloor \varepsilon n \rfloor}$.*

Proof. Let us write $\Gamma = \text{Aut}(G)$. We say that Γ is *small* if $|\Gamma| \leq n!/2^{\lfloor \varepsilon n \rfloor}$. Let us first show that either Γ is small or G contains a pseudo-clique on at least $n - \lfloor \varepsilon n \rfloor$ vertices (in a second step, we will show that this pseudo-clique, indeed, is fully symmetric).

We start by observing that Γ is a subgroup of $\text{Sym}(k_1) \times \cdots \times \text{Sym}(k_m)$, where k_i are the sizes of the orbits of the action on G . If $k_i < n - \lfloor \varepsilon n \rfloor$ for all i , then $|\Gamma| < (n - \lfloor \varepsilon n \rfloor)! \cdot \lfloor \varepsilon n \rfloor!$ and Γ is small because of

$$\frac{n!}{|\Gamma|} > \frac{n!}{(n - \lfloor \varepsilon n \rfloor)! \cdot \lfloor \varepsilon n \rfloor!} = \binom{n}{\lfloor \varepsilon n \rfloor} \geq \left(\frac{n}{\lfloor \varepsilon n \rfloor}\right)^{\lfloor \varepsilon n \rfloor} > 2^{\lfloor \varepsilon n \rfloor}.$$

So from now on there is one orbit $C \subseteq V$ consisting of at least $n - \lfloor \varepsilon n \rfloor$ vertices. Then we have $\Gamma \leq \text{Sym}(C) \times \text{Sym}(\bar{C})$ and we denote by $\varphi: \Gamma \rightarrow \text{Sym}(C)$ the projection to the first coordinate.

Suppose $\tilde{\Gamma} = \varphi(\Gamma)$ does not act primitively on C meaning that there is an $\tilde{\Gamma}$ -invariant partition of C with r classes each of which consists of $1 < m < |C|$ vertices (as $\tilde{\Gamma}$ acts transitively on C , it must act transitively on the classes; hence, they all have the same size). Thus, $\tilde{\Gamma}$ is isomorphic to a subgroup of the wreath product $\text{Sym}(m) \wr \text{Sym}(r)$ with $rm = |C|$ (see [10, Theorem 1.8]). Hence,

$$\begin{aligned} \frac{|C|!}{|\tilde{\Gamma}|} &\geq \frac{|C|!}{(m!)^r \cdot r!} = \frac{1 \cdot \dots \cdot m \cdot \dots \cdot |C|}{(1 \cdot \dots \cdot m) \cdot 1 \cdot (1 \cdot \dots \cdot m) \cdot 2 \cdot \dots \cdot (1 \cdot \dots \cdot m) \cdot r} = \frac{\prod_{i=1}^{r-1} \prod_{j=1}^{m-1} (im + j)}{((m-1)!)^{r-1}} \\ &\geq 2^{(m-1)(r-1)} \geq 2^{|C|/4} \geq 2^{\lfloor \varepsilon n \rfloor}. \end{aligned}$$

Here the last inequality is because $\varepsilon < 1/8$ (in particular $1 - \varepsilon \geq 1/2$), the second last inequality is due to the assumption $\varepsilon \leq 1/4$ and $m - 1 \geq m/2$ and $r - 1 \geq r/2$. The third last inequality is because $(\prod_{j=1}^{m-1} (im + j))/(m-1)! = \prod_{j=1}^{m-1} (im + j)/j \geq 2^{m-1}$ as $i \geq 1$. Since the index of Γ in $\text{Sym}(V)$ is at least the index of $\tilde{\Gamma} = \varphi(\Gamma)$ in $\text{Sym}(C)$, again Γ is small.

Hence, it remains to consider the case that $\varphi(\Gamma) \leq \text{Sym}(C)$ acts primitively on C . Thus, writing $k = |C|$, according to [7, 35] (see also [2]), there are three possibilities: $\varphi(\Gamma)$ is either A_k (the alternating group on k elements) or $S_k \cong \text{Sym}(C)$ or $|\varphi(\Gamma)| \leq 4^k$. First, let us consider the last case. As $n \geq 13$, we have $k \geq n - \lfloor \varepsilon n \rfloor \geq 11$. Therefore, we conclude that we have $|\varphi(\Gamma)| \leq 4^k \leq k!/2^{k/4}$ (which holds for all $k \geq 11$) and, as above, the index of Γ in $\text{Sym}(V)$ is at least $2^{k/4} \geq 2^{(n - \lfloor \varepsilon n \rfloor)/4} \geq 2^{\lfloor \varepsilon n \rfloor}$, meaning that Γ is small.

In the former two cases (i.e., that $\varphi(\Gamma)$ is A_k or S_k), $\varphi(\Gamma)$ acts set-transitively on C (meaning that for each pair of subsets $A, B \subseteq C$ with $|A| = |B|$ there is a permutation π mapping A to B); hence, C is a pseudo-clique.

To see that C is, indeed, fully symmetric if Γ is not small, we proceed as follows: Now, let $N \leq \Gamma$ denote the kernel of the projection $\Gamma \rightarrow \text{Sym}(\bar{C})$ to the second component (i.e. the pointwise stabilizer of \bar{C}). Then we have $\varphi(N) = N$ (when identifying $\text{Sym}(C)$ with the corresponding subgroup of $\text{Sym}(C) \times \text{Sym}(\bar{C})$). As A_k is simple and the index of N is at most $(n - k)!$ in $\varphi(\Gamma)$ (which is either A_k or S_k), we have $N = A_k$ or $N = S_k$ (as N is also normal in $\varphi(\Gamma)$). In both cases N acts set-transitively on C and; hence, C is fully symmetric (which, by Fact 9, also excludes the case $N = A_k$). ◀

4.2 Reduction Based on the Degree Decreasing Lemma

One of the few examples of lower bounds for circuits using modulo counting are due to Grolmusz and Tardos [23, 22]. The authors prove lower bounds for $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits with restrictions put on connections between AND_d layer and MOD_p gates. More precisely, [22] shows that if the number of multiplications needed to compute the polynomial corresponding to each $\text{MOD}_p \circ \text{AND}_d$ subcircuit is bounded by cn for small enough c , then a $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit requires exponential size to compute AND_n . One of their key tools is the so-called Degree Decreasing Lemma:

► **Lemma 11** (Degree Decreasing Lemma). *Let $p \neq q$ be prime numbers. Then every function $f : \mathbb{F}_p^3 \mapsto \mathbb{F}_q$ represented by a 3-ary $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_2$ expression $\mathbf{b}(\gamma \cdot z_1 \cdot z_2 + y; t)$ can be also represented by an expression of the form*

$$\sum_{(j_1, j_2, j_3) \in \mathbb{F}_p^3} \sum_{r \in \mathbb{F}_p} \beta_{j_1, j_2, j_3}^{(r)} \cdot \mathbf{b}(j_1 z_1 + j_2 z_2 + j_3 y; r)$$

where $\beta_{j_1, j_2, j_3}^{(r)}$ are some coefficients from \mathbb{F}_q (also depending on γ and t).

The Lemma is a consequence of the result by Grolmusz [22, Lemma 6] (note that the statement there does not include the factor γ , so formally, to obtain Lemma 11, one needs to apply [22, Lemma 6] several times). One can also see it as a consequence of [29, Fact 3.3]. This very simple lemma allows us to navigate through the space of different representations for a given function f by allowing a local change of its corresponding expression. The power of the lemma comes from the fact that we can substitute arbitrary polynomials for z_1, z_2, y and obtain many different equivalences.

We will need a more regular version of the Degree Decreasing Lemma when the multiplication inside \mathbf{b} has bigger arity. The price we pay for a nicer form is that the represented function has a smaller (partially Boolean) domain, which slightly reduces the scope of applicability of the lemma (as we cannot substitute any polynomial for the variables); however, it still suffices for our purposes.

► **Lemma 12** (Symmetric Degree Decreasing Lemma). *Let $p \neq q$ be prime. Let $\gamma \in \mathbb{F}_p \setminus \{0\}$. Then every function $f : \{0, 1\}^d \times \mathbb{F}_p \mapsto \mathbb{F}_q$ represented by a $d + 1$ -ary $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression*

$$\mathbf{b}(\gamma \cdot x_1 \cdot \dots \cdot x_d + y; t)$$

can be also represented by an expression

$$\mathbf{h}(\bar{x}, y; t) = \mathbf{b}(y; t) + \sum_{r \in \mathbb{F}_p} \beta_{t,r} \sum_{S \subseteq [d]} \alpha_{|S|} \cdot \mathbf{b}(\gamma \cdot \sum_{s \in S} s + y; r)$$

for $\alpha_{|S|} = (-1)^{|S|}$ and some coefficients $\beta_{t,r} \in \mathbb{F}_q$.

The key property of the formula \mathbf{h} is that it is invariant under permutations of the variables x_1, \dots, x_d , which is not the case for original Degree Decreasing Lemma of [22]. The next step in the proof is to apply the Symmetric Degree Decreasing Lemma (Lemma 12) to expressions generated by highly-symmetric hypergraphs in order to obtain an even nicer representation defined as follows:

► **Definition 13.** *We call an \mathbb{F}_p -labeled d -hypergraph $G = (V, \lambda)$ symmetry-purified with respect to $C \subseteq V$ if*

1. C is fully symmetric in G ,

58:12 Violating Constant Degree Hypothesis Requires Breaking Symmetry

2. if $\lambda(e) \neq 0$, then e is completely contained either in C or in \overline{C} (i.e., every edge e is fully contained either in C or in \overline{C}),
3. if $\lambda(e) \neq 0$ and $e \subseteq \overline{C}$, then $|e| = 1$ (i.e., every edge e with $e \cap C = \emptyset$ satisfies $|e| = 1$).

Moreover, if the graph satisfies only conditions 1 and 2, we will call it partially symmetry purified. We write $\text{sp}(V, C)$ for the set of all symmetry-purified d -hypergraphs with respect to $C \subseteq V$ and $\text{psp}(V, C)$ for the set of partially symmetry-purified d -hypergraphs with respect to $C \subseteq V$ (note that d and p are implicitly defined from the context for $\text{sp}(V, C)$ and $\text{psp}(V, C)$).

The next crucial lemma allows us to restrict our attention only to expressions $\mathbf{s}(G; u)$ over symmetry-purified graphs, which have a very regular and much easier to analyze structure. This enables a later combinatorial analysis of the periodic behaviour of such $\mathbf{s}(G; u)$. The proof of the lemma relies on carefully applying both Lemma 11 and Lemma 12 to alter the graphs while preserving the symmetry of the corresponding expression.

► **Lemma 14.** *Let $p \neq q$ be prime numbers, let $u \in \mathbb{F}_p$, and let $G = (V, \lambda)$ be an \mathbb{F}_p -labeled d -hypergraph. Moreover, let $C \subseteq V$ be a maximal fully symmetric subset with $|C| > |V|/2$. Then there are constants $\beta_{H,r} \in \mathbb{F}_q$ such that*

$$\mathbf{s}(G; u) \equiv \sum_{H \in \text{sp}(V, C)} \sum_{r \in \mathbb{F}_p} \beta_{H,r} \mathbf{s}(H, r).$$

Proof sketch. We will first represent a function computed by $\mathbf{s}(G; u)$ by a sum of expressions $\mathbf{s}(H, r)$ for H being partially symmetry purified. Let e be an edge in G , such that $e^C = e \cap C \neq \emptyset$ and $e^{\overline{C}} = e \cap \overline{C} \neq \emptyset$. We would like to remove the edge e from G . To do so, we apply Lemma 11 to replace e by a linear combination of e^C and $e^{\overline{C}}$. However, if we do this only for e we may destroy the symmetry of the resulting expression. For this reason, we pick not only e but its entire orbit $O = \text{Aut}(G) \cdot e = \{e_1, \dots, e_\ell\}$ and apply Lemma 11 simultaneously to it. Indeed, setting $P = \text{Aut}(G) \cdot e^C$ and $Q = \text{Aut}(G) \cdot e^{\overline{C}}$, since C is fully symmetric, we have

$$e_1 + \dots + e_\ell = \left(\sum_{w \in P} w \right) \left(\sum_{v \in Q} v \right).$$

Since all the e_i must have the same label γ , we have

$$\mathbf{s}(G; u) = \mathbf{s}(\gamma(e_1 + \dots + e_\ell) + G'; u) = \mathbf{s}(\gamma \cdot z_1 \cdot z_2 + G'; u)$$

where $z_1 = \sum_{w \in P} w$, $z_2 = \sum_{v \in Q} v$ and $G' = G - \gamma \cdot (e_1 + \dots + e_\ell)$. Thus, after applying Lemma 11 to each summand of \mathbf{s} (as in the formula (4)), we get a symmetric expression which is a linear combination of subexpressions of the form $\mathbf{b}^{\pi_i}(j_1 z_1 + j_2 z_2 + j_3 G'; r)$. Hence, we have replaced $\mathbf{s}(G, u)$ with a sum $\sum_{H,r} \mathbf{s}(H, r)$, where each graph H does not contain the edge e anymore and has no new edges intersecting both C and \overline{C} non-trivially. We apply this reasoning recursively for $\mathbf{s}(H, r)$ as long as there is any edge in any graph in the sum that intersects non-trivially with C and \overline{C} . At the very end there are no such edges left, so we managed to represent $\mathbf{s}(G, u)$ as a sum of expressions $\mathbf{s}(H, r)$ over partially symmetry purified graphs H .

Next, assume that G is already partially symmetry purified with respect to C but not yet symmetry purified. Pick an edge $e \subseteq \overline{C}$ of size at least 2 and denote its orbit as $\text{Aut}(G) \cdot e = \{e_1, e_2, \dots, e_\ell\}$. We have

$$\mathbf{s}(G, u) = \sum_{i=1}^k \mathbf{b}^{\pi_i}(\gamma \cdot e_1 + \dots + \gamma \cdot e_\ell + G'; u).$$

Now by applying Lemma 12 subsequently to the edges e_1, \dots, e_ℓ , we can replace all these edges with linear combinations of its vertices. Because the formula in Lemma 12 is symmetric, we end up with a symmetric expression at the end. So we have replaced $\mathbf{s}(G, u)$ by a sum of $\mathbf{s}(H, r)$, but now each H has less bad edges. We apply this reasoning recursively to write $\mathbf{s}(G, u)$ as a sum of $\mathbf{s}(H, r)$ where each H is symmetry purified. ◀

4.3 Period of Symmetry-Purified Expressions

For a fixed input $\bar{b} \in \{0, 1\}^n$ and an n -ary symmetric expression \mathbf{f} , we can compute the value $\mathbf{f}(\bar{b})$ only knowing the Hamming weight of the input, i.e. the number of 1s in \bar{b} . This means that \mathbf{f} represents not only a function $\{0, 1\}^n \rightarrow D$, but we can also view it as a function $\{0, 1, \dots, n\} \rightarrow D$. It turns out that a relatively small $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit can compute only functions with a relatively small period. Here, by a period of \mathbf{f} we mean an integer $r \in \mathbb{N} \setminus \{0\}$ that satisfies $\mathbf{f}(m+r) = \mathbf{f}(m)$ for all m in the range $[0, n-r]$. Note that all functions have periods $> n$, so we are mainly interested in finding periods in the range $[1, n]$. Note that the AND_n function, which is of our particular interest, does not have any period (less than $n+1$). Thus, proving an upper bound for a period of a function computed by a relatively short expression will give us a lower bound for the length of representation of AND_n . This is in line with some of the previous research [3, 22, 37]. As any $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ can be transformed into a symmetric expression over symmetry-purified graphs, we only need to concentrate on these special graphs. Indeed, any common period among all the elements of the sum, transfers to the sum itself.

For the following theorem, we need a careful analysis how an expression $\mathbf{s}(G; u)$ for some symmetry purified graph G is computed. We rely on the fact that, for fixed $s \in \mathbb{N}$, the function $m \mapsto \binom{m}{s} \bmod p$ is periodic with period p^k for each $k \in \mathbb{N}$ such that $p^k > s$ (see for instance [29, Proof of Fact 3.4]). Recall that a multinomial coefficient $\binom{n}{s_1, \dots, s_l}$ counts the number of ordered partition of n elements set into sequences of disjoint subsets of sizes s_1, \dots, s_l . More formally, for $s_1 + \dots + s_l = n$ we have

$$\binom{n}{s_1, \dots, s_l} = \binom{s_1}{s_1} \cdot \binom{s_1 + s_2}{s_2} \cdot \dots \cdot \binom{s_1 + \dots + s_l}{s_l}. \tag{5}$$

▶ **Lemma 15.** *Let p be a prime. Let s_1, \dots, s_{l-1} be a sequence of integers. Let k be such that $p^k > s_1 + \dots + s_{l-1}$. Then the function*

$$f(n) = \binom{n}{s_1, \dots, s_l(n)} \bmod p$$

is periodic with period p^k where $s_l(n) = n - (s_1 + s_2 + \dots + s_{l-1})$.

Proof. We use the formula (5) for computing multinomial coefficient. Note that the first $l-1$ factors of the above product are constants, while the last one satisfies

$$\binom{s_1 + \dots + s_l}{s_l(n)} = \binom{n}{s_l(n)} = \binom{n}{n - s_l(n)} = \binom{n}{s_1 + \dots + s_{l-1}}.$$

So, periodicity of multinomial coefficient comes from periodicity of the binomial coefficients. ◀

▶ **Theorem 16.** *Let $p \neq q$ be prime numbers, $r \in \mathbb{F}_p$ and let G be an \mathbb{F}_p -labeled d -hypergraph on n vertices that is symmetry purified with respect to a maximal fully symmetric subset of vertices C of size $|C| > n/2$. Then $p^{k_p} \cdot q^{k_q}$ is a period of $\mathbf{s}(G; r)$ where k_p is the smallest integer satisfying $p^{k_p} > d$ and k_q is the smallest integer satisfying $q^{k_q} > n - |C|$.*

58:14 Violating Constant Degree Hypothesis Requires Breaking Symmetry

Note that, if $p^{k_p} \cdot q^{k_q} > n$, Theorem 16 establishes no non-trivial periods.

Proof. Note that the induced subgraph on the set C is a pseudo-clique. Moreover, as $|C| > \frac{|V|}{2}$, we can reconstruct the graph G up to isomorphism having the following information

1. the size l_C of the largest pseudo-clique C in G ,
2. the type $\vec{t} = (t_1, \dots, t_d) \in \mathbb{F}_p^d$ of the pseudo-clique, where t_i is the label of every i -ary edge in the pseudo-clique
3. sizes l_0, l_1, \dots, l_{p-1} , where l_i is the number of vertices j that are not in the pseudo-clique C and have a unary edge with label $i = \lambda(\{j\})$. Vertices corresponding to label i in G will be denoted L_i (thus, $l_i = |L_i|$).

Using this characterization of a symmetry purified hypergraph $G = (V, \lambda)$, we obtain

► **Fact 17.** $\text{Aut}(G) = \text{Sym}(C) \times \text{Sym}(L_0) \times \dots \times \text{Sym}(L_{p-1})$

In particular, we have at most $p + 1$ orbits under the automorphism groups (note that we have less than $p + 1$ orbits if some of the l_i are 0)

Now we evaluate $\mathbf{s}(G; r)$ on some integer m (and denote this by $\mathbf{s}(G; r)(m)$). In order to do it, pick \vec{b} which has ones on the first m coordinates, i.e $\vec{b} = 1^m \cdot 0^{n-m}$. Hence,

$$\mathbf{s}(G; r)(m) = \mathbf{s}(G; r)(\vec{b}) = \sum_{i=1}^k \mathbf{b}(\pi_i(G); r)(\vec{b})$$

where, as before, π_1, \dots, π_k is a transversal of $\text{Sym}(V)/\text{Aut}(G)$. Each summand $\mathbf{b}(\pi_i(G); r)(\vec{b})$ evaluates to 1 or 0, depending on the mapping π_i . Being more precise, if π_i maps s_0 elements of L_0 to $[1..m]$, \dots , and s_{p-1} elements of L_{p-1} to $[1..m]$, then $\pi_i(G)(\vec{b})$ evaluates to

$$\sum_{j=1}^d t_j \cdot \binom{s_C(m)}{j} + \sum_{j=1}^{q-1} j \cdot s_j \pmod{p}. \quad (6)$$

where $s_C(m)$ denotes the number of elements of C mapped to $[1..m]$, which is computed according to formula $s_C(m) = m - (s_0 + \dots + s_{p-1})$. Recall that $\mathbf{b}(\pi_i(G); r)(\vec{b}) = 1$ if and only if $\pi_i(G)(\vec{b}) = r$.

Let $\chi[G; r](m)$ denote the set of $\vec{s} = (s_0, \dots, s_{p-1})$ that make the sum (6) evaluate to r . Observe that we have natural inequalities $0 \leq s_i \leq l_i$ for all $i \in \{0, \dots, p-1\}$ and $0 \leq s_C(m) \leq l_C$. Hence, the feasible $\vec{s} \in \mathbb{N}^p$ can be described by

$$\vec{s} \in \chi[G; r](m) \iff \begin{cases} \sum_{j=1}^d t_j \cdot \binom{s_C(m)}{j} + \sum_{j=1}^{p-1} j \cdot s_j \pmod{p} = r \\ 0 \leq s_i \leq l_i \text{ for } i \in \{0, \dots, p-1\} \\ 0 \leq s_C(m) \leq l_C. \end{cases} \quad (7)$$

Moreover, let $\#[\vec{s}](m)$ denote the number of permutations in $\{\pi_1, \dots, \pi_k\}$ that map $s_C(m)$ elements of C to $[m]$ and s_i elements of L_i to $[m]$ (for $i = 0, \dots, p-1$). Hence, we have

$$\mathbf{s}[G; r](m) = \left(\sum_{(\vec{s}) \in \chi[G; r](m)} \#[\vec{s}](m) \right) \pmod{q}.$$

Next, let us determine $\#[\vec{s}](m)$. Note that each permutation π_i in $\{\pi_1, \dots, \pi_k\}$ maps each of the sets L_0, \dots, L_{p-1}, C to some subsets $L'_0, \dots, L'_{p-1}, C' \subseteq [n]$ with $|L'_0| = |L_0|, \dots, |L'_{p-1}| = |L_{p-1}|$, and $|C'| = |C|$. Now, if two mappings π_i, π_j output the same image $\pi_i L_0 = \pi_j L_0, \dots, \pi_i L_{p-1} = \pi_j L_{p-1}, \pi_i C = \pi_j C$, then $\pi_i G = \pi_j G$ and hence π_i and

π_j must belong to the same coset of $\text{Aut}(G)$ in $\text{Sym}(V)$. Since $\{\pi_1, \dots, \pi_k\}$ were chosen to be a transversal of $\text{Sym}(V)/\text{Aut}(G)$, this is only possible when $\pi_i = \pi_j$. So, the mapping $\pi_i \mapsto (L'_0, \dots, L'_{p-1}, C')$ is injective.

In fact, the mapping is also surjective as for any particular $(L'_0, \dots, L'_{p-1}, C')$ with $|L'_0| = |L_0|, \dots, |L'_{p-1}| = |L_{p-1}|, |C'| = |C|$ we can find some $\pi \in \{\pi_1, \dots, \pi_k\}$ which maps $\pi(L_i) = L'_i$ and $\pi(C) = C'$. Indeed, just pick any $\sigma \in \text{Sym}(V)$ satisfying $\sigma(L_i) = L'_i$ for all i and $\sigma(C) = C'$ and take its representative in the equivalence class modulo $\text{Aut}(G)$ as π_i . So we have a bijective mapping between permutations $(\pi_i)_{i=1..p-1}$ and ordered partitions $(L'_0, \dots, L'_{p-1}, C')$ of $[n]$ satisfying $|L'_0| = |L_0|, \dots, |L'_{p-1}| = |L_{p-1}|$, and $|C'| = |C|$.

Thus, if we want to count $\#(\bar{s})$, we need to count the number of proper partitions satisfying $|L'_1 \cap [1..m]| = s_1, \dots, |L'_{p-1} \cap [1..m]| = s_{p-1}$, and $|C' \cap [1..m]| = s_C(m)$. In other words, we partition an m -element set into disjoint subsets of sizes $s_0, s_1, \dots, s_{p-1}, s_C(m)$ and an $n - m$ -element set into disjoint subsets of sizes $l_0 - s_0, \dots, l_{p-1} - s_{p-1}, l_C - s_C(m)$; this leads to the following formula

$$\#[\bar{s}](m) = \binom{m}{s_0, \dots, s_{p-1}, s_C(m)} \cdot \binom{n-m}{l_0 - s_0, \dots, l_{p-1} - s_{p-1}, l_C - s_C(m)}. \quad (8)$$

Now, when fixing s_0, \dots, s_{p-1} , we will use Lemma 15 to show that the function $m \mapsto \#[\bar{s}](m) \pmod q$ is periodic with period q^{k_q} in the interval $\{0, \dots, n\}$ where k_q is the smallest integer such that $q^{k_q} > l_0 + \dots + l_{p-1} = n - |C|$. The periodicity is immediate for the first element of the product by Lemma 15. Let $s'_i = l_0 - s_0$. One can see that the values of the second element of the product produce, in the interval $[0..n]$, a reversed sequence compared to the one produced by

$$\binom{m}{s'_0, \dots, s'_{p-1}, m - (s'_0 + \dots + s'_{p-1})}$$

Indeed, $l_C - s_C(m) = n - (l_0 + \dots + l_{p-1}) - (m - (s_0 + \dots + s_{p-1})) = (n - m) - (s'_0 + \dots + s'_{p-1})$ and $n - m$ plays the role of m in the reversed sequence. As periodicity of any given sequence on the interval $[0..n]$ is preserved under reversing, and as $q^{k_q} > l_0 + \dots + l_{p-1} \geq s'_0 + \dots + s'_{p-1}$, we get the desired periodicity of $\#[\bar{s}](m)$ (as the period transfers to the product).

Now, one could argue that then the sum

$$s[G; r](m) = \left(\sum_{(\bar{s}) \in \chi[G; r](m)} \#[\bar{s}](m) \right) \pmod q$$

must be periodic, as it is just a sum of elements that are periodic. Unfortunately, $\chi[G; r](m)$ selects elements of the sum depending on m (i.e. the s_i depend on m). We need to address this issue.

Note that in (7) we can drop the condition $0 \leq s_C(m) \leq l_C$ because whenever $s_C(m) < 0$ or $s_C(m) > l_C$ the formula (8) returns value 0 anyway (as multinomial coefficient takes value 0 whenever $s_0 + \dots + s_{p-1} > m$ or $s'_0 + \dots + s'_{p-1} > m$). So we can effectively get rid of $s_C(m)$ in χ to get an updated definition

$$\bar{s} \in \chi'[G; r](m) \iff \begin{cases} \sum_{j=1}^d t_j \cdot \binom{s_C(m)}{j} + \sum_{j=1}^{q-1} j \cdot s_j \pmod p = r \\ 0 \leq s_i \leq l_i \text{ for } i \in \{0, \dots, p-1\} \end{cases} \quad (9)$$

and maintain the value of the sum, i.e.

$$\left(\sum_{(\bar{s}) \in \chi'[G; r](m)} \#[\bar{s}](m) \right) = \left(\sum_{(\bar{s}) \in \chi[G; r](m)} \#[\bar{s}](m) \right).$$

58:16 Violating Constant Degree Hypothesis Requires Breaking Symmetry

Let K be the smallest integer satisfying $p^K > \max(l_0 + l_1 + \dots + l_{p-1}, d)$. We further modify the definition of χ' to create χ^* in the following way

$$\bar{s} \in \chi^*[G; r](m) \iff \begin{cases} \sum_{j=1}^d t_j \cdot \binom{s_C(m) + p^K}{j} + \sum_{j=1}^{q-1} j \cdot s_j \pmod{p} = r \\ 0 \leq s_i \leq l_i \text{ for } i \in \{0, \dots, p-1\} \end{cases} \quad (10)$$

We claim that for all m

$$\left(\sum_{(\bar{s}) \in \chi'[G; r](m)} \#[\bar{s}](m) \right) \pmod{q} = \left(\sum_{(\bar{s}) \in \chi^*[G; r](m)} \#[\bar{s}](m) \right) \pmod{q}$$

There are 2 cases we need to consider.

1. When some fixed \bar{s} belongs to both $\chi'[G; r](m)$ and $\chi^*[G; r](m)$, then $\#[\bar{s}](m)$ cancels out from both sides of the equation. Similarly if \bar{s} does not belong to either of the sets, we do not have $\#[\bar{s}](m)$ on either of sides of the equation.
2. If $\bar{s} \in \chi'[G; r](m)$ and $\bar{s} \notin \chi^*[G; r](m)$ or $\bar{s} \notin \chi'[G; r](m)$ and $\bar{s} \in \chi^*[G; r](m)$, there must be some j such that $\binom{s_C(m)}{j} \neq \binom{s_C(m) + p^K}{j}$. This can only be the case if $s_C(m)$ is negative: otherwise, because $p^K > d \geq j$, from the periodicity of function $a \mapsto \binom{a}{j} \pmod{p}$ (for natural numbers $a \geq 0$), we would get that $\binom{s_C(m)}{j} = \binom{s_C(m) + p^K}{j} \pmod{p}$ and, hence, the conditions for $\chi'[G; r](m)$ and $\chi^*[G; r](m)$ would be identical from the perspective of \bar{s} . But when $s_C(m) < 0$, then $\#[\bar{s}](m)$ is zero due to definition of multinomial coefficient, so it does not contribute to any of the sides anyway.

So we obtain that

$$\mathbf{s}(G; r)(m) = \left(\sum_{(\bar{s}) \in \chi^*[G; r](m)} \#[\bar{s}](m) \right) \pmod{q}.$$

But now the formula (10) gives ranges $0 \leq s_j \leq l_j$ for $j \in \{0, \dots, p-1\}$; thus, we conclude that $s_C(m) + p^K$ is always positive (since $s_C(m) + p^K = m + p^K - (s_0 + \dots + s_{p-1}) \geq m \geq 0$). So $m \mapsto \binom{s_C(m) + p^K}{j} \pmod{p}$ is periodic with period $p^{k_p} > d$ where k_p is the smallest integer with $p^{k_p} > d$. Looking at the definition of χ^* we conclude:

► **Fact 18.** *Let $\bar{s} \in \mathbb{N}^p$. For $m \geq 0$ we have*

$$\bar{s} \in \chi^*[G; r](m) \iff \bar{s} \in \chi^*[G; r](m + p^{k_p})$$

Hence, the condition $\bar{s} \in \chi^*(G; r)$ depends not really on m , but on the remainder of m modulo p^{k_p} . So now, for all integers $j \in \{0, \dots, p^{k_p} - 1\}$ we define $\chi_j^*[G; r]$ as $\chi^*[G; r](j)$ in order to obtain that $\chi^*[G; r](m) = \chi_j^*[G; r]$ for $j = m \pmod{p^{k_p}}$. Now we can see that $\mathbf{s}(G; r)(m)$ is periodic with period $p^{k_p} \cdot q^{k_q}$:

$$\begin{aligned} \mathbf{s}(G; r)(m + p^{k_p} \cdot q^{k_q}) &= \sum_{(\bar{s}) \in \chi_j^*[G; r]} \#[\bar{s}](m + p^{k_p} \cdot q^{k_q}) \\ &= \sum_{(\bar{s}) \in \chi_j^*[G; r]} \#[\bar{s}](m) \\ &= \mathbf{s}(G; r)(m) \pmod{q}. \end{aligned}$$

The first and the third equality comes from periodicity of the condition χ^* and the fact that m and $m + p^{k_p} \cdot q^{k_q}$ give the same rest modulo p^{k_p} and the second one comes from the equality of rests modulo q^{k_q} and periodicity of $\#(s)(m)$. ◀

4.4 Main Theorems

Now we have all the necessary components to prove our main theorems.

Proof of Theorem 1. As discussed in Section 3, any $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit has a corresponding symmetric $\Sigma_q \circ \text{MOD}_p \circ \text{AND}_d$ expression \mathbf{f} with no periods smaller than the circuit we started with (note that there can happen some blow-up in size, but this does not matter as we argue below). By Fact 7, \mathbf{f} can be written as a sum of expressions of the form $\mathbf{s}(G; r)$. Hence, from now on let us consider one of these expressions $\mathbf{s}(G; r)$.

We choose ε such that $2s = 2^{\varepsilon \cdot n}$ (meaning that $\varepsilon \cdot n = \log s + 1$ and $\varepsilon < 1/8$). If G does not contain a fully symmetric set $|C|$ of size at least $n - \lfloor \varepsilon n \rfloor$, by Lemma 10, it satisfies $|\text{Sym}([n]) / \text{Aut}(G)| \geq 2^{\lfloor \varepsilon n \rfloor}$. Thus, writing $\mathbf{s}(G; r) = \sum_{i=0}^k \mathbf{b}^{\pi_i}(G, r)$ as in Equation (4), it follows that $k \geq 2^{\lfloor \varepsilon n \rfloor}$. As all the different terms in this sum get their inputs from different graphs $\pi_i(G)$, also for each term in the sum there must have been a different gate in the original circuit we started with. This is a contradiction as $2^{\lfloor \varepsilon n \rfloor} > s$.

Hence, all the subexpressions $\mathbf{s}(G; r)$ contain a fully symmetric set C of size at least $n - \lfloor \varepsilon n \rfloor$. Now, Lemma 14 tells us that we can write \mathbf{f} as a sum of expressions of the form $\mathbf{s}(G; r)$ where G is symmetry-purified with respect to C . Then, Theorem 16 implies that each such $\mathbf{s}(G; r)$ has a period $p^{k_p} \cdot q^{k_q}$, where k_p is the smallest integer such that $p^{k_p} > d$ and k_q is the smallest integer such that $q^{k_q} > n - |C| = \lfloor \varepsilon n \rfloor$. As \mathbf{f} is a sum of different $\mathbf{s}(G; r)$, which all share the period $p^{k_p} \cdot q^{k_q}$, it itself has period $p^{k_p} \cdot q^{k_q}$. ◀

The following result shows that the estimate of size which can be derived from Theorem 1 is asymptotically (almost) precise.

▶ **Proposition 19.** *Let $p \neq q$ be primes. Let k_p, k_q be natural numbers. For every symmetric function f with a period $p^{k_p} \cdot q^{k_q}$ there is a symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit of size $O(d \cdot \binom{n}{d} + d \cdot l^2 \cdot \binom{n}{l})$ which computes f , where $d = p^{k_p} - 1$ and $l = q^{k_q} - 1$ and $d, l < \frac{n}{2}$.*

The proof, which is a rather straightforward application of known facts, can be found in the full version on arXiv.

▶ **Corollary 20.** *Let $p \neq q$ be primes and $d : \mathbb{N} \rightarrow \mathbb{N}$ with $d(n) \leq n/2$ for all n . A function $f = (f_n)_{n \in \mathbb{N}}$ (with $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$) can be computed by a family of symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_{d(n)}$ circuits of quasipolynomial size if and only if, for each n , f_n has a period $p^{k_p(n)} q^{k_q(n)} \in \log^{O(1)}(n)$ for some functions $k_p, k_q : \mathbb{N} \rightarrow \mathbb{N}$.*

Moreover, if $d = p^{k_p} - 1$ is a constant, then f can be computed by symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits of quasipolynomial size if and only if, for each n , f_n has a period $p^{k_p} q^{k_q(n)} \in \log^{O(1)}(n)$ for some function $k_q : \mathbb{N} \rightarrow \mathbb{N}$.

Proof. If f_n has period $p^{k_p(n)} q^{k_q(n)} \in \log^{O(1)}(n)$, by Proposition 19, f_n is computed by a $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit of size $O(d \cdot \binom{n}{d} + d \cdot l^2 \cdot \binom{n}{l})$ where $l = q^{k_q}$. As $\binom{n}{\log^{O(1)}(n)} \subseteq 2^{\log^{O(1)}(n)}$, it follows that f can be computed by a family of quasipolynomial-size $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_{d(n)}$ circuits. In the case that $d \in \{p^k - 1 \mid k \in \mathbb{N}\}$ is a constant, Proposition 19 still can be applied with the same outcome.

On the other hand, if f is computed by a family of quasipolynomial-size $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_{d(n)}$ circuits, we first observe that without loss of generality $d(n) \in \log^{O(1)}(n)$. Indeed, if for some n the circuits use an AND_d gate for some $d \geq \log^k(n)$, then the size of the circuit, because of the symmetry property, is at least $\binom{n}{d} \geq \left(\frac{n}{d}\right)^d \geq 2^d \geq 2^{\log^k(n)}$. If this holds for every $k \in \mathbb{N}$ and infinitely many n , then the circuit family is not of quasipolynomially bounded size.

58:18 Violating Constant Degree Hypothesis Requires Breaking Symmetry

Now, it remains to apply Theorem 1, which tells us that f_n has period $p^{k_p} q^{k_q}$ where k_p is the smallest integer with $p^{k_p} > d(n)$ and k_q is the smallest integer with $q^{k_q} > \log s(n) + 1$ where $s(n) \in 2^{\log^{O(1)}(n)}$ is the size of the n -input circuit. As $p^{k_p} q^{k_q} \in \log^{O(1)}(n)$, both parts of the corollary follow (for the second part, observe that p^{k_p} is the smallest p -power greater than d). ◀

Instead of directly proving Theorem 3, let us derive the following slightly more explicit and general variant of the theorem:

► **Theorem 21.** *Let $p \neq q$ be primes and let $n \geq \max\{13, 4p^2 q^2\}$ and $d \leq n - \sqrt{n}$. Then every symmetric $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit computing the AND_n function has size at least $2^{\max\{n/(2dpq), \sqrt{n}\}}$.*

Proof. Let us write $V = \{x_1, \dots, x_n\}$. First consider the case that $d \geq \sqrt{n}$ and there is actually an AND_k gate v with $n - \sqrt{n} \geq k \geq \sqrt{n}$ inputs. Since for any $\pi \in \text{Sym}(V)$ also $\pi(v)$ must be a gate in the circuit, we obtain different AND_k gates for each k -element subset of V . As there are $\binom{n}{k} \geq \max\{(n/k)^k, (n/(n-k))^{n-k}\} \geq 2^{\sqrt{n}} \geq 2^{n/d}$ many such subsets, the theorem holds in this case (almost trivially).

Therefore, in the following, we assume $d < \sqrt{n} \leq n/(2pq)$ and consider an arbitrary n -input $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuit \mathcal{C} of size $s \leq 2^{n/(2pqd)}$. By Theorem 1, the function computed by \mathcal{C} has period $p^{k_p} q^{k_q}$ where k_p is the smallest integer with $p^{k_p} > d$ and k_q is the smallest integer with $q^{k_q} > \log s + 1 \geq n/(2pqd) + 1$. Notice that $p^{k_p} \leq d \cdot p$ and $q^{k_q} \leq (n/(2pqd) + 1) \cdot q$ and, hence, we have

$$p^{k_p} \cdot q^{k_q} \leq d \cdot p \cdot (n/(2pqd) + 1) \cdot q = n/2 + dpq < n.$$

Thus, \mathcal{C} does not compute the AND_n function as AND_n does not have any non-trivial period. ◀

5 Further Perspectives

Arguably one of the strongest applications of the Degree Decreasing Lemma is Theorem 4 in [22]. It implies that, if all the polynomials over \mathbb{F}_p that compose the top levels of a $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ -circuit can be written with a sublinear number of (binary) multiplications, then the circuit can be replaced with a $\text{MOD}_q \circ \text{MOD}_p$ circuit with only a subexponential blow-up in size. We argue that this kind of theorem cannot be applied in the context of our proof.

Note that a large pseudo-clique in the symmetry-purified expressions are (arbitrary) symmetric polynomials. Most of symmetric polynomials over \mathbb{F}_p require at least a linear number of multiplications in any formula (circuit) defining them. To see it, consider the example $p(\bar{x}) = \sum_{i < j} x_i \cdot x_j$ as a polynomial over \mathbb{F}_2 . One can easily check that it represents a function with smallest period 4. But now, if it could be written with a sub-linear number of multiplications, by Theorem 4 in [22], it could be represented by a sub-exponential size $\text{MOD}_3 \circ \text{MOD}_2$ circuit. However, this contradicts [23, Theorem 2.4] as subexponential size $\text{MOD}_3 \circ \text{MOD}_2$ circuits can only represent periodic functions with period of the form $2 \cdot 3^k$. This shows that that the Degree Decreasing Lemma cannot be used in this context, as it puts the limitations on its own applicability, by providing arithmetic circuit lower bounds. Such lower bounds can be proved for all non-trivial symmetric polynomials over \mathbb{F}_p with $d \geq p$ using a similar period analysis. Thus, our symmetry purification technique as well as combinatorial analysis contained in the proof of Theorem 16 constitute a substantial improvement over the Degree Decreasing Lemma and its accompanying techniques.

The results of the present paper indicate what type of symmetric functions might be computable by small, but not necessarily symmetric, $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits. It is natural to believe that the optimal (or nearly optimal) representation of the symmetric function should also be symmetric. Thus, we state the following

► **Conjecture 22.** *For fixed d, p, q , the only symmetric functions that can be represented by $\text{MOD}_q \circ \text{MOD}_p \circ \text{AND}_d$ circuits of subexponential size have to be periodic with some period of the form $p^{k_p} \cdot q^{k_q}$, for k_p being the smallest integer with $p^{k_p} \geq d$ and k_q be such that $p^{k_p} \cdot q^{k_q} \leq n$.*

References

- 1 Miklós Ajtai. Σ_1^1 -formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.
- 2 László Babai. Primitive coherent configurations and the order of uniprimitive permutation groups, 2018. URL: <https://people.cs.uchicago.edu/~laci/papers/uni-update.pdf>.
- 3 David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4:367–382, 1994. doi:10.1007/BF01263424.
- 4 David A. Mix Barrington and Howard Straubing. Complex polynomials and circuit lower bounds for modular counting. *Computational Complexity*, 4:325–338, 1994. doi:10.1007/BF01263421.
- 5 David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990. doi:10.1016/0890-5401(90)90007-5.
- 6 Abhishek Bhrushundi, Kaave Hosseini, Shachar Lovett, and Sankeerth Rao. Torus polynomials: An algebraic approach to ACC lower bounds. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, volume 124 of *LIPICs*, pages 13:1–13:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ITCS.2019.13.
- 7 Alfred Bochert. Ueber die Zahl der verschiedenen Werthe, die eine Function gegebener Buchstaben durch Vertauschung derselben erlangen kann. *Mathematische Annalen*, 33(4):584–590, 1889. doi:10.1007/BF01444035.
- 8 Joshua Brakensiek, Sivakanth Gopi, and Venkatesan Guruswami. Constraint satisfaction problems with global modular constraints: Algorithms and hardness via polynomial representations. *SIAM Journal on Computing*, 51(3):577–626, 2022. doi:10.1137/19m1291054.
- 9 Bettina Brustmann and Ingo Wegener. The complexity of symmetric functions in bounded-depth circuits. *Information Processing Letters*, 25(4):217–219, 1987. doi:10.1016/0020-0190(87)90163-3.
- 10 Peter J. Cameron. *Permutation Groups*. London Mathematical Society Student Texts. Cambridge University Press, 1999. doi:10.1017/CB09780511623677.
- 11 Brynmor Chapman and Ryan Williams. Smaller ACC^0 circuits for symmetric functions. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPICs*, pages 38:1–38:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.38.
- 12 Arkadev Chattopadhyay, Navin Goyal, Pavel Pudlak, and Denis Thérien. Lower bounds for circuits with MOD_m gates. In *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006*, pages 709–718, 2006. doi:10.1109/FOCS.2006.46.
- 13 Anuj Dawar and Gregory Wilsenach. Symmetric arithmetic circuits. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPICs*, pages 36:1–36:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.36.

- 14 Larry Denenberg, Yuri Gurevich, and Saharon Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control*, 70(2/3):216–240, 1986. doi:10.1016/S0019-9958(86)80006-7.
- 15 Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM Journal on Computing*, 40(4):1154–1178, 2011. doi:10.1137/100804322.
- 16 Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 577–584. ACM, 2015. doi:10.1145/2746539.2746546.
- 17 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012. doi:10.1137/090772721.
- 18 Ronald Fagin, Maria M. Klawe, Nicholas Pippenger, and Larry J. Stockmeyer. Bounded-depth, polynomial-size circuits for symmetric functions. *Theoretical Computer Science*, 36:239–250, 1985. doi:10.1016/0304-3975(85)90045-3.
- 19 Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17:13–27, 1984. doi:10.1007/BF01744431.
- 20 Parikshit Gopalan. Constructing Ramsey graphs from Boolean function representations. *Combinatorica*, 34:173–206, 2014. doi:10.1007/s00493-014-2367-1.
- 21 Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit Ramsey graphs. *Combinatorica*, 20(1):71–86, 2000. doi:10.1007/s004930070032.
- 22 Vince Grolmusz. A degree-decreasing lemma for $(\text{MOD}_p\text{-MOD}_m)$ circuits. *Discrete Mathematics and Theoretical Computer Science*, 4(2):247–254, 2001. doi:10.46298/dmtcs.289.
- 23 Vince Grolmusz and Gábor Tardos. Lower bounds for $(\text{MOD}_p\text{-MOD}_m)$ circuits. *SIAM Journal on Computing*, 29(4):1209–1222, 2000. doi:10.1137/S0097539798340850.
- 24 Kristoffer Arnsfelt Hansen. Computing symmetric boolean functions by circuits with few exact threshold gates. In *Computing and Combinatorics, 13th Annual International Conference, COCOON 2007, Proceedings*, volume 4598 of *Lecture Notes in Computer Science*, pages 448–458. Springer, 2007. doi:10.1007/978-3-540-73545-8_44.
- 25 Kristoffer Arnsfelt Hansen and Michal Koucký. A new characterization of ACC^0 and probabilistic CC^0 . *Computational Complexity*, 19(2):211–234, 2010. doi:10.1007/s00037-010-0287-z.
- 26 Johan Håstad. *Computational limitations for small depth circuits*. PhD thesis, Massachusetts Institute of Technology, 1986.
- 27 William He and Benjamin Rossman. Symmetric formulas for products of permutations. In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023*, volume 251 of *LIPICs*, pages 68:1–68:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ITCS.2023.68.
- 28 Paweł M. Idziak, Piotr Kawalek, and Jacek Krzaczkowski. Intermediate problems in modular circuits satisfiability. In *Proceedings of 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020*, pages 578–590, 2020. doi:10.1145/3373718.3394780.
- 29 Paweł M. Idziak, Piotr Kawalek, and Jacek Krzaczkowski. Complexity of modular circuits. In *Proceedings of 37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2022*, pages 32:1–32:11, 2022. doi:10.1145/3531130.3533350.
- 30 Paweł M. Idziak, Piotr Kawalek, Jacek Krzaczkowski, and Armin Weiß. Satisfiability Problems for Finite Groups. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, volume 229 of *LIPICs*, pages 127:1–127:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.127.
- 31 Paweł M. Idziak and Jacek Krzaczkowski. Satisfiability in multivalued circuits. *SIAM Journal on Computing*, 51(3):337–378, 2022. doi:10.1137/18m1220194.
- 32 Piotr Kawalek and Armin Weiß. Violating constant degree hypothesis requires breaking symmetry. *CoRR*, abs/2311.17440, 2023. doi:10.48550/arXiv.2311.17440.
- 33 Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 699–708, 2018. doi:10.1145/3188745.3188936.

- 34 Chi-Jen Lu. An exact characterization of symmetric functions in $\text{qAC}^0[2]$. *Theoretical Computer Science*, 261(2):297–303, 2001. doi:10.1016/S0304-3975(00)00145-6.
- 35 Cheryl E. Praeger and Jan Saxl. On the orders of primitive permutation groups. *The Bulletin of the London Mathematical Society*, 12(4):303–307, 1980. doi:10.1112/blms/12.4.303.
- 36 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC 1987*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.
- 37 Howard Straubing and Denis Thérien. A note on MOD_p - MOD_m circuits. *Theory of Computing Systems*, 39(5):699–706, 2006.
- 38 Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, FOCS 1985*, pages 1–10. IEEE Computer Society, 1985. doi:10.1109/SFCS.1985.49.
- 39 Zhi-Li Zhang, David A. Mix Barrington, and Jun Tarui. Computing symmetric functions with AND/OR circuits and a single MAJORITY gate. In *Proceedings of 10th Annual Symposium on Theoretical Aspects of Computer Science, STACS 1993*, volume 665 of *Lecture Notes in Computer Science*, pages 535–544. Springer, 1993. doi:10.1007/3-540-56503-5_53.

Online Matching with Delays and Size-Based Costs

Yasushi Kawase   

The University of Tokyo, Japan

Tomohiro Nakayoshi  

The University of Tokyo, Japan

Abstract

In this paper, we introduce the problem of *Online Matching with Delays and Size-based Costs (OMDSC)*. The OMDSC problem involves m requests arriving online. At any time, a group can be formed by matching any number of requests that have been received but remain unmatched. The cost associated with each group is determined by the waiting time for each request within the group and size-dependent cost. The size-dependent cost is specified by a penalty function. Our goal is to partition all the incoming requests into multiple groups while minimizing the total associated cost. This problem is an extension of the TCP acknowledgment problem proposed by Dooly et al. (J. ACM, 2001). It generalizes the cost model for sending acknowledgments. This study reveals the competitive ratios for a fundamental case, in which the penalty function takes only values of either 0 or 1. We classify such penalty functions into three distinct cases: (i) a fixed penalty of 1 regardless of the group size, (ii) a penalty of 0 if and only if the group size is a multiple of a specific integer k , and (iii) other situations. The problem in case (i) is equivalent to the TCP acknowledgment problem, for which Dooly et al. proposed a 2-competitive algorithm. For case (ii), we first show that natural algorithms that match all remaining requests are $\Omega(\sqrt{k})$ -competitive. We then propose an $O(\log k / \log \log k)$ -competitive deterministic algorithm by carefully managing the match size and timing, and prove its optimality. For any penalty function in case (iii), we demonstrate the non-existence of a competitive online algorithm. Additionally, we discuss competitive ratios for other typical penalty functions that are not restricted to take values of 0 or 1.

2012 ACM Subject Classification Theory of computation \rightarrow Online algorithms

Keywords and phrases Online matching, competitive analysis, delayed service

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.59

Related Version *Full Version*: <https://arxiv.org/abs/2408.08658> [22]

Funding *Yasushi Kawase*: JST ERATO Grant Number JPMJER2301, JST PRESTO Grant Number JPMJPR2122, JSPS KAKENHI Grant Number JP20K19739, and Value Exchange Engineering, a joint research project between Mercari, Inc. and the RIISE.

Acknowledgements The authors thank anonymous reviewers for useful comments.

1 Introduction

In online gaming platforms, the challenge of matching players for an optimal gaming experience lies in balancing minimal waiting times and high match quality. For instance, consider an online gaming platform hosting a k -player game, such as chess ($k = 2$), Mahjong ($k = 4$), or battle royal-style shooting games (e.g., $k = 60$ for Apex Legends). On such platforms, players enter a waiting queue sequentially, and the platform selects k players from the queue to start a match. To address scenarios with an insufficient number of players, the platform may opt to fill in groups with computer (AI) players. Although this approach ensures prompt matchmaking, it potentially compromises the quality of the gaming experience. Conversely, waiting to gather the full quota of the required k players may significantly increase the waiting time, potentially reducing player satisfaction.



© Yasushi Kawase and Tomohiro Nakayoshi;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 59; pp. 59:1–59:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Motivated by the above challenge, we introduce the *Online Matching with Delays and Size-based Costs (OMDSC)* problem. In this problem, requests (corresponding to players) are presented sequentially in real time. At any moment, it is possible to match a group composed of any number of previously received but unmatched requests. The total cost associated with each match is defined by the waiting time for each request within the group and the cost that depends on the group size. The *penalty function* specifies the cost based on the size of the group.

The OMDSC problem has applications beyond gaming. For instance, in online shopping services, product orders arrive sequentially, and the products can be dispatched together at any time. While it is preferable to dispatch as many orders as possible to minimize costs, customers may become frustrated if their wait times are too long. This situation can also be modeled as an OMDSC problem, where the waiting time and the size of each dispatch jointly determine the total cost.

Notably, our problem is closely related to the TCP acknowledgment problem [14], the online weighted cardinality *Joint Replenishment Problem (JRP)* with delays [12], and the online *Min-cost Perfect Matching with Delays (MPMD)* [16]. The TCP acknowledgment problem (without look-ahead) is equivalent to the OMDSC problem with a constant penalty function. Various generalizations of the TCP acknowledgment problem partially capture the OMDSC problem. Specifically, the online weighted cardinality JRP with delays considers the costs dependent on the (weighted) cardinality of each group. In these generalizations, the penalty function is assumed to be monotonically nondecreasing. However, the OMDSC problem treats penalty functions as nonmonotonic. In the MPMD problem, requests arrive on a metric space, and matching is restricted to groups of exactly size 2. The online *Min-cost Perfect k -way Matching with Delays (k -MPMD)* [27] extends the group size to k . For further details, please refer to Section 1.2.

1.1 Our Results

In this study, we determine the competitive ratios of the OMDSC problem for fundamental and critical scenarios in which the penalty function takes only values of either 0 or 1. In the OMDSC problem, dividing a group into multiple smaller groups can reduce penalties. Therefore, it is sufficient to consider a modified penalty function obtained by optimally dividing a group into subgroups. With this modification, we classify penalty functions into three cases: (i) always 1, (ii) 0 if the size is a multiple of k , and (iii) all other scenarios.

In case (i), the OMDSC problem is equivalent to the TCP acknowledgment problem. Dooly et al. [14] proposed a 2-competitive online algorithm that matches all remaining requests when the total waiting cost increases by 1. For case (ii), we first examine natural algorithms that match all the remaining requests whenever a match incurs a positive cost, which we refer to as *match-all-remaining* algorithms. We demonstrate that every match-all-remaining algorithm is $\Omega(\sqrt{k})$ -competitive (Theorem 6). Consequently, to obtain an $o(\sqrt{k})$ -competitive algorithm, we must consider both the timing and size of the requests to be matched. By carefully managing remaining requests, we propose an $O(\log k / \log \log k)$ -competitive online algorithm (Theorem 8). We also prove that this competitive ratio is the best possible (Theorem 15). It is worth mentioning that the competitive ratio for case (ii) with $k = 1$ is trivial (Theorem 3) and that with $k = 2$ can be obtained from the results of Emek et al. [17]. For any penalty function in case (iii), we prove that no competitive online algorithm exists (Theorem 2). The results are summarized in Table 1. Furthermore, the competitive ratios for other typical penalty functions that are not restricted to take values of 0 or 1 are discussed in Section 2.3.

■ **Table 1** Competitive ratios for the OMDSC problem.

Penalty function (with modification)	Competitive ratio	Reference
(i) always 1	2	Dooly et al. [14]
(ii) 0 if the size is a multiple of k	$k = 1$	Theorem 3
	$k = 2$	Emek et al. [17]
	general k	$\Theta(\log k / \log \log k)$ Theorems 8 and 15
(iii) other scenarios	unbounded	Theorem 2

1.2 Related Work

An online version of the matching problem was first introduced by Karp et al. [21]. In their study, arriving requests are matched immediately upon arrival, primarily focusing on bipartite matching. Additionally, research has been conducted to minimize matching costs, both in settings where vertices have determined costs and in settings that consider distances in a metric space. Mehta et al. [26] proposed an application to AdWords. Other applications, such as food delivery, have also been considered. For further details, please refer to [15, 25].

In applications such as online game matchmaking and ride-sharing services, service providers can delay user matches. Emek et al. [16] modeled this scenario as an MPMD problem. In this setting, the requests can be retained by incurring waiting costs. Several online algorithms have been proposed for solving the MPMD problem [1, 3, 8]. Subsequently, Melnyk et al. [27] extended the MPMD problem to the k -MPMD problem, which requires exactly matching the size k . To address this problem, deterministic and randomized algorithms have been proposed [19, 27].

Dooly et al. [14] introduced the TCP acknowledgment problem that involves acknowledging TCP packets. In this problem, the aim is to minimize the sum of the acknowledgment costs and the costs of delaying TCP packets. Optimal deterministic and randomized algorithms were proposed by Dooly et al. [14] and Karlin et al. [20], respectively. The TCP acknowledgment problem is equivalent to the OMDSC problem, where the penalty function falls under case (i). Conversely, the OMDSC problem can be viewed as a generalization of the acknowledgment cost of the TCP acknowledgment problem.

Various extensions of the TCP acknowledgment problem have been proposed [2, 5–7, 9–12, 29]. In particular, Chen et al. [12] introduced the problem of an online weighted cardinality JRP with delays. Their model can handle penalties based on the size of the match. However, unlike in our study, their penalty function is restricted to monotonically nondecreasing. They proposed a constant-competitive algorithm to solve this problem.

The objectives of the MPMD and k -MPMD problems are to pair requests and form groups of exact size k , respectively. In contrast, the proposed OMDSC problem does not impose constraints on the number of requests in each group. Emek et al. [16] introduced a problem called MPMDfp, which allows the clearance of a request at a cost. The MPMDfp problem on a single source is equivalent to the OMDSC problem where the penalty function corresponds to case (ii) with $k = 2$. Additionally, the MPMDfp problem on a single source can be reduced to an MPMD problem using two sources [16]. For MPMD on two sources, Emek et al. [17] proposed a deterministic algorithm, and He et al. [18] proposed a randomized algorithm (see Section 4 for more details).

Another approach to extend the MPMD problem is to modify waiting costs. Liu et al. [23] generalized the waiting costs from linear to convex. Other variations of waiting costs have also been studied [4, 13, 24]. Pavone et al. [28] investigated *Online Hypergraph Matching with*

Deadlines problem. Although hypergraph matching does not impose constraints on group size, it differs from the OMDSC problem in that it employs deadlines instead of delays, and requests arrive at one at each unit of time.

2 Preliminaries

We denote the set of real numbers, nonnegative real numbers, integers, nonnegative integers, and positive integers by \mathbb{R} , \mathbb{R}_+ , \mathbb{Z} , \mathbb{Z}_+ , and \mathbb{Z}_{++} , respectively. In addition, for a positive integer $k \in \mathbb{Z}_{++}$, we define \mathbb{Z}_k as the set of integers from 0 to $k - 1$.

2.1 Problem Settings

In this section, we formally define the OMDSC problem. An instance of the problem is specified by a penalty function $f: \mathbb{Z}_{++} \rightarrow \mathbb{R}_+$ and m requests $V = \{v_1, v_2, \dots, v_m\}$ that arrive in real time. Each request v_i arrives at time t_i , where $0 \leq t_1 \leq t_2 \leq \dots \leq t_m$ is assumed.

An online algorithm initially knows only the function f ; it does not have prior knowledge of the arrival times or the total number of requests. Each request v_i is revealed at time t_i . At each time τ , the algorithm can match any subset S_j of requests that appeared by time τ and have not yet been matched. The cost of matching requests in S_j at time τ_j is defined as

$$f(|S_j|) + \sum_{i: v_i \in S_j} (\tau_j - t_i),$$

where the first term represents the *size cost* and the second term represents the *waiting cost*. In addition, the *waiting cost* at time τ is defined as $\sum_{i: v_i \in V'} (\tau - t_i)$, where V' is the set of unmatched but previously presented requests at that time.

The objective is to design an online algorithm that matches all requests while minimizing the total cost. We use the *competitive ratio* to evaluate the performance of the online algorithm. For a given instance σ , let $\text{ALG}(\sigma)$ represent the cost incurred by the online algorithm and let $\text{OPT}(\sigma)$ denote the optimal cost with prior knowledge of all requests in the instance. The competitive ratio of ALG for an instance σ is defined as $\text{ALG}(\sigma)/\text{OPT}(\sigma)$, interpreting $0/0$ as 1. In addition, the competitive ratio of ALG for a problem is defined as the supremum of the competitive ratios over all instances, that is, $\sup_{\sigma} \text{ALG}(\sigma)/\text{OPT}(\sigma)$. We call an online algorithm ρ -competitive if the competitive ratio of that algorithm is ρ .

The competitive ratio defined above is the *strict* competitive ratio. We can also define the *asymptotic* competitive ratio as $\limsup_{\text{ALG}(\sigma) \rightarrow \infty} \text{ALG}(\sigma)/\text{OPT}(\sigma)$. However, in our problem, the strict and asymptotic competitive ratios of the optimal algorithm coincide. Indeed, if an algorithm is strictly ρ -competitive, then it is also asymptotically at most ρ -competitive by definition. Moreover, if no strictly ρ -competitive algorithm exists, we can construct an instance for each algorithm in which the strict competitive ratio is at least ρ . Thus, by repeatedly constructing and providing such instances, we can conclude that no algorithm has an asymptotic competitive ratio better than ρ . Hence, we only consider the strict competitive ratio hereafter.

2.2 Binary Penalty Function

This study focuses primarily on penalty functions that take values only of either 0 or 1. We discuss other penalty functions in Section 2.3. For such a binary penalty function f , we may be able to match n requests with size cost 0, even if $f(n) = 1$, by appropriately partitioning the requests. For instance, if $f(2) = f(3) = 0$ and $f(7) = 1$, we can match 7 requests with a size cost of 0 by partitioning them into groups of sizes 2, 2, and 3. We introduce a notation for the set of numbers of requests that can be matched with a size cost of 0 as follows:

► **Definition 1.** For the penalty function $f: \mathbb{Z}_{++} \rightarrow \{0, 1\}$, we define the zero-penalty set B_f as a set of quantities that can be matched with a size cost of 0 by optimally dividing requests into subsets and matching them. Formally,

$$B_f := \left\{ n \in \mathbb{Z}_{++} \mid \exists s \in \mathbb{Z}_{++}, \exists n_1, \dots, n_s \in \mathbb{Z}_{++} \text{ s.t. } \sum_{i=1}^s n_i = n, f(n_1) = \dots = f(n_s) = 0 \right\}.$$

For example, if $f(1) = 0$, then $B_f = \mathbb{Z}_{++}$. Alternatively, if $f(1) = 1$ and $f(2) = f(3) = 0$, then $B_f = \mathbb{Z}_{++} \setminus \{1\}$. We can interpret the size cost of matching n requests as 0 if $n \in B_f$ and 1 if $n \notin B_f$.

We classify binary penalty functions into the following three types: (i) $B_f = \emptyset$, (ii) $B_f = \{kn \mid n \in \mathbb{Z}_{++}\}$ for some $k \in \mathbb{Z}_{++}$, and (iii) all other scenarios. Case (i) is the situation in which the penalty for matching requests is always 1 (i.e., $f(n) = 1$ for all $n \in \mathbb{Z}_{++}$). Case (ii) is a situation in which a set of requests can be matched without a size cost only if the size is a multiple of $k = \min\{n \in \mathbb{Z}_{++} \mid f(n) = 0\}$ (i.e., $f(k) = 0$ and $f(n) = 1$ for all $n \in \mathbb{Z}_{++} \setminus \{kn' \mid n' \in \mathbb{Z}_{++}\}$). This case is applicable in the context of an online gaming platform that hosts a k -players game, as described in Introduction.

2.3 Other Penalty Functions

If the range of the penalty function is $\{0, \mu\}$ with a positive real μ , it can be treated as a binary penalty function by scaling the increase rates of the waiting costs μ times slower. This is because the cost of matching requests in S_j at time τ_j can be expressed as $\mu \cdot (f(|S_j|)/\mu + \sum_{i: v_i \in S_j} (\tau_j/\mu - t_i/\mu))$. For example, if $\mu = 60$ and the unit time is one minute in the original problem, the range of the penalty function can be treated as $\{0, 1\}$ by adjusting the unit time to one hour. Let $\mu, \lambda \in \mathbb{R}$ with $0 < \mu \leq \lambda$ and consider a penalty function that takes values of 0 or within the range $[\mu, \lambda]$. By applying our ρ -competitive algorithm to the problem while treating positive penalties as if they were μ , we can obtain a $(\rho \cdot \lambda/\mu)$ -competitive algorithm. In addition, our impossibility result for case (iii) can be transferred in the same manner.

Another interesting penalty function is $f(n) = \lceil n/k \rceil$ for a specific integer k . This penalty function appears when a service can process up to k requests simultaneously, and each processing costs a fixed amount. For this penalty function, an algorithm similar to that in (i) is 2-competitive for any $k \geq 2$. We also prove that this competitive ratio is best possible for every $k \geq 2$ (see the full version [22]). For $k = 1$, the algorithm that matches each request upon its arrival is 1-competitive.

3 Zero-penalty Set is Nonempty and Non-representable as Multiples

In this section, we examine case (iii), where the penalty function f satisfies $B_f \neq \emptyset$ and $B_f \neq \{kn \mid n \in \mathbb{Z}_{++}\}$ for any $k \in \mathbb{Z}_{++}$. We demonstrate that, in this case, no algorithm has a finite competitive ratio.

To illustrate this intuitively, let us consider the case where $f(1) = 1$ and $f(2) = f(3) = 0$, representing a poker table for at least two players. Imagine two players arriving initially, followed potentially by a third. If we match the first two players immediately upon their arrival, matching the subsequent player incurs a cost. Alternatively, if we wait to match the first two, an unnecessary waiting cost is incurred when no additional player appears. We can construct similar instances for every penalty function in case (iii).

► **Theorem 2.** *For the OMDSC problem with a penalty function f that satisfies both $B_f \neq \emptyset$ and $B_f \neq \{kn \mid n \in \mathbb{Z}_{++}\}$ for every $k \in \mathbb{Z}_{++}$, the competitive ratio of any randomized algorithm is unbounded against an oblivious adversary.*

Proof. Let $k^* := \min B_f$, where such a value must exist by the assumption that $B_f \neq \emptyset$. Additionally, let $\ell := \min(B_f \setminus \{k^*n \mid n \in \mathbb{Z}_{++}\})$, where such a value must exist because $\{k^*n \mid n \in \mathbb{Z}_{++}\} \subseteq B_f$ and $B_f \neq \{k^*n \mid n \in \mathbb{Z}_{++}\}$. We fix an arbitrary online algorithm ALG and take a sufficiently small $\varepsilon > 0$. Consider an instance where k^* requests are given at time 0, and afterward, there may be arrivals of $\ell - k^*$ additional requests at time ε depending on the behavior of ALG. Suppose that ALG matches all the initial requests before ε with probability p .

If $p \geq 1/2$, consider an instance where $\ell - k^*$ additional requests are given at time ε . Since $\ell - k^* \notin B_f$, ALG incurs an expected size cost of at least $p (\geq 1/2)$. In contrast, the minimum total cost is $k^*\varepsilon$ by matching all $k^* + (\ell - k^*) = \ell$ requests at time ε . Thus, the competitive ratio is at least $p/(k^*\varepsilon) \geq 1/(2k^*\varepsilon)$.

Conversely, if $p < 1/2$, consider the instance where no additional requests are presented. Then, the (expected) waiting cost for ALG is at least $(1 - p)k^*\varepsilon > k^*\varepsilon/2$, while the offline optimal cost is 0 by matching all requests at time 0. Hence, the competitive ratio is unbounded.

Therefore, in both scenarios, the competitive ratio is unbounded as ε goes to 0, proving that the competitive ratio for any online algorithm is unbounded. ◀

4 Zero-penalty Set is Multiples of an Integer

In this section, we investigate the case (ii), where the penalty function f satisfies $B_f = \{kn \mid n \in \mathbb{Z}_{++}\}$ for some $k \in \mathbb{Z}_{++}$.

When $k = 1$ (i.e., $B_f = \mathbb{Z}_{++}$), we have $f(1) = 0$. Thus, the algorithm that immediately matches each request individually upon its arrival incurs a cost of 0 and is 1-competitive.

► **Theorem 3.** *For the OMDSC problem with a penalty function f such that $f(1) = 0$, there exists a 1-competitive deterministic online algorithm.*

For the case $k = 2$, the OMDSC problem is equivalent to the problem of MPMDfp in a metric space consisting of a single point. It is known that the MPMDfp problem can be reduced to the MPMD problem by doubling the number of points on the metric space [16]. By this reduction, the OMDSC problem with $k = 2$ can be reduced to the problem of MPMD on two sources by setting the distance between two sources to 2 and giving requests for two sources simultaneously when a request is given in the OMDSC problem. A matching across two sources in the MPMD problem corresponds to a matching of a single request in the OMDSC problem. The optimal cost of the reduced MPMD problem is twice that of the original OMDSC problem.

Emek et al. [17] provided a 3-competitive online algorithm for the online MPMD problem on two sources and demonstrated that this is best possible. In their instances constructed for the proof of the lower bound, requests are always given to two sources simultaneously, which can be reduced to the OMDSC problem with $k = 2$. Therefore, we obtain the following theorem.

► **Theorem 4** (Emek et al. [17]). *For the OMDSC problem with a penalty function f such that $B_f = \{2n \mid n \in \mathbb{Z}_{++}\}$, there exists a 3-competitive deterministic algorithm. Moreover, no deterministic online algorithm has a competitive ratio smaller than 3.*

He et al. [18] proposed a 2-competitive randomized algorithm against an oblivious adversary for the MPMD problem on two sources. Thus, this leads to a 2-competitive randomized algorithm for the OMDSC problem with $B_f = \{2n \mid n \in \mathbb{Z}_{++}\}$.

In the remainder of this section, we conduct an asymptotic analysis with respect to k .

Firstly, we define a type of algorithm as below and demonstrate the difficulty of case (ii) compared to case (i).

► **Definition 5.** We call an algorithm for the OMDSC problem match-all-remaining if, whenever it makes a match, it matches all remaining requests or the size is in B_f .

For case (i), where a penalty to match requests is always 1, a match-all-remaining algorithm achieves the optimal competitive ratio. However, for case (ii), we show that every match-all-remaining algorithm has a competitive ratio of $\Omega(\sqrt{k})$, where the proof can be found in the full version [22].

► **Theorem 6.** For the OMDSC problem with a penalty function f that satisfies $B_f = \{kn \mid n \in \mathbb{Z}_{++}\}$, every match-all-remaining algorithm has a competitive ratio of $\Omega(\sqrt{k})$.

In Section 4.1, we provide an $O(\log k / \log \log k)$ -competitive algorithm by utilizing partial matchings of remaining requests. Thus, no match-all-remaining algorithm is optimal. Subsequently, in Section 4.2, we establish the lower bound of $\Omega(\log k / \log \log k)$.

Before proceeding, we introduce some notations. Recall that $\mathbb{Z}_k = \{0, 1, \dots, k-1\}$.

► **Definition 7.** We write $\bar{x} \in \mathbb{Z}_k$ to represent the remainder of $x \in \mathbb{Z}$ when divided by k . Additionally, for $\ell, r \in \mathbb{Z}_+$, we define the cyclic interval

$$[[\ell, r]]_k := \begin{cases} \{x \in \mathbb{Z}_k \mid \bar{\ell} \leq x \leq \bar{r}\} = \{\bar{\ell}, \bar{\ell} + 1, \dots, \bar{r}\} & (\text{if } \bar{\ell} \leq \bar{r}), \\ \{x \in \mathbb{Z}_k \mid x \leq \bar{r} \text{ or } \bar{\ell} \leq x\} = \{\bar{\ell}, \bar{\ell} + 1, \dots, k-1, 0, 1, \dots, \bar{r}\} & (\text{if } \bar{\ell} > \bar{r}). \end{cases}$$

With this notation, $a \equiv b \pmod{k}$ can be expressed as $\bar{a} = \bar{b}$. Note that we have $|\llbracket \ell, r \rrbracket_k| = \bar{r} - \bar{\ell} + 1$ if $\bar{\ell} \leq \bar{r}$ and $|\llbracket \ell, r \rrbracket_k| = k + \bar{r} - \bar{\ell} + 1$ if $\bar{\ell} > \bar{r}$.

Let α be a real number such that $\alpha^\alpha = k$. Then, we have $\alpha = \Theta(\log k / \log \log k)$ because the ratio $\frac{\alpha}{\log k / \log \log k} = \frac{\alpha}{\log \alpha^\alpha / \log \log \alpha^\alpha} = \frac{\log \alpha + \log \log \alpha}{\log \alpha}$ approaches 1 as $k \rightarrow \infty$ (i.e., $\alpha \rightarrow \infty$). Thus, our task is to provide upper and lower bounds of $O(\alpha)$ and $\Omega(\alpha)$, respectively.

4.1 Upper Bound

The objective of this subsection is to prove the following theorem.

► **Theorem 8.** For the OMDSC problem with a penalty function f that satisfies $B_f = \{kn \mid n \in \mathbb{Z}_{++}\}$, there exists an $O(\log k / \log \log k)$ -competitive deterministic online algorithm.

According to Theorem 6, an $O(\alpha)$ ($= O(\log k / \log \log k)$)-competitive algorithm must consider both the timing and the size of requests to be matched. To address this requirement, we propose an algorithm that references the costs of other algorithms. The execution of the algorithm is divided into phases. In each phase, the goal is to ensure that any competing algorithm incurs a cost of at least 1, while our algorithm's cost remains at most $O(\alpha)$. We categorize competing algorithms based on the number of unmatched requests they carry over from the previous phase. As for the current phase, we assume that algorithms only perform matches of sizes that are multiples of k , since otherwise, their cost immediately reaches at least 1. Furthermore, it is sufficient to focus on “greedy” algorithms that immediately match k requests whenever at least k unmatched requests exist.

The first phase starts at the beginning. In each phase, our algorithm runs with the following $k+2$ variables.

► **Definition 9.** At each time t within each phase, the variables $W_0(t), W_1(t), \dots, W_{k-1}(t)$, $s(t)$, and $a(t)$ are defined as follows:

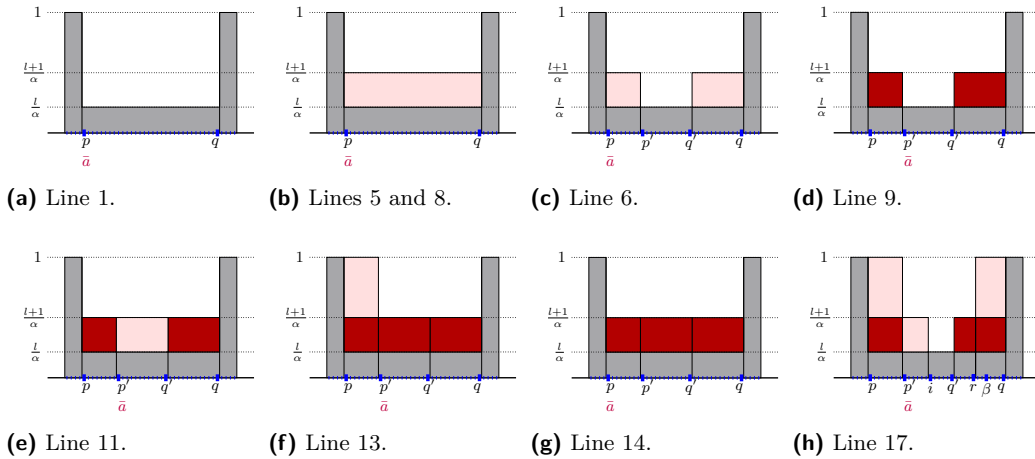
- $W_i(t)$ ($i \in \mathbb{Z}_k$): the waiting cost incurred by time t within the phase on the algorithm that greedily performs matches with no size cost, assuming $(k-i)$ unmatched requests are carried over to the current phase.
- $s(t)$: the number of requests given by time t in the phase.
- $a(t)$: the number of requests that our algorithm has matched up to, but not including, time t during the phase.

We will omit the argument t when there is no confusion.

Note that, if $(k-i)$ unmatched requests are carried over to the current phase, the optimum cost incurred thus far within the phase is at least $\min\{1, W_i\}$. At the end of the phase, our algorithm ensures $W_i \geq 1$ for all $i \in \mathbb{Z}_k$. This means that any algorithm incurs a cost of at least 1 per phase because matching a number of requests that is not a multiple of k results in a cost of 1.

During each phase, our algorithm recursively processes a subroutine **recurring** $(\llbracket p, q \rrbracket_k, l)$, which takes as parameters a cyclic interval $\llbracket p, q \rrbracket_k$ and an integer l . When the subroutine is called, it ensures that $W_i \geq 1$ for all $i \notin \llbracket p, q \rrbracket_k$ and $W_i \geq l/\alpha$ for all $i \in \llbracket p, q \rrbracket_k$. In each iteration of the subroutine, the interval size $|\llbracket p, q \rrbracket_k|$ is reduced by a factor of $\Theta(1/\alpha)$ or l is increased by one, with incurring cost at most $O(1)$. As a result, all W_i reach at least 1 after $O(\alpha)$ recursions by $\alpha^\alpha = k$.

At the beginning of each phase, the variables W_0, W_1, \dots, W_{k-1} , s , and a are 0 and **recurring** $(\llbracket 0, k-1 \rrbracket_k, 0)$ is called. Throughout the phase, the algorithm updates the values of W_0, W_1, \dots, W_{k-1} , s , and a , appropriately. Moreover, if at least k unmatched requests exist (i.e., $s - a \geq k$), it immediately matches k of them.



■ **Figure 1** Lower bounds of W_i at each line of Algorithm 1. Gray areas indicate lower bounds of W_i at the beginning of **recurring** $(\llbracket p, q \rrbracket_k, l)$. Red areas illustrate increments earlier than the last wait in **recurring** $(\llbracket p, q \rrbracket_k, l)$. Pink areas represent increments during the last wait.

We now describe the subroutine **recurring** $(\llbracket p, q \rrbracket_k, l)$. See Algorithm 1 for a formal description. Figure 1 illustrates lower bound of each W_i at each line.

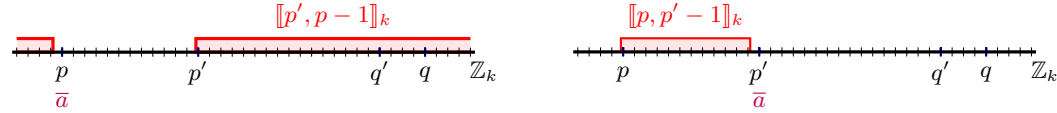
When **recurring** $(\llbracket p, q \rrbracket_k, l)$ is called, it is guaranteed that $p, q \in \mathbb{Z}_k$, $l \in \{0, 1, \dots, \lceil \alpha \rceil\}$, $W_i \geq 1$ for all $i \notin \llbracket p, q \rrbracket_k$, $W_i \geq l/\alpha$ for all $i \in \llbracket p, q \rrbracket_k$, and $\bar{a} = p$ (recall that $\bar{a} \in \mathbb{Z}_k$ is the remainder of a when divided by k , see also Figure 1a). The recursion ends (line 3) when $|\llbracket p, q \rrbracket_k|$ equals to 1 or when l is at least α . Then, it matches all remaining requests and proceeds to the next phase.

■ **Algorithm 1** The pseudo-code of $\text{recurring}(\llbracket p, q \rrbracket_k, l)$.

```

/*  $W_0, W_1, \dots, W_{k-1}, s, a$  are variables that change as defined in Definition 9 */
1 def recurring( $\llbracket p, q \rrbracket_k, l$ ):
  /* Ensure:  $p, q \in \mathbb{Z}_k, l \in \mathbb{Z}_+, W_i \geq 1 (\forall i \notin \llbracket p, q \rrbracket_k), W_i \geq \frac{l}{\alpha} (\forall i \in \llbracket p, q \rrbracket_k), \bar{a} = p$  */
  2 Wait until  $W_{\bar{a}} (= W_p)$  increases by 2; // Cost: 2
  3 if  $|\llbracket p, q \rrbracket_k| = 1$  or  $l \geq \alpha$  then
  4   Match all remaining requests and Proceed to the next phase; // Cost:  $\leq 1$ 
  5   if  $W_i \geq \frac{l+1}{\alpha}$  for all  $i \in \llbracket p, q \rrbracket_k$  then call recurring( $\llbracket p, q \rrbracket_k, l+1$ );
  6   Let  $\llbracket p', q' \rrbracket_k \subseteq \llbracket p, q \rrbracket_k$  be the smallest cyclic interval where  $W_i \geq \frac{l+1}{\alpha}$  for all
      $i \notin \llbracket p', q' \rrbracket_k$ ;
  7   Wait until  $\bar{s} \in \llbracket p', p-1 \rrbracket_k$  or  $W_{\bar{a}} (= W_p)$  increases by  $1/\alpha$ ; // Cost:  $\leq 1$ 
  8   if  $W_{\bar{a}} (= W_p)$  increased by  $1/\alpha$  at line 7 then call recurring( $\llbracket p, q \rrbracket_k, l+1$ );
  9   Match  $\overline{p'-p}$  requests; // Cost: 1
  10  Wait until  $W_{\bar{a}} (= W_{p'})$  increases by 2; // Cost: 2
  11  if  $W_i \geq \frac{l+1}{\alpha}$  for all  $i \in \llbracket p', q' \rrbracket_k$  then
  12    Wait until  $\bar{s} \in \llbracket p', p'-1 \rrbracket_k$  or  $W_{\bar{a}} (= W_{p'})$  increases by 1; // Cost:  $\leq 1$ 
  13    if  $W_{\bar{a}} (= W_{p'})$  increased by 1 at line 12 then call recurring( $\llbracket p', q' \rrbracket_k, l+1$ );
  14    Match  $\overline{p-p'}$  requests; // Cost:  $\leq 1$ 
  15    Call recurring( $\llbracket p, q \rrbracket_k, l+1$ );
  16  else
  17    Let  $\llbracket p', r \rrbracket_k \subseteq \llbracket p', q' \rrbracket_k$  be the smallest cyclic interval where  $W_i \geq 1$  for all
      $i \notin \llbracket p', r \rrbracket_k$ ;
  18    Call recurring( $\llbracket p', r \rrbracket_k, l$ );

```



(a) Situation at line 7.

(b) Situation at line 12.

■ **Figure 2** Relative positions of p, q, p', q' , and \bar{a} in Algorithm 1.

At the beginning of each recursion, the algorithm waits until $W_{\bar{a}}$ increases by 2 (line 2). Afterward, if $W_i \geq (l+1)/\alpha$ for all $i \in \llbracket p, q \rrbracket_k$, it calls $\text{recurring}(\llbracket p, q \rrbracket_k, l+1)$ (Figure 1b). Now, assume that $W_i < (l+1)/\alpha$ for some $i \in \llbracket p, q \rrbracket_k$. In this case, there exists a smaller interval $\llbracket p', q' \rrbracket_k (\subseteq \llbracket p, q \rrbracket_k)$ such that $W_i \geq (l+1)/\alpha$ for all $i \in \llbracket p, q \rrbracket_k \setminus \llbracket p', q' \rrbracket_k$ (Figure 1c). The algorithm picks the smallest such cyclic interval $\llbracket p', q' \rrbracket_k$ (line 6).

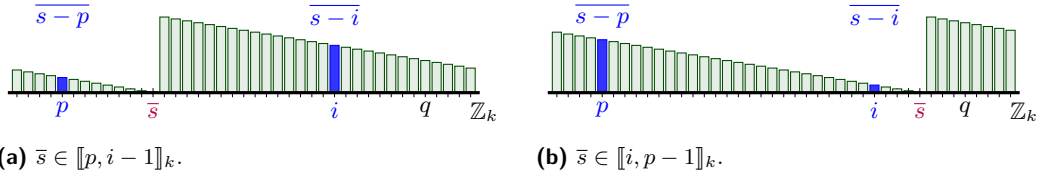
Next, it attempts to change \bar{a} from p to p' by matching $\overline{p'-p}$ requests. To achieve this, it waits to satisfy $\bar{s} \in \llbracket p', p-1 \rrbracket_k$ (see Figure 2a) until $W_{\bar{a}} (= W_p)$ increases by $1/\alpha$ (line 7). If $W_{\bar{a}}$ increases by $1/\alpha$, then $W_i \geq (l+1)/\alpha$ holds for every $i \in \llbracket p, q \rrbracket_k$ as we will show later (Figure 1b). Consequently, the algorithm calls $\text{recurring}(\llbracket p, q \rrbracket_k, l+1)$ (line 8). Now, suppose that \bar{s} in $\llbracket p', p-1 \rrbracket_k$ at some point. In that case, it changes \bar{a} from p to p' by matching $\overline{p'-p}$ requests (line 9 and see Figure 1d).

Then, the algorithm waits until $W_{\bar{a}} (= W_{p'})$ increases by 2 (line 10). We consider two cases: (♠) $W_i \geq (l+1)/\alpha$ for all $i \in \llbracket p', q' \rrbracket_k$ (Figure 1e) and (♡) $W_i < (l+1)/\alpha$ for some $i \in \llbracket p', q' \rrbracket_k$ (Figure 1h). In case (♠), the algorithm attempts to move \bar{a} from p' to p to call $\text{recurring}(\llbracket p, q \rrbracket_k, l+1)$. To do this, the algorithm waits until $\bar{s} \in \llbracket p, p'-1 \rrbracket_k$ (see Figure 2b)

or until $W_{\bar{a}} (= W_{p'})$ increases by 1 (line 12). If $W_{p'}$ increases by 1, then $W_i \geq 1$ holds for every $i \in \llbracket p, p' \rrbracket_k$ as we will show later (Figure 1f). Then, it calls **recurring**($\llbracket p', q \rrbracket_k, l + 1$) (line 18). Otherwise, the algorithm changes \bar{a} from p' to p by matching $\overline{p - p'}$ requests (line 14 and see Figure 1g). Consequently, it calls **recurring**($\llbracket p, q \rrbracket_k, l + 1$) (line 15). In case (\heartsuit), W_i becomes at least 1 for i in not a cyclic interval $\llbracket p', r \rrbracket_k$ (Figure 1h). Then, the algorithm calls **recurring**($\llbracket p', r \rrbracket_k, l$) (line 18).

After completing the call of **recurring**, we have $W_i \geq 1$ for all $i \in \mathbb{Z}_k$. Then, the algorithm moves to the next phase.

To analyze the competitive ratio of this algorithm, we present several lemmas. The value of W_i changes continuously within each phase, and its rate of increase over time is given by $dW_i(t)/dt = \overline{s(t) - i}$. The increase rate of W_i for each i is illustrated in Figure 3.



■ **Figure 3** The increase rates $dW_p/dt = \overline{s - p}$ and $dW_i/dt = \overline{s - i}$.

We demonstrate that if W_p increases significantly while W_i increases only slightly, then W_p increases significantly during periods when $\overline{s - i}$ is small.

► **Lemma 10.** *Let $p, q \in \mathbb{Z}_k$ and $i \in \llbracket p + 1, q \rrbracket_k$. For two times x, y ($x < y$) in the same phase, suppose that $W_p(y) - W_p(x) \geq 2$ and $W_i(y) - W_i(x) < 1/\alpha$. Then, the increment of W_p in time t with $x \leq t \leq y$ and $s(t) \in \llbracket i, p - 1 \rrbracket_k \cap \llbracket i, i + \lceil (\overline{i - p}) / (\alpha - 1) \rceil - 1 \rrbracket_k$ is more than 1.*

Proof. Let $\beta \in \mathbb{Z}_k$ be the index such that $\llbracket i, \beta \rrbracket_k = \llbracket i, p - 1 \rrbracket_k \cap \llbracket i, i + \lceil (\overline{i - p}) / (\alpha - 1) \rceil - 1 \rrbracket_k$. Define $T := \{t \mid x \leq t \leq y \text{ and } \overline{s(t)} \in \llbracket i, \beta \rrbracket_k\}$ and $\hat{T} := \{t \mid x \leq t \leq y \text{ and } \overline{s(t)} \notin \llbracket i, \beta \rrbracket_k\}$.

Fix $t \in \hat{T}$. Then, we have either $\overline{s(t)} \notin \llbracket i, p - 1 \rrbracket_k$ or $\overline{s(t) - i} \geq \lceil (\overline{i - p}) / (\alpha - 1) \rceil$. If $\overline{s(t)} \notin \llbracket i, p - 1 \rrbracket_k$, then $\frac{dW_i(t)}{dt} \geq \frac{dW_p(t)}{dt}$. Otherwise (i.e. $\overline{s(t)} \in \llbracket i, p - 1 \rrbracket_k$ and $\overline{s(t) - i} \geq \lceil (\overline{i - p}) / (\alpha - 1) \rceil$), we have

$$\frac{dW_i(t)}{dt} = \overline{s(t) - i} = \frac{(\overline{s(t) - i}) + (\alpha - 1) \cdot \overline{s(t) - i}}{\alpha} \geq \frac{(\overline{s(t) - i}) + (\overline{i - p})}{\alpha} = \frac{(\overline{s(t) - p})}{\alpha} = \frac{1}{\alpha} \cdot \frac{dW_p(t)}{dt}.$$

Thus, in either case, $dW_p(t)/dt \leq \alpha \cdot dW_i(t)/dt$.

Suppose to the contrary that the increment of W_p within $t \in T$ is at most 1. Under this condition, W_p must increase by at least 1 within $t \in \hat{T}$ because $W_p(y) - W_p(x) \geq 2$. Therefore, we have

$$\frac{1}{\alpha} \cdot 1 \leq \frac{1}{\alpha} \cdot \int_{\hat{T}} \frac{dW_p(t)}{dt} dt \leq \int_{\hat{T}} \frac{dW_i(t)}{dt} dt \leq \int_x^y \frac{dW_i(t)}{dt} dt = W_i(y) - W_i(x),$$

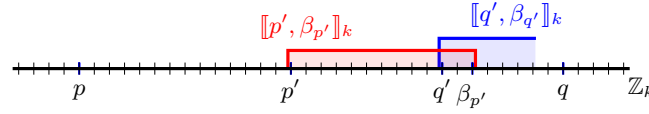
where the second inequality holds by $dW_p(t)/dt \leq \alpha \cdot dW_i(t)/dt$ for $t \in \hat{T}$. This contradicts the assumption that $W_i(y) - W_i(x) < 1/\alpha$. Therefore, the increment of W_p within $t \in T$ is more than 1. ◀

Next, by using Lemma 10, we show that W_i increases by at least $1/\alpha$ except for some small interval after W_p increases by 2. This indicates that $|\llbracket p', q' \rrbracket_k|$ is small in the situation depicted in Figure 1c.

► **Lemma 11.** *Let $p, q \in \mathbb{Z}_k$. For two times x, y ($x < y$) in the same phase, suppose that $W_p(y) - W_p(x) = 2$. Then, there exist some $p', q' \in \mathbb{Z}_k$ such that $\llbracket p', q' \rrbracket_k \subseteq \llbracket p, q \rrbracket_k$, $|\llbracket p', q' \rrbracket_k| \leq \lceil |\llbracket p, q \rrbracket_k|/\alpha \rceil$, and $W_i(y) - W_i(x) \geq 1/\alpha$ for all $i \in \llbracket p, q \rrbracket_k \setminus \llbracket p', q' \rrbracket_k$.*

Proof. If $W_i(y) - W_i(x) \geq 1/\alpha$ for all $i \in \llbracket p, q \rrbracket_k$, then the conditions are satisfied by setting $p' = q' = p$. Hence, in what follows, we assume that $W_i(y) - W_i(x) < 1/\alpha$ for some $i \in \llbracket p, q \rrbracket_k$. Define $\llbracket p', q' \rrbracket_k$ ($\subseteq \llbracket p, q \rrbracket_k$) to be the minimum cyclic interval such that $W_i(y) - W_i(x) \geq 1/\alpha$ for all $i \in \llbracket p, q \rrbracket_k \setminus \llbracket p', q' \rrbracket_k$.

What is left is to show that $|\llbracket p', q' \rrbracket_k| \leq \lceil |\llbracket p, q \rrbracket_k|/\alpha \rceil$. Let $\beta_i \in \mathbb{Z}_k$ be the index such that $\llbracket i, \beta_i \rrbracket_k = \llbracket i, p-1 \rrbracket_k \cap \llbracket i, i + \lceil (i-p)/(\alpha-1) \rceil - 1 \rrbracket_k$ and $T_i := \{t \mid x \leq t \leq y \text{ and } s(t) \in \llbracket i, \beta_i \rrbracket_k\}$ for $i \in \{p', q'\}$. Then, by Lemma 10, the increments of W_p within $t \in T_{p'}$ and $t \in T_{q'}$ are more than 1, respectively. Since the total increment of W_p is 2, the intervals $\llbracket p', \beta_{p'} \rrbracket_k$ and $\llbracket q', \beta_{q'} \rrbracket_k$ must overlap. Thus, we have $q' \in \llbracket p', \beta_{p'} \rrbracket_k$ (see Figure 4).



■ **Figure 4** Relative positions of p, q, p', q' , and $\beta_{p'}$ in Lemma 11.

Now, we are ready to prove $|\llbracket p', q' \rrbracket_k| \leq \lceil |\llbracket p, q \rrbracket_k|/\alpha \rceil$. If $\overline{p' - p} \leq (\alpha - 1) \cdot |\llbracket p, q \rrbracket_k|/\alpha$, then we have $|\llbracket p', q' \rrbracket_k| \leq |\llbracket p', \beta_{p'} \rrbracket_k| \leq \lceil (\overline{p' - p})/(\alpha - 1) \rceil \leq \lceil |\llbracket p, q \rrbracket_k|/\alpha \rceil$. Otherwise (i.e., $\overline{p' - p} > (\alpha - 1) \cdot |\llbracket p, q \rrbracket_k|/\alpha$), we have

$$\begin{aligned} |\llbracket p', q' \rrbracket_k| &\leq |\llbracket p', q \rrbracket_k| = |\llbracket p, q \rrbracket_k| - |\llbracket p, p-1 \rrbracket_k| = |\llbracket p, q \rrbracket_k| - \overline{p' - p} \\ &< |\llbracket p, q \rrbracket_k| - (\alpha - 1) \cdot |\llbracket p, q \rrbracket_k|/\alpha = |\llbracket p, q \rrbracket_k|/\alpha \leq \lceil |\llbracket p, q \rrbracket_k|/\alpha \rceil. \end{aligned} \quad \blacktriangleleft$$

Now, we demonstrate that **recurring** satisfies desired properties.

► **Lemma 12.** *When **recurring**($\llbracket p, q \rrbracket_k, l$) is called, the following three conditions are satisfied: (i) $W_i \geq 1$ for all $i \notin \llbracket p, q \rrbracket_k$, (ii) $W_i \geq l/\alpha$ for all $i \in \llbracket p, q \rrbracket_k$, and (iii) $\bar{a} = p$. Moreover, if **recurring**($\llbracket \hat{p}, \hat{q} \rrbracket_k, \hat{l}$) is called next, it satisfies either (a) $|\llbracket \hat{p}, \hat{q} \rrbracket_k| \leq |\llbracket p, q \rrbracket_k|$ and $\hat{l} = l + 1$, or (b) $|\llbracket \hat{p}, \hat{q} \rrbracket_k| \leq \lceil |\llbracket p, q \rrbracket_k|/(\alpha - 1) \rceil$ and $\hat{l} = l$.*

Proof. We prove these by induction.

Initially, when **recurring**($\llbracket 0, k-1 \rrbracket_k, 0$) is called at the beginning of a phase, the conditions are satisfied as $W_i = 0 \geq 0/\alpha$ for all $i \in \llbracket 0, k-1 \rrbracket_k$ and $\bar{a} = p = 0$.

As an induction hypothesis, suppose the conditions are satisfied when **recurring**($\llbracket p, q \rrbracket_k, l$) is called. Our goal is to show that these conditions continue to be satisfied at the next recursive call. We analyze this based on where the recursive call is made in the conditional branching of Algorithm 1. We denote by L the cardinality of $\llbracket p, q \rrbracket_k$.

If the condition of line 5 is true and recurring($\llbracket p, q \rrbracket_k, l + 1$) is called Conditions (i) and (iii) are satisfied as $\bar{a} = p$. Condition (ii) is also satisfied since the condition of line 5 is true (see Figure 1b). Moreover, the next call satisfies conditions (a).

If the condition of line 5 is false Since the proposed algorithm waits until $W_{\bar{a}}$ increases by 2 at line 2, the interval $\llbracket p', q' \rrbracket_k$ chosen at line 6 satisfies $|\llbracket p', q' \rrbracket_k| \leq \lceil L/\alpha \rceil$ by Lemma 11 (see Figure 1c).

If the condition of line 8 is true and recurring($\llbracket p, q \rrbracket_k, l + 1$) is called Conditions (i) and (iii) are satisfied as $\bar{a} = p$. As W_p increases by $1/\alpha$ while $\bar{s} \in \llbracket p, p' - 1 \rrbracket_k$, W_i also increases by at least $1/\alpha$ for all $i \in \llbracket p', p - 1 \rrbracket_k$ by $\bar{s} - i \geq \bar{s} - p$ (see Figure 1c). Thus, for all $i \in \llbracket p', q' \rrbracket_k$, W_i increases by $1/\alpha$, and hence condition (ii) is also satisfied (see Figure 1b). Additionally, the next call satisfies condition (a).

If the condition of line 8 is false Since $\bar{s} \in \llbracket p', p - 1 \rrbracket_k$, the proposed algorithm can match $(\overline{p - a}) = (\overline{p' - p})$ requests at line 9 (see Figure 1d). This match results in $\bar{a} = p'$.

If the condition of line 11 is true In this scenario, **recurring**($\llbracket p', q \rrbracket_k, l + 1$) at line 13 or **recurring**($\llbracket p, q \rrbracket_k, l + 1$) at line 15 is called. In both cases, condition (a) is satisfied. In the first case, $W_{\bar{a}} (= W_{p'})$ increases by 1 at line 12, which indicates $\bar{s} \in \llbracket p', p - 1 \rrbracket_k$ during the increase. Thus, for any $i \in \llbracket p, p' - 1 \rrbracket_k$, W_i increases by at least 1, and $\bar{a} = p'$ (see Figure 1f). Then, it calls **recurring**($\llbracket p', q \rrbracket_k, l + 1$) at line 13. In the second case, \bar{s} is in $\llbracket p, p' - 1 \rrbracket_k$ and the proposed algorithm can match $(\overline{p - a}) = (\overline{p - p'})$ requests, making $\bar{a} = p$ (see Figure 1g). Then, it calls **recurring**($\llbracket p, q \rrbracket_k, l + 1$) at line 15. These calls satisfy conditions (i) and (iii) in both cases and condition (ii) is satisfied because the condition of line 11 is true.

If the condition of line 11 is false and recurring($\llbracket p', r \rrbracket_k, l$) is called There exists an index $i \in \llbracket p', q' \rrbracket_k$ such that W_i increases by less than $1/\alpha$ at line 10 (Figure 1h). Let $\beta \in \mathbb{Z}_k$ be the index such that $\llbracket i, \beta \rrbracket_k = \llbracket i, p - 1 \rrbracket_k \cap \llbracket i, i + \lceil (i - p)/(\alpha - 1) \rceil - 1 \rrbracket_k$. Lemma 10 ensures that $W_{p'}$ increases by more than 1 during $\bar{s} \in \llbracket i, \beta \rrbracket_k$. Therefore, for all $j \in \llbracket \beta + 1, p' - 1 \rrbracket_k$, W_j also increases by at least 1. As $(\overline{i - p'}) \leq (\overline{q' - p'}) = |\llbracket p', q' \rrbracket_k| - 1 \leq L/\alpha$, we obtain

$$\begin{aligned} \llbracket p', \beta \rrbracket_k &\subseteq \llbracket p', i + \lceil (i - p')/(\alpha - 1) \rceil - 1 \rrbracket_k \\ &\subseteq \llbracket p', p' + (i - p') + \lceil \frac{1}{\alpha - 1} \cdot (i - p') \rceil - 1 \rrbracket_k \\ &= \llbracket p', p' + \lceil \frac{\alpha}{\alpha - 1} \cdot (i - p') \rceil - 1 \rrbracket_k \subseteq \llbracket p', p' + \lceil L/(\alpha - 1) \rceil - 1 \rrbracket_k. \end{aligned}$$

As $\llbracket p', r \rrbracket_k \subseteq \llbracket p', \beta \rrbracket_k$, we have $|\llbracket p', r \rrbracket_k| \leq \lceil L/(\alpha - 1) \rceil$ and condition (b) is satisfied. Since $\bar{a} = p'$, conditions (i) and (iii) are satisfied, and by the induction hypothesis, condition (ii) is also satisfied. \blacktriangleleft

We can bound the number of recursions called by the proposed algorithm from Lemma 12, giving us the upper bound of the cost incurred by the proposed algorithm during each phase.

► **Lemma 13.** *The proposed algorithm incurs a cost of $O(\alpha)$ per phase.*

Proof. It is not difficult to see that the cost incurred by the algorithm in each recursive call is no more than 8. Thus, it is sufficient to prove that the number of recursions is at most $O(\alpha)$ in each phase.

As our goal is asymptotic evaluation, we may assume that $\alpha \geq 4$, which implies $\alpha \leq (\alpha - 1)^2$. By Lemma 12, each call of **recurring**($\llbracket p, q \rrbracket_k, l$) either increments l by 1 or reduces the number of elements in $\llbracket p, q \rrbracket_k$ to at most $\lceil |\llbracket p, q \rrbracket_k|/(\alpha - 1) \rceil$. Recall that the recursion starts with $|\llbracket p, q \rrbracket_k| = k (= \alpha^\alpha)$ and $l = 0$ and ends when $|\llbracket p, q \rrbracket_k| = 1$ or $l \geq \alpha$.

Let L_n be the number of elements in $\llbracket p, q \rrbracket_k$ after the interval reduction has occurred n times. Then, we have $L_n \leq \lceil L_{n-1}/(\alpha - 1) \rceil \leq L_{n-1}/(\alpha - 1) + 1$, which implies

$$\left(L_n - \frac{\alpha - 1}{\alpha - 2}\right) \leq \frac{1}{\alpha - 1} \cdot \left(L_{n-1} - \frac{\alpha - 1}{\alpha - 2}\right) \leq \dots \leq \frac{1}{(\alpha - 1)^n} \cdot \left(L_0 - \frac{\alpha - 1}{\alpha - 2}\right) \leq \frac{L_0}{(\alpha - 1)^n}.$$

Thus, the number of elements in the interval after $2\lceil \alpha \rceil + 1$ iterations is at most

$$L_{2\lceil \alpha \rceil + 1} \leq \frac{\alpha^\alpha}{(\alpha - 1)^{2\alpha + 1}} + \frac{\alpha - 1}{\alpha - 2} \leq \frac{1}{\alpha - 1} + \frac{\alpha - 1}{\alpha - 2} = 1 + \frac{1}{\alpha - 1} + \frac{1}{\alpha - 2} < 2,$$

where the second inequality holds by $\alpha \leq (\alpha - 1)^2$ and the third inequality holds by $1/(\alpha - 1) < 1/(\alpha - 2) \leq 1/2$ from the assumption that $\alpha \geq 4$. Consequently, the number of elements in $\llbracket p, q \rrbracket_k$ is reduced to 1 or l increases to $\lceil \alpha \rceil$ in at most $3\lceil \alpha \rceil = O(\alpha)$ iterations. Thus, the number of recursive calls is $O(\alpha)$. ◀

Next, we provide a lower bound on the cost incurred by any algorithm during each phase.

► **Lemma 14.** *At the end of a phase, the cost incurred within the phase by any algorithm is at least 1 (if we treat the waiting costs as imposed sequentially at each moment, rather than at the time of matching).*

Proof. Recall that, if $(k - i)$ unmatched requests are carried over to a phase, the optimum cost incurred thus far within the phase is at least $\min\{1, W_i\}$. Thus, it is sufficient to prove that $W_i \geq 1$ for all $i \in \mathbb{Z}_k$ at the end of the phase.

The phase ends at line 4. At this point, we have $|\llbracket p, q \rrbracket_k| = 1$ or $l \geq \alpha$, according to the condition at line 3. If $|\llbracket p, q \rrbracket_k| = 1$, then we have $p = q$, and $W_i \geq 1$ for all $i \neq p$ by Lemma 12. Since $W_p \geq 1$ due to the wait performed at line 2, it follows that $W_i \geq 1$ for all $i \in \mathbb{Z}_k$. Conversely, if $l \geq \alpha$, we have $W_i \geq 1$ for all $i \notin \llbracket p, q \rrbracket_k$ and $W_i \geq l/\alpha \geq 1$ for all $i \in \llbracket p, q \rrbracket_k$ by Lemma 12. Thus, in either case, we obtain $W_i \geq 1$ for all $i \in \mathbb{Z}_k$ at the end of the phase. This proves the lemma. ◀

Based on the lemmas above, we now prove Theorem 8.

Proof of Theorem 8. Suppose that the proposed algorithm completes p (≥ 1) phases for the given instance. Then, the cost for the proposed algorithm is at most $(p + 1) \cdot O(\alpha)$ by Lemma 13, while the cost for any algorithm is at least p by Lemma 14. Therefore, the competitive ratio for this instance is at most $(p + 1) \cdot O(\alpha)/p = O(\alpha)$.

Conversely, suppose that the instance ends during the first phase. If the optimal offline algorithm incurs a cost of at least 1, then the competitive ratio is at most $O(\alpha)$, as the cost incurred by the proposed algorithm during the first phase is $O(\alpha)$. If the cost incurred by the optimal offline algorithm is less than 1, then only waiting costs are incurred, as no requests are carried over to the first phase. This means that the total cost of the optimal offline algorithm is equal to W_0 . Since the proposed algorithm continues to wait until W_0 reaches 2 at line 2, the total cost of the proposed algorithm is also equal to W_0 , resulting in a competitive ratio of 1 for such an instance.

Therefore, the proposed algorithm is $O(\alpha)$ ($= O(\log k / \log \log k)$)-competitive. ◀

4.2 Lower Bound

In this subsection, we prove the following lower bound of the competitive ratio.

► **Theorem 15.** *For the OMDSC problem with a penalty function f that satisfies $B_f = \{kn \mid n \in \mathbb{Z}_{++}\}$, every deterministic online algorithm has a competitive ratio of $\Omega(\log k / \log \log k)$.*

Recall that α is a positive real such that $\alpha^\alpha = k$. We fix an arbitrary online algorithm ALG and construct an adversarial instance according to the behavior of ALG, denoted as σ_{ALG} . The instance σ_{ALG} is composed of $\Theta(\alpha)$ rounds. To construct σ_{ALG} , we introduce variables similar to Definition 9.

► **Definition 16.** *For the instance σ_{ALG} and time t , the variables $W_0(t), W_1(t), \dots, W_{k-1}(t), s(t)$, and $a(t)$ are defined as follows:*

- $W_i(t)$ ($i \in \mathbb{Z}_k$): *the waiting cost incurred by time t for an algorithm that initially matches $(i - 1)$ requests (i.e., leaving $(k - i)$ requests) at time 0 and subsequently performs matches of size k immediately for every k unmatched requests accumulated.*

- $s(t)$: the number of requests given by time t .
- $a(t)$: the number of requests matched by ALG up to, but not including, time t .

We will omit the argument t when there is no confusion.

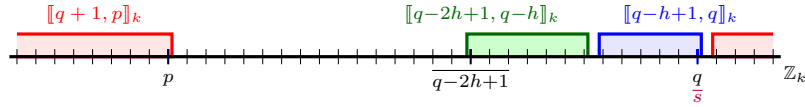
The instance σ_{ALG} contains $k - 1$ requests with arrival time at 0. For σ_{ALG} and any $i \in \mathbb{Z}_k$, there exists an offline algorithm that incurs a cost of at most $2 + W_i(x_{n^*})$, where x_{n^*} is the time when the last requests are given in σ_{ALG} . This can be achieved by matching $\overline{(i - 1)}$ requests at time 0, greedily matching a multiple of k requests in the middle, and matching all the remaining requests at time x_{n^*} . For each round n , the starting time is x_n , and there is a (non-empty) cyclic interval $\llbracket p_n, q_n \rrbracket_k$ such that $W_i(x_n) \leq 2\alpha/n$ for every $i \in \llbracket p_n, q_n \rrbracket_k$. As rounds progress, the interval is gradually narrowed, and ALG incurs a cost of $\Omega(1)$ per round. The instance σ_{ALG} consists of $\Theta(\alpha)$ rounds. This implies that while there exists an offline algorithm with a cost of $O(1)$, ALG incurs a cost of $\Omega(\alpha)$, indicating that the competitive ratio of ALG is $\Omega(\alpha) = \Omega(\log k / \log \log k)$.

Initially, the instance gives $k - 1$ requests at time 0 (i.e. $\overline{a(0)} = 0$ and $\overline{s(0)} = k - 1$). Let n be the variable that indicates the round, initialized to 0. Define $p_0 = 0$, $q_0 = k - 1$, and $x_0 = 0$.

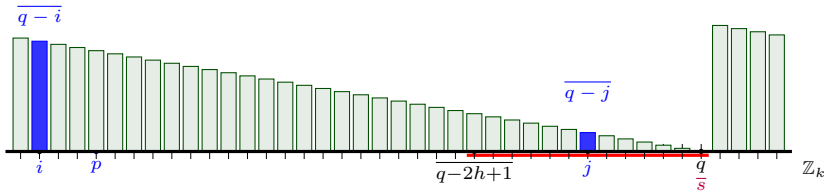
For the n th round ($n \geq 0$), if $|\llbracket p_n, q_n \rrbracket_k| < \alpha^2$, set n^* to be n and finalize the instance. Otherwise, the round continues until time $x_{n+1} = x_n + 1/(s(x_n) - a(x_n))$. Requests in round n are provided at time x_{n+1} based on the number of remaining requests immediately before time x_{n+1} (i.e., $\overline{s(x_n) - a(x_{n+1}))}$).

Let h_n be $\lceil |\llbracket p_n, q_n \rrbracket_k| / \alpha^2 \rceil$. If $\overline{a(x_{n+1})} \notin [q_n - h_n + 1, q_n]_k$, then set (p_{n+1}, q_{n+1}) to be $(q_n - h_n + 1, q_n)$ and give no requests. Otherwise, let $(p_{n+1}, q_{n+1}) = (q_n - 2h_n + 1, q_n - h_n)$, and give $\overline{q_{n+1} - q_n}$ requests so that $\overline{s(x_{n+1})} = q_{n+1}$ at time x_{n+1} (see Figure 5). Then, increase round count n by 1 and proceed to the next round.

This instance ensures that (i) $k/\alpha^{2n} \leq |\llbracket p_n, q_n \rrbracket_k| \leq k/\alpha^n$, (ii) $W_i(x_n) \leq 2n/\alpha$ for all $i \in \llbracket p_n, q_n \rrbracket_k$, and (iii) $\overline{a(x_n)} \in [q_n + 1, p_n]_k$ and $\overline{s(x_n)} = q_n$.



■ **Figure 5** Relative positions of the cyclic intervals.



■ **Figure 6** Increment rates $dW_i/dt = \overline{q - i}$ and $dW_j/dt = \overline{q - j}$.

Now, we show a lemma that establishes a lower bound on the number of elements in a cyclic interval where the increment of W_i in the interval is at most $2/\alpha$ times the increment of W_j for specific j .

► **Lemma 17.** For a cyclic interval $\llbracket p, q \rrbracket_k$, let $L := |\llbracket p, q \rrbracket_k|$ and $h := \lceil L/\alpha^2 \rceil$. Suppose that $L \geq \alpha^2$ and $\alpha \geq 4$. Then, for any $i \in [q + 1, p]_k$ and $j \in [q - 2h + 1, q]_k$, the increment of W_j is at most $2/\alpha$ times the increment of W_i in the state where $\overline{s} = q$.

Proof. Figure 6 depicts the increment rates $dW_i/dt = \overline{q-i}$ and $dW_j/dt = \overline{q-j}$. By the assumption that $L \geq \alpha^2$ and $\alpha \geq 4$, we have

$$\frac{2}{\alpha} \cdot (L-1) - \left(\frac{2L}{\alpha^2} + 1 \right) = \frac{2L}{\alpha} \cdot \left(1 - \frac{1}{\alpha} \right) - \frac{2}{\alpha} - 1 \geq \frac{2\alpha^2}{\alpha} \cdot \left(1 - \frac{1}{4} \right) - \frac{2}{4} - 1 > 0. \quad (1)$$

Therefore, for any $i \in \llbracket q+1, p \rrbracket_k$ and $j \in \llbracket q-2h+1, q \rrbracket_k$, we get

$$\frac{dW_j}{dt} = \overline{q-j} \leq 2 \cdot \left\lceil \frac{L}{\alpha^2} \right\rceil - 1 \leq 2 \cdot \frac{L}{\alpha^2} + 1 \leq \frac{2}{\alpha} \cdot (L-1) = \frac{2}{\alpha} \cdot \overline{q-p} = \frac{2}{\alpha} \cdot \frac{dW_i}{dt},$$

where the third inequality follows from (1). Since the rate of increase is constant while $\bar{s} = q$, the increment of W_j is at most $2/\alpha$ times the increment of W_i . ◀

By using this lemma, we bound the number of elements in $\llbracket p, q \rrbracket_k$ after n rounds from below and bound the values of W_i for all $i \in \llbracket p, q \rrbracket_k$ from above.

► **Lemma 18.** *Let $n \in \{0, 1, 2, \dots, n^*\}$ and suppose that $\alpha \geq 4$. At time x_n , the following three conditions are satisfied: (i) $k/\alpha^{2n} \leq |\llbracket p_n, q_n \rrbracket_k| \leq k/\alpha^n$, (ii) $W_i(x_n) \leq 2n/\alpha$ for all $i \in \llbracket p_n, q_n \rrbracket_k$, and (iii) $\overline{a(x_n)} \in \llbracket q_n+1, p_n \rrbracket_k$ and $\overline{s(x_n)} = q_n$.*

Proof. We prove this by induction on n .

At time $x_0 = 0$, we have $\llbracket p_0, q_0 \rrbracket_k = \llbracket 0, k-1 \rrbracket_k = k = k/\alpha^0$, and $W_i(x_0) = 0 \leq 2 \cdot 0/\alpha$ for all $i \in \llbracket 0, k-1 \rrbracket_k = \llbracket p_0, q_0 \rrbracket_k$. In addition, we have $\overline{a(x_0)} = 0 \in \llbracket 0, 0 \rrbracket_k = \llbracket k, 0 \rrbracket_k$ and $\overline{s(x_0)} = k-1$. Thus, conditions (i), (ii), and (iii) are satisfied.

Let $n' \in \{0, 1, 2, \dots, n^*-1\}$. Suppose that conditions (i), (ii), and (iii) are satisfied for $n = n'$. We show that these conditions are also satisfied for $n = n' + 1$.

Recall that $h_n = \lceil |\llbracket p_n, q_n \rrbracket_k|/\alpha^2 \rceil$. Then, there are two cases where (p_{n+1}, q_{n+1}) is set to be $(q_n - h_n + 1, q_n)$ or $(q_n - 2h_n + 1, q_n - h_n)$. In both cases, we prove the inductive step by using Lemma 17. Note that $|\llbracket p_{n+1}, q_{n+1} \rrbracket_k| = h_n$, $\overline{a(x_{n+1})} \notin \llbracket p_{n+1}, q_{n+1} \rrbracket_k$ and $\overline{s(x_{n+1})} = q_{n+1}$ in both cases by the definition of the procedure. Thus, the condition (iii) is met. By the induction hypothesis, we have $k/\alpha^{2n} \leq |\llbracket p_n, q_n \rrbracket_k| \leq k/\alpha^n$ and $\overline{a(x_n)} \in \llbracket q_n+1, p_n \rrbracket_k$. Also, as $n < n^*$, we have $|\llbracket p_n, q_n \rrbracket_k| \geq \alpha^2$. Thus, we have

$$|\llbracket p_{n+1}, q_{n+1} \rrbracket_k| = \left\lceil \frac{|\llbracket p_n, q_n \rrbracket_k|}{\alpha^2} \right\rceil \geq \left\lceil \frac{k/\alpha^{2n}}{\alpha^2} \right\rceil \geq \frac{k}{\alpha^{2n+2}}.$$

Also, we have

$$|\llbracket p_{n+1}, q_{n+1} \rrbracket_k| = \left\lceil \frac{|\llbracket p_n, q_n \rrbracket_k|}{\alpha^2} \right\rceil \leq 1 + \frac{|\llbracket p_n, q_n \rrbracket_k|}{\alpha^2} \leq \frac{2|\llbracket p_n, q_n \rrbracket_k|}{\alpha^2} \leq \frac{|\llbracket p_n, q_n \rrbracket_k|}{\alpha} \leq \frac{k}{\alpha^{n+1}},$$

where the second inequality holds by $|\llbracket p_n, q_n \rrbracket_k| \geq \alpha^2$ and the third inequality holds by $\alpha \geq 4$. Thus, the condition (i) is satisfied in both cases. Moreover, we have

$$W_{p_n}(x_{n+1}) - W_{p_n}(x_n) = (x_{n+1} - x_n) \cdot \overline{(s(x_n) - p_n)} = \frac{(s(x_n) - p_n)}{(s(x_n) - a(x_n))} \leq 1,$$

because $\overline{s(x_n)} = q_n$ and $\overline{a(x_n)} \in \llbracket q_n+1, p_n \rrbracket_k$ (see Figure 6). Thus, the increment of W_j is less than $2/\alpha$ for all $j \in \llbracket p_{n+1}, q_{n+1} \rrbracket_k \subseteq \llbracket q_n - 2h_n + 1, q_n \rrbracket_k$ by Lemma 17 with setting $(p, q) = (p_n, q_n)$ and $i = p_n$. Therefore, combining with the induction hypothesis, we obtain $W_j(x_{n+1}) \leq 2(n+1)/\alpha$ for all $j \in \llbracket p_{n+1}, q_{n+1} \rrbracket_k$. This means that the condition (ii) is satisfied.

Therefore, the conditions (i), (ii), and (iii) are satisfied for $n = n' + 1$. This completes the proof by induction. ◀

In what follows, we bound the cost incurred by any online algorithm ALG on σ_{ALG} is at least $\Omega(\alpha)$ and the cost for the optimal offline algorithm on σ_{ALG} is at most a constant. As our goal is asymptotic evaluation, we assume that $\alpha \geq 4$ in the following.

► **Lemma 19.** *The number of rounds n^* is $\Theta(\alpha)$.*

Proof. For $n \in \{0, 1, 2, \dots, n^*\}$, let $L_n = \lfloor [p_n, q_n]_k \rfloor$. From Lemma 18, we have $k/\alpha^{2n} \leq L_n \leq k/\alpha^n$. By the termination condition, we have $L_{n^*} < \alpha^2$ and $L_{n^*-1} \geq \alpha^2$. As $\alpha^2 > L_{n^*} \geq k/\alpha^{2n^*} = \alpha^{\alpha-2n^*}$, we have $n^* \geq (\alpha - 2)/2 = \alpha/2 - 1$. In addition, as $\alpha^2 \leq L_{n^*-1} \leq k/\alpha^{n^*-1} = \alpha^{\alpha-n^*+1}$, we have $n^* \leq \alpha - 1 \leq \alpha$. Therefore, we obtain $\alpha/2 - 1 \leq n^* \leq \alpha$, and hence $n^* = \Theta(\alpha)$. ◀

From Lemma 19, we can prove the following lemmas.

► **Lemma 20.** *For any deterministic online algorithm ALG, there exists an offline algorithm that incurs a cost of at most a constant for the instance σ_{ALG} .*

Proof. Let i^* be an arbitrary element in $\lfloor [p_{n^*}, q_{n^*}]_k \rfloor$. Consider the algorithm that initially matches $(i^* - 1)$ requests at time 0, subsequently performs matches of size k immediately for every k unmatched requests accumulated, and finally matches all the remaining unmatched requests at x_{n^*} . Then, the total waiting cost of this algorithm is at most $W_{i^*}(x_{n^*}) \leq 2n^*/\alpha \leq 2$ by Lemmas 18 and 19. Moreover, the total size cost is at most 2, as it matches at most twice with sets of requests of size not a multiple of k . Therefore, The total cost incurred by this algorithm is at most 4. ◀

► **Lemma 21.** $\text{ALG}(\sigma_{\text{ALG}}) = \Omega(\alpha)$ for any deterministic online algorithm ALG.

Proof. For the n th round, we divide into two cases in which ALG matches a non-multiple of k requests during a period $[x_n, x_{n+1})$ or not. If ALG matches a non-multiple of k requests, it incurs a size cost of 1. Otherwise, $a(t) = a(x_n)$ for all $t \in [x_n, x_{n+1})$, leading to that the increment of $W_{\frac{a(x_n)}{s(x_n)}}$ during $[x_n, x_{n+1})$ is 1 because $x_{n+1} - x_n = 1/(s(x_n) - a(x_n))$ and its increase rate is $(s(x_n) - a(x_n))$. In both cases, the algorithm incurs at least a cost of 1 in each round.

Therefore, the total cost incurred over the instance is at least $1 \cdot n^* = \Omega(\alpha)$ by Lemma 19. ◀

Now, we are ready to prove Theorem 15.

Proof of Theorem 15. By combining Lemmas 20 and 21, the competitive ratio of ALG is at least

$$\sup_{\sigma} \frac{\text{ALG}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{\text{ALG}(\sigma_{\text{ALG}})}{\text{OPT}(\sigma_{\text{ALG}})} \geq \frac{\Omega(\alpha)}{O(1)} = \Omega(\alpha) = \Omega\left(\frac{\log k}{\log \log k}\right). \quad \blacktriangleleft$$

References

- 1 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1051–1061, 2017. doi:10.1137/1.9781611974782.67.
- 2 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay – Clairvoyance is not required. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA 2020)*, pages 8:1–8:21, 2020. doi:10.4230/LIPIcs.ESA.2020.8.

- 3 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. *Theory Comput. Syst.*, 64(4):572–592, 2020. doi:10.1007/s00224-019-09963-7.
- 4 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In *Proceedings of the 2021 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 301–320, 2021. doi:10.1137/1.9781611976465.20.
- 5 Yossi Azar and Noam Touitou. General framework for metric optimization problems with delay or with deadlines. In *Proceedings of the 2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 60–71, 2019. doi:10.1109/FOCS.2019.00013.
- 6 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Luk'avs Folwarczn'y, Lukasz Jez, Jiri Sgall, Kim Thang Nguyen, and Pavel Vesely. Online algorithms for multi-level aggregation. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016)*, pages 12:1–12:17, 2016. doi:10.4230/LIPIcs.ESA.2016.12.
- 7 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, Dorian Nogneng, and Jiri Sgall. Better approximation bounds for the joint replenishment problem. In *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 42–54, 2014. doi:10.1137/1.9781611973402.4.
- 8 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In *Proceedings of the 16th Workshop on Approximation and Online Algorithms (WAOA 2018)*, pages 51–68, 2018. doi:10.1007/978-3-030-04693-4_4.
- 9 Niv Buchbinder, Moran Feldman, Joseph S. Naor, and Ohad Talmon. $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1235–1244, 2017. doi:10.1137/1.9781611974782.80.
- 10 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: Primal-dual competitive algorithms. *Oper. Res.*, 61(4):1014–1029, 2013. doi:10.1287/opre.2013.1188.
- 11 Rodrigo A. Carrasco, Kirk Pruhs, Cliff Stein, and José Verschae. The online set aggregation problem. In *Proceedings of the 13th Latin American Symposium on Theoretical Informatics (LATIN 2018)*, pages 245–259, 2018. doi:10.1007/978-3-319-77404-6_19.
- 12 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, pages 40:1–40:18, 2022. doi:10.4230/LIPIcs.ICALP.2022.40.
- 13 Lindsey Deryckere and Seeun William Umboh. Online matching with set and concave delays. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, pages 17:1–17:17, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.17.
- 14 Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. On-line analysis of the TCP acknowledgment delay problem. *J. ACM*, 48(2):243–273, 2001. doi:10.1145/375827.375843.
- 15 Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani. *Online and Matching-Based Market Design*. Cambridge University Press, 2023. doi:10.1017/9781108937535.
- 16 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: Haste makes waste! In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 333–344, 2016. doi:10.1145/2897518.2897557.
- 17 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. *Theor. Comput. Sci.*, 754:122–129, 2019. doi:10.1016/j.tcs.2018.07.004.
- 18 Kun He, Sizhe Li, Enze Sun, Yuyi Wang, Roger Wattenhofer, and Weihao Zhu. Randomized algorithm for MPMD on two sources. In *Proceedings of the 19th Conference On Web And InterNet Economics (WINE 2023)*, pages 348–365, 2023. doi:10.1007/978-3-031-48974-7_20.

- 19 Naonori Kakimura and Tomohiro Nakayoshi. Deterministic primal-dual algorithms for online k -way matching with delays. In *Proceedings of the 29th International Computing and Combinatorics Conference (COCOON 2023)*, pages 238–249, 2023. doi:10.1007/978-3-031-49193-1_18.
- 20 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgement and other stories about $e/(e-1)$. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 502–509, 2001. doi:10.1145/380752.380845.
- 21 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC 1990)*, pages 352–358, 1990. doi:10.1145/100216.100262.
- 22 Yasushi Kawase and Tomohiro Nakayoshi. Online matching with delays and size-based costs. *arXiv:2408.08658*, 2024. doi:10.48550/arXiv.2408.08658.
- 23 Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Impatient online matching. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC 2018)*, pages 62:1–62:12, 2018. doi:10.4230/LIPIcs.ISAAC.2018.62.
- 24 Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Online matching with convex delay costs. *arXiv:2203.03335*, 2022. doi:10.48550/arXiv.2203.03335.
- 25 Aranyak Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, 2013. doi:10.1561/04000000057.
- 26 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. AdWords and generalized online matching. *J. ACM*, 54(5):22:1–22:19, 2007. doi:10.1145/1284320.1284321.
- 27 Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. Online k -way matching with delays and the H -metric. *arXiv:2109.06640*, 2021. doi:10.48550/arXiv.2109.06640.
- 28 Marco Pavone, Amin Saberi, Maximilian Schiffer, and Matthew W. Tsao. Technical note—online hypergraph matching with delays. *Oper. Res.*, 70(4):2194–2212, 2022. doi:10.1287/opre.2022.2277.
- 29 Noam Touitou. Nearly-tight lower bounds for set cover and network design with deadlines/delay. In *Proceedings of the 32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, pages 53:1–53:16, 2021. doi:10.4230/LIPIcs.ISAAC.2021.53.

Modular Counting CSP: Reductions and Algorithms

Amirhossein Kazeminia ✉

Simon Fraser University, Burnaby, Canada

Andrei A. Bulatov ✉

Simon Fraser University, Burnaby, Canada

Abstract

The Constraint Satisfaction Problem (CSP) is ubiquitous in various areas of mathematics and computer science. Many of its variations have been studied including the Counting CSP, where the goal is to find the number of solutions to a CSP instance. The complexity of finding the exact number of solutions of a CSP is well understood (Bulatov, 2013, and Dyer and Richerby, 2013) and the focus has shifted to other variations of the Counting CSP such as counting the number of solutions modulo an integer. This problem has attracted considerable attention recently. In the case of CSPs based on undirected graphs Bulatov and Kazeminia (STOC 2022) obtained a complexity classification for the problem of counting solutions modulo p for arbitrary prime p . In this paper we report on the progress made towards a similar classification for the general CSP, not necessarily based on graphs.

We identify several features that make the general case very different from the graph case such as a stronger form of rigidity and the structure of automorphisms of powers of relational structures. We provide a solution algorithm in the case $p = 2$ that works under some additional conditions and prove the hardness of the problem under some assumptions about automorphisms of the powers of the relational structure. We also reduce the general CSP to the case that only uses binary relations satisfying strong additional conditions.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Constraint and logic programming

Keywords and phrases Constraint Satisfaction Problem, Modular Counting

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.60

Related Version *Full Version*: <https://arxiv.org/abs/2501.04224> [29]

Funding *Andrei A. Bulatov*: NSERC Discovery Grant.

1 Introduction

Counting problems in general have been intensively studied since the pioneering work by Valiant [34, 33]. One of the most interesting and well studied problems in this area is the Counting Constraint Satisfaction Problem ($\#$ CSP), which provides a generic framework for a wide variety of counting combinatorial problems that arise frequently in multiple disciplines such as logic, graph theory, and artificial intelligence. The counting CSP also allows for generalizations including weighted CSPs and partition functions [2, 7] that yield connections with areas such as statistical physics, see, e.g. [27, 32]. While the complexity of exact counting solutions of a CSP is now well-understood [13, 3, 14, 12], modular counting such as finding the parity of the number of solutions remains widely open.

Homomorphisms and the Constraint Satisfaction Problem. The most convenient way to introduce CSPs is through homomorphisms of relational structures. A *relational signature* σ is a collection of *relational symbols* each of which is assigned a positive integer, the *arity* of the symbol. A *relational structure* \mathcal{H} with signature σ is a set H and an *interpretation* $\mathcal{R}^{\mathcal{H}}$



© Amirhossein Kazeminia and Andrei A. Bulatov;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 60; pp. 60:1–60:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



of each $\mathcal{R} \in \sigma$, where $\mathcal{R}^{\mathcal{H}}$ is a relation or a predicate on H whose arity equals that of \mathcal{R} . The set H is said to be the *base set* or the *universe* of \mathcal{H} . We will use the same letter for the base set as for the structure, only in the regular font. A structure with signature σ is often called a σ -*structure*. Structures with the same signature are called *similar*.

Let \mathcal{G}, \mathcal{H} be similar structures with signature σ . A *homomorphism* from \mathcal{G} to \mathcal{H} is a mapping $\varphi : G \rightarrow H$ such that for any $\mathcal{R} \in \sigma$, say, of arity r , if $\mathcal{R}^{\mathcal{G}}(a_1, \dots, a_r)$ is true for $a_1, \dots, a_r \in G$, then $\mathcal{R}^{\mathcal{H}}(\varphi(a_1), \dots, \varphi(a_r))$ is also true. The set of all homomorphisms from \mathcal{G} to \mathcal{H} is denoted $\text{Hom}(\mathcal{G}, \mathcal{H})$. The cardinality of $\text{Hom}(\mathcal{G}, \mathcal{H})$ is denoted by $\text{hom}(\mathcal{G}, \mathcal{H})$. A homomorphism φ is an *isomorphism* if it is bijective and the inverse mapping φ^{-1} is a homomorphism from \mathcal{H} to \mathcal{G} . A homomorphism of a structure to itself is called an *endomorphism*, and an isomorphism to itself is called an *automorphism*.

Following Feder and Vardi [17], in a CSP, the goal is, given similar relational structures \mathcal{G}, \mathcal{H} , to decide whether there is a homomorphism from \mathcal{G} to \mathcal{H} . The restricted problem in which \mathcal{H} is fixed and only \mathcal{G} is given as an input is denoted by $\text{CSP}(\mathcal{H})$. The complexity of such problems is well understood [4, 35].

Counting CSP. In the (exact) Counting CSP the goal is to find the number $\text{hom}(\mathcal{G}, \mathcal{H})$ of homomorphisms from a structure \mathcal{G} to a similar structure \mathcal{H} . Restricted versions of the Counting CSP can be introduced in the same way as for the decision one. In the counting version of $\text{CSP}(\mathcal{H})$ denoted $\#\text{CSP}(\mathcal{H})$ the goal is to find $\text{hom}(\mathcal{G}, \mathcal{H})$ for a given structure \mathcal{G} as an input.

The complexity class the Counting CSP belongs to is $\#\text{P}$, the class of problems of counting accepting paths of polynomial time nondeterministic Turing machines. There are several ways to define reductions between counting problems, but the most widely used ones are parsimonious reductions and Turing reductions. A *parsimonious reduction* from a counting problem $\#A$ to a counting problem $\#B$ is an algorithm that, given an instance I of $\#A$ produces (in polynomial time) an instance I' of $\#B$ such that the answers to I and I' are equal. A *Turing reduction* is a polynomial time algorithm that solves $\#A$ using $\#B$ as an oracle. Completeness in $\#\text{P}$ is then defined in the standard way. This paper and all the papers we cite predominantly use Turing reductions.

A complexity classification of counting CSPs of the form $\#\text{CSP}(\mathcal{H})$ was obtained by Bulatov [3] and was further improved and simplified by Dyer and Richerby [14]. Bulatov's proof makes heavy use of techniques of universal algebra. Dyer and Richerby's proof, on the other hand, uses combinatorial and structural properties of relational structures. The more general version of the counting CSP, the weighted CSP, has also been thoroughly studied. Cai and Chen [10] obtained a complexity classification for weighted CSP, where each homomorphism has a complex weight. One of the main features of counting with complex weights is the phenomenon of cancellation, when complex weights of homomorphisms cancel each other rather than add up. This, of course, never happens in exact unweighted counting problems, but is frequently encountered in modular counting.

Modular Counting. Another natural variation of counting problems is counting modulo some integer. In this paper we consider the problem of computing the number of solutions of a CSP modulo a prime number p . If a relational structure \mathcal{H} is fixed, this problem is denoted $\#_p\text{CSP}(\mathcal{H})$. More precisely, in $\#_p\text{CSP}(\mathcal{H})$ the objective is, given a relational structure \mathcal{G} , to find the number of homomorphisms from \mathcal{G} to \mathcal{H} modulo p .

There are several complexity classes related to modular counting. The more established type of classes is Mod_kP , the class of problems of deciding whether the number of accepting paths of a polynomial time nondeterministic Turing machine is *not* divisible by k , [11, 25].

In particular, if $k = 2$ then $\text{Mod}_k P$ is the class $\oplus P$. However, problems of counting accepting paths naturally belong to classes of the form $\#_k P$, introduced by Faben in [15] that contain problems of counting accepting paths modulo k . The standard notion of reduction is again Turing reduction. Faben in [15] studied the basic properties of such classes, in particular, he identified several $\#_k P$ -complete problems.

In the case of the CSP, the research has mostly been focused on graph homomorphisms. The only exceptions we are aware of are a result of Faben [15], who characterized the complexity of counting the solutions of a Generalized Satisfiability problem modulo an integer, and a generalization of [15] to problems with weights by Guo et al. [21]. The study of modular counting of graph homomorphisms has been much more vibrant.

Before discussing the existing results on modular counting and the results of this study, we need to mention some features of the automorphism group of a relational structure. The automorphisms of a relational structure \mathcal{H} form a group with respect to composition denoted $\text{Aut}(\mathcal{H})$. The order of an automorphism $\pi \in \text{Aut}(\mathcal{H})$ is the smallest number k such that π^k is the identity permutation. An element $a \in H$ is a fixed point of $\pi \in \text{Aut}(\mathcal{H})$ if $\pi(a) = a$. The set of all fixed points of π is denoted by $\text{Fix}(\pi)$.

A systematic study of counting homomorphisms in graphs was initiated by Faben and Jerrum in [15]. They observed that the automorphism group $\text{Aut}(\mathcal{H})$, particularly the automorphisms of order p , plays a crucial role in the complexity of the problem $\#_p \text{Hom}(\mathcal{H})$. This insight extends to relational structures, as discussed in [9]. Specifically, for a homomorphism φ from a relational structure \mathcal{G} to \mathcal{H} , composing φ with an automorphism from $\text{Aut}(\mathcal{H})$ yields another homomorphism from \mathcal{G} to \mathcal{H} . Thus, any automorphism of \mathcal{H} acts on the set $\text{Hom}(\mathcal{G}, \mathcal{H})$ of all homomorphisms from \mathcal{G} to \mathcal{H} . If $\text{Aut}(\mathcal{H})$ contains an automorphism π of order p , the size of the orbit of φ is divisible by p unless $\pi \circ \varphi = \varphi$, making this orbit contribute 0 modulo p to the total homomorphism count from \mathcal{G} to \mathcal{H} . If $\pi \circ \varphi = \varphi$, the range of φ lies within the set of fixed points $\text{Fix}(\pi)$ of π . This observation motivates the following construction: let \mathcal{H}^π denote the substructure of \mathcal{H} induced by $\text{Fix}(\pi)$. We write $\mathcal{H} \rightarrow_p \mathcal{H}'$ there exists $\pi \in \text{Aut}(\mathcal{H})$ such that \mathcal{H}' is isomorphic to \mathcal{H}^π . Furthermore, we write $\mathcal{H} \rightarrow_p^* \mathcal{H}'$ if there exist structures $\mathcal{H}_1, \dots, \mathcal{H}_k$ such that \mathcal{H} is isomorphic to \mathcal{H}_1 , \mathcal{H}' is isomorphic to \mathcal{H}_k , and $\mathcal{H}_1 \rightarrow_p \mathcal{H}_2 \rightarrow_p \dots \rightarrow_p \mathcal{H}_k$.

Relational structures without order p automorphisms will be called *p-rigid*.

► **Lemma 1** ([9, 16]). *Let \mathcal{H} be a relational structure and p a prime. Then*

- (a) *Up to an isomorphism there exists a unique p -rigid structure \mathcal{H}^{*p} such that $\mathcal{H} \rightarrow_p^* \mathcal{H}^{*p}$.*
- (b) *For any relational structure \mathcal{G} it holds that $\text{hom}(\mathcal{G}, \mathcal{H}) \equiv \text{hom}(\mathcal{G}, \mathcal{H}^{*p}) \pmod{p}$.*

By Lemma 1 it suffices to determine the complexity of $\#_p \text{CSP}(\mathcal{H})$ for p -rigid structures \mathcal{H} .

The existing results on modular counting. As we mentioned before, the research in modular counting CSPs has mostly been aimed at counting graph homomorphisms. The complexity of the problem $\#_p \text{Hom}(H)$ of counting homomorphism to a fixed graph H modulo a prime number p has received significant attention in the last ten years. Faben and Jerrum in [16] posed a conjecture that up to order p automorphism reduction \rightarrow_p the complexity of this problem is the same as that for exact counting. More precisely, they conjectured that $\#_p \text{Hom}(H)$ is solvable in polynomial time if and only if $\text{Hom}(H^{*p})$ is. By the result of Dyer and Greenhill [13] $\#_p \text{Hom}(H)$ is solvable in polynomial time if and only if every connected component of H^{*p} is a complete graph with all loops present or a complete bipartite graph. Therefore, proving that if a p -rigid H does not satisfy these conditions then $\#_p \text{Hom}(H)$ is $\#_p P$ -hard suffices to confirm the conjecture of Faben and Jerrum. Over several years the hardness of $\#_p \text{Hom}(H)$ was established for various graph classes [16, 22, 19, 20, 28, 18, 31]. Finally, it was proved for arbitrary graphs in [9].

► **Theorem 2** ([9]). *For any prime p and any graph H the problem $\#_p\text{Hom}(H)$ is solvable in polynomial time if and only if $\text{Hom}(H^{*p})$ is solvable in polynomial time. Otherwise it is $\#_pP$ -complete.*

Our Results. In this paper we begin a systematic study of the problem $\#_p\text{CSP}(\mathcal{H})$ for general relational structures \mathcal{H} . Note that to the best of our knowledge, it is the first paper attempting at such a general modular counting problem. The ultimate goal is to obtain a complexity classification similar to Theorem 2 for arbitrary relational structures. The full version of the paper can be found in [29].

The contribution of the paper is twofold. First, we analyse the existing techniques such as those from [9], and the methods used in exact counting [3, 14, 10], and their applicability to the general case. We conclude that few of them work. More specifically, Theorem 2 asserts that the complexity of modular counting for p -rigid graphs is the same as of exact counting. We, however, suggest a relational structure, a digraph \mathcal{T}_p , that is p -rigid, its exact counting problem is hard, but modular counting is easy, see Example 18. Another important ingredient of the proof of Theorem 2 is a structural theorem on automorphisms of products of graphs [24]. No such result exists for products of relational structures. Moreover, in Example 18 in Section 6 we suggest an example (again, a digraph) showing that nothing similar to such a structural result can be true. Some of the standard techniques in counting CSPs involve properties of relations and relational structures such as rectangularity, permutability, balancedness, the presence of a Mal'tsev polymorphism. In the case of exact counting these concepts are closely related to each other and make efficient algorithms possible. Unfortunately, these concepts are of little help for modular counting. We introduce their modular equivalents, but then a series of examples show that no tight connections are possible in this case. This makes algorithm design very difficult.

On the positive side, we obtain some results to somewhat remedy the situation. The first step is to convert the problem into a richer framework of multi-sorted relational structures and CSPs. The main idea is, given a CSP instance \mathcal{G} , to try to identify the possible images of each vertex of \mathcal{G} , and then treat vertices with different ranges as having different types and map them to different disjoint domains. In Section 4 we call this process refinement and propose two types of refinement, one is based on local propagation techniques, and the other on solving the decision version of the problem. The main benefit of using multi-sorted structures and CSPs is the richer structure of their automorphisms. This often allows stronger reductions than single-sorted structures do. In particular, if the digraph \mathcal{T}_p mentioned above is subjected to this process, it results in a multi-sorted structure that is no longer p -rigid, the corresponding reduced structure is very simple and can easily be solved. We are not aware of any examples of a structure whose multi-sorted refinement is p -rigid, but that would give rise to an easy modular counting problem.

In the next line of research we follow the approach of [9] to expand the relational structure \mathcal{H} by adding relations to \mathcal{H} that are primitive positive (pp-)definable in \mathcal{H} , that is, relations that can be derived from the relations of \mathcal{H} using equality, conjunction and existential quantifiers. However, expanding the general relational structure by pp-definable relations does not work as well as for graphs. To overcome this obstacle, we introduce a new form of expansion which uses p -modular quantifiers instead of regular existential quantifiers. The semantics of a p -modular quantifier is “there are non-zero modulo p values of a variable” rather than “there exists at least one value of a variable” as the regular existential quantifier asserts. Every relational structure is associated with a *relational clone* $\langle \mathcal{H} \rangle$ that consists of all relations pp-definable in \mathcal{H} . The new concept gives rise to new definitions of pp-formulas and

relational clones. If regular existential quantifiers in pp-formulas are replaced with p -modular quantifiers, we obtain p -modular primitive positive formulas (p -mpp-formulas, for short). Then, similar to pp-definitions, a relation \mathcal{R} is said to be p -mpp-definable in a structure \mathcal{H} if there is a p -mpp-formula in \mathcal{H} expressing \mathcal{R} . The p -modular clone $\langle \mathcal{H} \rangle_p$ associated with \mathcal{H} is the set of all relations p -mpp-definable in \mathcal{H} . We show in Theorem 10 (see also Theorem 7) that, similar to the result of Bulatov and Dalmau [6], expanding a structure by a p -mpp-definable relation does not change the complexity of the problem $\#_p\text{CSP}(\mathcal{H})$.

► **Theorem 3.** *Let p be a prime number and \mathcal{H} a p -rigid relational structure. If \mathcal{R} is p -mpp-definable in \mathcal{H} , then $\#_p\text{CSP}(\mathcal{H} + \mathcal{R})$ is polynomial time reducible to $\#_p\text{CSP}(\mathcal{H})$.*

In the remaining part of the paper we identify a number of conditions under which it is possible to design an algorithm or to prove the hardness of the problem. One such case is $\#_2\text{CSP}(\mathcal{H})$ when \mathcal{H} satisfies both 2-rectangularity and the usual strong rectangularity conditions (or, equivalently, has a Mal'tsev polymorphism). It starts with applying the known techniques [5, 14] to find a concise representation or a frame of the set of solutions of a given CSP. However, such a representation cannot be used directly to find the parity of the number of solutions. The algorithm performs multiple recomputations of the frame to exclude the parts that produce an even number of solutions. Unfortunately, this algorithm does not generalize to larger p even under very strong assumptions, because the structure of finite fields start playing a role.

While studying the structure of automorphisms of products of relational structures such as $\text{Aut}(\mathcal{H}^n)$ may be a difficult problem, in Section 7 we make a step forward by reducing the class of structures \mathcal{H} for which such structural results are required. More precisely, for any relational structure $\mathcal{H} = (H; \mathcal{R}_1, \dots, \mathcal{R}_k)$ we construct its binarization $b(\mathcal{H})$ whose relations are binary and rectangular. This makes such structures somewhat closer to graphs and the hope is that it will be easier to study the structure of $\text{Aut}(b(\mathcal{H})^n)$ than $\text{Aut}(\mathcal{H}^n)$ itself. We prove that \mathcal{H} and $b(\mathcal{H})$ share many important properties.

► **Theorem 4.** *Let \mathcal{H} be a relational structure. Then \mathcal{H} is strongly rectangular (p -strongly rectangular, p -rigid, has a Mal'tsev polymorphism) if and only if $b(\mathcal{H})$ is strongly rectangular (p -strongly rectangular, p -rigid, has a Mal'tsev polymorphism).*

2 Preliminaries

Let $[n]$ denote the set $\{1, 2, \dots, n\}$. Let H^n be the Cartesian product of the set H with itself n times and $H_1 \times \dots \times H_n$ the Cartesian product of sets H_1, \dots, H_n . We denote the members of H^n and $H_1 \times \dots \times H_n$ using bold font, $\mathbf{a} \in H^n$, $\mathbf{a} \in H_1 \times \dots \times H_n$. The i -th element of \mathbf{a} is denoted by $\mathbf{a}[i]$ or \mathbf{a}_i .

For $I = \{i_1, \dots, i_k\} \subseteq [n]$ we use $\text{pr}_I \mathbf{a}$ to denote $(\mathbf{a}_{i_1}, \dots, \mathbf{a}_{i_k})$; and for $\mathcal{R} \subseteq H_1 \times \dots \times H_n$ we use $\text{pr}_I \mathcal{R}$ to denote $\{\text{pr}_I \mathbf{a} \mid \mathbf{a} \in \mathcal{R}\}$. For $\mathbf{a} \in H^s$ by $\#\text{ext}_{\mathcal{R}}(\mathbf{a})$ we denote the number of assignments $\mathbf{b} \in H^{n-s}$ such that $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}$. (Note that the order of elements in \mathbf{a} and \mathbf{b} and \mathcal{R} might differ, hence we slightly abuse the notation here.) We denote the number of assignments mod p by $\#_p \text{ext}_{\mathcal{R}}(\mathbf{a})$. Moreover, $\text{pr}_I^p \mathcal{R}$ denotes the set $\{\text{pr}_I \mathbf{a} \mid \mathbf{a} \in \mathcal{R} \text{ and } \#_p \text{ext}_{\mathcal{R}}(\text{pr}_I \mathbf{a}) \neq 0\}$. Often, we treat relations $\mathcal{R} \subseteq H_1 \times \dots \times H_n$ as predicates $\mathcal{R} : H_1 \times \dots \times H_n \rightarrow \{0, 1\}$.

2.1 Multi-Sorted Sets and Relational Structures

We begin with introducing *multi-sorted* or *typed* sets. Let $H = \{H_i\}_{i \in [k]} = \{H_1, \dots, H_k\}$ be a collection of sets. We will assume that the sets H_1, \dots, H_k are disjoint.

The cardinality of a multi-sorted set H equals $|H| = \sum_{i \in [k]} |H_i|$. A mapping φ between two multi-sorted sets $G = \{G_i\}_{i \in [k]}$ and $H = \{H_i\}_{i \in [k]}$ is defined as a collection of mappings $\varphi = \{\varphi_i\}_{i \in [k]}$, where $\varphi_i : G_i \rightarrow H_i$, that is, φ_i maps elements of the sort i in G to elements of the sort i in H . A mapping $\varphi = \{\varphi_i\}_{i \in [k]}$ from $\{G_i\}_{i \in [k]}$ to $\{H_i\}_{i \in [k]}$ is injective (bijective), if for all $i \in [k]$, φ_i is injective (bijective).

A *multi-sorted relational signature* σ over a set of types $\{1, \dots, k\}$ is a collection of *relational symbols*, a symbol $\mathcal{R} \in \sigma$ is assigned a positive integer $\ell_{\mathcal{R}}$, the *arity* of the symbol, and a tuple $(i_1, \dots, i_{\ell_{\mathcal{R}}}) \subseteq [k]^{\ell_{\mathcal{R}}}$, the *type* of the symbol. A *multi-sorted relational structure* \mathcal{H} with signature σ is a multi-sorted set $\{H_i\}_{i \in [k]}$ and an *interpretation* $\mathcal{R}^{\mathcal{H}}$ of each $\mathcal{R} \in \sigma$, where $\mathcal{R}^{\mathcal{H}}$ is a relation or a predicate on $H_{i_1} \times \dots \times H_{i_{\ell_{\mathcal{R}}}}$. The multi-sorted structure \mathcal{H} is finite if H and σ are finite. All structures in this paper are finite. The set H is said to be the *base set* or the *universe* of \mathcal{H} . For the base set we will use the same letter as for the structure, only in the regular font. Multi-sorted structures with the same signature and type are called *similar*.

Let \mathcal{G}, \mathcal{H} be similar multi-sorted structures with signature σ . A *homomorphism* φ from \mathcal{G} to \mathcal{H} is a collection of mappings $\varphi_i : G_i \rightarrow H_i$, $i \in [k]$, from G to H such that for any $\mathcal{R} \in \sigma$ with type $(i_1, \dots, i_{\ell_{\mathcal{R}}})$, if $\mathcal{R}^{\mathcal{G}}(a_1, \dots, a_{\ell_{\mathcal{R}}})$ is true for $(a_1, \dots, a_{\ell_{\mathcal{R}}}) \in G_{i_1} \times \dots \times G_{i_{\ell_{\mathcal{R}}}}$, then $\mathcal{R}^{\mathcal{H}}(\varphi_{i_1}(a_1), \dots, \varphi_{i_{\ell_{\mathcal{R}}}}(a_{\ell_{\mathcal{R}}}))$ is true as well. The set of all homomorphisms from \mathcal{G} to \mathcal{H} is denoted $\text{Hom}(\mathcal{G}, \mathcal{H})$. The cardinality of $\text{Hom}(\mathcal{G}, \mathcal{H})$ is denoted by $\text{hom}(\mathcal{G}, \mathcal{H})$. For a multi-sorted structure \mathcal{H} , the corresponding *counting CSP*, $\#_p\text{CSP}(\mathcal{H})$, is the problem of computing $\text{hom}(\mathcal{G}, \mathcal{H})$ modulo a prime number p for a given structure \mathcal{G} . A homomorphism φ is an *isomorphism* if it is bijective and the inverse mapping φ^{-1} is a homomorphism from \mathcal{H} to \mathcal{G} . If \mathcal{H} and \mathcal{G} are isomorphic, we write $\mathcal{H} \cong \mathcal{G}$. A homomorphism of a structure to itself is called an *endomorphism*, and an isomorphism to itself is called an *automorphism*. As is easily seen, automorphisms of a structure \mathcal{H} form a group denoted $\text{Aut}(\mathcal{H})$.

The *direct product* of multi-sorted σ -structures \mathcal{H}, \mathcal{G} , denoted $\mathcal{H} \times \mathcal{G}$ is the multi-sorted σ -structure with the base set $H \times G = \{H_i \times G_i\}_{i \in [k]}$, the interpretation of $\mathcal{R} \in \sigma$ is given by $\mathcal{R}^{\mathcal{H} \times \mathcal{G}}((a_1, b_1), \dots, (a_{\ell_{\mathcal{R}}}, b_{\ell_{\mathcal{R}}})) = 1$ if and only if $\mathcal{R}^{\mathcal{H}}(a_1, \dots, a_{\ell_{\mathcal{R}}}) = 1$ and $\mathcal{R}^{\mathcal{G}}(b_1, \dots, b_{\ell_{\mathcal{R}}}) = 1$. By \mathcal{H}^{ℓ} we will denote the ℓ th power of \mathcal{H} , that is, the direct product of ℓ copies of \mathcal{H} .

For a prime number p we say that π has order p if π is not the identity in $\text{Aut}(\mathcal{H})$ and has order p in $\text{Aut}(\mathcal{H})$. In other words, each of the π_j 's is either the identity mapping or has order p , and at least one of the π_j 's is not the identity mapping. Structure \mathcal{H} is said to be *p-rigid* if it has no automorphism of order p . Similar to regular relational structures we can introduce reductions of multi-sorted structures by their automorphisms of order p .

A *substructure* \mathcal{H}' of \mathcal{H} induced by a collection of sets $\{H'_i\}_{i \in [k]}$, where $H'_i \subseteq H_i$ is the relational structure given by $(\{H'_i\}_{i \in [k]}; \mathcal{R}'_1, \dots, \mathcal{R}'_m)$, where $\mathcal{R}'_j = \mathcal{R}_j \cap (H'_{i_1} \times \dots \times H'_{i_{\ell}})$ and (i_1, \dots, i_{ℓ}) is the type of \mathcal{R}_j . By $\text{Fix}(\pi)$ we denote the collection $\{\text{Fix}(\pi_i)\}_{i \in [k]}$ of sets of fixed points of the π_i 's. Let \mathcal{H}^{π} denote the substructure of \mathcal{H} induced by $\text{Fix}(\pi)$. We write $\mathcal{H} \rightarrow_p \mathcal{H}'$ if there is $\pi \in \text{Aut}(\mathcal{H})$ of order p such that \mathcal{H}' is isomorphic to \mathcal{H}^{π} . We also write $\mathcal{H} \rightarrow_p^* \mathcal{H}'$ if there are structures $\mathcal{H}_1, \dots, \mathcal{H}_t$ such that \mathcal{H} is isomorphic to \mathcal{H}_1 , \mathcal{H}' is isomorphic to \mathcal{H}_t , and $\mathcal{H}_1 \rightarrow_p \mathcal{H}_2 \rightarrow_p \dots \rightarrow_p \mathcal{H}_t$. Let also \mathcal{H}^{*p} be a p -rigid structure such that $\mathcal{H} \rightarrow_p^* \mathcal{H}^{*p}$.

► **Proposition 5.** *Let \mathcal{H} be a multi-sorted structure and p a prime. Then up to an isomorphism there exists a unique p -rigid multi-sorted structure \mathcal{H}^{*p} such that $\mathcal{H} \rightarrow_p^* \mathcal{H}^{*p}$. Moreover, for any relational structure \mathcal{G} it holds $\text{hom}(\mathcal{G}, \mathcal{H}) \equiv \text{hom}(\mathcal{G}, \mathcal{H}^{*p}) \pmod{p}$.*

The proof of Proposition 5 follows the same lines as the analogous statement in the single-sorted case.

We complete this section with a definition of polymorphisms. Let $\mathcal{R} \subseteq H^n$ be a relation over a set H . A k -ary *polymorphism* of \mathcal{R} is a mapping $f : H^k \rightarrow H$ such that for any choice of $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathcal{R}$, it holds that $f(\mathbf{a}_1, \dots, \mathbf{a}_k) \in \mathcal{R}$ (computed component-wise). The mapping f is a polymorphism of a (single-sorted) relational structure $\mathcal{H} = (H, \mathcal{R}_1, \dots, \mathcal{R}_m)$ if it is a polymorphism of each relation $\mathcal{R}_1, \dots, \mathcal{R}_m$. In the multi-sorted case the definitions are a bit more complicated. Let $\mathcal{R} \subseteq H_1 \times \dots \times H_n$ be a multi-sorted relation. Instead of a single mapping we consider a family of mappings $f = \{f_i\}_{i \in [n]}$, $f_i : H_i^k \rightarrow H_i$. The family f is said to be a polymorphism of \mathcal{R} if it satisfies the same condition: for any choice of $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathcal{R}$, it holds that $f(\mathbf{a}_1, \dots, \mathbf{a}_k) \in \mathcal{R}$, only in this case the mapping f_i is applied in the i th coordinate position, $i \in [n]$. A polymorphism of a multi-sorted structure $\mathcal{H} = (\{H_i\}_{i \in [q]}; \mathcal{R}_1, \dots, \mathcal{R}_m)$ is again a family $f = \{f_i\}_{i \in [q]}$: $H_i^k \rightarrow H_i$ such that for each $j \in [m]$, f (or rather its appropriate subfamily) is a polymorphism of \mathcal{R}_j . For a complete introduction into polymorphisms the reader is referred to [1].

The following special type of polymorphisms often occurs in the study of counting CSPs. For a set H a mapping $\varphi : H^3 \rightarrow H$ is said to be a *Mal'tsev operation* if for any $a, b \in H$ it satisfies the conditions $\varphi(a, a, b) = \varphi(b, a, a) = b$. In the multi-sorted case a family $f = \{f_i\}_{i \in [n]}$ is Mal'tsev, if every f_i is Mal'tsev. A Mal'tsev operation (or a family of operations) that is a polymorphism of a relation \mathcal{R} or a relational structure \mathcal{H} is said to be a Mal'tsev polymorphism of \mathcal{R} or \mathcal{H} .

2.2 Expansion of relational structures

One of the standard techniques when studying constraint problems is to identify ways to expand the target relational structure or constraint language with additional relations without changing the complexity of the problem.

Let \mathcal{H} be a relational structure with signature σ and $\mathcal{H}^=$ its expansion by adding a binary relational symbol $=$ interpreted as $=_H$, the equality relation on H . The following reduction is straightforward.

► **Lemma 6** ([9]). *For any relational structure \mathcal{H} and any prime p , $\#_p\text{CSP}(\mathcal{H}^=) \leq_T \#_p\text{CSP}(\mathcal{H})$.*

A constant relation over a set H is a unary relation $C_a = \{a\}$, $a \in H$. For a relational structure \mathcal{H} by \mathcal{H}^c we denote the expansion of \mathcal{H} by all the constant relations C_a , $a \in H$. Theorem 7 was proved for exact counting in [6], for modular counting of graph homomorphisms in [16, 19, 20], and for general modular counting CSP in [9].

► **Theorem 7** ([9]). *Let \mathcal{H} be a p -rigid σ -structure. Then $\#_p\text{CSP}(\mathcal{H}^c)$ is polynomial time reducible to $\#_p\text{CSP}(\mathcal{H})$.*

Lemma 6 and Theorem 7 can be generalized to the multi-sorted case. Let $\mathcal{H} = \{\mathcal{H}_i\}_{i \in [k]}$ be a multi-sorted structure with signature σ and $\mathcal{H}^=$ its expansion by adding a family of binary relational symbols $=_{H_i}$ (one for each type) interpreted as the equality relation on H_i , $i \in [k]$.

A constant relation over a set $\{H_i\}_{i \in [k]}$ is a unary relation $C_a = \{a\}$, $a \in H_i, i \in [k]$ (such a predicate can only be applied to variables of type i). For a structure \mathcal{H} by \mathcal{H}^c we denote the expansion of \mathcal{H} by all the constant relations C_a , $a \in H_i, i \in [k]$.

► **Theorem 8.** *Let \mathcal{H} be a multi-sorted relational structure and p prime.*

- (1) $\#_p\text{CSP}(\mathcal{H}^=)$ is Turing reducible to $\#_p\text{CSP}(\mathcal{H})$;
- (2) Let \mathcal{H} be p -rigid. Then $\#_p\text{CSP}(\mathcal{H}^c)$ is Turing reducible to $\#_p\text{CSP}(\mathcal{H})$.

Yet another way to expand a relational structure is by primitive positive definable (pp-definable for short) relations. Primitive-positive definitions have played a major role in the study of the CSP. It has been proved in multiple circumstances that expanding a structure with pp-definable relations does not change the complexity of the corresponding CSP. This has been proved for the decision CSP in [26, 8] and the exact Counting CSP [6]. The reader is referred to [1] for details about pp-definitions and their use in the study of the CSP.

Conjunctive definitions are a special case of pp-definitions that do not use quantifiers. Let \mathcal{H} be a structure with signature σ . A *conjunctive formula* Φ over variables $\{x_1, \dots, x_k\}$ is a conjunction of atomic formulas of the form $\mathcal{R}(y_1, \dots, y_\ell)$, where $\mathcal{R} \in \sigma$ is an (ℓ -ary) symbol and $y_1, \dots, y_\ell \in \{x_1, \dots, x_k\}$. A k -ary predicate \mathcal{Q} is conjunctive definable by Φ if $(a_1, \dots, a_k) \in \mathcal{Q}$ if and only if $\Phi(a_1, \dots, a_k)$ is true.

► **Lemma 9** ([9]). *Let \mathcal{H} be a relational structure with signature σ , \mathcal{R} be conjunctive definable in \mathcal{H} , and $\mathcal{H} + \mathcal{R}$ denote the expansion of \mathcal{H} by a predicate symbol \mathcal{R} that is interpreted as the relation \mathcal{R} in \mathcal{H} . Then $\#_p \text{CSP}(\mathcal{H} + \mathcal{R}) \leq_T \#_p \text{CSP}(\mathcal{H})$.*

We now extend the concept of primitive-positive definability to the multi-sorted case. Let \mathcal{H} be a multi-sorted relational structure with the base set H . As before *primitive positive* (pp-) formula in \mathcal{H} is a first-order formula $\exists y_1, \dots, y_s \Phi(x_1, \dots, x_k, y_1, \dots, y_s)$, where Φ is a conjunction of atomic formulas of the form $z_1 =_H z_2$ or $\mathcal{R}(z_1, \dots, z_\ell)$, $z_1, \dots, z_\ell \in \{x_1, \dots, x_k, y_1, \dots, y_s\}$, and \mathcal{R} is a predicate of \mathcal{H} . However, every variable in Φ is now assigned a type $\tau(x_i), \tau(y_j)$ in such a way that for every atomic formula $z_1 =_H z_2$ it holds that $\tau(z_1) = \tau(z_2)$, and for any atomic formula $\mathcal{R}(z_1, \dots, z_\ell)$ the sequence $(\tau(z_1), \dots, \tau(z_\ell))$ matches the type of \mathcal{R} . We say that \mathcal{H} *pp-defines* a predicate \mathcal{Q} if there exists a pp-formula such that

$$\mathcal{Q}(x_1, \dots, x_k) = \exists y_1, \dots, y_s \Phi(x_1, \dots, x_k, y_1, \dots, y_s).$$

For $\mathbf{a} \in \mathcal{R}$ by $\# \text{ext}_\Phi(\mathbf{a})$ we denote the number of assignments $\mathbf{b} \in H_{\tau(y_1)} \times \dots \times H_{\tau(y_s)}$ to y_1, \dots, y_s such that $\Phi(\mathbf{a}, \mathbf{b})$ is true. We denote the number of such assignments mod p by $\#_p \text{ext}_\Phi(\mathbf{a})$.

While when \mathcal{H} is a graph it is possible to prove a statement similar to Lemma 9 for pp-definable relations [9], we will later see that it is unlikely to be true for general relational structures.

Finally, for a relational structure \mathcal{H} (single- or multi-sorted) $\langle \mathcal{H} \rangle$ denotes the relational clone of \mathcal{H} , that is, the set of all relations pp-definable in \mathcal{H}

2.3 Modular Expansion of Relational Structures

We follow the approach of [9] by expanding the relational structure \mathcal{H} by adding pp-definable relations, but doing in a manner friendly to modular counting.

We introduce a new form of expansion which is using p -modular quantifiers instead of regular existential quantifiers. The semantics of a p -modular quantifier is “there are non-zero modulo p values of a variable” rather than “there exists at least one value of a variable” as the regular existential quantifier asserts. The new concept gives rise to new definitions of pp-formulas. If regular existential quantifiers in pp-formulas are replaced with p -modular quantifiers, we obtain p -modular primitive positive formulas (p -mpp-formulas, for short). The p -modular quantifier is denoted $\exists^{\equiv p}$, and so p -mpp-formulas have the form

$$\exists^{\equiv p} y_1, \dots, y_{\ell_1} \dots \exists^{\equiv p} y_{\ell_1 + \dots + \ell_{s-1} + 1}, \dots, y_k \Phi(z_1, \dots, z_m).$$

Note the more complicated form of the quantifier prefix: modular quantification is not as robust as the regular one and quantifying variables away in groups or one-by-one may change the result. For example, let $\mathcal{R} = \{(1, 0, 0), (1, 1, 0), (1, 1, 1), (2, 2, 2)\}$ be a relation on $\{0, 1, 2\}$. Then formulas $\exists^{\equiv 3} y \exists^{\equiv 3} z \mathcal{R}(x, y, z)$ and $\exists^{\equiv 3} y, z \mathcal{R}(x, y, z)$ define sets $\{1, 2\}$ and $\{2\}$, respectively.

Every relational structure is associated with a relational clone $\langle \mathcal{H} \rangle$ that consists of all relations pp-definable in \mathcal{H} . Then, similar to pp-definitions, a relation \mathcal{R} is said to be p -mpp-definable in a structure \mathcal{H} if there is a p -mpp-formula in \mathcal{H} expressing \mathcal{R} . The p -modular clone $\langle \mathcal{H} \rangle_p$ associated with \mathcal{H} is the set of all relations p -mpp-definable in \mathcal{H} . Similar to the result of Bulatov and Dalmau [6], expanding a structure by a p -mpp-definable relation does not change the complexity of the problem $\#_p \text{CSP}(\mathcal{H})$.

► **Theorem 10.** *Let \mathcal{H} be a σ -structure, and p a prime. Let \mathcal{R} be a relation that is defined by $R(x_1, \dots, x_k) = \exists^{\equiv p} y_1, \dots, y_\ell \Phi(x_1, \dots, x_k, y)$, then $\#_p \text{CSP}(\mathcal{H} + \mathcal{R})$ is polynomial time reducible to $\#_p \text{CSP}(\mathcal{H})$.*

3 Structural Properties for Counting

3.1 Rectangularity, Permutability, and Friends

The key properties of relational structures heavily exploited in [3, 14, 10] to obtain characterizations of the complexity of exact counting are rectangularity, balancedness, congruence permutability, and the presence of a Mal'tsev polymorphism. Indeed, according to [14] $\# \text{CSP}(\mathcal{H})$ is polynomial time solvable if and only if \mathcal{H} is strongly balanced. However, in order for the solution algorithm to work, it requires a Mal'tsev polymorphism to be applied over and over again to construct a *frame*, that is, a compact representation of the set of solutions, [3, 14].

Using modular pp-definitions, we can modify these properties' definitions accordingly to obtain the properties of strong p -rectangularity, p -balancedness, and p -permutability. Modular-pp-definitions preserve the complexity of modular problems, however, they destroy the connections between the concepts above.

p -Rectangularity. Recall that a binary relation $\mathcal{R} \subseteq H_1 \times H_2$ is said to be *rectangular* if $(a, c), (a, d), (b, c) \in \mathcal{R}$ implies $(b, d) \in \mathcal{R}$ for any $a, b \in H_1, c, d \in H_2$. A relation $\mathcal{R} \subseteq H_1 \times \dots \times H_n$ for $n \geq 2$ is rectangular if for every $I \subsetneq [n]$, the relation R is rectangular when viewed as a binary relation, a subset of $\text{pr}_I \mathcal{R} \times \text{pr}_{[n]-I} \mathcal{R}$. A relational structure \mathcal{H} is *strongly rectangular* if every relation $\mathcal{R} \in \langle \mathcal{H} \rangle$ of arity at least 2 is rectangular. Finally, a relational structure \mathcal{H} is said to be *strongly p -rectangular*, if every $\mathcal{R} \in \langle \mathcal{H} \rangle_p$ is rectangular.

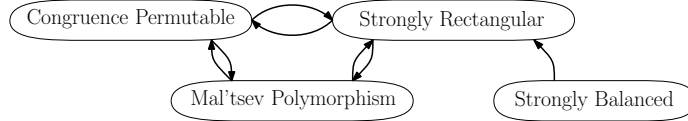
p -Balancedness. An n -by- m matrix M is said to be a *rank-1 block matrix* if by permuting rows and columns it can be transformed to a block-diagonal matrix (not necessarily square), where every nonzero diagonal block has rank at most 1. Note that the rank can also be computed in \mathbb{Z}_p , in which case we use the term *rank-1 block matrix modulo p* .

Let $\mathcal{R}(x, y, z)$ be a ternary relation, a subset of $H_1 \times H_2 \times H_3$. We call \mathcal{R} *balanced* if the matrix $M_{\mathcal{R}} \in \mathbb{Z}^{|H_1| \times |H_2|}$, where $M_{\mathcal{R}}[x, y] = |\{z \in H_3 : (x, y, z) \in \mathcal{R}\}|$ such that $x \in H_1$ and $y \in H_2$ is a rank-1 block matrix. It is *p -balanced* if $M_{\mathcal{R}}$ is a rank-1 block matrix modulo p . A relation \mathcal{R} of arity $n > 3$ is balanced if every representation of \mathcal{R} as a ternary relation, a subset of $H^k \times H^\ell \times H^{n-k-\ell}$, is balanced. Similarly, A relation \mathcal{R} on H of arity $n > 3$ is *p -balanced* if every representation of \mathcal{R} as a ternary relation, a subset of $H^k \times H^\ell \times H^{n-k-\ell}$, is p -balanced.

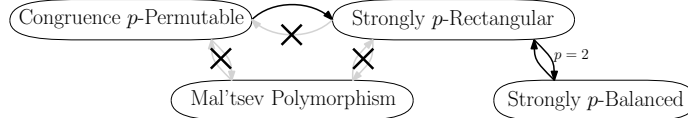
A relational structure \mathcal{H} is called *strongly balanced* if every relation $\mathcal{R} \in \langle \mathcal{H} \rangle$ is balanced. Similarly, a relational structure \mathcal{H} is called *strongly p -balanced* if every relation $\mathcal{R} \in \langle \mathcal{H} \rangle_p$ is p -balanced.

p -Permutability. A *congruence* of a relational structure \mathcal{H} is an equivalence relation θ on H that is pp-definable in \mathcal{H} . More generally, let $\mathcal{R} \in \langle \mathcal{H} \rangle$ be a k -ary relation. A congruence of \mathcal{R} is a $2k$ -ary relation pp-definable in \mathcal{H} that is an equivalence relation on \mathcal{R} . We denote the set of all congruences of \mathcal{H} (of \mathcal{R}) by $\text{Con}(\mathcal{H})$ (respectively, by $\text{Con}(\mathcal{R})$). By \circ we denote the product of binary relations: $(a, b) \in \mathcal{R} \circ \mathcal{Q}$ if and only if there is c such that $(a, c) \in \mathcal{R}$ and $(c, b) \in \mathcal{Q}$. We say \mathcal{H} is *congruence permutable* if for all $\alpha, \beta \in \text{Con}(\mathcal{R})$, where $\mathcal{R} \in \langle \mathcal{H} \rangle$, it holds that $\alpha \circ \beta = \beta \circ \alpha$.

Congruence p -permutability is defined as follows: A *p -congruence* of \mathcal{H} or of $\mathcal{R} \in \langle \mathcal{H} \rangle_p$ is an equivalence relation on H or \mathcal{R} , respectively, that is p -mpp-definable in \mathcal{H} . We denote the set of all p -congruences of \mathcal{H} (\mathcal{R}) by $\text{Con}_p(\mathcal{H})$ ($\text{Con}_p(\mathcal{R})$). By \textcircled{p} we denote the product of binary relations: $(a, b) \in \mathcal{R} \textcircled{p} \mathcal{Q}$ if and only if the number of elements c such that $(a, c) \in \mathcal{R}$ and $(c, b) \in \mathcal{Q}$ is not a multiple of p . We say that \mathcal{H} is *congruence p -permutable* if for all $\alpha, \beta \in \text{Con}_p(\mathcal{R})$, where $\mathcal{R} \in \langle \mathcal{H} \rangle_p$, we have $\alpha \textcircled{p} \beta = \beta \textcircled{p} \alpha$. Figure 1 demonstrates the connection between congruence permutability, strong rectangularity, the existence of a Mal'tsev polymorphism, strong balancedness and their modular counterparts. A collection of statements and examples proving these connections or lack thereof will be given in the full version of the paper. Below we give one such example showing that the existence of a Mal'tsev polymorphism does not guarantee 2-rectangularity.



(a) Congruence Permutability, Strong Rectangularity, and Mal'tsev polymorphism are equivalent (See [23, 14, 3]). Also, Strong Balancedness implies Strong Rectangularity(See [14]).



(b) The only connection that is preserved for the modular case is Congruence p -Permutability implies Strong p -rectangularity. Also, Strong 2-rectangularity is equivalence to Strong 2-Balancedness.

■ **Figure 1** The connection between 4 properties is shown above. Figure (1a) shows the connection for the exact counting. Figure (1b) shows the the connection for the modular counterparts.

► **Example 11.** Let $H = \{0, 1, 2\}$, $p = 2$, and $\mathcal{H} = (H; \mathcal{R})$, where \mathcal{R} is the following ternary relation, (triples are written vertically), $\mathcal{R} = \{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 0, 2), (1, 1, 0)\}$. As is easily see, $\exists \equiv^2 z \mathcal{R}(x, y, z)$ is the relation $\{(0, 0), (0, 1), (1, 1)\}$, which is not rectangular, and so, \mathcal{H} is not strongly 2-rectangular. We now show that \mathcal{H} is strongly rectangular by presenting a Mal'tsev polymorphism of \mathcal{H} . Let $f(x, y, z) = x + y + z$, where addition is modulo 2, is an operation on $\{0, 1\}$. For $\mathbf{a} = (a_1, a_2, a_3) \in H^3$, let also $\mathbf{a}' \in \{0, 1\}^3$ denote the triple obtained by replacing 2's with 1's. Then let $g(x, y, z)$ be given by

$$g(\mathbf{a}) = \begin{cases} 2, & \text{if } \mathbf{a} \in \{(2, a, a), (a, a, 2) \mid a \in H\}, \\ f(\mathbf{a}') & \text{otherwise.} \end{cases}$$

It is straightforward that g is a Mal'tsev operation and is a polymorphism of \mathcal{H} .

4 Rigidity and Multi-sorted Structures

We start with applying a well-known framework of multi-sorted relational structures to modular counting. While multi-sorted structures is a standard tool in the study of decision CSPs, it usually only used to simplify arguments and streamline algorithms. Here we use this framework in a more fundamental way, to strengthen the main (hypothetical) tractability condition.

The classification result in Theorem 2 asserts that $\#_p \text{Hom}(H)$ for a p -rigid graph H is hard whenever the exact counting problem is hard. In the following example, we briefly show that this is no longer the case for general relational structures.

► **Example 12.** Let p be a prime number and $T_p = \{0, \dots, p-1, p, p+1\}$. A relational structure \mathcal{T}_p has the base set T_p and predicates $\mathcal{R}, C_0, \dots, C_{p+1}$, where C_a is the constant relation $\{(a)\}$ and $\mathcal{R} = T_p^2 - \{(0, p), \dots, (p-1, p)\}$. The structure \mathcal{T}_p is p -rigid as it contains all the constant relations and every automorphism must preserve them, implying that every element of T_p is a fixed point. By [4, 35] the decision $\text{CSP}(\mathcal{T}_p)$ is solvable in polynomial time, while the exact counting problem $\#\text{CSP}(\mathcal{T}_p)$ is $\#P$ -complete by [6], as it does not have a Mal'tsev polymorphism and \mathcal{R} is not rectangular (see below). However, if \mathcal{G} is a structure similar to \mathcal{T}_p and there is a homomorphism $\varphi : \mathcal{G} \rightarrow \mathcal{T}_p$ such that some vertex v of \mathcal{G} is mapped to $a \in \{0, \dots, p-1\}$ then, unless v is bound by C_a , the mapping that differs from φ only at v by sending it to any $b \in \{0, \dots, p-1\}$ is also a homomorphism. Since $|\{0, \dots, p-1\}| = p$, this means that the elements $0, \dots, p-1$ can be effectively eliminated from \mathcal{T}_p , and the resulting structure is somewhat trivial. Therefore $\#_p \text{CSP}(\mathcal{T}_p)$ can be solved in polynomial time.

We introduce a different concept of rigidity that is stronger than the one used before. In particular, it will explain the tractability of the problem from Example 12.

While Proposition 5 allows one to reduce CSPs over non- p -rigid structures, it is sometimes possible to go further and reduce a CSP to one with a richer structure, and a richer set of automorphisms. Let $\mathcal{H} = (\{H_i\}_{i \in [k]}; \mathcal{R}_1, \dots, \mathcal{R}_m)$ be a multi-sorted relational structure. We say that a structure \mathcal{G} is a *refinement* of \mathcal{H} if it satisfies the following conditions:

- (a) $\mathcal{G} = (\{G_i\}_{i \in [q]}; \mathcal{Q}_1, \dots, \mathcal{Q}_t)$, where the G_i ' are pairwise disjoint,
 - (b) for every $i \in [q]$, there is an injective mapping $\xi_i : G_i \rightarrow H_{i'}$ for some $i' \in [k]$,
 - (c) for every $j \in [t]$ there is $j' \in [m]$ with $\mathcal{R}_{j'} \subseteq H_{i_1} \times \dots \times H_{i_\ell}$ and $\mathcal{Q}_j = \{(a_1, \dots, a_\ell) \in G_{i_1} \times \dots \times G_{i_\ell} \mid (\xi_{i_1}(a_1), \dots, \xi_{i_\ell}(a_\ell)) \in \mathcal{R}_{j'}\}$, where G_{i_r} is such that $\xi_{i_r}(G_{i_r}) \subseteq H_{i_r} \cap \text{pr}_r \mathcal{R}_{j'}$.
- Condition (a) is required because the domains of a multi-sorted structure have to be disjoint. Condition (b) basically says that G_i consists of copies of some elements of $H_{i'}$. Condition (c) amounts to saying that $\mathcal{Q}_j \subseteq \mathcal{R}_{j'}$, except it uses copies of the elements of \mathcal{H} . In the notation of item (c) we use $\xi(a_1, \dots, a_\ell)$ to denote $(\xi_{i_1}(a_1), \dots, \xi_{i_\ell}(a_\ell))$, we use $\xi(\mathcal{Q}_j)$ to denote $\{\xi(a_1, \dots, a_\ell) \mid (a_1, \dots, a_\ell) \in \mathcal{Q}_j\}$, and $\xi^{-1}(\mathcal{R}_{j'})$ for $\{(a_1, \dots, a_\ell) \in G_{i_1} \times \dots \times G_{i_\ell} \mid \xi(a_1, \dots, a_\ell) \in \mathcal{R}_{j'}\}$.

It is possible that while \mathcal{H} is p -rigid, its refinement is not and Proposition 5.

In order to relate refinement structures with reductions between counting problems we introduce two special types of refinement. First of all, we will need an alternative approach to pp-definitions based on homomorphisms, see [17, 30].

► **Lemma 13** ([17, 30]). *A predicate $\mathcal{R}(x_1, \dots, x_k)$ is pp-definable in a multi-sorted structure \mathcal{H} containing the equality predicate if and only if there exists a similar structure $\mathcal{G}_{\mathcal{R}}$ containing vertices x_1, \dots, x_k such that for any $(a_1, \dots, a_k) \in \mathcal{R}$ it holds that $(a_1, \dots, a_k) \in \mathcal{R}$ if and only if there is a homomorphism from $\mathcal{G}_{\mathcal{R}}$ to \mathcal{H} that maps x_i to a_i , $i \in [k]$. We will say that $\mathcal{G}_{\mathcal{R}}$ defines \mathcal{R} in \mathcal{H} .*

Let $\mathcal{H} = (\{H_i\}_{i \in [k]}; \mathcal{R}_1, \dots, \mathcal{R}_m)$ be a multi-sorted relational structure. The *Gaifman graph* of \mathcal{H} is the graph $G(\mathcal{H}) = (V, E)$, where $V = \bigcup_{i \in [k]} H_i$ and $(a, b) \in E$ if and only if there is $j \in [m]$ and $\mathbf{a} \in \mathcal{R}_j$ such that $\mathbf{a}[s] = a$, $\mathbf{a}[t] = b$ for some coordinate positions s, t of \mathcal{R}_j . The structure \mathcal{H} has *treewidth* d if $G(\mathcal{H})$ has treewidth d .

We say that a refinement $\mathcal{G} = (\{G_i\}_{i \in [q]}; \mathcal{Q}_1, \dots, \mathcal{Q}_t)$ of \mathcal{H} is *width d definable* if for every $i \in [q]$ there is a structure \mathcal{G}_i of treewidth at most d that defines $\xi_i(G_i)$ in \mathcal{H} . In a similar way, we say that \mathcal{G} is a *definable refinement* if for every $i \in [q]$ the set $\xi_i(G_i)$ is pp-definable in \mathcal{H} . Finally, we say that \mathcal{G} is the *full width d definable refinement* (respectively, *full definable refinement*) if it satisfies the following conditions.

- (1) It is a width d definable refinement (respectively, a definable refinement).
- (2) For every unary relation U definable in \mathcal{H} by a structure of treewidth at most d (respectively, pp-definable unary relation) there is $i \in [q]$ such that $\xi_i(G_i) = U$.
- (3) For every relation S obtained from some relation \mathcal{R}_j by restricting it to domains definable by structures of width d (respectively by pp-definable domains), there is $\mathcal{Q}_{j'}$ such that $\xi(\mathcal{Q}_{j'}) = S$.

Since the original domains H_i , $i \in [k]$, have trivial pp-definitions, they (or rather their copies) are always among the G_j 's, and copies of the original relations are also among the \mathcal{Q}_j 's, although they may be over different, smaller domains than the \mathcal{R}_i 's.

Next, we extend refinements to CSP instances. Let $\mathcal{G} = (\{G_i\}_{i \in [q]}; \mathcal{Q}_1, \dots, \mathcal{Q}_t)$ be a refinement of \mathcal{H} and $\mathcal{P} = (V, \mathcal{C})$ an instance of $\text{CSP}(\mathcal{H})$. Recall that every variable $v \in V$ is assigned a sort $\sigma(v) \in [k]$. Let $\sigma' : V \rightarrow [q]$ be such that $\xi_{\sigma'(v)} : G_{\sigma'(v)} \rightarrow H_{\sigma(v)}$ for each $v \in V$. The instance $\mathcal{P}^{\sigma'} = (V, \mathcal{C}^{\sigma'})$ is said to be a *refinement for \mathcal{G}* of \mathcal{P} with the sort function σ' if it satisfies the following two conditions

- (a) every $v \in V$ is assigned the sort $\sigma'(v)$;
- (b) for every $C = \langle \mathbf{s}, \mathcal{R} \rangle \in \mathcal{C}$, $\mathbf{s} = (v_1, \dots, v_\ell)$, it holds that $\xi_{\sigma'(v_i)}(G_{\sigma'(v_i)}) \subseteq \text{pr}_i \mathcal{R}$, $i \in [\ell]$, and there is $C' = \langle \mathbf{s}, \xi^{-1}(\mathcal{R}) \rangle \in \mathcal{C}^{\sigma'}$.

The refinement $\mathcal{P}^{\sigma'}$ is *lossless* if for every solution φ of \mathcal{P} the mapping $\xi^{-1} \circ \varphi$ is a solution of $\mathcal{P}^{\sigma'}$. Suppose \mathcal{G} is full pp-definable. The refinement $\mathcal{P}^{\sigma'}$ is *minimal lossless* if it is lossless and for each $v \in V$, $\sigma'(v)$ is minimal (with respect to inclusion of $\xi_{\sigma'(v)}(G_{\sigma'(v)})$ in the original domain). If \mathcal{G} is full of treewidth d , the definition is bit more complicated. The instance \mathcal{P} can also be viewed as a structure \mathcal{F} with vertex set V and such that the solutions of \mathcal{P} are exactly the homomorphisms from \mathcal{F} to \mathcal{H} , see [17]. Then $\mathcal{P}^{\sigma'}$ is *minimal lossless of width d* if it is lossless and for every $v \in V$, $\xi_{\sigma'(v)}(G_{\sigma'(v)})$ is minimal with respect to inclusion among unary relations defined by a structure \mathcal{G}_v of treewidth d with a designated variable x and such that there is a homomorphism from \mathcal{G}_v to \mathcal{F} mapping x to v . In fact, a minimal lossless of width d structure $\mathcal{P}^{\sigma'}$ is produced from \mathcal{P} by applying an appropriate local propagation algorithm [30].

► **Proposition 14.** *Let \mathcal{H} be a multi-sorted relational structure.*

- (1) *Let \mathcal{G} be the full width d definable refinement of \mathcal{H} . For any instance \mathcal{P} of $\text{CSP}(\mathcal{H})$, its minimal lossless refinement of width d for \mathcal{G} can be found in polynomial time.*
- (2) *Let \mathcal{H} be a relational structure containing all the constant relations and such that $\text{CSP}(\mathcal{H})$ is solvable in polynomial time, and \mathcal{G} the full definable refinement of \mathcal{H} . Then for any instance \mathcal{P} of $\text{CSP}(\mathcal{H})$, its minimal lossless refinement for \mathcal{G} can be found in polynomial time.*

5 An Algorithm for Parity

In this section we outline an algorithm that solves $\#_2\text{CSP}(\mathcal{H})$, that is, finds the parity of the number of homomorphisms to \mathcal{H} , provided the structure \mathcal{H} satisfies some additional conditions.

► **Theorem 15.** *Let \mathcal{H} be a 2-rigid, strongly 2-rectangular, and $\langle \mathcal{H} \rangle_2$ has a Mal'tsev polymorphism¹. Then $\#_2\text{CSP}(\mathcal{H})$ can be solved in time polynomial time.*

In order to prove Theorem 15 we apply some of the existing techniques such as compact representations of relations with a Mal'tsev polymorphism, but in a novel way that is very different from its original use.

Frames and Witness Functions. Suppose that \mathcal{R} is an n -ary relation with a Mal'tsev polymorphism φ . For each $i \in [n]$ we define the following relation \sim_i on $\text{pr}_i\mathcal{R}$: $a \sim_i b$ if there exist tuples $\mathbf{x} \in H^{i-1}$ and $\mathbf{y}_a, \mathbf{y}_b \in H^{n-i}$ such that $(\mathbf{x}, a, \mathbf{y}_a) \in \mathcal{R}$ and $(\mathbf{x}, b, \mathbf{y}_b) \in \mathcal{R}$. For the case $i = 1$, we have $a \sim_1 b$ for all $a, b \in \text{pr}_1\mathcal{R}$ because they share the common empty prefix ε . The relations \sim_i will be called *frame relations*. The following observations are straightforward corollaries of the rectangularity of \mathcal{R} and were used in [14].

► **Lemma 16 (Folklore).** *Let \mathcal{R} be a relation with a Mal'tsev polymorphism.*

- (1) \sim_i is an equivalence relation for all $i \in [n]$.
- (2) If $a \sim_i b$ and $\mathbf{x} \in \mathcal{R}$ with $\mathbf{x}_i = a$, then there is a $\mathbf{y} \in \mathcal{R}$ with $\mathbf{y}_i = b$ and $\text{pr}_{[i-1]}\mathbf{x} = \text{pr}_{[i-1]}\mathbf{y}$.

A mapping $\omega : [n] \times H \rightarrow H^n \cup \{\perp\}$ is called a witness function of \mathcal{R} if

- (i) For any $i \in [n]$ and $a \in H - \text{pr}_i\mathcal{R}$, $\omega(i, a) = \perp$;
- (ii) For any $i \in [n]$ and $a \in \text{pr}_i\mathcal{R}$, $\omega(i, a) \in \mathcal{R}$ is a witness for (i, a) , i.e., $\text{pr}_i\omega(i, a) = a$;
- (iii) For any $i \in [n]$ and $a, b \in \text{pr}_i\mathcal{R}$ with $a \sim_i b$, we have $\text{pr}_{[i-1]}\omega(i, a) = \text{pr}_{[i-1]}\omega(i, b)$.

A witness function ω provides a concise representation of \mathcal{R} . Let $F = \{\omega(i, a) \mid i \in [n], a \in \text{pr}_i\mathcal{R}\}$. Such a set of tuples is called a *frame* of \mathcal{R} . A witness function (or a frame) can be found in polynomial time given a conjunctive definition (i.e. a CSP instance) of a relation in a relational structure with a Mal'tsev polymorphism. This is the property that makes them essential for solving CSPs. The following transformations of frames can be performed in polynomial time.

► **Proposition 17 ([5, 14]).** *Let \mathcal{H} be a relational structure with a Mal'tsev polymorphism and a relation \mathcal{R} has a conjunctive definition $\mathcal{R}(x_1, \dots, x_n) = \bigwedge_{i \in [m]} \mathcal{R}_i(x_{i_1}, \dots, x_{i_t})$ in \mathcal{H} .*

- (1) *A frame and a witness function for \mathcal{R} can be computed in $O(mn^4)$.*
- (2) *Let $I \subseteq [n]$. Given a frame F for \mathcal{R} , a frame for $\mathcal{R}(x_1, x_2, \dots, x_n) \wedge (\bigwedge_{s \in I} C_{a_s}(x_s))$ (i.e., x_s is the constant $a_s \in H$, $s \in I$) can be constructed in $O(n^3)$ time.*
- (3) *Given a frame F for \mathcal{R} , a frame for $\mathcal{Q}(x_1, x_2, \dots, x_{n-1}) = \exists y \mathcal{R}(x_1, \dots, x_{n-1}, y)$, can be constructed in $O(n)$ time.*

Overview of the algorithm. The exact counting algorithm in [14] first finds a frame of a conjunctive defined relation (a.k.a. the set of solutions of a CSP instance), and then uses the frame and the condition of balancedness to compute the number of solutions. Unfortunately, this approach does not work in our case, because according to the results of Section 3.1 the property of 2-balancedness that we need here does not correlate well with the existence of a Mal'tsev polymorphism. We therefore choose a different approach.

¹ Note that the latter condition does not follow from \mathcal{H} having a Mal'tsev polymorphism, because 2-mpp-definitions do not preserve polymorphisms.

60:14 Modular Counting CSP

Let \mathcal{H} be a structure satisfying the conditions of Theorem 15 and $\mathcal{R} \in \langle \mathcal{H} \rangle_2$. Since $\langle \mathcal{H} \rangle_2$ has a Mal'tsev polymorphism, a frame for \mathcal{R} can be computed in polynomial time by Proposition 17(1). Suppose that \mathcal{R} is n -ary. It is not hard to see that

$$|\mathcal{R}| \equiv |\exists^{\equiv 2} y \mathcal{R}(x_1, \dots, x_{n-1}, y)| \pmod{2}.$$

Therefore, if it is possible to find a frame of $\tilde{\mathcal{R}} = \exists^{\equiv 2} y \mathcal{R}(x_1, \dots, x_{n-1}, y)$, we could repeat this process $n - 1$ times eventually obtaining a unary relation \mathcal{R}' such that $|\mathcal{R}'| \equiv |\mathcal{R}| \pmod{2}$ and then just count the elements in \mathcal{R}' using its frame. Unfortunately, it is not that straightforward, because Proposition 17(3) does not work for modular quantifiers. Instead, we use a more convoluted method.

Let $\text{PAR}_{\mathcal{R}}(\mathbf{x}, y) = \mathcal{R}(\mathbf{x}, y) \wedge (\exists^{\equiv 2} z \mathcal{R}(\mathbf{x}, z))$. This relation contains essentially the same tuples as $\tilde{\mathcal{R}}$, except it also keeps their extensions to the last coordinate position. This means that $\tilde{\mathcal{R}} = \exists y \text{PAR}_{\mathcal{R}}(\mathbf{x}, y)$, and if we know a frame of $\text{PAR}_{\mathcal{R}}(\mathbf{x}, y)$, a frame of $\tilde{\mathcal{R}}$ can be found by Proposition 17(3). Finding a frame for $\text{PAR}_{\mathcal{R}}(\mathbf{x}, y)$ is the crux of the algorithm.

Finding a frame for $\text{PAR}_{\mathcal{R}}(\mathbf{x}, y)$. Let \sim_i and ω be frame relations and a witness function (a frame) of the relation \mathcal{R} found in the previous step. Let \mathcal{E}_i denote the collection of the equivalence classes of \sim_i , and $\mathcal{E}_i = \{\mathcal{E}_{i,1}, \dots, \mathcal{E}_{i,\ell_i}\}$, $\mathcal{E}_{i,j} \subseteq \text{pr}_i \mathcal{R}$, where $j \in [\ell_i]$. We often refer to these classes as *frame classes*. By \sim'_i, ω' , and \mathcal{E}'_i , $i \in [n]$, we denote yet unknown frame relations, witness function, and frame classes of $\text{PAR}_{\mathcal{R}}$ (clearly, the number of classes in \mathcal{E}'_i may differ from that of \mathcal{E}_i).

First, we observe that by definition a tuple $(\mathbf{x}, a) \in \text{PAR}_{\mathcal{R}}$ if and only if there is a class $\mathcal{E}_{n,s} \in \mathcal{E}_n$, $s \in [\ell_n]$, such that $a \in \mathcal{E}_{n,s}$ and $|\mathcal{E}_{n,s}| \equiv 1 \pmod{2}$. This makes finding \sim'_n and $\omega'(n, *)$ easy, they are just restrictions of \sim_n and $\omega(n, *)$ on the union of odd classes from \mathcal{E}_n . For $k \in [n - 1]$ and $a \in \text{pr}_k \mathcal{R}$ we use Proposition 17(2) to find a witness function $\omega^{k \leftarrow a}$ and frame classes $\mathcal{E}_i^{k \leftarrow a}$ of $\mathcal{R}(x_1, x_2, \dots, x_n) \wedge C_a(x_k)$. We examine $\mathcal{E}_{n,s}^{k \leftarrow a}$ for $s \in [|\mathcal{E}_n^{k \leftarrow a}|]$. We check whether there exists $s \in [|\mathcal{E}_n^{k \leftarrow a}|]$ such that $|\mathcal{E}_{n,s}^{k \leftarrow a}| \equiv 1 \pmod{2}$. If we find such an s , we select $b \in \mathcal{E}_{n,s}^{k \leftarrow a}$ and set $\omega'(k, a) = \omega^{k \leftarrow a}(n, b)$. By construction $\text{pr}_k \omega'(k, a) = \text{pr}_k \omega^{k \leftarrow a}(n, b) = a$, and by the observation above $\omega'(k, a) \in \text{PAR}_{\mathcal{R}}$. Finally, in order to check whether for some $b \in \text{pr}_k \mathcal{R}$ it holds that $a \sim'_k b$ it suffices to complete the following steps. Use Proposition 17(2) to compute a witness function ω'' and frame classes of

$$\mathcal{R}(x_1, \dots, x_n) \wedge C_b(x_k) \wedge \left(\bigwedge_{s=1}^{k-1} C_{d_s}(x_s) \right),$$

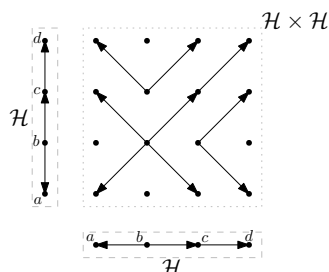
where $(d_1, \dots, d_n) = \omega^{k \leftarrow a}(k, a)$. It can be shown that $a \sim'_k b$ if and only if there exists $t \in [|\mathcal{E}_n''|]$ such that $|\mathcal{E}_{n,t}''| \equiv 1 \pmod{2}$.

This completes the outline of the algorithm.

6 Hardness and Automorphisms of Direct Products of Structures

Another crucial component of Theorem 2 is the structure of automorphisms of direct products of graphs [24]. It essentially asserts that every automorphism of a direct product $H_1 \times \dots \times H_n$ can be thought of as a composition of a permutation of factors in the product and automorphisms of those factors. In the following example we show that this breaks down already for digraphs.

► **Example 18.** let $\mathcal{H} = (V, E)$ be a directed graph where $V = \{a, b, c, d\}$ with (directed) edge set $E = \{(b, a), (b, c), (c, d)\}$. This digraph is rigid. However, the automorphism group of \mathcal{H}^2 , see Figure 2, has a complicated structure. As is easily seen \mathcal{H}^2 has a large number of automorphisms of order 2 and 3, not all of which have a simple representation mentioned above.



■ **Figure 2** The structure of \mathcal{H} and \mathcal{H}^2 .

In [9] one of the important applications of the structural theorem for automorphisms of graph products is that it allows one to prove that $\#_p \text{Hom}(\mathcal{H} + \mathcal{R})$, where $\mathcal{H} + \mathcal{R}$ denotes the expansion of \mathcal{H} by a relation \mathcal{R} pp-definable in \mathcal{H} , is polynomial time reducible to $\#_p \text{Hom}(\mathcal{H})$. The example above indicates that this result may no longer be true for general relational structures.

Next, we explore what implications of a structural result similar to that in [24] that is used in [9] about $\text{Aut}(\mathcal{H}^n)$ can be. A rectangularity obstruction is a violation of the rectangularity or p -rectangularity property, that is, a (n -ary) relation \mathcal{R} pp- or p -mpp-definable in a structure \mathcal{H} , $k \in [n]$, and tuples $\mathbf{a}, \mathbf{b} \in \text{pr}_{[k]}\mathcal{R}$, $\mathbf{c}, \mathbf{d} \in \text{pr}_{[n]-[k]}\mathcal{R}$ such that $(\mathbf{a}, \mathbf{c}), (\mathbf{a}, \mathbf{d}), (\mathbf{b}, \mathbf{d}) \in \mathcal{R}$, but $(\mathbf{b}, \mathbf{c}) \notin \mathcal{R}$. A *generalized rectangularity obstruction* are the relation \mathcal{R} and sets $A_{1,1}, A_{1,2} \subseteq \text{pr}_{[k]}\mathcal{R}$, $A_{2,1}, A_{2,2} \subseteq \text{pr}_{[n]-[k]}\mathcal{R}$ such that $A_{1,1} \cap A_{1,2} = \emptyset$, $A_{2,1} \cap A_{2,2} = \emptyset$, and any $\mathbf{a} \in A_{1,1}, \mathbf{b} \in A_{1,2}, \mathbf{c} \in A_{2,1}, \mathbf{d} \in A_{2,2}$ form a rectangularity obstruction.

At the first glance, if such an obstruction exists, it should be possible to prove the hardness of $\#_p \text{CSP}(\mathcal{H})$. Indeed, \mathcal{R} can be viewed as a bipartite graph $K_{\mathcal{R}}$, whose parts of the bipartition are $\text{pr}_{[k]}\mathcal{R}$ and $\text{pr}_{[n]-[k]}\mathcal{R}$, and as the rectangularity obstruction shows, this graph is not complete bipartite implying that $\# \text{Hom}(K_{\mathcal{R}})$ is $\#P$ -complete. However, the hardness of $\#_p \text{Hom}(K_{\mathcal{R}})$ also involves the requirement that $K_{\mathcal{R}}$ is p -rigid. p -rigidity is achieved by restricting the problem to induced subgraphs of $K_{\mathcal{R}}$ as in Lemma 1. However, such a subgraph may avoid the (generalized) rectangularity obstruction rendering it useless.

The obstruction $A_{1,1}, A_{1,2} \subseteq \text{pr}_{[k]}\mathcal{R}$, $A_{2,1}, A_{2,2} \subseteq \text{pr}_{[n]-[k]}\mathcal{R}$ is said to be *protected* in \mathcal{R} if, after applying a p -reduction to \mathcal{R} under a sequence of p -automorphisms from $\text{Aut}(\mathcal{R})$, for the resulting relation $\tilde{\mathcal{R}}$ it holds that $\text{pr}_{[k]}\tilde{\mathcal{R}} \cap A_{1,1}, \text{pr}_{[k]}\tilde{\mathcal{R}} \cap A_{1,2} \neq \emptyset$, $\text{pr}_{[n]-[k]}\tilde{\mathcal{R}} \cap A_{2,1}, \text{pr}_{[n]-[k]}\tilde{\mathcal{R}} \cap A_{2,2} \neq \emptyset$. In fact, Theorem 4.2 [9] implies, although implicitly, that any p -rigid graph that is not a complete bipartite graph contains a protected rectangularity obstruction. One case of a protected rectangularity obstruction is when it is protected in $K_{\mathcal{R}}$, that is, survives p -reductions of $K_{\mathcal{R}}$ itself. In this case we say that the obstruction is *graph-protected*.

► **Proposition 19.** *Let \mathcal{H} be a (multi-sorted) relational structure and p a prime number. If \mathcal{H} has a graph protected generalized rectangularity obstruction modulo p , $\#_p \text{CSP}(\mathcal{H})$ is $\#_p P$ -complete.*

We consider a special case of graph-protected generalized rectangularity obstructions, standard hardness gadgets, that have to satisfy the additional condition $A_{1,1} \cup A_{1,2} = \text{pr}_{[k]}\mathcal{R}$, $A_{2,1} \cup A_{2,2} = \text{pr}_{[n]-[k]}\mathcal{R}$. It can be proved that a standard hardness gadget is indeed a graph-protected obstruction.

Standard hardness gadgets provide a fairly limited condition for the hardness of $\#_p\text{CSP}(\mathcal{H})$. In fact, it is possible to prove that $\#_p\text{CSP}(\mathcal{H})$ is $\#_pP$ -complete whenever \mathcal{H} has any protected rectangularity obstruction, not necessarily a standard gadget. However, it cannot be done using Theorem 2 as a black box, and is outside of the scope of this paper.

7 Binarization

While studying the structure of $\text{Aut}(\mathcal{H}^k)$ for a relational structure \mathcal{H} and an integer k may be a difficult problem, in this Section we make a step forward by reducing the class of structures \mathcal{H} for which such a characterization is required. In particular, we show that it suffices to obtain a characterization for structures with only binary rectangular relations. More precisely, for any relational structure $\mathcal{H} = (H; \mathcal{R}_1, \dots, \mathcal{R}_k)$ we construct its binarization $b(\mathcal{H})$ as follows. The structure $b(\mathcal{H})$ is multi-sorted, and the domains are the relations $\mathcal{R}_1, \dots, \mathcal{R}_k$ viewed as sets of tuples, thus, $b(\mathcal{H})$ has k domains. For every pair $i, j \in [k]$ (i, j are allowed to be equal) and any $s \in [\ell_i], t \in [\ell_j]$, where ℓ_i, ℓ_j are the arities of $\mathcal{R}_i, \mathcal{R}_j$, the structure $b(\mathcal{H})$ contains a binary relation $\mathcal{Q}_{st}^{ij} = \{(\mathbf{a}, \mathbf{b}) \mid \mathbf{a} \in \mathcal{R}_i, \mathbf{b} \in \mathcal{R}_j, \mathbf{a}[s] = \mathbf{b}[t]\}$. We show that \mathcal{H} and $b(\mathcal{H})$ share many important properties.

► **Theorem 20.** *Let \mathcal{H} be a relational structure. Then \mathcal{H} is strongly rectangular (p -strongly rectangular, p -rigid, has a Mal'tsev polymorphism) if and only if $b(\mathcal{H})$ is strongly rectangular (p -strongly rectangular, p -rigid, has a Mal'tsev polymorphism).*

In addition to Theorem 20 every relation of $b(\mathcal{H})$ is binary and rectangular. This makes such structures somewhat closer to graphs and the hope is that it will be easier to study the structure of $\text{Aut}(b(\mathcal{H})^n)$ than $\text{Aut}(\mathcal{H}^n)$ itself.

References

- 1 Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In *Dagstuhl Follow-Ups*, volume 7. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/DFU.VOL7.15301.1.
- 2 Alexander I. Barvinok. *Combinatorics and Complexity of Partition Functions*, volume 30 of *Algorithms and combinatorics*. Springer, 2016. doi:10.1007/978-3-319-51829-9.
- 3 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013. doi:10.1145/2528400.
- 4 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In Chris Umans, editor, *FOCS*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.
- 5 Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006. doi:10.1137/050628957.
- 6 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Information and Computation*, 205(5):651–678, 2007. doi:10.1016/J.IC.2006.09.005.
- 7 Andrei A. Bulatov and Martin Grohe. The complexity of partition functions. *Theor. Comput. Sci.*, 348(2-3):148–186, 2005. doi:10.1016/J.TCS.2005.09.011.
- 8 Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.

- 9 Andrei A. Bulatov and Amirhossein Kazeminia. Complexity classification of counting graph homomorphisms modulo a prime number. In *STOC*, pages 1024–1037. ACM, 2022. doi:10.1145/3519935.3520075.
- 10 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. *J. ACM*, 64(3), June 2017. doi:10.1145/2822891.
- 11 Jin-yi Cai and Lane A. Hemachandra. On the power of parity polynomial time. In Burkhard Monien and Robert Cori, editors, *STACS*, volume 349 of *Lecture Notes in Computer Science*, pages 229–239. Springer, 1989. doi:10.1007/BFB0028987.
- 12 Víctor Dalmau and Peter Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004. doi:10.1016/J.TCS.2004.08.008.
- 13 M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17:260–289, 2000. doi:10.1002/1098-2418(200010/12)17:3/4<3C260::AID-RSA5%3E3.O.CO;2-W.
- 14 Martin Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. on Comp.*, 42(3):1245–1274, 2013. doi:10.1137/100811258.
- 15 John Faben. The complexity of counting solutions to generalised satisfiability problems modulo k , 2008. arXiv:0809.1836.
- 16 John Faben and Mark Jerrum. The complexity of parity graph homomorphism: an initial investigation. *arXiv preprint arXiv:1309.4033*, 2013. arXiv:1309.4033.
- 17 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.
- 18 Jacob Focke, Leslie Ann Goldberg, Marc Roth, and Stanislav Zivný. Counting homomorphisms to k_4 -minor-free graphs, modulo 2. In Dániel Marx, editor, *SODA*, pages 2303–2314. SIAM, 2021. doi:10.1137/1.9781611976465.137.
- 19 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. Counting homomorphisms to square-free graphs, modulo 2. *ACM Transactions on Computation Theory (TOCT)*, 8(3):1–29, 2016. doi:10.1145/2898441.
- 20 Andreas Göbel, J. A. Gregor Lagodzinski, and Karen Seidel. Counting homomorphisms to trees modulo a prime. In *MFCS*, volume 117, pages 49:1–49:13, 2018. doi:10.4230/LIPICS.MFCS.2018.49.
- 21 Heng Guo, Sangxia Huang, Pinyan Lu, and Mingji Xia. The Complexity of Weighted Boolean #CSP Modulo k . In *STACS*, volume 9, pages 249–260, 2011. doi:10.4230/LIPICS.STACS.2011.249.
- 22 Andreas Göbel, Leslie Ann Goldberg, and David Richerby. The complexity of counting homomorphisms to cactus graphs modulo 2. *ACM Trans on Comp Th.*, 6(4):1–29, 2014. doi:10.1145/2635825.
- 23 J. Hagemann and A. Mitschke. On n -permutable congruences. *Algebra Universalis*, 3:8–12, 1973.
- 24 Richard Hammack, Wilfried Imrich, and Sandi Klavzar. *Handbook of Product Graphs, Second Edition*. CRC Press, Inc., USA, 2nd edition, 2011.
- 25 Ulrich Hertrampf. Relations among mod-classes. *Theor. Comput. Sci.*, 74(3):325–328, 1990. doi:10.1016/0304-3975(90)90081-R.
- 26 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998. doi:10.1016/S0304-3975(97)00230-2.
- 27 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.
- 28 Amirhossein Kazeminia and Andrei A Bulatov. Counting homomorphisms modulo a prime number. In *MFCS*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 29 Amirhossein Kazeminia and Andrei A. Bulatov. Modular counting csp: Reductions and algorithms, 2025. arXiv:2501.04224.

- 30 Phokion G Kolaitis. Constraint satisfaction, complexity, and logic. In *Hellenic Conference on Artificial Intelligence*, pages 1–2. Springer, 2004. doi:10.1007/978-3-540-24674-9_1.
- 31 J. A. Gregor Lagodzinski, Andreas Göbel, Katrin Casel, and Tobias Friedrich. On counting (quantum-)graph homomorphisms in finite fields of prime order. *CoRR*, abs/2011.04827, 2021. arXiv:2011.04827.
- 32 E.H. Lieb and A.D. Sokal. A general Lee-Yang theorem for one-component and multicomponent ferromagnets. *Communications in Mathematical Physics*, 80(2):153–179, 1981.
- 33 L. Valiant. The complexity of computing the permanent. *Theoretical Computing Science*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 34 L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.
- 35 Dmitriy Zhuk. A proof of the CSP dichotomy conjecture. *J. ACM*, 67(5):30:1–30:78, 2020. doi:10.1145/3402029.

Efficiently Computing the Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

Stefan Kiefer ✉ 

Department of Computer Science, University of Oxford, UK

Andrew Ryzhikov ✉ 

Department of Computer Science, University of Oxford, UK

Abstract

A zero-one matrix is a matrix with entries from $\{0, 1\}$. We study monoids containing only such matrices. A finite set of zero-one matrices generating such a monoid can be seen as the matrix representation of an unambiguous finite automaton, an important generalisation of deterministic finite automata which shares many of their good properties.

Let \mathcal{A} be a finite set of $n \times n$ zero-one matrices generating a monoid of zero-one matrices, and m be the cardinality of \mathcal{A} . We study the computational complexity of computing the minimum rank of a matrix in the monoid generated by \mathcal{A} . By using linear-algebraic techniques, we show that this problem is in NC and can be solved in $\mathcal{O}(mn^4)$ time. We also provide a combinatorial algorithm finding a matrix of minimum rank in $\mathcal{O}(n^{2+\omega} + mn^4)$ time, where $2 \leq \omega \leq 2.4$ is the matrix multiplication exponent. As a byproduct, we show a very weak version of a generalisation of the Černý conjecture: there always exists a straight line program of size $\mathcal{O}(n^2)$ describing a product resulting in a matrix of minimum rank.

For the special case corresponding to complete DFAs (that is, for the case where all matrices have exactly one 1 in each row), the minimum rank is the size of the smallest image of the set of states under the action of a word. Our combinatorial algorithm finds a matrix of minimum rank in time $\mathcal{O}(n^3 + mn^2)$ in this case.

2012 ACM Subject Classification Theory of computation \rightarrow Formal languages and automata theory; Computing methodologies \rightarrow Symbolic and algebraic manipulation

Keywords and phrases matrix monoids, minimum rank, unambiguous automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.61

Funding *Andrew Ryzhikov*: Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT).

Acknowledgements We thank the anonymous reviewers for their helpful comments that improved the presentation of the paper.

1 Introduction

Matrix monoids are a rich and versatile object used in formal verification, program analysis, dynamical systems and weighted automata. However, many of their properties are in general undecidable. One such example is the well-studied matrix mortality problem. Given a finite set \mathcal{A} of $n \times n$ matrices, it asks if the monoid generated by \mathcal{A} (that is, the set of all products of matrices from \mathcal{A}) contains the zero matrix. This problem is undecidable already for 3×3 integer matrices [40], and was studied for several decidable special cases, see e.g. [16, 6, 47].

Even if \mathcal{A} is a set of zero-one matrices (that is, matrices with entries in $\{0, 1\}$), matrix mortality is PSPACE-complete [47]. We thus restrict our attention to the case where the whole monoid generated by \mathcal{A} consists of zero-one matrices; in this case, matrix mortality becomes decidable in polynomial time [34]. We call such monoids *zero-one matrix monoids*. Intuitively, when multiplying any two matrices from such a monoid, we never get $1 + 1$ as a subexpression. Zero-one matrix monoids have a rich structure while still admitting good



© Stefan Kiefer and Andrew Ryzhikov;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 61; pp. 61:1–61:22



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



algorithmic properties. They correspond precisely to unambiguous finite automata, and find applications in formal verification [3], variable-length codes [9] and symbolic dynamics [38]. They are also an interesting special case of finite monoids of rational matrices (studied in, e.g., [39, 28, 1, 13]), monoids of nonnegative matrices (studied in, e.g., [41, 10, 51, 22]), and, in the case where they do not contain the zero matrix, of matrix monoids with constant spectral radius [42].

In this paper, we consider a problem that can be seen as a natural generalisation of matrix mortality: given a finite set \mathcal{A} generating a zero-one monoid, find the minimum real rank of a matrix in this monoid. By the real rank of a matrix we mean the dimension of the subspace generated by its columns over the reals. Clearly, this rank is zero if and only if the monoid contains the zero matrix. The minimum real rank of a matrix in a zero-one matrix monoid is a much more tractable problem than deciding other similar properties: for example, checking if a zero-one matrix monoid contains a matrix of a given real rank was shown to be NP-hard¹ [24].

The goal of our paper is threefold. Firstly, we present efficient algorithms for analysing monoids of zero-one matrices and unambiguous finite automata. Secondly, to obtain these algorithms, we provide new structural properties of such monoids and automata that might be interesting on their own. Thirdly, we strengthen the connections between the areas of synchronising automata, weighted automata and matrix semigroups by transferring methods and tools between them. We also highlight open problems in the intersection of these areas.

2 Existing results and our contributions

Throughout the paper, we always assume that matrix monoids are defined by sets of generators, and all matrices are square zero-one unless stated otherwise.

Complete DFAs

An $n \times n$ zero-one matrix with exactly one 1 in every row can be equivalently seen as a transformation of a set Q of size n . A set of such matrices generates a zero-one matrix monoid, and can be seen as a complete deterministic finite (semi-)automaton² (complete DFA) $\mathcal{A} = (Q, \Sigma, \delta)$. Here, Σ is a finite alphabet whose letters correspond to the generating matrices, and $\delta : Q \times \Sigma \rightarrow Q$ is the transition function defined in such a way that for each $a \in \Sigma$, $\delta(_, a)$ is the transformation of Q induced in the natural way by the matrix corresponding to a . Thus, words over Σ correspond to products of the generating matrices.

The *rank* of a word w in \mathcal{A} is the size of the image of Q under the transformation corresponding to w . Equivalently, it is the real rank of the matrix corresponding to w . The *rank* of a complete DFA is the minimum among the ranks of all its words. This concept was studied from the perspectives of automata theory [43, 30] and the theory of transformation semigroups [48, 31]. It is the subject of the rank conjecture (called the Černý-Pin conjecture in [43]), which states that every complete DFA of rank r admits a word of rank r having length at most $(n - r)^2$. The Černý conjecture, one of the oldest open problems in combinatorial automata theory [50], is a special case with $r = 1$. We refer to surveys [49, 4, 30, 50] for the vast literature on the Černý conjecture. Underlying digraphs of complete DFAs of a given rank were studied in [12, 4] in the context of the road colouring problem.

¹ In fact, it is PSPACE-complete, which follows directly from [8, Theorem 3]: add a fresh state and define all yet undefined transitions to lead to this state.

² In this paper, all automata are semi-automata, meaning that they do not have any initial or accepting states, and do not recognise any languages. Following the usual conventions (as in, e.g., [9]), we omit “semi-”, in particular because it would make abbreviations like DFA less recognisable.

The rank of an n -state complete DFA over an alphabet of size m can be found in $\mathcal{O}(m^4 n^4)$ time [43, Theorem 1]. In contrast, for any fixed $r \geq 2$, the problem of checking if a complete DFA admits a word of rank r is NP-hard [24]. Checking if an n -state complete DFA over an alphabet of size m has rank one is NL-complete [26, 50], and can be done in $\mathcal{O}(mn^2)$ time [20, 49]. For each complete DFAs of rank r , there exists a word of rank r of length at most $\frac{(n-r)^3}{6} + \mathcal{O}((n-r)^2)$ [37], and if $r = 1$, finding a word of rank one can be done in $\mathcal{O}(n^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space [20].

Unambiguous finite automata

Generalising the case of complete DFAs, a set \mathcal{A} of $n \times n$ zero-one matrices generating a zero-one matrix monoid can be equivalently seen as an unambiguous nondeterministic finite (semi-)automaton (UFA). Let $Q = \{q_1, \dots, q_n\}$ be its set of states. To each matrix in \mathcal{A} we again associate a letter in the alphabet Σ , and the transition relation $\Delta \subseteq Q \times \Sigma \times Q$ is defined so that $(q_i, a, q_j) \in \Delta$ if and only if the entry (i, j) in the matrix corresponding to a is equal to one. Just as in the complete DFA case, words over Σ naturally correspond to products of matrices from \mathcal{A} .

The obtained NFA then has the property that is sometimes called *diamond-free*: for every two states p, q and every word w , there is at most one path from p to q labelled by w . A simple reachability argument shows that the length of a shortest word labelling two such paths, if it exists, is at most quadratic in the dimension of the matrices. Hence, deciding whether an NFA is a UFA (and thus whether a set of zero-one matrices generates a zero-one monoid) is in $\text{coNL} = \text{NL}$. It is actually NL-complete as described in the next subsection.

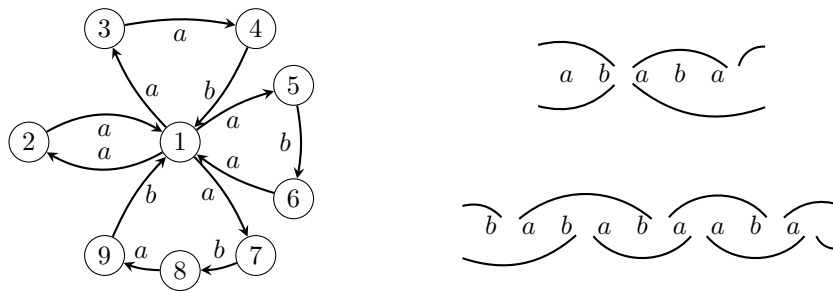
A UFA is called complete if it does not admit a word whose matrix is the zero matrix. For an n -state UFA the length of such a word if it exists is at most n^5 [34]. The best known lower bound is quadratic in n , and is achieved by a series of DFAs [44]. For UFAs, the quadratic upper bound was conjectured to be tight [43, Conjecture 2]. Checking if a UFA is complete can be done in NC^2 [34].

The *real rank* of a UFA is the minimum among the real ranks of the matrices corresponding to words. It was shown in [14] that for an n -state UFA of real rank $r \geq 1$ there always exists a word of minimum rank of length $\mathcal{O}(rn^3)$. For n -state strongly connected Eulerian UFAs of rank one, a subclass with remarkably nice properties, there always exists a word of length at most $(n-1)^2$ of rank one [15, Corollary 4]. All mentioned constructions also provide polynomial time algorithms that construct words with the required properties (in particular, with a length within the stated bounds).

Applications to variable-length codes

A *variable-length code* (or simply a *code*) is a set X of finite words over an alphabet Σ such that every finite word over Σ has at most one factorisation over X . In other words, a code is a basis of a free submonoid of Σ^* .

The definitions of both UFAs and codes rely, intuitively, on the uniqueness of certain representations. In fact, UFAs and codes are tightly related. Let us illustrate this relationship. If the cardinality of a code X is finite, one can construct its flower automaton, which is a UFA with a chosen state s such that, for each word from X , there is a separate cycle containing s and labelled by this word, see Figure 1 (left) for an example. More generally, codes that are regular languages correspond precisely to strongly connected UFAs in a similar way, see [9, Chapter 4] for the details.



■ **Figure 1** The flower automaton of the code $X = \{aa, aab, aba, abab\}$ (left), two adjacent interpretations of $ababa$ over X (top right), and two disjoint interpretations of $bababaaba$ over X (bottom right). Note that this code is not complete, but still illustrates all the discussed properties.

A useful corollary of the construction of the flower automaton is the fact that deciding if a set of zero-one matrices generates a zero-one monoid is NL-hard. Indeed, a finite set of words is a code if and only if its flower automaton is unambiguous [9]. Deciding if a finite set of words is a code is NL-complete [45], and the flower automaton can be constructed in AC^0 .

A code X over Σ is called *complete* if every word over Σ is a factor of a concatenation of codewords, that is, for every word $w \in \Sigma^*$ there exist $u, v \in \Sigma^*$ with $uwv \in X^*$. A code that is a regular language is complete if and only if the corresponding UFA is complete [9]. For complete codes that are regular languages, the real rank of the corresponding UFA is equal to a natural and important parameter called the degree of a code [9, Proposition 9.6.1].

Let us explain the intuition behind the notion of degree, see [9, Chapter 9.6] for the formal definitions. For each word w we can consider all possible factorisations over X of all its extensions $uwv \in X^*$ with $u, v \in \Sigma^*$, called *interpretations* of w . Two such interpretations either match in at least one position (as in Figure 1 (top right) between the second and the third letter), or do not match in any position (as in Figure 1 (bottom right)), in which case they are called *disjoint*. The degree of a word is the number of pairwise disjoint interpretations of this word. The degree of a code X is the minimum nonzero degree of all words $w \in \Sigma^*$.

A particularly important case is when a complete code has degree one. Then there exists a word $w \in X^*$ (called a synchronising word) such that for any concatenation of codewords $uwvw \in X^*$ with $u, v \in \Sigma^*$ we have $uw, vw \in X^*$. Intuitively, this means that the two halves uw and vw can be decoded separately and independently.

Computational complexity classes

In this paper, we characterise the computational complexity of problems by showing that they belong to the classes $NL \subseteq NC^2 \subseteq NC \subseteq P$, see [2, 23] for their formal definitions. NL is the class of problems solvable in nondeterministic logarithmic time. NC^k is the class of problems solvable by $\mathcal{O}((\log n)^k)$ -depth polynomial-size bounded fan-in Boolean circuits, and NC is the union of these classes for all $k \geq 1$. The class NC represents problems that have efficient parallel algorithms, and is a subclass of problems solvable in polylogarithmic space [2]. Intuitively, NC is the class of problems that can be solved using local computations, as opposed to P-complete problems, which are inherently sequential and thus require storing the entire structure in the memory unless $NC = P$. Showing that a problem is in NC also allows to obtain a polynomial space algorithm for the case of exponential-size input computed by a PSPACE-transducer (as done, e.g., in [3]), which is not necessarily true for arbitrary problems solvable in polynomial time.

NC^2 is an especially important class in computational algebra. To quote [23, page 468], “ NC^2 is the habitat of most natural problems in linear algebra”. Indeed, matrix multiplication, computing the determinant, inverse and rank of a matrix belong to NC^2 [11, 18, 7, 19].

Our contributions

The known results about reachability properties of zero-one matrix monoids (including the special case of complete DFAs), such as [14, 20, 46, 34], mostly construct a product of minimum rank iteratively, with each iteration decreasing the number of different rows or the rank of a matrix. Such an approach is inherently sequential, since the matrix in the new iteration has to depend on the previous one, which thus has to be constructed explicitly. In particular, this requires matrix multiplication at every step, which heavily increases the time complexity. In this paper, we take a different direction by strongly relying on linear algebra. While linear-algebraic arguments are used widely in the synchronising automata literature, they mostly serve to decrease the number of iterations in the iterative scheme described above. Our approach is to instead relate the rank of a zero-one matrix monoid to efficiently computable linear-algebraic properties, without explicitly constructing a matrix of minimum rank.

Our first main result is that computing the rank of a zero-one matrix monoid provided in the input by a generating set of m matrices of dimension n (or, equivalently, by a UFA with n states and m letters) is in NC^2 (Theorem 18) and can be done in time $\mathcal{O}(mn^4)$ (Theorem 22). Previously, it was not known that this problem is in NC , not even for complete DFAs or finite complete codes. Moreover, the naive implementation of the polynomial time algorithm from the literature works in time $\mathcal{O}(n^{4+\omega} + mn^4)$ [14].

These results rely on a new concept of weight of the matrices in a complete zero-one monoid. This theory of matrix weight, which we develop in Section 4, is our main technical contribution. Matrix weight is a natural generalisation of an existing notion of weight of columns of matrices in complete DFAs, which was used, e.g., in connection with the road colouring problem [21, 29, 25]. We show that all matrices in a zero-one matrix monoid have the same weight, and that this weight is tightly related to both the rank of the monoid and to the maximal weight of the columns and rows of its matrices (Section 4.3). This connection allows us to reduce the computation of the monoid rank to the computation of maximal column and row weight. Then we show that we can instead compute the weight of “maximal pseudo-columns” and “maximal pseudo-rows”, as they have the same weight as maximal columns and rows, respectively (Section 4.4). Finally, we transfer linear-algebraic techniques from the literature on weighted automata to compute those weights, and thus the rank of the monoid, efficiently (Section 5 and Section 6.2).

We complement the linear-algebraic algorithms with a combinatorial algorithm, our second main contribution. While the latter has higher time complexity of $\mathcal{O}(n^{2+\omega} + mn^4)$ in the general case (Theorem 23), it also constructs a matrix of minimum rank in addition to computing the rank of the monoid. For complete DFAs, our combinatorial algorithm runs in time $\mathcal{O}(n^3 + mn^2)$ (Theorem 24), thus outmatching the linear-algebraic counterpart and improving upon the $\mathcal{O}(m^4n^4)$ algorithm known before [43]. The two key technical ingredients of our combinatorial algorithm are explained in the beginnings of Section 6.3 and Section 6.4. Our results on the time complexity of computing the rank are summarised in the table below.

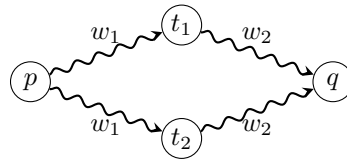
class	previous best	linear-algebraic algorithm	combinatorial algorithm
UFA	$\mathcal{O}(n^{4+\omega} + mn^4)$ [14]	$\mathcal{O}(mn^4)$ (Theorem 22)	$\mathcal{O}(n^{2+\omega} + mn^4)$ (Theorem 23)
complete DFA	$\mathcal{O}(m^4n^4)$ [43]	$\mathcal{O}(mn^3)$ (see Section 6.2)	$\mathcal{O}(n^3 + mn^2)$ (Theorem 24)

3 Main definitions

Let Q be a finite set, which we view as a set of states. For $S \subseteq Q$ we write $[S]$ for the column vector $x \in \{0, 1\}^Q$ such that $x(q) = 1$ if and only if $q \in S$. We may write $[q]$ for $[\{q\}]$. For a column vector $x \in \{0, 1\}^Q$ we write x^T for the transpose, a row vector. For two column vectors $x_1, x_2 \in \mathbb{R}^Q$ we write $x_1 \geq x_2$ if the inequality holds component-wise. We view the elements of $\mathbb{R}^{Q \times Q}$ (and similar sets) as matrices. Vector and matrix addition and multiplication are defined in the usual way (over \mathbb{R}). We denote by $\langle X \rangle$ the span of a set X of vectors, i.e., the set of all linear combinations of X with real coefficients. The *real rank* of a matrix $A \in \mathbb{R}^{Q \times Q}$ is, as usual, the dimension of the column space of A over the field of the reals (which equals the dimension of the row space); i.e., $\text{rank}_{\mathbb{R}}(A) = \dim \langle A[q] \mid q \in Q \rangle = \dim \langle [q]^T A \mid q \in Q \rangle$.

Let $\mathcal{A} = \{A_1, \dots, A_m\}$ be a set of matrices from $\{0, 1\}^{Q \times Q}$, and $\Sigma = \{a_1, \dots, a_m\}$ be a finite alphabet. We associate the letters with the matrices by setting $M(a_i) = A_i$ for $1 \leq i \leq m$. Throughout this paper, when speaking about computational complexity, we assume that the input is the function $M: \Sigma \rightarrow \{0, 1\}^{Q \times Q}$ from letters to zero-one matrices. We can extend $M: \Sigma \rightarrow \{0, 1\}^{Q \times Q}$ naturally (and often implicitly) to $M: \Sigma^* \rightarrow \mathbb{Z}_{\geq 0}^{Q \times Q}$ by defining $M(a_1 \dots a_k) = M(a_1) \dots M(a_k)$. Thus, M is a monoid homomorphism from Σ^* to the matrix monoid $M(\Sigma^*)$ generated by $\mathcal{A} = M(\Sigma)$. Note that $M(\varepsilon) = I$, where ε denotes the empty word and I the $Q \times Q$ identity matrix. In this paper, we consider only monoid morphisms $M: \Sigma^* \rightarrow \mathbb{Z}_{\geq 0}^{Q \times Q}$ that are *unambiguous*, i.e., $M: \Sigma^* \rightarrow \{0, 1\}^{Q \times Q}$. If M is unambiguous, $\mathcal{A} = M(\Sigma)$ generates a finite matrix monoid $M(\Sigma^*) \subseteq \{0, 1\}^{Q \times Q}$.

Viewing the matrices as transition matrices of an automaton, we obtain a *nondeterministic finite (semi-)automaton (NFA)* (Q, Σ, Δ) with transition relation $\Delta = \{(p, a, q) \in Q \times \Sigma \times Q \mid [p]^T M(a)[q] = 1\}$. Recall that in this paper automata do not have dedicated initial or accepting states, see footnote 2 on page 2. We can extend Δ from letters to words in the usual way so that we have $\Delta = \{(p, w, q) \in Q \times \Sigma^* \times Q \mid [p]^T M(w)[q] \geq 1\}$. An NFA (Q, Σ, Δ) is *unambiguous* (or *diamond-free*) if for every two states p, q and for every two words w_1, w_2 there exists at most one $t \in Q$ with $(p, w_1, t) \in \Delta$ and $(t, w_2, q) \in \Delta$; see Figure 2 for an illustration of the forbidden configuration. We denote unambiguous NFAs as UFAs. Recall from the previous section that deciding if an NFA is unambiguous is NL-complete. In the following, we often identify $M: \Sigma^* \rightarrow \{0, 1\}^{Q \times Q}$ with the corresponding UFA (Q, Σ, Δ) . In particular, a monoid homomorphism is unambiguous if and only if the corresponding NFA is unambiguous.



■ **Figure 2** The configuration that is forbidden in a UFA.

When M (or, equivalently, Δ) is clear from the context, we may write $p \cdot w = \{q \in Q \mid (p, w, q) \in \Delta\}$. Then $[p \cdot w]^T = [p]^T M(w)$. Similarly, we may write $w \cdot q = \{p \in Q \mid (p, w, q) \in \Delta\}$, so that $[w \cdot q] = M(w)[q]$. We call M *strongly connected* if for all $p, q \in Q$ there is $w \in \Sigma^*$ with $p \cdot w \ni q$. We call M *complete* if $0 \notin M(\Sigma^*)$, where 0 is the zero matrix. The *real rank* of M (and of $M(\Sigma^*)$) is $\text{rank}_{\mathbb{R}}(M) := \min\{\text{rank}_{\mathbb{R}}(M(w)) \mid w \in \Sigma^*\}$. Note that M is complete if and only if $\text{rank}_{\mathbb{R}}(M) \neq 0$.

Suppose that $|p \cdot a| = 1$ holds for every $p \in Q$ and $a \in \Sigma$, or, equivalently, that every matrix in \mathcal{A} has exactly one 1 in each row. Then $|p \cdot w| = 1$ holds for every $p \in Q$ and $w \in \Sigma^*$. We call such UFAs *complete deterministic finite (semi-)automata (complete DFAs)*

and we may write δ instead of Δ to highlight that it is a transition function $\delta: Q \times \Sigma \rightarrow Q$ instead of a transition relation. A complete DFA (Q, Σ, δ) is complete in the sense defined above (i.e., $0 \notin M(\Sigma^*)$), and for any $w \in \Sigma^*$ we have that $\text{rank}_{\mathbb{R}}(M(w))$ is the number of nonzero columns in $M(w)$.

4 Main concepts and the linear algebra toolbox

In this section, we introduce the main tools that we will use for both linear-algebraic and combinatorial algorithms in later sections. Until Section 4.5, we fix an unambiguous, complete, and strongly connected monoid morphism M . In Section 4.5 we will show that the case where M is not strongly connected can be easily reduced to the strongly connected case.

4.1 Columns, rows and the structure of minimum rank matrices

The concept of maximum columns and rows plays a crucial role in dealing with reachability problems in unambiguous monoid morphisms. Abusing language slightly in the following, by *column* we refer to column vectors of the form $[w \cdot q] = M(w)[q] \in \{0, 1\}^Q$ where $w \in \Sigma^*$ and $q \in Q$. Similarly, a *row* is of the form $[q \cdot w]^T = [q]^T M(w)$. See Figure 3 for an example. In the case of complete DFAs, all rows are of the form $[q]^T$. This fact makes complete DFAs significantly simpler to deal with than general complete UFAs.

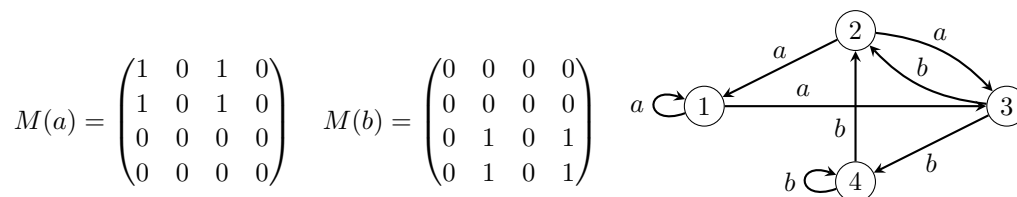


Figure 3 $[a \cdot 3] = M(a)[3] = [\{1, 2\}]$ is a column; $[2 \cdot a]^T = [2]^T M(a) = [\{1, 3\}]^T$ is a row.

A column $[C]$ is called *maximal* if there is no column $[C']$ such that $[C'] \neq [C]$ and $[C'] \geq [C]$ (that is, $C \subset C'$). Maximal rows are defined in the same way. Recall that the inequalities are taken component-wise.

Let $A \in \{0, 1\}^{m \times n}$ be a zero-one matrix. One can view $\text{rank}_{\mathbb{R}}(A)$ as the least number r such that there are matrices $C \in \mathbb{R}^{m \times r}$ and $R \in \mathbb{R}^{r \times n}$ with $A = CR$. Define the *unambiguous rank* $\text{rank}_{\text{un}}(A)$ as the least number r such that there are matrices $C \in \{0, 1\}^{m \times r}$ and $R \in \{0, 1\}^{r \times n}$ such that $A = CR$. Analogously to $\text{rank}_{\mathbb{R}}(M)$, define also $\text{rank}_{\text{un}}(M) := \min\{\text{rank}_{\text{un}}(M(w)) \mid w \in \Sigma^*\}$. Clearly, $\text{rank}_{\mathbb{R}}(A) \leq \text{rank}_{\text{un}}(A)$, and the inequality can be strict, but in Corollary 2 below we show that $\text{rank}_{\mathbb{R}}(M) = \text{rank}_{\text{un}}(M)$. The reason we are interested in the unambiguous rank is that Theorem 1 below implies that there is always a matrix with a very simple structure such that its unambiguous rank is equal to its real rank and both ranks are minimum.

In the following let us write $r := \text{rank}_{\text{un}}(M)$ when M is understood. A word $u \in \Sigma^*$ is of *minimum unambiguous rank* if $\text{rank}_{\text{un}}(M(u)) = r$. If $u \in \Sigma^*$ is of minimum unambiguous rank then so is vuw for all $v, w \in \Sigma^*$.

Words of unambiguous rank one, known as synchronising words, play an especially important role due to their applications in the theory of codes, as explained in Section 2. It is easy to see that a word w has unambiguous rank one if and only if there exist $C, R \subseteq Q$ such that w maps a state p to a state q if and only if $p \in C$ and $q \in R$. For complete DFAs, we moreover have that $C = Q$ and R has cardinality one.

► **Theorem 1** (Césari [17]). *Let $u \in \Sigma^*$ be of minimum unambiguous rank. There are pairwise disjoint sets $C_1, \dots, C_r \subseteq Q$ and pairwise disjoint sets $R_1, \dots, R_r \subseteq Q$ such that*

$$M(u) = \sum_{i=1}^r [C_i][R_i]^T.$$

Moreover, each $[C_i]$ and $[R_i]^T$ is, respectively, a maximal column and a maximal row.

This theorem will play a central role. A proof can be found in [5, Proposition 4]. In the case of a complete DFA, R_1 is a singleton and Theorem 1 is fairly obvious.

In Theorem 1, since the C_i are pairwise disjoint and the R_i are pairwise disjoint, each $[C_i][R_i]^T$ forms, intuitively, a “combinatorial rectangle”, and no such rectangle shares a row or a column with any other rectangle. The column vectors $[C_i]$ are exactly the nonzero columns of $M(u)$ and linearly independent, and the row vectors $[R_i]^T$ are exactly the nonzero rows of $M(u)$ and linearly independent. Thus, r is the number of distinct nonzero columns and also the number of distinct nonzero rows in $M(u)$. It follows that $r = \text{rank}_{\text{un}}(M(u)) = \text{rank}_{\mathbb{R}}(M(u))$. Thus we have:

► **Corollary 2.** *We have $r = \text{rank}_{\text{un}}(M) = \text{rank}_{\mathbb{R}}(M)$.*

We can thus define $\text{rank}(M)$ as $\text{rank}_{\text{un}}(M) = \text{rank}_{\mathbb{R}}(M)$. For words $w \in \Sigma^*$ that are not of minimum unambiguous rank, we may have $\text{rank}_{\mathbb{R}}(M(w)) < \text{rank}_{\text{un}}(M(w))$, but the rank of such matrices will rarely play a role in the following. In what follows, we call words of minimum unambiguous rank simply words of minimum rank. Since we never refer to the real rank of words below, this will not lead to any confusion.

4.2 The weight of columns and rows

The results in this subsection, about the column and row vectors that appear in the matrices $M(w)$, are mostly due to [17]; see also [5, Section 3]. Since a notion of column and row weight will be crucial for us in the later development, we phrase and prove the results around these concepts, but we do not view the lemmas of this subsection as novel.

Define $\bar{A} = \frac{1}{|\Sigma|} \sum_{a \in \Sigma} M(a) \in [0, 1]^{Q \times Q}$. Since M is strongly connected, \bar{A} is irreducible. Since M is unambiguous, the spectral radius of \bar{A} is at most 1, and since M is complete, it is at least 1. Thus, the spectral radius of \bar{A} equals 1. Since \bar{A} is irreducible, it follows from basic Perron-Frobenius theory that \bar{A} has an eigenvalue 1 and every right eigenvector with eigenvalue 1 is a multiple of a strictly positive vector, say $\beta \in \mathbb{R}_{>0}^Q$. Since \bar{A} has only rational entries, we can assume $\beta \in \mathbb{Q}_{>0}^Q$. Similarly for left eigenvectors. Therefore, there are $\alpha, \beta \in \mathbb{Q}_{>0}^Q$ with $\alpha^T \bar{A} = \alpha^T$ and $\bar{A} \beta = \beta$. Without loss of generality, we assume that $\alpha^T \beta = 1$.

In the complete DFA case, since $M(a)[Q] = [Q]$ for all $a \in \Sigma$, we have $\bar{A}[Q] = [Q]$ and so it is natural to take $\beta = [Q]$. In that case, $\alpha^T [Q] = \alpha^T \beta = 1$ means that $\alpha^T = \alpha^T \bar{A}$ is the (unique) stationary distribution of the Markov chain whose transition probabilities are given by the row-stochastic matrix \bar{A} ; intuitively, in this Markov chain a letter $a \in \Sigma$ is picked uniformly at random in every step.

Define the *weight* of a column y and of a row x^T by $\alpha^T y \in \mathbb{R}$ and $x^T \beta \in \mathbb{R}$, respectively. Denote the maximum column weight and the maximum row weight by mcw and mrw , respectively, i.e.,

$$\text{mcw} := \max\{\alpha^T y \mid y \text{ is a column}\} \quad \text{and} \quad \text{mrw} := \max\{x^T \beta \mid x^T \text{ is a row}\}.$$

A column y is called *of maximum weight* if $\alpha^T y = \text{mcw}$, and analogously for rows. In the complete DFA case, every row is of the form $[q]^T$ for some $q \in Q$, hence every row is of maximum weight.

► **Lemma 3.** *A column (respectively, row) is maximal if and only if it is of maximum weight.*

An important property that we will need later is that the set of maximal columns is closed under left multiplication by matrices from the monoid, as stated in the following lemma. Note that this is no longer true without the completeness assumption, and is the key reason why the case of complete matrix monoids is easier to deal with.

► **Lemma 4.** *Let $v \in \Sigma^*$ and $q \in Q$ be such that $[v \cdot q]$ is a maximal column. Then $[uv \cdot q]$ is a maximal column for all $u \in \Sigma^*$.*

The following lemma will be useful later to construct minimum rank matrices from maximal columns and rows.

► **Lemma 5.** *Let $w \in \Sigma^*$ be such that all non-zero columns and rows in $M(w)$ are maximal. Then w is of minimum rank.*

4.3 Weight preservation property and minimum rank

Every word w of minimum rank in a complete DFA induces a partition of the state set into subsets of states mapped by w to the same state (that is, into columns). It was observed by Friedman in [21] that all sets of such a partition have the same weight. This observation has many applications to variations of the road colouring problem [21, 29, 25, 30]. Moreover, it was proved in [25, Theorem 6], again in connection with road colouring, that for every w the weights of all columns in $M(w)$ sum up to 1 (assuming $\beta = [Q]$ as suggested previously). This can be seen as a weight preservation property: the total weight of columns in the matrix of a word is preserved under multiplication by any matrix from the monoid. As a result we get that $1 = r \cdot \text{mcw}$, and hence $r = \frac{1}{\text{mcw}}$. The proof of the weight preservation property for complete DFAs is quite simple and relies on the fact that for a state q and a word w the set $q \cdot w$ is always a singleton. For complete UFAs this is no longer true; in particular, $q \cdot w$ can be the empty set, thus permanently “losing” some weight collected in q . Hence, a more refined property is required. The following result provides such a property. It also turns out that its proof requires more sophisticated techniques than in the complete DFA case.

► **Theorem 6.** *For all $w \in \Sigma^*$ we have $1 = \alpha^T M(w) \beta = r \cdot \text{mcw} \cdot \text{mrw}$.*

Similarly to the complete DFA case, this result allows us to reduce computing r to computing mcw and mrw , which we will use later in our algorithms. Recall that we have defined α^T and β so that $\alpha^T \beta = 1$.

Towards a proof of Theorem 6 we first prove the following lemma.

► **Lemma 7.** *Let $u \in \Sigma^*$ be of minimum rank. Then $\alpha^T M(u) \beta = r \cdot \text{mcw} \cdot \text{mrw}$.*

Proof. Let $M(u) = \sum_{i=1}^r [C_i][R_i]^T$ be as in Theorem 1. Each $[C_i]$ and each $[R_i]$ is of maximum weight. Thus,

$$\alpha^T M(u) \beta = \sum_{i=1}^r \alpha^T [C_i][R_i]^T \beta = \sum_{i=1}^r \text{mcw} \cdot \text{mrw} = r \cdot \text{mcw} \cdot \text{mrw}. \quad \blacktriangleleft$$

We also need the following proposition.

61:10 Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

► **Proposition 8.** *Let $x \in \mathbb{R}^Q$ and $c \in \mathbb{R}$ be such that $x^T M(u)\beta = c$ holds for all $u \in \Sigma^*$ of minimum rank. Then $x^T M(w)\beta = c$ holds for all $w \in \Sigma^*$.*

Proof. Let $w \in \Sigma^*$. Let $u \in \Sigma^*$ be of minimum rank. Recall that every word that contains u as a factor is of minimum rank. For every $k \geq 0$, partition Σ^k into sets $W_0(k)$ and $W_1(k)$ so that $W_0(k) = \Sigma^k \cap (\Sigma^* u \Sigma^*)$ and $W_1(k) = \Sigma^k \setminus (\Sigma^* u \Sigma^*)$; i.e., $W_0(k), W_1(k)$ are the sets of length- k words that do or do not contain u as a factor, respectively. For all $v \in W_0(k)$ both v and wv are of minimum rank. Thus, we have $x^T M(wv)\beta = c$ for all $v \in W_0(k)$. It follows that

$$\sum_{v \in W_0(k)} \frac{x^T M(wv)\beta}{|W_0(k)|} = c \quad \text{for all } k \geq 0. \quad (1)$$

Let $d > 0$ be such that $|x^T A\beta| \leq d$ for all $A \in \{0, 1\}^{Q \times Q}$. Then we have

$$\sum_{v \in W_1(k)} \frac{|x^T M(wv)\beta|}{|W_1(k)|} \leq d \quad \text{for all } k \geq 0. \quad (2)$$

Let $m \geq 0$. Define $p_1(m) := \frac{|W_1(m|u)|}{|\Sigma|^{m|u}|}$. We can view $p_1(m)$ as the probability of picking a word in $W_1(m|u)$ when a word of length $m|u|$ is picked uniformly at random. We have $p_1(m) \leq \left(1 - \frac{1}{|\Sigma|^{|u|}}\right)^m$, as in order to avoid u as a factor, it has to be avoided in each of the m consecutive blocks of length $|u|$. Thus, $\lim_{m \rightarrow \infty} p_1(m) = 0$. We have

$$\begin{aligned} x^T M(w)\beta &= x^T M(w)\bar{A}\beta = x^T M(w)\bar{A}^{m|u}\beta = \frac{1}{|\Sigma|^{m|u}|} \sum_{v \in \Sigma^{m|u}} x^T M(wv)\beta \\ &= \frac{|W_0(m|u)|}{|\Sigma|^{m|u}|} \sum_{v \in W_0(m|u)} \frac{x^T M(wv)\beta}{|W_0(m|u)|} + \frac{|W_1(m|u)|}{|\Sigma|^{m|u}|} \sum_{v \in W_1(m|u)} \frac{x^T M(wv)\beta}{|W_1(m|u)|} \\ &= (1 - p_1(m)) \cdot c + p_1(m) \cdot \sum_{v \in W_1(m|u)} \frac{x^T M(wv)\beta}{|W_1(m|u)|} \quad (\text{by Equation (1)}). \end{aligned}$$

With Equation (2) it follows that $|x^T M(w)\beta - c| \leq p_1(m)(|c| + d)$. Since this holds for all $m \geq 0$ and $\lim_{m \rightarrow \infty} p_1(m) = 0$, we conclude that $x^T M(w)\beta = c$. ◀

Now we prove Theorem 6.

Proof of Theorem 6. It follows from Lemma 7 and Proposition 8 that

$$\alpha^T M(w)\beta = r \cdot mcw \cdot mrw \quad \text{for all } w \in \Sigma^*.$$

With $w = \varepsilon$ we obtain $1 = \alpha^T \beta = \alpha^T M(\varepsilon)\beta = r \cdot mcw \cdot mrw$, as required. ◀

4.4 Maximal pseudo-columns

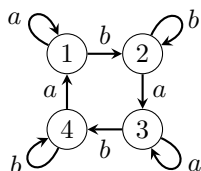
In this subsection, we define maximal pseudo-columns, which are vectors that can be seen as a relaxation of the notion of maximal columns. We show that the weight of a maximal pseudo-column is equal to the weight of a maximal column, and a maximal pseudo-column is a solution of a system of linear equations, and thus can be computed efficiently. By invoking Theorem 6, this will allow us to efficiently compute r .

Denote by $\text{MCol} \subseteq \{0, 1\}^Q$ the set of maximal columns. By Theorem 1 (bearing in mind also Corollary 2), the vector space spanned by all maximum columns, $\langle \text{MCol} \rangle$, is at least r -dimensional:

► **Proposition 9.** *We have $r \leq \dim \langle \text{MCol} \rangle$.*

One might hypothesise that $r = \dim \langle \text{MCol} \rangle$ or even that all minimum-rank matrices have the same r nonzero (hence, maximum) columns. The following example shows that neither is the case in general, not even for complete DFAs.

► **Example 10.** Consider the complete DFA with $\Sigma = \{a, b\}$ and

$$M(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M(b) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$


By symmetry, we have $\alpha^T = (\frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{4})$. Since no word maps states 1 and 3 to the same state, we have $r = 2$; i.e., a and b are both minimum rank. Further, MCol consists exactly of the four nonzero columns in $M(a)$ and $M(b)$. Their span $\langle \text{MCol} \rangle$ is 3-dimensional, as $(1 \quad -1 \quad 1 \quad -1)$ is orthogonal to each maximum column. Thus, $r = 2 < 3 = \dim \langle \text{MCol} \rangle < 4 = |\text{MCol}|$.

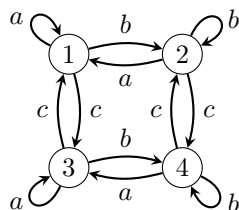
Define the vector space $U := \langle \alpha^T M(w) - \alpha^T \mid w \in \Sigma^* \rangle$. Intuitively, it is the set of all differences of weight distributions over the states before and after a word is applied. Notice that for all $w_1, w_2 \in \Sigma^*$ we have $\alpha^T M(w_1) - \alpha^T M(w_2) \in U$. Later (see the proof of Lemma 19 below) we show that U is closed under post-multiplication with $M(a)$ for all $a \in \Sigma$. Such “forward spaces” play an important role in weighted automata; see, e.g., [32]. Denote the orthogonal complement of U by U^\perp ; i.e., $U^\perp = \{y \in \mathbb{R}^Q \mid \forall w \in \Sigma^* : \alpha^T M(w)y = \alpha^T y\}$. Intuitively, it is the set of vectors whose weight does not change under pre-multiplication with $M(w)$ for any w (where by the weight of a vector y we understand $\alpha^T y$). Clearly, $\dim U + \dim U^\perp = |Q|$. The following proposition follows immediately from Lemma 4.

► **Proposition 11.** *We have $\text{MCol} \subseteq U^\perp$.*

It follows that $\langle \text{MCol} \rangle$ is a subspace of U^\perp . With Proposition 9, we have $r \leq \dim \langle \text{MCol} \rangle \leq \dim U^\perp$. One might hypothesise that $\langle \text{MCol} \rangle = U^\perp$. The following example shows that this is not the case in general, not even for complete DFAs.

► **Example 12.** Consider the DFA with $\Sigma = \{a, b, c\}$ and

$$M(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad M(b) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad M(c) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$



61:12 Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

We have $\text{Mer}(1) = \text{Mer}(2) = \{1, 2\}$ and $\text{Mer}(3) = \text{Mer}(4) = \{3, 4\}$. Thus, $\text{MCol} = \{(1 \ 1 \ 0 \ 0)^T, (0 \ 0 \ 1 \ 1)^T\}$. Hence, $\dim \langle \text{MCol} \rangle = 2$.

On the other hand, by symmetry we have $\alpha^T = (\frac{1}{4} \ \frac{1}{4} \ \frac{1}{4} \ \frac{1}{4})$. For any $w \in \Sigma^*$,

$$\alpha^T M(w) = \begin{cases} \left(\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \right) & \text{if } w \in \{c\}^* \\ \left(\frac{1}{2} & 0 & \frac{1}{2} & 0 \right) & \text{if the last non-}c \text{ letter in } w \text{ is } a \\ \left(0 & \frac{1}{2} & 0 & \frac{1}{2} \right) & \text{if the last non-}c \text{ letter in } w \text{ is } b. \end{cases}$$

It follows that $U = \langle (1 \ -1 \ 1 \ -1) \rangle$. Thus, $\dim U^\perp = 4 - 1 = 3 > 2 = \dim \langle \text{MCol} \rangle$, and $\langle \text{MCol} \rangle$ is a strict subspace of U^\perp . For example, the vector $(1 \ 0 \ 0 \ 1)^T$ is in U^\perp but not in $\langle \text{MCol} \rangle$.

Although the dimension of U^\perp does not generally equal r , the vector space U^\perp turns out useful for computing r . Recall that, by Theorem 6, we can obtain r by computing mcw (and, symmetrically, mrw). For $q \in Q$ define

$$\text{Mer}(q) := \{q' \in Q \mid \exists S \supseteq \{q, q'\} \text{ such that } [S] \text{ is a column}\}.$$

Intuitively, $\text{Mer}(q)$ consists of the states that can “appear” in a column together with q , or, equivalently, the states that are “mergeable” with q (that is, can be mapped to the same state by a word). Note that $q \in \text{Mer}(q)$. We will need the following lemma which is easy to prove.

► **Lemma 13.** *Let $v \in \Sigma^*$ and $q \in Q$ be such that $[v \cdot q]$ is a maximal column. Then $v \cdot q' = \emptyset$ holds for all $q' \in \text{Mer}(q) \setminus \{q\}$.*

We call a vector $y \in U^\perp$ a *maximal pseudo-column* if there is $q \in Q$ with $y(q) = 1$ and $y(q') = 0$ for all $q' \notin \text{Mer}(q)$. This notion, which is closely related to the “pseudo-cuts” from [35], can be seen as a relaxation of the notion of a maximal column: clearly, every maximal column is a maximal pseudo-column, but the converse is not true, since a maximal pseudo-column is not necessarily a vector over $\{0, 1\}$, let alone a *column* in the strict sense, i.e., of the form $[w \cdot p]$. The following lemma however shows that the weight of a maximal pseudo-column is equal to the weight of a maximal column. We will later show that computing the former can be done in NC^2 .

► **Lemma 14.** *Let y be a maximal pseudo-column. Then $\alpha^T y = \text{mcw}$.*

Proof. Let $q \in Q$ be such that $y(q) = 1$ and $y(q') = 0$ for all $q' \notin \text{Mer}(q)$. Let $w \in \Sigma^*$ be such that $[w \cdot q]$ is a maximal column. We have

$$\begin{aligned} \alpha^T y &= \alpha^T M(w)y && (y \in U^\perp) \\ &= \sum_{q' \in Q} y(q') \alpha^T [w \cdot q'] \\ &= \sum_{q' \in \text{Mer}(q)} y(q') \alpha^T [w \cdot q'] && (y(q') = 0 \text{ for } q' \notin \text{Mer}(q)) \\ &= \alpha^T [w \cdot q] + \sum_{q' \in \text{Mer}(q) \setminus \{q\}} y(q') \alpha^T [w \cdot q'] && (y(q) = 1) \\ &= \alpha^T [w \cdot q] && (\text{Lemma 13}) \\ &= \text{mcw} && (\text{by the choice of } w, q). \quad \blacktriangleleft \end{aligned}$$

► **Example 15.** We continue Example 10. We have $U = \langle (1 \ -1 \ 1 \ -1) \rangle$ and $\text{Mer}(2) = \{1, 2, 3\}$. Let $y = (4/3 \ 1 \ -1/3 \ 0)^T$. Then $y \in U^\perp$. Since $y(2) = 1$ and $y(4) = 0$, vector y is a maximal pseudo-column. Thus, by Lemma 14, $\text{mcw} = \alpha^T y = (\frac{1}{4} \ \frac{1}{4} \ \frac{1}{4} \ \frac{1}{4}) y = \frac{1}{2}$.

► **Theorem 16.** Let Γ be a basis of U , and let $q \in Q$. Then the following linear system for $y \in \mathbb{R}^Q$ has a solution, and all its solutions are maximal pseudo-columns:

$$\begin{aligned} \gamma^T y &= 0 \quad \text{for all } \gamma^T \in \Gamma \\ y(q) &= 1 \\ y(q') &= 0 \quad \text{for all } q' \notin \text{Mer}(q). \end{aligned}$$

Proof. By Proposition 11, any maximal column solves the linear system. Let $y \in \mathbb{R}^Q$ be a solution of the linear system. The equations on the first line guarantee that $y \in U^\perp$. Then, the equations on the second and third line guarantee that y is a maximal pseudo-column. ◀

4.5 Dealing with the non-strongly connected case

The following lemma shows that in order to compute the minimum rank we can focus on the strongly connected case.

► **Proposition 17.** Let $M : \Sigma \rightarrow \{0, 1\}^{Q \times Q}$ be an unambiguous matrix monoid morphism. Suppose that $Q_1 \cup Q_2 = Q$ is a partition of Q such that for all $w \in \Sigma^*$ it holds that $[Q_2]^T M(w) [Q_1] = 0$; i.e., for all $w \in \Sigma^*$ matrix $M(w)$ has the block form $M(w) = \begin{pmatrix} M_1(w) & M_{12}(w) \\ 0 & M_2(w) \end{pmatrix}$, where $M_1(w) \in \{0, 1\}^{Q_1 \times Q_1}$ and $M_{12}(w) \in \{0, 1\}^{Q_1 \times Q_2}$ and $M_2(w) \in \{0, 1\}^{Q_2 \times Q_2}$. We have $\text{rank}_{\mathbb{R}}(M) = \text{rank}_{\mathbb{R}}(M_1) + \text{rank}_{\mathbb{R}}(M_2)$ and $\text{rank}_{\text{un}}(M) = \text{rank}_{\text{un}}(M_1) + \text{rank}_{\text{un}}(M_2)$.

By a straightforward induction it follows from Proposition 17 that the minimum rank of an unambiguous matrix monoid is the sum of the minimum ranks of its strongly connected components (where “incomplete” components count as having rank 0).

5 Computing the rank in NC^2

In this section, we prove our first main result, which is as follows.

► **Theorem 18.** The problem of computing the (real) rank of an unambiguous matrix monoid is in NC^2 .

In order to use Theorem 16, we need the following lemma. We use the notation defined in the previous section. Recall that we defined $U := \langle \alpha^T M(w) - \alpha^T \mid w \in \Sigma^* \rangle$.

► **Lemma 19.** If M is strongly connected, one can compute a basis of U in NC^2 .

For each $a \in \Sigma$ define $M'(a) := M(a) - I \in \{-1, 0, 1\}^{Q \times Q}$ and extend M' to $M' : \Sigma^* \rightarrow \mathbb{Z}^{Q \times Q}$ by defining $M'(a_1 \cdots a_k) = M'(a_1) \cdots M'(a_k)$. Define $U' := \langle \alpha^T M'(w) \mid w \in \Sigma^+ \rangle$. Note that here w ranges over Σ^+ , i.e., nonempty words, only. By definition, U' is closed under right multiplication by $M'(a)$ for all $a \in \Sigma$. We first show the following lemma.

► **Lemma 20.** We have $U = U'$.

61:14 Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

Proof. For the inclusion $U \subseteq U'$, we prove by induction on $i \geq 0$ that for all length- i words $w \in \Sigma^i$ we have $\alpha^T(M(w) - I) \in U'$. Concerning the induction base, $i = 0$, we have $\alpha^T(M(\varepsilon) - I) = 0 \in U'$. Concerning the induction step, let $i \geq 0$, and let $w \in \Sigma^i$ and $a \in \Sigma$. We have

$$\begin{aligned} \alpha^T(M(wa) - I) &= \alpha^T(M(w) - I)M(a) + \alpha^T(M(a) - I) \\ &= \alpha^T(M(w) - I)(M(a) - I) + \alpha^T(M(w) - I) + \alpha^T(M(a) - I) \\ &= \alpha^T(M(w) - I)M'(a) + \alpha^T(M(w) - I) + \alpha^T M'(a). \end{aligned}$$

It holds that $\alpha^T M'(a) \in U'$, and, by the induction hypothesis, $\alpha^T(M(w) - I) \in U'$. It follows that $\alpha^T(M(wa) - I) \in U'$.

For the converse, $U' \subseteq U$, we proceed similarly by induction. Concerning the induction base, $i = 1$, for all $a \in \Sigma$ we have $\alpha^T M'(a) = \alpha^T(M(a) - I) \in U$. Concerning the induction step, let $i \geq 1$, and let $w \in \Sigma^i$ and $a \in \Sigma$. By the induction hypothesis there are $n \leq |Q|$ and $w_1, \dots, w_n \in \Sigma^*$ and $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ such that $\alpha^T M'(w) = \sum_{i=1}^n \lambda_i \alpha^T(M(w_i) - I)$. Thus, we have

$$\begin{aligned} \alpha^T M'(wa) &= \alpha^T M'(w)(M(a) - I) = \sum_{i=1}^n \lambda_i \alpha^T(M(w_i) - I)(M(a) - I) \\ &= \sum_{i=1}^n \lambda_i \alpha^T((M(w_i a) - I) - (M(a) - I) - (M(w_i) - I)) \in U, \end{aligned}$$

as required. \blacktriangleleft

Proof of Lemma 19. For each $a \in \Sigma$ define $U'_a := \langle \alpha^T M'(a) M'(w) \mid w \in \Sigma^* \rangle$. Using the technique from [33, Section 4.2] (see [32, Proposition 5.2] for a clearer explanation), for each $a \in \Sigma$ one can compute³ a basis of U'_a in NC^2 . The union of these bases, say $\Gamma = \{\gamma_1^T, \dots, \gamma_n^T\}$ for some $n \leq |\Sigma||Q|$, spans U' , which equals U by Lemma 20. To shrink Γ to a basis of U , for each $i \in \{1, \dots, n\}$ include γ_i^T in the basis if and only if $\dim \langle \gamma_1^T, \dots, \gamma_{i-1}^T \rangle < \dim \langle \gamma_1^T, \dots, \gamma_i^T \rangle$. The latter (rank) computation can be done in NC^2 [27]. \blacktriangleleft

Now we can prove Theorem 18.

Proof of Theorem 18. Let $M : \Sigma \rightarrow \{0, 1\}^{Q \times Q}$ be an unambiguous monoid morphism. Its strongly connected components can be computed in $\text{NL} \subseteq \text{NC}^2$. It follows from the proof of [34, Proposition 3] that one can check each component for completeness in NC^2 , since a zero-one monoid contains the zero matrix if and only if the joint spectral radius of the set of its generators is strictly less than one [34]. Therefore, using Proposition 17, we can assume in the rest of the proof that M is complete and strongly connected.

We use the fact that one can compute a solution of a possibly singular linear system of equations in NC^2 [11, Section 5]. First, compute in NC^2 vectors $\alpha, \beta \in \mathbb{Q}_{>0}^Q$ with $\alpha^T \bar{A} = \alpha^T$ and $\bar{A} \beta = \beta$ and $\alpha^T \beta = 1$. Using Lemma 19 compute in NC^2 a basis of U . Choose an arbitrary $q \in Q$ and compute $\text{Mer}(q)$ in $\text{NL} \subseteq \text{NC}^2$ with a reachability analysis. Then, solve the linear system from Theorem 16 to compute in NC^2 a maximal pseudo-column $y \in \mathbb{Q}^Q$. Hence, using Lemma 14, we can compute $\text{mcw} = \alpha^T y$ in NC^2 . Symmetrically, we can compute mrw in NC^2 . Finally, by Theorem 6, we can compute $r = \frac{1}{\text{mcw} \cdot \text{mrw}}$ in NC^2 . \blacktriangleleft

³ In [33, 32] only membership in NC is claimed, but the bottleneck computations are matrix powering and rank computation, which can in fact be done in $\text{DET} \subseteq \text{NC}^2$; see [18]. The computations in [32, Proposition 5.2] are on polynomially larger matrices, but this does not impact the membership in NC^2 , as $\log^2(\text{poly}(n)) = O(\log^2(n))$.

► **Example 21.** We continue Examples 10 and 15. Since M is a complete DFA, it is natural to take $\beta = [Q]$. Note that $\alpha^T \beta = 1$. Since every row is of the form $[q]^T$ for some q , we have $\text{mrw} = 1$. Recall from Example 15 that $\text{mcw} = \frac{1}{2}$. With Theorem 6 we conclude that $r = \frac{1}{\text{mcw} \cdot \text{mrw}} = 2$, as observed in Example 10.

6 Time and space complexity

In this section, we study the time and space complexity of computing the rank of a zero-one matrix monoid and finding a matrix of minimum rank in it. We provide two approaches. The first one relies on the linear-algebraic tools developed in Section 4. It turns out to be faster than the second, combinatorial, approach, but is limited to only computing the rank. This is due to the fact that we never explicitly construct a maximal column in this approach, which is required in order to find a matrix of minimum rank by Theorem 1. Moreover, it is not known if one can find a maximal column in NC. In contrast, the combinatorial approach explicitly constructs a matrix of minimum rank step by step. In a way, this is exactly the reason why it is slower than the linear-algebraic approach, since this requires performing a large number of matrix multiplications. In the complete DFAs case, where direct matrix multiplication can be avoided due to the special shape of transformation matrices, it becomes much more efficient, and outmatches its linear-algebraic counterpart by a factor of the alphabet size.

The three main results of this section are as follows.

► **Theorem 22.** *The (real) rank of an n -state UFA over an alphabet of size m can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

► **Theorem 23.** *A matrix of minimum (real) rank in an n -state UFA over an alphabet of size m can be found in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

► **Theorem 24.** *A matrix of minimum (real) rank in an n -state complete DFA over an alphabet of size m can be found in $\mathcal{O}(n^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space.*

Until the end of the section, fix a strongly connected complete UFA $\mathcal{A} = (Q, \Sigma, \Delta)$. Denote $n = |Q|$, $m = |\Sigma|$. Section 4.5 shows that strong connectivity can be assumed without loss of generality.

6.1 Square automaton and square digraph

We will need the construction of the square automaton of an NFA. The *square automaton* $\mathcal{A}^{(2)} = (Q^{(2)}, \Sigma, \Delta^{(2)})$ of \mathcal{A} is defined as follows. Let $Q^{(2)} = \{(p, q) \mid p, q \in Q\}$, and for $p, q \in Q$ and $a \in \Sigma$, the transitions are defined component-wise, that is,

$$\Delta^{(2)} = \{((p, q), a, (p', q')) \in Q^{(2)} \times \Sigma \times Q^{(2)} \mid (p, a, p'), (q, a, q') \in \Delta\}.$$

Note that the square automaton of a complete DFA is also a complete DFA.

We call states of the form (q, q) in $\mathcal{A}^{(2)}$ *singletons*. Observe that the restriction of $\mathcal{A}^{(2)}$ to singletons is equal to \mathcal{A} . We denote by $G^{(2)} = (V^{(2)}, E^{(2)})$ the underlying digraph of $\mathcal{A}^{(2)}$ obtained by forgetting the labels of the transitions. Note that $|E^{(2)}| = \mathcal{O}(mn^4)$, and there exists an infinite series of complete UFAs over a two-letter alphabet with $|E^{(2)}| = \Theta(n^4)$ [36, Appendix A]. If \mathcal{A} is a complete DFA, then $|E^{(2)}| = mn^2$.

6.2 Minimum rank in $\mathcal{O}(mn^4)$ time

We now perform the steps of the linear-algebraic algorithm described in Section 5, but implement them efficiently in terms of time and space complexity.

► **Lemma 25.** *For a state p , the set $\text{Mer}(p)$ can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

Proof. Perform a multi-source backwards digraph search starting from all singletons in $G^{(2)}$ and label all states $q \in Q$ such that (p, q) or (q, p) is visited during this search. This search can be performed in time linear in the number of edges of $|E^{(2)}|$, and $|E^{(2)}| = \mathcal{O}(mn^4)$. Observe that only the vertices of this digraph have to be stored in the memory explicitly, since the edges can be computed on the fly from the input without increasing the time complexity, hence the space complexity of the algorithm is $\mathcal{O}(n^2)$. ◀

► **Lemma 26.** *A maximal pseudo-column can be found in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

Proof. To use Theorem 16 we first need to set up the linear system of equations described there. The average matrix \bar{A} can be computed in time $\mathcal{O}(mn^2)$. The weight vectors $\alpha, \beta \in \mathbb{Q}^Q$ can then be computed in time $\mathcal{O}(n^3)$ by solving a system of linear equations. Then, using Lemma 25, we compute in $\mathcal{O}(mn^4)$ time the set $\text{Mer}(p)$ for some state p . We also need to compute a basis of the vector space U defined in Section 4.4. As in the proof of Lemma 19, we compute a basis of $U = U' = \langle \alpha^T M'(w) \mid w \in \Sigma^+ \rangle$, which is the smallest vector space that contains $\alpha^T M(a)$ for all $a \in \Sigma$ and is closed under post-multiplication with $M(a)$ for all $a \in \Sigma$. This can be done in $\mathcal{O}(mn^3)$ time and $\mathcal{O}(n^2)$ space using a worklist algorithm and keeping a basis in echelon form using Gaussian elimination, as described, e.g., in [32, Section 2]. Finally, we solve the system of linear equations from Theorem 16, which can be done in $\mathcal{O}(n^3)$ time. Each step requires at most $\mathcal{O}(n^2)$ space. ◀

By Lemma 14, we thus get that mcw and mrw can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space, which together with Theorem 6 proves Theorem 22. We remark that for complete DFAs the proof of Lemma 26 gives $\mathcal{O}(mn^3)$ time for computing the rank. The combinatorial algorithm provided below will improve it to $\mathcal{O}(n^3 + mn^2)$ (see Section 6.5), while additionally finding a matrix of minimum rank.

6.3 Efficiently constructing a maximal column

We now consider a more general problem of finding a matrix of minimum rank in a zero-one matrix monoid. As mentioned above, by Theorem 1 we have to explicitly construct a maximal column for that, which turns out to be a more difficult task in terms of the time complexity. We will see in the next subsection that we only need one maximal column (together with a word constructing it) to get a matrix of minimum rank. The goal of this subsection is thus to show how to efficiently compute a short representation of a word constructing a maximal column, since the word itself may be longer than the time complexity we are aiming for. We do so by reusing repeating subwords in such a word, and describing their occurrences with a straight line program.

In [20, Section 5], an algorithm for constructing a word of rank one in complete DFAs in $\mathcal{O}(m^3 + mn^2)$ time and $\mathcal{O}(n^2)$ space was suggested. Our approach for finding a maximal column and a word constructing it follows a similar direction, with a few key differences. Firstly, we observe that it is enough to only compute words merging states with one chosen state p , instead of finding all pairs of mergeable states. This both simplifies the algorithm

(in [20] an additional step is required to decrease the space complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$, which we get for free) and allows to present our results in terms of straight line programs, giving a better insight into the regularities present in the constructed word of minimum rank.

We define set straight line programs (set-SLPs), which are just SLPs with multiple initial symbols, and thus encode a set of words instead of one word. Formally, a *set-SLP* is a tuple $(\mathcal{V}, \Sigma, R, \mathcal{S})$, where \mathcal{V} and Σ are disjoint finite sets of *nonterminal* and *terminal* symbols respectively, $R: \mathcal{V} \rightarrow (\mathcal{V} \cup \Sigma)^*$ is a function defining a derivation rule for each nonterminal symbol, and $\mathcal{S} \subseteq \mathcal{V}$ is a set of *initial* symbols. For $v \in \mathcal{V}$, we write $R(v)$ as $v \rightarrow w$ with $w \in (\mathcal{V} \cup \Sigma)^*$, and we call v and w the left- and right-hand sides of this derivation rule respectively. The *length* of a set-SLP is the total length of the right-hand sides of all the derivation rules. The semantics of a set-SLP is defined as follows. Given an initial symbol $s \in \mathcal{S}$, we recursively replace each symbol in $R(s)$ with the right-hand side of its derivation rule until we obtain a word over Σ , which is called *the word encoded by s* . We require that each initial symbol produces a unique word over Σ as a result of such derivation. Namely, we require that there exists a total linear order \leq on the set \mathcal{V} such that for all v with $v \rightarrow w$, w does not contain $v' \in \mathcal{V}$ with $v' \leq v$. The (multi-)set of words encoded by all initial symbols is called *the set of words encoded by a set-SLP*.

► **Example 27.** Consider a set-SLP $(\{w_1, w_3, w_5, u_1, u_2, u_3\}, \{a, b\}, R, \{w_1, w_3, w_5\})$ with

$$w_1 \rightarrow u_1, w_3 \rightarrow u_3 u_2, w_5 \rightarrow u_2, u_1 \rightarrow aab, u_2 \rightarrow aab, u_3 \rightarrow aaba.$$

This set-SLP encodes the (multi-)set $\{aab, aabaaab, aab\}$, and illustrates the reason why we are using set-SLPs: they allow to construct sets of words out of smaller “pieces” without having to explicitly repeat these “pieces” multiple times (in our example, we are reusing u_2). Note that the set-SLPs that we construct below only encode sets of words whose total length is polynomial in the size of the set-SLPs.

► **Lemma 28.** *Given a state p , a set-SLP of length $\mathcal{O}(n^2)$ defining a set $\{w_q \mid q \in \text{Mer}(p)\}$, where w_q is a word with $p \in p \cdot w_q$ and $p \in q \cdot w_q$, can be computed in $\mathcal{O}(mn^4)$ time and $\mathcal{O}(n^2)$ space.*

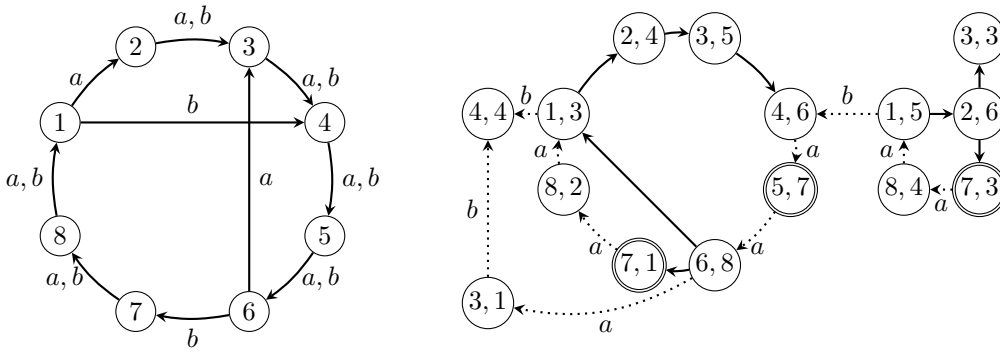
Proof sketch. Call vertices (p, q) with $q \in \text{Mer}(p)$ *merging*. The idea is to construct, by a digraph search of $G^{(2)}$, a directed tree T rooted in (p, p) and containing a path from each merging vertex to the root, and then use the joint subpaths of these paths in the tree to obtain a short set-SLP describing these paths. See Figure 5 for an example. ◀

► **Lemma 29.** *For a given state p of \mathcal{A} , an SLP of length $\mathcal{O}(n^2)$ encoding a word w such that $[w \cdot p]$ is a maximal column can be computed in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

Proof sketch. Compute the matrices of the words encoded by the set-SLP from Lemma 28. We rely on the property, already used in some form in [14], that if for all $q \in \text{Mer}(p)$, $q \neq p$, the vector $[w \cdot q]$ is zero, then $[w \cdot p]$ is a maximal column. To construct a word with this property, we iteratively concatenate the words w_q depending on nonzero columns in the matrix in each iteration. The number of iterations is bounded by n . ◀

6.4 Finding a matrix of minimum rank

We now use the results of the previous section to construct a matrix of minimum rank. The key idea is as follows: since the set of maximal columns is stable under left multiplication by matrices from the monoid, we can iteratively make each column of the matrix maximal or



■ **Figure 5** An example of \mathcal{A} (left), and a part of the underlying digraph $G^{(2)}$ of its square automaton (right). Merging vertices are doubly circled. The edges of T for $p = 7$ are represented by dotted edges, and these edges are labelled with one of the letters labelling the corresponding transition in $\mathcal{A}^{(2)}$. Furthermore, we have $\rho_1 = (7, 1) \rightarrow (8, 2) \rightarrow (1, 3) \rightarrow (4, 4)$, $\rho_2 = (5, 7) \rightarrow (6, 8) \rightarrow (3, 1) \rightarrow (4, 4)$, $\rho_3 = (7, 3) \rightarrow (8, 4) \rightarrow (1, 5) \rightarrow (4, 6) \rightarrow (5, 7)$. A set-SLP encoding the labels of these path is presented in Example 27, with u_i labelling the path ρ_i , $i \in \{1, 2, 3\}$.

zero by, intuitively, applying the same word (together with a short “reachability” word) to a state in each column. This simple observation significantly decreases the time complexity of our algorithm compared to the naive implementation of the algorithm from [14] constructing a word of minimum rank. Indeed, in the approach of [14], the word is constructed letter by letter, and requires to know the result of applying the last letter at every step. Since the constructed word has length $\mathcal{O}(rn^3) = \mathcal{O}(n^4)$, where n is the number of states and r is the rank, this results in $\mathcal{O}(n^{4+\omega})$ time complexity. By efficiently constructing only one maximal column (as described in the previous section) and reusing it for the whole set of states (as described in this section), we decrease the time complexity to $\mathcal{O}(n^{2+\omega})$.

► **Proposition 30.** *An SLP of length $\mathcal{O}(n^2)$ encoding a word w of minimum rank can be computed in $\mathcal{O}(n^{2+\omega} + mn^4)$ time and $\mathcal{O}(n^3)$ space.*

Proof sketch. Compute the matrix of the word w encoded by the SLP from Lemma 29. Iteratively, for each $q \in Q$, concatenate w with a word of length at most n mapping p to a state corresponding to a nonzero element of the row in the current iteration. Denote by w_n the resulting word, which has the property that all nonzero columns of $M(w_n)$ are maximal. Symmetrically compute w'_n for rows. Then by Lemma 5 the word $w_n w'_n w_n w'_n$ matrix has minimum rank. ◀

Given an SLP of length $\mathcal{O}(n^2)$ encoding a word w , we can compute the matrix of w by computing the matrices of words occurring in the derivation of w from bottom to top in time $\mathcal{O}(n^{2+\omega})$. Thus we prove Theorem 23. We also get the following result, which can be seen as a proof of a very weak version of the Černý conjecture generalised from rank one words in complete DFAs to minimum rank words in complete UFAs.

► **Theorem 31.** *For every n -state complete UFA, there exists an SLP of length $\mathcal{O}(n^2)$ encoding a word of minimum rank.*

We remark that the length of the word encoded by the constructed SLP asymptotically matches the best known upper bound for words of minimum rank: $\mathcal{O}(n^4)$ for complete UFAs [14] and $\mathcal{O}(n^3)$ for complete DFAs [37]. In particular, one can efficiently compute words of minimum rank within these bounds.

6.5 Complete DFAs

For complete DFAs, we follow the same algorithms as in the proof of Theorem 23, but exploit the fact that elementary matrix operations can be performed more efficiently. Namely, if \mathcal{A} is a complete DFA, then each word defines a transformation on Q . By storing matrices of words as transformations, we get that matrix multiplication can be performed in $\mathcal{O}(n)$ time, and each matrix requires $\mathcal{O}(n)$ space. Moreover, we have $|E^{(2)}| = mn^2$. By taking these improvements into account, we get the proof of Theorem 24.

7 Conclusions and open problems

We list a few open questions that follow directly from our work.

- In [20], it was asked if a word of rank one for a complete DFA can be found in NC. Similarly, can a matrix of minimum rank for a complete DFA be computed in NC?
- Given an unambiguous morphism $M: \Sigma \rightarrow \{0, 1\}^{Q \times Q}$ and a vector $\alpha \in \mathbb{Q}_{>0}^Q$, can a basis of $\langle \alpha^T M(w) \mid w \in \Sigma^* \rangle$ be computed faster than in $\mathcal{O}(|Q|^3)$ time? This would improve algorithms for several fundamental problems for weighted automata [32]. Similarly, for complete DFAs, computing a basis of U from Section 4.4 in subcubic time (see the proof of Lemma 26) would allow to compute the minimum rank of a complete DFA faster than in cubic time.
- The bottleneck in the time complexity in Theorem 22 is the very first step, computing $\text{Mer}(q)$ via digraph search in the square digraph of \mathcal{A} . The number of edges of this digraph can be quadratic in the number of its vertices [36, Appendix A], hence of order $|Q|^4$. Can $\text{Mer}(q)$ be computed faster than in time $\mathcal{O}(|Q|^4)$? Very little seems to be known about general properties of square automata of DFAs or UFAs.
- One of the bottlenecks of the combinatorial algorithm is performing matrix multiplication. Can it be done faster for matrices from a zero-one matrix monoid? If not, are there any subclasses of UFAs (other than DFAs) where it can be done more efficiently?

References

- 1 Jorge Almeida and Benjamin Steinberg. Matrix mortality and the Černý-Pin conjecture. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory*, pages 67–80, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02737-6_5.
- 2 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- 3 Christel Baier, Stefan Kiefer, Joachim Klein, David Müller, and James Worrell. Markov chains and unambiguous automata. *Journal of Computer and System Sciences*, 136:113–134, 2023. doi:10.1016/J.JCSS.2023.03.005.
- 4 M.-P. Béal and D. Perrin. Synchronised automata. In Valérie Berthé and Michel Rigo, editors, *Combinatorics, Words and Symbolic Dynamics*, Encyclopedia of Mathematics and its Applications, pages 213–240. Cambridge University Press, 2016. doi:10.1017/CB09781139924733.008.
- 5 Marie-Pierre Béal, Eugen Czeizler, Jarkko Kari, and Dominique Perrin. Unambiguous automata. *Math. Comput. Sci.*, 1(4):625–638, 2008. doi:10.1007/S11786-007-0027-1.
- 6 Paul C. Bell, Igor Potapov, and Pavel Semukhin. On the mortality problem: From multiplicative matrix equations to linear recurrence sequences and beyond. *Information and Computation*, 281:104736, 2021. doi:10.1016/J.IC.2021.104736.
- 7 Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

- 8 Mikhail V. Berlinkov. On two algorithmic problems about synchronizing automata (short paper). In Arseny M. Shur and Mikhail V. Volkov, editors, *Developments in Language Theory – 18th International Conference, DLT 2014, Ekaterinburg, Russia, August 26-29, 2014. Proceedings*, volume 8633 of *Lecture Notes in Computer Science*, pages 61–67. Springer, 2014. doi:10.1007/978-3-319-09698-8_6.
- 9 Jean Berstel, Dominique Perrin, and Christophe Reutenauer. *Codes and automata*, volume 129. Cambridge University Press, 2010.
- 10 Vincent D. Blondel, Raphaël M. Jungers, and Alex Olshevsky. On primitivity of sets of matrices. *Automatica*, 61:80–88, 2015. doi:10.1016/J.AUTOMATICA.2015.07.026.
- 11 Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982. doi:10.1016/S0019-9958(82)90766-5.
- 12 Greg Budzban and Philip Feinsilver. The generalized road coloring problem and periodic digraphs. *Applicable Algebra in Engineering, Communication and Computing*, 22:21–35, 2011. doi:10.1007/S00200-010-0135-Z.
- 13 Georgina Bumpus, Christoph Haase, Stefan Kiefer, Paul-Ioan Stoienescu, and Jonathan Tanner. On the size of finite rational matrix semigroups. 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 115:1–115:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.115.
- 14 Arturo Carpi. On synchronizing unambiguous automata. *Theoretical Computer Science*, 60:285–296, 1988. doi:10.1016/0304-3975(88)90114-4.
- 15 Arturo Carpi and Flavio D’Alessandro. Strongly transitive automata and the Černý conjecture. *Acta Informatica*, 46(8):591–607, 2009. doi:10.1007/S00236-009-0106-7.
- 16 Julien Cassaigne, Vesa Halava, Tero Harju, and François Nicolas. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more. *CoRR*, abs/1404.0644, 2014. arXiv:1404.0644.
- 17 Yves Césari. Sur l’application du théorème de Suschkewitsch à l’étude des codes rationnels complets. In Jacques Loeckx, editor, *Automata, Languages and Programming, 2nd Colloquium, University of Saarbrücken, Germany, July 29 - August 2, 1974, Proceedings*, volume 14 of *Lecture Notes in Computer Science*, pages 342–350. Springer, 1974. doi:10.1007/3-540-06841-4_73.
- 18 Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Inf. Control*, 64(1-3):2–21, 1985. doi:10.1016/S0019-9958(85)80041-3.
- 19 Laszlo Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976. doi:10.1137/0205040.
- 20 David Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990. doi:10.1137/0219033.
- 21 Joel Friedman. On the road coloring problem. *Proceedings of the American Mathematical Society*, 110(4):1133–1135, 1990.
- 22 Balázs Gerencsér, Vladimir V. Gusev, and Raphaël M. Jungers. Primitive sets of nonnegative matrices and synchronizing automata. *SIAM Journal on Matrix Analysis and Applications*, 39(1):83–98, 2018. doi:10.1137/16M1094099.
- 23 Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. doi:10.1017/CB09780511804106.
- 24 Pavel Goralčík and Václav Koubek. Rank problems for composite transformations. *International Journal of Algebra and Computation*, 05(03):309–316, 1995. doi:10.1142/S0218196795000185.
- 25 Vladimir V. Gusev and Elena V. Pribavkina. On synchronizing colorings and the eigenvectors of digraphs. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, volume 58 of *LIPICs*, pages 48:1–48:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.MFCS.2016.48.

- 26 Markus Holzer and Sebastian Jakobi. On the computational complexity of problems related to distinguishability sets. *Information and Computation*, 259(2):225–236, 2018. doi:10.1016/J.IC.2017.09.003.
- 27 Oscar H. Ibarra, Shlomo Moran, and Louis E. Rosier. A note on the parallel complexity of computing the rank of order n matrices. *Information Processing Letters*, 11(4/5):162, 1980. doi:10.1016/0020-0190(80)90042-3.
- 28 Gérard Jacob. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science*, 5(2):183–204, 1977. doi:10.1016/0304-3975(77)90006-8.
- 29 Jarkko Kari. A counter example to a conjecture concerning synchronizing words in finite automata. *Bulletin of the EATCS*, 73:146, 2001.
- 30 Jarkko Kari, Andrew Ryzhikov, and Anton Varonka. Words of minimum rank in deterministic finite automata. In Piotrek Hofman and Michal Skrzypczak, editors, *Developments in Language Theory – 23rd International Conference, DLT 2019, Warsaw, Poland, August 5-9, 2019, Proceedings*, volume 11647 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2019. doi:10.1007/978-3-030-24886-4_5.
- 31 Nasim Karimi. Reaching the minimum ideal in a finite semigroup. *Semigroup Forum*, 94(2):390–425, 2017.
- 32 Stefan Kiefer. Notes on equivalence and minimization of weighted automata. <https://arxiv.org/abs/2009.01217>, 2020. arXiv:arXiv:2009.01217.
- 33 Stefan Kiefer, Ines Marusic, and James Worrell. Minimisation of multiplicity tree automata. *Logical Methods in Computer Science*, 13(1), 2017. doi:10.23638/LMCS-13(1:16)2017.
- 34 Stefan Kiefer and Corto N. Mascle. On nonnegative integer matrices and short killing words. *SIAM Journal on Discrete Mathematics*, 35(2):1252–1267, 2021. doi:10.1137/19M1250893.
- 35 Stefan Kiefer and Cas Widdershoven. Efficient analysis of unambiguous automata using matrix semigroup techniques. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPICs*, pages 82:1–82:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.MFCS.2019.82.
- 36 Stefan Kiefer and Cas Widdershoven. Efficient analysis of unambiguous automata using matrix semigroup techniques. *CoRR*, abs/1906.10093, 2019. arXiv:1906.10093.
- 37 A.A. Klyachko, I.K. Rystsov, and M.A. Spivak. An extremal combinatorial problem associated with the bound on the length of a synchronizing word in an automaton. *Cybernetics*, 23(2):165–171, 1987.
- 38 Douglas A. Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge university press, 2021.
- 39 Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theoretical Computer Science*, 5(2):101–111, 1977. doi:10.1016/0304-3975(77)90001-9.
- 40 Mike Paterson. Unsolvability in 3×3 matrices. *Studies in Applied Mathematics*, 49:105–107, 1970.
- 41 Vladimir Yu. Protasov. Analytic methods for reachability problems. *Journal of Computer and System Sciences*, 120:1–13, 2021. doi:10.1016/J.JCSS.2021.02.007.
- 42 Vladimir Yu. Protasov and Andrey S. Voynov. Matrix semigroups with constant spectral radius. *Linear Algebra and its Applications*, 513:376–408, 2017.
- 43 Igor Rystsov. Rank of a finite automaton. *Cybernetics and Systems Analysis*, 28(3):323–328, 1992.
- 44 Igor K. Rystsov. Reset words for commutative and solvable automata. *Theoretical Computer Science*, 172(1-2):273–279, 1997. doi:10.1016/S0304-3975(96)00136-3.
- 45 Wojciech Rytter. The space complexity of the unique decipherability problem. *Information Processing Letters*, 23(1):1–3, 1986. doi:10.1016/0020-0190(86)90121-3.
- 46 Andrew Ryzhikov. Mortality and synchronization of unambiguous finite automata. In Robert Mercas and Daniel Reidenbach, editors, *Combinatorics on Words – 12th International Conference, WORDS 2019, Loughborough, UK, September 9-13, 2019, Proceedings*, volume 11682 of *Lecture Notes in Computer Science*, pages 299–311. Springer, 2019. doi:10.1007/978-3-030-28796-2_24.

61:22 Minimum Rank of a Matrix in a Monoid of Zero-One Matrices

- 47 Andrew Ryzhikov. On shortest products for nonnegative matrix mortality. In Laura Kovács and Ana Sokolova, editors, *Reachability Problems – 18th International Conference, RP 2024, Vienna, Austria, September 25-27, 2024, Proceedings*, volume 15050 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2024. doi:10.1007/978-3-031-72621-7_8.
- 48 Sujin Shin and Jisang Yoo. A note on the rank of semigroups. *Semigroup Forum*, 81(2):335–343, 2010.
- 49 Mikhail V. Volkov. Synchronizing automata and the Černý conjecture. In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008.
- 50 Mikhail V. Volkov. Synchronization of finite automata. *Russian Mathematical Surveys*, 77(5):819–891, 2022. doi:10.4213/rm10005e.
- 51 Yaokun Wu and Yinfeng Zhu. Primitivity and Hurwitz primitivity of nonnegative matrix tuples: A unified approach. *SIAM Journal on Matrix Analysis and Applications*, 44(1):196–211, 2023. doi:10.1137/22M1471535.

Faster Algorithms on Linear Delta-Matroids

Tomohiro Koana  

Utrecht University, The Netherlands

Research Institute for Mathematical Sciences, Kyoto University, Japan

Magnus Wahlström  

Royal Holloway, University of London, UK

Abstract

We present new algorithms and constructions for linear delta-matroids. Delta-matroids are generalizations of matroids that also capture structures such as matchable vertex sets in graphs and path-packing problems. As with matroids, an important class of delta-matroids is given by linear delta-matroids, which generalize linear matroids and are represented via a “twist” of a skew-symmetric matrix. We observe an alternative representation, termed a *contraction representation* over a skew-symmetric matrix. This representation is equivalent to the more standard twist representation up to $O(n^\omega)$ -time transformations (where n is the dimension of the delta-matroid and $\omega < 2.372$ the matrix multiplication exponent), but it is much more convenient for algorithmic tasks. For instance, the problem of finding a max-weight feasible set now reduces directly to finding a max-weight basis in a linear matroid. Supported by this representation, we provide new algorithms and constructions for linear delta-matroids. In particular, we show that the *union* and *delta-sum* of linear delta-matroids are again linear delta-matroids, and that a representation for the resulting delta-matroid can be constructed in randomized time $O(n^\omega)$ (or more precisely, in $O(n^\omega)$ field operations, over a field of size at least $\Omega(n \cdot (1/\varepsilon))$, where $\varepsilon > 0$ is an error parameter). Previously, it was only known that these operations define delta-matroids. We also note that every *projected* linear delta-matroid can be represented as an *elementary* projection. This implies that several optimization problems over (projected) linear delta-matroids, including the coverage, delta-coverage, and parity problems, reduce (in their decision versions) to a single $O(n^\omega)$ -time matrix rank computation. Using the methods of Harvey, previously applied by Cheung, Lao and Leung for linear matroid parity, we furthermore show how to solve the search versions in the same time. This improves on the $O(n^4)$ -time augmenting path algorithm of Geelen, Iwata and Murota, albeit with randomization. Finally, we consider the maximum-cardinality delta-matroid intersection problem (equivalently, the maximum-cardinality delta-matroid matching problem). Using Storjohann’s algorithms for symbolic determinants, we show that such a solution can be found in $O(n^{\omega+1})$ time. This provides the first (randomized) polynomial-time solution for the problem, thereby solving an open question of Kakimura and Takamatsu.

2012 ACM Subject Classification Mathematics of computing → Matroids and greedoids

Keywords and phrases Delta-matroids, Randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.62

Related Version *Full Version*: <https://arxiv.org/abs/2402.11596>

Funding *Tomohiro Koana*: Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (project CRACKNP under grant agreement No. 853234). Supported by JSPS KAKENHI Grant Number JP20H05967.

1 Introduction

Matroids are important unifying structures in many parts of computer science and discrete mathematics, abstracting and generalizing notions from linear vector spaces and graph theory; see, e.g., Oxley [30] and Schrijver [32]. Formally, a matroid is a collection of *independent sets*, subject to particular axioms (see below). A maximum independent set is a *basis*. Among



© Tomohiro Koana and Magnus Wahlström;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 62; pp. 62:1–62:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



other things, matroids are a very useful source of algorithmic meta-results, since there are many problems on matroids which admit efficient, general-purpose algorithms – such as the greedy algorithm for finding a max-weight basis, generalizing algorithms for max-weight spanning trees; or MATROID INTERSECTION, the problem of finding a maximum common feasible set in two given matroids, which generalizes bipartite matching.

An important class of matroids are *linear matroids*, where independence in the matroid is represented by the column space of a matrix. Linear matroids enjoy many properties not shared by generic matroids. For example, the famous MATROID PARITY problem, which generalizes the matching problem in general graphs, is known to be intractable in the general case but efficiently solvable over linear matroids [26]. In addition, linear representations, as a compact representation of combinatorial information, have seen many applications in parameterized complexity for purposes of *sparsification* and *kernelization* [24, 25], and algebraic algorithms over linear matroids have proven a very useful general-purpose tool in FPT algorithms [14, 12] (cf. [8, 15]).

Delta-matroids are a generalization of matroids, where, informally, the notion of bases is replaced by the notion of *feasible sets*, which satisfy an exchange axiom similar to matroid bases but need not all have the same cardinality. Delta-matroids were introduced by Bouchet (although similar structures were independently defined by others), and have connections to multiple areas of computer science such as structural and topological graph theory [28], constraint satisfaction problems [13, 22], matching and path-packing problems and more. Like with matroids, there is also a notion of *linear delta-matroids*, where the feasible sets are represented through a skew-symmetric matrix. These generalize linear matroids, although this fact (or indeed the fact that skew-symmetric matrices define delta-matroids) is not elementary [17]. Delta-matroids (linear or otherwise) are remarkably flexible structures, in that there are many ways to modify or combine given delta-matroids into new delta-matroids, including twisting (partial dualization), contraction and deletion, existential projection, and unions and delta-sums of delta-matroids (all described below).

Similarly to matroids, there is also a range of generic problems that have been considered over delta-matroids, including delta-matroid intersection, partition, and parity problems. Unfortunately, due to the generality of delta-matroids, these problems are all intractable in the general case, since they generalize matroid parity. However, they are tractable on linear delta-matroids, where Geelen et al. [17] gave an algorithm (and a corresponding min-max theorem) with a running time of $O(n^4)$, and $O(n^{\omega+1})$ using fast matrix multiplication. However, other variants remain open. Kakimura and Takamatsu [21] considered the maximum cardinality version of delta-matroid parity (as opposed to the result of Geelen et al. [17], which is more of a *feasibility* or *minimum error* version). They gave a solution for a restricted class of linear and projected linear delta-matroid, but left the general case open. Furthermore, the natural weighted optimization variants of the above appear completely open.

In this paper, we show new constructions of linear delta-matroids and new and faster randomized algorithms for the aforementioned problems on linear and projected linear delta-matroids. In particular, we show a new representation variant for linear delta-matroids – dubbed *contraction representation*, as opposed to the standard *twist representation* – which appears more amenable to efficient algorithms. Using this representation, we show for the first time that unions and delta-sums of linear delta-matroids (represented over a common field \mathbb{F}) define *linear* delta-matroids, and that a representation can be constructed in randomized polynomial time. We also show new algorithmic results, including solving the search version of LINEAR DELTA-MATROID PARITY (LINEAR DM PARITY for short) in $O(n^\omega)$ field operations¹

¹ Throughout, we give our running times as field operations. If the field has size $n^{O(1)}$, as in most applications, then this is just polylogarithmic overhead.

and giving the first (randomized) polynomial-time algorithm for the maximum cardinality version of the problem in $O(n^{\omega+1})$ field operations, thereby settling an open question from Kakimura and Takamatsu [21].

1.1 Introduction to delta-matroids

Before we describe our results in detail, let us review some background on delta-matroids. For more material, we refer to the survey by Moffatt [28].

Like matroids, delta-matroids are formally defined as set systems satisfying particular axioms. Formally, a delta-matroid is a pair $D = (V, \mathcal{F})$ where V is a ground set and $\mathcal{F} \subseteq 2^V$ a non-empty collection of subsets of V , referred to as *feasible sets* in D , subject to the following *symmetric exchange axiom*:

For all $F_1, F_2 \in \mathcal{F}$ and $x \in F_1 \Delta F_2$ there exists $y \in F_1 \Delta F_2$ such that $F_1 \Delta \{x, y\} \in \mathcal{F}$,

where Δ denotes symmetric difference.

It should be enlightening to compare this to the definition of matroids. Formally, a matroid is most commonly defined as a collection of *independent sets*; i.e., a matroid is defined as a pair $M = (V, \mathcal{I})$ where V is the ground set and $\mathcal{I} \subseteq 2^V$ is a collection of sets, referred to as independent sets in M , subject to (1) $\emptyset \in \mathcal{I}$; (2) if $B \in \mathcal{I}$ and $A \subset B$ then $A \in \mathcal{I}$; and (3) if $A, B \in \mathcal{I}$ with $|A| < |B|$ then there exists an element $x \in B \setminus A$ such that $A + x \in \mathcal{I}$. The second condition encodes that (V, \mathcal{I}) is an *independence system*. However, matroids can also be equivalently defined from just the collection of *maximal* independent sets, known as *bases*. Under this definition, a matroid is a pair $M = (V, \mathcal{B})$ where $\mathcal{B} \subseteq 2^V$ is a non-empty collection of bases, subject to the *basis exchange property*:

For all $A, B \in \mathcal{B}$ and $x \in A \setminus B$ there exists $y \in B \setminus A$ such that $A \Delta \{x, y\} \in \mathcal{B}$.

In particular, all bases of a matroid have the same cardinality. Thus, delta-matroids can be seen as the relaxation of matroids where the feasible sets (analogous to the bases) need not all have the same cardinality. In fact, a delta-matroid where all feasible sets have the same cardinality is precisely a matroid (represented as a set of bases). A similar statement holds for independent sets – the independent sets of a matroid form the feasible sets of a delta-matroid, and a delta-matroid which is an independence system is precisely a matroid in this sense – but the formulation from the set of bases is standard and more convenient.

As a further illustration, consider the case of graph matchings. Let $G = (V, E)$ be a graph. The *matching matroid* of G is a matroid over ground set V where a set $B \subseteq V$ is a basis if and only if it is the set of endpoints of a maximum matching of G . Correspondingly, the independent sets $S \subseteq V$ of the matching matroid are vertex sets that can be covered by a matching. On the other hand, the *matching delta-matroid* over G is the delta-matroid where a set $S \subseteq V$ is feasible if and only if $G[S]$ has a perfect matching. Thus, the maximal feasible sets of the matching delta-matroid form the bases of the matching matroid, but clearly, the matching delta-matroid captures more of the structure of G than the matching matroid does.

Linear delta-matroids. As with matroids, an important class of delta-matroids are *linear delta-matroids*. A matrix A is *skew-symmetric* if $A^T = -A$. Let A be a skew-symmetric matrix with rows and columns indexed by a set V . Then A defines a delta-matroid $\mathbf{D}(A) = (V, \mathcal{F})$ where for $S \subseteq V$ we have $S \in \mathcal{F}$ if and only if the principal submatrix of A indexed by S , denoted by $A[S]$, is non-singular. We refer to $\mathbf{D}(A)$ as a *directly represented* linear delta-matroid. More generally, the *twist* of a delta-matroid $D = (V, \mathcal{F})$ by a set $S \subseteq V$, denoted $D \Delta S$, is the delta-matroid with feasible sets

$$\mathcal{F} \Delta S := \{F \Delta S \mid F \in \mathcal{F}\}.$$

It is easy to check that $D\Delta S$ is a delta-matroid. The twisting operation is also known as *partial dualization*, since the twist $D^* := D\Delta V$ corresponds to the dualization M^* of a matroid M . A general representation of a linear delta-matroid is given as $D = \mathbf{D}(A)\Delta S$ for some skew-symmetric matrix A and twisting set S . A delta-matroid D is *even* if all feasible sets have the same cardinality modulo 2; all linear delta-matroids are even. In addition, we consider *projected linear delta-matroids*, which is a delta-matroid $D = (V, \mathcal{F})$ defined via existential projection over a set X from a larger linear delta-matroid $D' = \mathbf{D}(A)\Delta S$ over ground set $V \cup X$. We denote this $D = D'|X$, where D has the feasible set

$$\mathcal{F} = \{F \setminus X \mid F \in \mathcal{F}(D')\}.$$

As a canonical example, the matching delta-matroid of a graph G is directly represented by the Tutte matrix over G . The set of bases of a linear matroid forms a linear delta-matroid, and the independent sets form a projected linear delta-matroid, under natural representations (see Section 3.1 of the full version of the paper).

Underpinning algorithms on linear delta-matroids are a number of fundamental operations on skew-symmetric matrices. For a skew-symmetric matrix A indexed by V and a set $S \subseteq V$ such that $A[S]$ is non-singular, there is a *pivoting* operation that constructs a new skew-symmetric matrix $A' = A * S$ such that for any $U \subseteq V$, $A'[U]$ is non-singular if and only if $A[S\Delta U]$ is. Via this operation, linear delta-matroids are closed under the *contraction* operation D/T as well as *deletion* $D \setminus T$ (see Section 2 for the definitions). Another fundamental property of skew-symmetric matrices is the *Pfaffian*, defined as follows. Let A be a skew-symmetric matrix with rows and columns indexed by V . The *support graph* of A is the graph $G = (V, E)$ where $uv \in E$ if and only if $A[u, v] \neq 0$. Then the Pfaffian of A is defined as

$$\text{Pf } A = \sum_M \sigma(M) \prod_{e \in M} A[u, v],$$

where M ranges over all perfect matchings in G and $\sigma(M) \in \{1, -1\}$ is a sign term. It holds that $\det A = (\text{Pf } A)^2$, thus A is non-singular if and only if $\text{Pf } A \neq 0$. Via this connection to matchings, the Pfaffian forms a link between the combinatorial and algebraic aspects of linear delta-matroids, in a way that is often exploited in this paper. The Pfaffian also enjoys some useful algebraic properties, such as the Pfaffian sum formula and the Ishikawa-Wakayama formula, with clear combinatorial interpretations. See Section 2 for details.

1.2 Our results

We show a range of results regarding the representation and construction of linear delta-matroids, and new and faster algorithms for computational problems over them. We discuss these in turn.

Representations and constructions. Our first result, which supports the others, is the introduction of a new representation for linear delta-matroids. Recall that a linear delta-matroid $D = (V, \mathcal{F})$ is represented as $D = \mathbf{D}(A)\Delta S$ for a skew-symmetric matrix A with rows and columns indexed by V . We refer to this as a *twist representation*. Although this representation is intimately connected to the structure of delta-matroids, it is less convenient for algorithmic purposes. For this, we introduce the *contraction representation*, representing a linear delta-matroid $D = (V, \mathcal{F})$ as $D = \mathbf{D}(A)/T$ for a skew-symmetric matrix A indexed by $V \cup T$. Thus, a set $F \subseteq V$ is feasible in D if and only if $A[F \cup T]$ is non-singular.

We show that the representations are equivalent, and given a representation in one form, we can efficiently and deterministically construct one in the other; see Section 3.1. Thus contraction representations do not change the class of representable delta-matroids; however, we find that the contraction representation is more compatible with the algorithmic methods of linear algebra.

Next, we consider two methods of composing linear delta-matroids. Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be given delta-matroids (padding the ground sets with dummy elements if necessary so that they are defined over the same ground set V). The *union* $D_1 \cup D_2$ is the delta-matroid $D = (V, \mathcal{F})$ where

$$\mathcal{F} = \{F_1 \cup F_2 \mid F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, F_1 \cap F_2 = \emptyset\},$$

i.e., the feasible sets in D are the disjoint unions of feasible sets in D_1 and D_2 . Additionally, the *delta-sum* $D_1 \Delta D_2$ is defined as the delta-matroid $D = (V, \mathcal{F})$ with feasible sets

$$\mathcal{F} = \{F_1 \Delta F_2 \mid F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2\},$$

where Δ denotes symmetric difference. Bouchet [2] and Bouchet and Schwärzler [4] showed that $D_1 \cup D_2$ and $D_1 \Delta D_2$ are delta-matroids. Using properties of Pfaffians and the contraction representation, we show that furthermore, the union and delta-sum of linear delta-matroids are linear delta-matroids.

The construction is randomized, and takes an error parameter $\varepsilon > 0$ which controls the size of the field that the output delta-matroid is represented over. For this purpose, we say that an algorithm constructs an ε -*approximate representation* of a delta-matroid $D = (V, \mathcal{F})$ if it constructs a representation of a delta-matroid $D' = (V, \mathcal{F}')$ where $\mathcal{F}' \subseteq \mathcal{F}$ and for every $F \in \mathcal{F}$ the probability that $F \in \mathcal{F}'$ is at least $1 - \varepsilon$. Setting $\varepsilon = O(1/2^n)$ where $n = |V|$ gives a representation that with good probability is correct for all subsets. However, this leads to a prohibitive field size, with significant overhead cost per field operation. Thus, for algorithmic applications, a smaller value of ε may be faster and sufficient.

► **Theorem 1.** *Let D_1 and D_2 be delta-matroids represented over a common field \mathbb{F} , and let $\varepsilon > 0$ be given. Let \mathbb{F}' be an extension field of \mathbb{F} with at least $n \cdot \lceil 1/\varepsilon \rceil$ elements. Then the delta-matroid union $D_1 \cup D_2$ and delta-sum $D_1 \Delta D_2$ are linear, and ε -approximate representations over \mathbb{F}' can be constructed in $O(n^2)$ respectively $O(n^\omega)$ field operations.*

Algorithms. As a warm-up, we first consider the problem of finding a max-weight feasible set in a given delta-matroid $D = (V, \mathcal{F})$ with element weights $w: V \rightarrow \mathbb{R}$. Note that the weights may be negative, and since not all feasible sets have the same cardinality, unlike in matroids, they cannot simply be raised to be non-negative. Bouchet [3, 1] showed that there is a variant of greedy algorithm that solves this problem using only separation oracle calls. However, this requires $O(n)$ separation oracle calls, each of which requires $O(n^\omega)$ field operations in linear representation. We show that, using the contraction representation, the max-weight feasible set problem in a linear delta-matroid reduces to finding a max-weight column basis of an $O(n) \times O(n)$ matrix, which can be done significantly faster.

► **Theorem 2.** *Let $D = (V, \mathcal{F})$ be a linear or projected linear delta-matroid. In $O(n^\omega)$ field operations, we can find a max-weight feasible set in D .*

For more intricate questions, the literature contains a range of problems over delta-matroids [3, 17, 29]. The most important are arguably the following.

- DM INTERSECTION: Given delta-matroids $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$, find a common feasible set $F \in \mathcal{F}_1 \cap \mathcal{F}_2$.
- DM PARITY: Given a delta-matroid $D = (V, \mathcal{F})$ and a partition Π of V into pairs, is there a feasible set in D which is the union of pairs? More generally, find a feasible set $F \in \mathcal{F}$ to minimize the number of broken pairs, i.e., the number of pairs $p \in \Pi$ with $|p \cap F| = 1$.

Recall that the matroid versions differ in complexity; MATROID INTERSECTION is tractable in the oracle model while MATROID PARITY is intractable in general but tractable for linear matroids. However, due to the flexibility of delta-matroids, DM INTERSECTION and DM PARITY are equivalent under efficient transformations [17], hence both intractable in general. Given an instance (D_1, D_2) of DM INTERSECTION, a reduction to DM PARITY is immediate by constructing the disjoint union of D_1 and D_2 [17]. In the other direction, given a delta-matroid $D = (V, \mathcal{F})$ and a partition Π of V into pairs, let D_Π be the matching delta-matroid of the graph with edge set Π . Thus the feasible sets of D_Π are precisely the sets $S \subseteq V$ with no broken pairs, and the DM PARITY instance (D, Π) has a *perfect* solution – i.e., one with no broken pairs – if and only if D and D_Π have a common feasible set. For linear-delta-matroids the problems are solvable in $O(n^{\omega+1})$ time due to Geelen et al. [17].

Additionally, the following problems have been considered. Again, due to the flexibility of delta-matroids, these problems are related to one another; see [17, 29]. The previously fastest algorithm for all of them is by Geelen et al. [17] at $O(n^{\omega+1})$ time. Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be given delta-matroids.

- DM COVERING: Given D_1 and D_2 , find $F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2$ with $F_1 \cap F_2 = \emptyset$ to maximize $|F_1 \cup F_2|$
- DM DELTA-COVERING: Given D_1 and D_2 , find $F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2$ to maximize $|F_1 \Delta F_2|$
- DM PARTITION: Given D_1 and D_2 , find a partition $V = P \cup Q$ such that $P \in \mathcal{F}_1$ and $Q \in \mathcal{F}_2$

The variant of DM COVERING where the disjointness constraint is dropped reduces to MATROID UNION, since the maximal feasible sets of a delta-matroid form a matroid, hence is of less interest for delta-matroids.

Using the methods of the previous subsection, the decision versions of the above for linear and projected linear delta-matroids all reduce to computing the rank of an $O(n) \times O(n)$ skew-symmetric matrix. Indeed, consider DM DELTA-COVERING. Assume that D_1 and D_2 are given in some linear representation and let $D = \mathbf{D}(A)/T = D_1 \Delta D_2$. Then a set $F \subseteq V$ is a solution $F = F_1 \Delta F_2$ to the delta-covering problem if and only if $F \cup T$ is a basis of A . Similarly, DM COVERING reduces to finding a maximum feasible set in $D_1 \cup D_2$. DM INTERSECTION and DM PARTITION are asking whether \emptyset is feasible in $D_1 \Delta D_2$ and $D_1 \cup D_2$, respectively. For DM PARITY, as above let the input be (D, Π) and construct the delta-matroid D_Π . Then DM PARITY reduces to finding the rank of $D \Delta D_\Pi$ (or indeed $D \cup D_\Pi$). Thus the decision versions of all the above problems can be solved by a randomized algorithm using $O(n^\omega)$ field operations given linear representations of D resp. D_1 and D_2 thanks to Theorem 2.

For general delta-matroids, these problems are as hard as MATROID PARITY: Clearly, DM PARITY is as hard as MATROID PARITY. Given an instance (D, Π) of DM PARITY, let $D_1 = D$ and $D_2 = D_\Pi$. Then, the following is equivalent:

- The instance (D, Π) has a perfect solution for DM PARITY
- (D_1, D_2) has a solution of cardinality $V(D)$ for DM COVERING
- (D_1, D_2) has a solution of cardinality $V(D)$ for DM DELTA-COVERING
- (D_1, D_2) is a yes-instance of DM PARTITION

This shows that these problems are all intractable in the oracle model. For linear delta-matroids, achieving $O(n^\omega)$ time is arguably the best possible.

For the search versions, the situation changes slightly. Although these problems can be reduced to straightforward questions about non-singularity or rank of a skew-symmetric matrix A the resulting solution (e.g., a basis of A) provides only limited insight into the solution of the original problem. For example, if (D, Π) is an instance of DM PARITY, then the rank of $D\Delta D_\Pi$ tells us the number of broken pairs in an optimal solution (and a maximum feasible set tells us the set of broken pairs in such a solution), it does not give us the feasible set $F \in \mathcal{F}(D)$ that is the “actual” solution. Similarly, for DM COVERING and DM DELTA-COVERING the above reduction gives us the set $F = F_1 \cup F_2$ respectively $F = F_1 \Delta F_2$ but not the pair (F_1, F_2) . We refer to this (the set F for DM PARITY and the pair (F_1, F_2) for the other four problems) as the *witness*.

The algorithm of Geelen et al. [17] actually computes the witness for LINEAR DM PARITY, and can be implemented in $O(n^{\omega+1})$ field operations. Applying self-reducibility over the above-mentioned representation to find a witness would reproduce the same running time. Using methods of Harvey [18] we show the following improved result. These methods also underpin the currently fastest algorithm for linear matroid parity [6]. The condition of field size is for simplicity; given representations over a common field \mathbb{F} we can easily move to a large enough extension field of \mathbb{F} .

► **Theorem 3.** *Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be (projected) linear delta-matroids over a common field with $\Omega(n^3)$ elements. For a feasible set $F \subseteq V$ in $D_1 \Delta D_2$, we can, with high probability, find feasible sets $F_1 \in \mathcal{F}_1$ and $F_2 \in \mathcal{F}_2$ with $F_1 \Delta F_2 = F$ in $O(n^\omega)$ field operations.*

Via the reductions given above, this lets us find a witness for all problems considered. For LINEAR DM COVERING, we would first find the optimal set $F = F_1 \cup F_2$ as above, then solve LINEAR DM PARTITION for the instance induced by F ; for the remaining problems the solution is straightforward. See the full version of the paper for details.

► **Corollary 4.** *The following search problems can be solved in $O(n^\omega)$ field operations with high probability, given (projected) linear delta-matroids on ground sets of n elements represented over a common field with $\Omega(n^3)$ elements: DM COVERING; DM DELTA-COVERING; DM INTERSECTION; DM PARTITION; and DM PARITY.*

Finally, we consider the *weighted* versions of the above problems. Again, the picture becomes slightly different. For DM PARTITION, there is no sensible weighted version. For DM COVERING and DM DELTA-COVERING, the natural weight is the weight of the solution set F , in which case the problem is solved in strongly polynomial time of $O(n^\omega)$ field operations using Theorem 2 and 3. For DM PARITY, we would attach weights to the pairs, and consider two weighted versions. In the first version, we want to minimize the weight of the broken pairs; this problem is solved in $O(n^\omega)$ field operations using Theorem 2 and 3. For the more interesting version, we assume that there exists a perfect delta-matroid parity solution, and we wish to maximize the weight of such a set. This version, finally, is equivalent to the weighted version of DM INTERSECTION, which we focus on, and directly generalizes WEIGHTED MATROID PARITY. We use the matrix representation of the problem to construct a solution via algebraic methods.

As a special case, even the unit weight version, MAXIMUM LINEAR DM INTERSECTION, is an interesting problem which up to now has had no polynomial-time solution. Kakimura and Takamatsu [21] asked this as an open problem, and provided algorithms for some special cases of it. We solve the general case with a randomized algorithm.

► **Theorem 5.** *Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be (projected) linear delta-matroids over a common field with $\Omega(n^2)$ elements and let $w: V \rightarrow \{1, \dots, W\}$ be element weights. In $O(Wn^\omega)$ field operations we can find with high probability the maximum weight of a common feasible set $F \in \mathcal{F}_1 \cap \mathcal{F}_2$. In particular, we can find the maximum cardinality of $|F|$ in $O(n^\omega)$ field operations with high probability.*

By self-reducibility, the search version can thus be solved by a factor $O(n)$ overhead.

► **Corollary 6.** *MAXIMUM LINEAR DM INTERSECTION can be solved with high probability in $O(n^{\omega+1})$ field operations. WEIGHTED LINEAR DM INTERSECTION and WEIGHTED PERFECT DM PARITY with maximum element weight W can be solved with high probability in $O(Wn^{\omega+1})$ field operations.*

Removing the overhead on this result appears significantly harder. Indeed, a similar overhead exists for algorithms for weighted linear matroid parity [6], and even removing the overhead for WEIGHTED PERFECT MATCHING was significantly non-trivial [9]. We leave this as a (challenging) open question.

Applications. We now review some applications of the results. Not all of these results are new, but they serve to demonstrate the applicability of the setting.

One area of application is graph matching and factor problems. In the general factor problem, we are given a graph $G = (V, E)$, and a set of integers $f(v) \subseteq \mathbb{N}$ for each vertex $v \in V$. The task is to find a spanning subgraph $H = (V, F)$ such that $\deg_H(v) \in f(v)$ for every vertex $v \in V$. Cornuéjols [7] showed that this problem is polynomial-time solvable if each $f(v)$ has gaps of length at most 1, but becomes NP-hard otherwise.

For each $v \in V$, let $\delta(v)$ be the set of edges incident with v and define $\mathcal{F}_v = \{S \subseteq \delta(v) \mid |S| \in f(v)\}$. Under the gap-1 condition, $D_v = (\delta(v), \mathcal{F}_v)$ forms a delta-matroid for every $v \in V$. We refer to D_v as a *symmetric* delta-matroid.

The general factor problem can be reformulated as a DM INTERSECTION problem. We define two delta-matroids on the ground set $E_2 = \{(e, v) \mid e \in E, v \in e\}$, where each edge $uv \in E$ is represented by (uv, u) and (uv, v) . Let D_E be the matching delta-matroid of G' , where G' is the graph on E_2 with edges between the pairs (uv, u) and (uv, v) for each edge $uv \in E$. Furthermore, let D_f be the direct sum of symmetric delta-matroids $D_v = (\partial_v, \mathcal{F}_v)$ with $\partial_v = \{(uv, v) \mid uv \in E\}$ and $\mathcal{F}_v = \{S \subseteq \partial_v \mid |S| \in f(v)\}$. Then the intersection of D_E and D_f captures the general factor problem.²

When the symmetric delta-matroids correspond to cases where $f(v) = \{a, a + 2, \dots, b\}$ and $f(v) = \{a, a + 1, \dots, b\}$, they are linear and projected linear, respectively. The associated factor problems are known as *parity (a, b) -factors* and *(a, b) -factors*. Thus, these problems can be reduced to LINEAR DM INTERSECTION, and solved via Corollary 4 (although a naive formulation gives only time $O(m^\omega)$ here). The algebraic formulation of Gabow and Sankowski [16] for the f -factor problem, where $|f(v)| = 1$ for all $v \in V$, can essentially be derived from this method in a generic way using the Ishikawa-Wakayama formula (see Lemma 10), as the corresponding systems form matroids. See also recent work on *weighted* general factors [10, 23].

² We find it interesting that in this formulation, the GENERAL FACTOR problem is tractable if and only if every set system $D_v, v \in V$ forms a delta-matroid. For more such occurrences, see the BOOLEAN EDGE CSP [13, 22] and BOOLEAN PLANAR CSP [11, 22] problems, both of which appear (roughly speaking) to be interestingly tractable if and only if the constraints form delta-matroids in natural ways.

For another example, let $G = (V, E)$ be a graph and $T \subseteq V$ a set of terminals. Let \mathcal{S} be a partition of T . An \mathcal{S} -path packing is a vertex-disjoint packing of paths where every path has endpoints in distinct parts of \mathcal{S} and internal vertices disjoint from T . A classical theorem of Mader shows a min-max theorem characterizing the maximum number of paths in an \mathcal{S} -path packing, generalizing Menger’s theorem and the Tutte-Berge formula; see Schrijver [32]. Wahlström [36] recently showed the following. Let a set $F \subseteq T$ be *feasible* if there is an \mathcal{S} -path packing whose set of endpoints is precisely F . Then $D = (T, \mathcal{F})$ is a linear delta-matroid. Then, via LINEAR DM INTERSECTION as above, we can solve the following “ \mathcal{S} -factor” problem: Given G, \mathcal{S} and a prescribed set of degrees $f(T_i) \in \mathbb{N}$, $T_i \in \mathcal{S}$, is there an \mathcal{S} -path packing in (G, \mathcal{S}) where precisely $f(T_i)$ paths have an endpoint in T_i ? The generalizations to (a, b) -factors and (a, b) -parity factors can be handled similarly.

Structure of the paper. Section 2 contains all definitions. Section 3 gives results about new representations, and Section 4 gives the algorithmic results. Section 5 concludes the paper. For missing proofs (for results marked with \star) and additional results, including a more extensive related work review, see the full version of the paper.

2 Preliminaries

For two sets A, B , we let $A \Delta B = (A \setminus B) \cup (B \setminus A)$ denote their symmetric difference.

For a matrix A and a set of rows S and columns T , we denote by $A[S, T]$ the submatrix containing rows S and columns T . If S contains all rows (T contains all columns), then we use the shorthand $A[\cdot, T]$ ($A[S, \cdot]$, respectively). The $n \times m$ zero matrix and the $n \times n$ identity matrix is denoted by $O_{n \times m}$ and I_n , respectively. We often drop the subscript when clear from context.

Delta-matroids. A *delta-matroid* is a pair $D = (V, \mathcal{F})$ where V is a ground set and $\mathcal{F} \subseteq 2^V$ a collection of *feasible sets*, subject to the rule

$$\forall A, B \in \mathcal{F}, x \in A \Delta B \exists y \in A \Delta B : A \Delta \{x, y\} \in \mathcal{F}.$$

This is known as the *symmetric exchange axiom*. For $D = (V, \mathcal{F})$ we let $V(D) = V$ and $\mathcal{F}(D) = \mathcal{F}$. A delta-matroid is *even* if all feasible sets have the same parity. Note that in this case we must have $x \neq y$ in the symmetric exchange axiom, although this does not necessarily hold in general.

A *separation oracle* for a delta-matroid $D = (V, \mathcal{F})$ is an oracle that, given a pair (S, T) of disjoint subsets of V , reports whether there is a set $F \in \mathcal{F}$ such that $S \subseteq F$ and $F \cap T = \emptyset$. If so, the pair (S, T) is *separable*. A delta-matroid is *tractable* if it has a polynomial-time separation oracle.

For a delta-matroid $D = (V, \mathcal{F})$ and $S \subseteq D$, the *twisting* of D by S is the delta-matroid $D \Delta S = (V, \mathcal{F} \Delta S)$ where $\mathcal{F} \Delta S = \{F \Delta S \mid F \in \mathcal{F}\}$. This generalizes some common operations from matroid theory. The *dual* delta-matroid of D is $D \Delta V(D)$. For a set $S \subseteq V(D)$, the *deletion* of S from D refers to the set system $D \setminus S = (V \setminus S, \{F \in \mathcal{F} \mid F \subseteq V \setminus S\})$. The *contraction* of S refers to $D/S = (D \Delta S) \setminus S = (V \setminus S, \{F \setminus S \mid F \in \mathcal{F}, S \subseteq F\})$.

Skew-symmetric matrices. A square matrix A is *skew-symmetric* if $A = -A^T$. In the case that A is over a field of characteristic 2, we will additionally assume that it has zero diagonal, unless stated otherwise. For a skew-symmetric matrix A with rows and columns indexed by a set $V = [n]$, the *support graph* of A is the graph $G = (V, E)$ where $E = \{uv \mid A[u, v] \neq 0\}$. A fundamental tool for working with skew-symmetric matrices is the *Pfaffian*, defined for a skew-symmetric matrix A as $\text{Pf } A = \sum_M \sigma(M) \prod_{e \in M} A[u, v]$, where M ranges over all perfect

62:10 Faster Algorithms on Linear Delta-Matroids

matchings of the support graph of A and $\sigma(M) \in \{1, -1\}$ is the sign of the permutation: $\begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ v_1 & v'_1 & \cdots & v_{n/2} & v'_{n/2} \end{pmatrix}$, where $M = \{v_i v'_i \mid i \in [n/2]\}$ with $v_i < v'_i$ for all $i \in [n/2]$. It is known that $\det A = (\text{Pf } A)^2$, hence in particular $\text{Pf } A \neq 0$ if and only if A is non-singular. However, for many algorithms it will be more convenient to work directly with the Pfaffian. In fact, Pfaffian generalizes the notion of determinants as follows.

► **Lemma 7.** *For an $n \times n$ -matrix M , it holds that $\det M = (-1)^{n(n-1)/2} \text{Pf} \begin{pmatrix} O & M \\ -M^T & O \end{pmatrix}$.*

An important operation on skew-symmetric matrices is *pivoting*. Let $A \in \mathbb{F}^{n \times n}$ be skew-symmetric and let $S \subseteq [n]$ be such that $A[S]$ is non-singular. Order the rows and columns of A so that $A = \begin{pmatrix} B & C \\ -C^T & D \end{pmatrix}$, where $A[S] = B$. Then the *pivoting* of A by S is $A * S = \begin{pmatrix} B^{-1} & B^{-1}C \\ C^T B^{-1} & D + C^T B^{-1}C \end{pmatrix}$. Note that this is a well-defined, skew-symmetric matrix.

► **Lemma 8** ([34]). *It then holds, for any $X \subseteq [n]$, that $\det(A * S)[X] = \frac{\det A[X \Delta S]}{\det A[S]}$, In particular, $(A * S)[X]$ is non-singular if and only if $A[X \Delta S]$ is non-singular.*

Finally, let us note a formula on the Pfaffian of a sum of two skew-symmetric matrices:

► **Lemma 9** ([29, Lemma 7.3.20]). *For two skew-symmetric matrices A_1 and A_2 both indexed by $V = [n]$, we have*

$$\text{Pf}(A_1 + A_2) = \sum_{U \subseteq V} \sigma_U \text{Pf } A_1[U] \cdot \text{Pf } A_2[V \setminus U],$$

where $\text{Pf } A_i[\emptyset] = 1$ for $i = 1, 2$ and $\sigma_U \in \{1, -1\}$ is a sign of the permutation

$$\begin{pmatrix} 1 & 2 & \cdots & |U| & |U| + 1 & \cdots & |V| - 1 & |V| \\ u_1 & u_2 & \cdots & u_{|U|} & v_1 & \cdots & v_{|V \setminus U| - 1} & u_{|V \setminus U|} \end{pmatrix},$$

where u_i and v_i are the i -th largest elements of U and $V \setminus U$, respectively.

The following is a generalization of the Cauchy-Binet formula to skew-symmetric matrices. The algebraic approach of Lovász [26] for matroid parity can be derived from this formula (see [27]).

► **Lemma 10** (Ishikawa-Wakayama formula [19]). *For a skew-symmetric $2n \times 2n$ -matrix A and a $2k \times 2n$ -matrix B with $k \leq n$, we have*

$$\text{Pf } BAB^T = \sum_{U \in \binom{[2n]}{2k}} \det B[\cdot, U] \text{Pf } A[U].$$

Linear representation. A skew-symmetric matrix defines a delta-matroid as follows. For a skew-symmetric matrix $A \in \mathbb{F}^{V \times V}$ over a field \mathbb{F} , define $\mathcal{F} = \{X \subseteq V \mid A[X] \text{ is non-singular}\}$. Then, (V, \mathcal{F}) , which is denoted by $\mathbf{D}(A)$, is a delta-matroid. We say that a delta-matroid $D = (V, \mathcal{F})$ is *representable* over \mathbb{F} if there is a skew-symmetric matrix $A \in \mathbb{F}^{V \times V}$ and a twisting set $S \subseteq V$ such that $D = \mathbf{D}(A) \Delta S$. If $A[X]$ is non-singular, or equivalently $\emptyset \in \mathcal{F}(D)$, we say that D is *directly representable* over \mathbb{F} . Note that a directly representable delta-matroid D can be represented without a twisting set X , as $\mathbf{D}(A) \Delta X = \mathbf{D}(A * X)$. We

will say that D is *directly represented* by A if $D = \mathbf{D}(A)$. A delta-matroid is called *normal* if \emptyset is feasible. Note that every linear delta-matroid is even, and that a linear delta-matroid is directly representable if and only if it is normal. Linear delta-matroids are tractable [3].

In addition, we consider *projected* linear delta-matroids. Let $D = (V, \mathcal{F})$ be a delta-matroid and $X \subseteq V$. Then the projection $D|X$ is defined as $D|X = (V \setminus X, \mathcal{F}|X)$ where $\mathcal{F}|X = \{F \setminus X \mid F \in \mathcal{F}\}$. Then $D|X$ is a delta-matroid, although it is in general not even, hence not linear. When D is linear, then we refer to $D' = D|X$ as a *projected linear delta-matroid*. When $|X| = 1$ we refer to this as an *elementary projection*, following Geelen et al. [17].

As noted above, we assume that A has a zero diagonal (this naturally follows from the definition over all fields with a characteristic not 2). However, if \mathbb{F} is a field of characteristic 2, then linear delta-matroids over \mathbb{F} with a non-zero diagonal correspond to projected linear delta-matroids over \mathbb{F} ; see Geelen et al. [17].

For a matroid $M = (V, \mathcal{I})$ with the basis family \mathcal{B} , $D = (V, \mathcal{B})$ is a delta-matroid. If M is represented by A , D can be represented as follows. Fix a basis $B \in \mathcal{B}$. We may assume w.l.o.g. that $A[\cdot, B] = I$. Define

$$A' = \begin{matrix} & & B & & V \setminus B \\ & & O & & A[\cdot, V \setminus B] \\ & & -A^T[V \setminus B, \cdot] & & O \end{matrix} \begin{pmatrix} \\ \\ \end{pmatrix}.$$

Observe that for every $F \subseteq V$, $A'[F]$ is non-singular if and only if $A[F \cap B, F \setminus B]$ is non-singular. Since $A[\cdot, B] = I$, this is equivalent to $A[\cdot, (B \setminus F) \cup (F \setminus B)]$ being non-singular, and thus $D = \mathbf{D}(A')\Delta B$.

Conversely, let $D = (V, \mathcal{F})$ be a delta-matroid. Then the set of maximum-cardinality feasible sets in D forms the set of bases of a matroid $M = (V, \mathcal{I})$. Furthermore, if $D = \mathbf{D}(A)$ is directly represented, then A (as a column space) is also a representation of the matroid M . This is because if B is a column basis for a skew-symmetric matrix A , then $A[B]$ is non-singular (see e.g., [31] or [29, Proposition 7.3.6]).

To avoid intricate representation issues, we assume that every linear representation is given over some finite field. We note that a representation over the rationals can be efficiently transformed into an equivalent representation over a finite field.

Approximate linear representation. For a delta-matroid $D = (V, \mathcal{F})$, we say that a delta-matroid $D' = (V, \mathcal{F}')$ is an ε -*approximate representation* of D if $\mathcal{F}' \subseteq \mathcal{F}$ and for every $F \in \mathcal{F}$, the probability that $F \in \mathcal{F}'$ is at least $1 - \varepsilon$. For constructing an ε -approximate linear representation, the Schwartz-Zippel lemma [33, 37] (also referred to as the DeMillo-Lipton-Schwartz-Zippel lemma) comes in handy. It states that a polynomial $P(X)$ of total degree at most d over a field \mathbb{F} becomes nonzero with probability at least $1 - d/|\mathbb{F}|$ when evaluated at uniformly chosen elements from \mathbb{F} , unless $P(X)$ is identically zero.

Let $G = (V, E)$ be an undirected graph and let $\mathcal{F} \subseteq 2^V$ contain all sets $F \subseteq V$ such that $G[F]$ has a perfect matching. Then $D(G) = (V, \mathcal{F})$ is a delta-matroid referred to as the *matching delta-matroid* of G . The Tutte matrix gives rise to an approximate linear representation. Note that setting $\varepsilon = O(2^{-|V|})$ (or lower) gives a matrix of polynomial size which with high probability is a correct representation of $D(G)$. However, this will inflate the time needed for field operations over \mathbb{F} by at least $\Omega(n)$, so for efficiency reasons we work with ε -approximate representations where ε is a parameter.

► **Lemma 11 (★).** *Let $G = (V, E)$ be a graph on n vertices and \mathbb{F} be a field with at least $n \cdot \lceil 1/\varepsilon \rceil$ elements. We can construct a ε -approximate linear representation of the matching delta-matroid of G over \mathbb{F} .*

Operations in matrix product time. Determinant, rank, basis, inverse can be found in $O(n^\omega)$ time. Given an $n \times 2n$ -matrix, its row echelon form can be computed in $O(n^\omega)$. We can find a lexicographically smallest column basis in $O(n^\omega)$ time. See [35].

3 Contraction representation of linear delta-matroids

In this section, we introduce a novel linear representation for delta-matroids, called *contraction representation*. For the sake of clarity, we will say that the representation of a delta-matroid as $D = \mathbf{D}(A)\Delta S$ is a *twist representation*. As we will see in Section 4, the contraction representation is useful in the design of more efficient algorithms for linear delta-matroids. We also give further results, supported by the new representation. First, we show that the union and delta-sum of linear delta-matroids is linear. Previously, this was only known to define delta-matroids [2, 4]. We also use this to provide a compact representation of projected linear delta-matroids. All of these additional results will be useful in our algorithms.

3.1 Contraction representations

For a delta-matroid $D = (V, \mathcal{F})$, a *contraction representation* of D is a pair (A, T) where A is a skew-symmetric matrix over a field \mathbb{F} whose rows and columns are labelled by $V \cup T$, such that $D = \mathbf{D}(A)/T$, i.e., for every $F \subseteq V$, F is feasible in D if and only if $F \cup T$ is feasible in $\mathbf{D}(A)$. This is closely related to *strong maps* of delta-matroids. For two delta-matroids D and D° , D° is a *strong map* of D if there exists a delta-matroid $D^+ = (V \cup Z, \mathcal{F})$ such that $D = D^+ \setminus Z$ and $D^\circ = D^+ / Z$ (see Geelen et al. [17]). Hence, if $D = (V, \mathcal{F}) = \mathbf{D}(A)/T$ is a contraction representation of a delta-matroid D , then D is a strong map of the directly representable delta-matroid $\mathbf{D}(A[V])$. We show that the contraction and twist representations are equivalent.

► **Lemma 12.** *Given a delta-matroid D in twist representation, we can construct a contraction representation $D = \mathbf{D}(A)/T$ of D deterministically in $O(n^2)$ time.*

Proof. Let $D = (V, \mathcal{F})$ be given as $D = \mathbf{D}(A)\Delta S$, $S \subseteq V$. For a set T of size $|S|$, define a skew-symmetric matrix A' over $V \cup T$ by

$$A' = \begin{matrix} & T & S & V \setminus S \\ \begin{matrix} T \\ S \\ V \setminus S \end{matrix} & \begin{pmatrix} O & I & O \\ -I & A[S] & A[S, V \setminus S] \\ O & A[V \setminus S, S] & A[V \setminus S] \end{pmatrix} \end{matrix}$$

where I is an identity matrix. Note that the support graph of $A'[T \cup S]$ has a unique perfect matching (namely, every vertex in T has degree one), thus $\text{Pf } A'[T \cup S] = \pm 1$ and $A'[T \cup S]$ is non-singular. Thus, we can construct the matrix $A^* = A' * (T \cup S)$. Note that

$$(A'[T \cup S])^{-1} = \begin{pmatrix} O & I \\ -I & A[S] \end{pmatrix}^{-1} = \begin{pmatrix} A[S] & -I \\ I & O \end{pmatrix},$$

and consequently, the result of pivoting is

$$A^* = \begin{array}{c} T \\ S \\ V \setminus S \end{array} \begin{pmatrix} & T & S & V \setminus S \\ A[S] & -I & -A[S, V \setminus S] \\ I & O & O \\ -A[V \setminus S, S] & O & A[V \setminus S] \end{pmatrix}.$$

Clearly, A^* can be constructed in $O(n^2)$ time. Now by Lemma 8, for any $F \subseteq V$, $A^*[F \cup T]$ is non-singular if and only if $A'[(F \cup T)\Delta(S \cup T)] = A'[F\Delta S]$ is. Since $F\Delta S \subseteq V$ and $A'[V] = A[V]$, this is equivalent to $F \in \mathcal{F}(D)$, thereby showing that $\mathbf{D}(A^*)/T$ is a contraction representation of D . ◀

► **Lemma 13.** *Given a contraction representation $D = \mathbf{D}(A)/T$, we can find a twist representation of D deterministically using $O(n^\omega)$ field operations.*

Proof. Let $S \subseteq V$ be a set such that $A[S \cup T]$ is non-singular; such a set exists since $\mathcal{F}(D) \neq \emptyset$ by assumption, and can be found efficiently over A . Then $A' = A * (S \cup T)$ is well-defined. Let $A^* = A' \setminus T$. Observe that $D = \mathbf{D}(A)\Delta S$, that is, for every $F \subseteq V$, $A^*[F\Delta S] = A'[F\Delta S]$ is non-singular if and only if $A[(F\Delta S)\Delta(S \cup T)] = A[F \cup T]$ is non-singular by Lemma 8. All operations above can be performed in matrix multiplication time. ◀

We also observe that the contracted set T in a representation of a delta-matroid $D = (V, \mathcal{F})$ never needs to be larger than $|V|$.

► **Lemma 14 (★).** *Given a contraction representation $D = \mathbf{D}(A)/T$ of a delta-matroid $D = (V, \mathcal{F})$, in $O(n^\omega)$ field operations, where $n = |V| + |T|$, we can find a contraction representation $D = \mathbf{D}(A')/T'$ where $|T'| \leq |V|$.*

3.2 Constructions

We next consider an immediate way to combine two delta-matroids into a new delta-matroid, the *delta-matroid union* (surveyed in the introduction). Let $D_1 = (V_1, \mathcal{F}_1)$ and $D_2 = (V_2, \mathcal{F}_2)$ be two delta-matroids on not necessarily disjoint ground sets and let $V = V_1 \cup V_2$. Define $\mathcal{F} = \mathcal{F}_1 \uplus \mathcal{F}_2 := \{F_1 \cup F_2 \mid F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2, F_1 \cap F_2 = \emptyset\}$ as the collection of sets that can be produced as disjoint unions from \mathcal{F}_1 and \mathcal{F}_2 , and write $D = (V, \mathcal{F}) = D_1 \cup D_2$. Then Bouchet [2] showed that $D = (V, \mathcal{F})$ is a delta-matroid. We show that furthermore, if D_1 and D_2 are linear or projected linear then so is D , and an ε -approximate representation can be constructed in polynomial time. Note that we may as well assume that $V = V_1 = V_2$, by adding the missing elements to the respective delta-matroid as loops.

► **Lemma 15 (★).** *Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be linear (respectively projected linear) delta-matroids defined over a common field \mathbb{F} and given in contraction representation. Then the delta-matroid union $D = D_1 \cup D_2$ is a linear (respectively projected linear) delta-matroid, and an ε -approximate representation of D can be constructed in $O(n^2)$ field operations over an extension field of \mathbb{F} with at least $n \cdot \lceil 1/\varepsilon \rceil$ elements.*

Let $D_1 = (V_1, \mathcal{F}_1)$ and $D_2 = (V_2, \mathcal{F}_2)$ be delta-matroids on not necessarily disjoint ground sets. Let $V = V_1 \cup V_2$ and $\mathcal{F} = \{F_1 \Delta F_2 \mid F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2\}$. Then $D = (V, \mathcal{F})$ is called the *delta-sum* $D = D_1 \Delta D_2$ of D_1 and D_2 , and is itself a delta-matroid. Bouchet and Schwärzler [4] give a proof, citing unpublished work by Duchamp for the result. We show that the delta-sum of linear delta-matroids is linear when D_1 and D_2 are given as representations over a common field. By Lemma 12, we can work with contraction representations.

► **Lemma 16 (★).** *The delta-sum of (projected) linear delta-matroids over a common field is a (projected) linear delta-matroid, and a representation can be computed in randomized polynomial time. More precisely, let D_1 and D_2 be linear delta-matroids given in contraction representation over a common finite field \mathbb{F} . Let $\varepsilon > 0$ be given and let \mathbb{F}' be a field extension of \mathbb{F} with at least $n \cdot \lceil 1/\varepsilon \rceil$ elements. We can construct an ε -approximate contraction representation of $D_1 \Delta D_2$ in $O(n^\omega)$ field operations over \mathbb{F}' .*

Proof sketch. Let $D_1 = (V, \mathcal{F}_1) = \mathbf{D}(A_1)/T_1$ and $D_2 = (V, \mathcal{F}_2) = \mathbf{D}(A_2)/T_2$, w.l.o.g. represented over the same ground set and with $T_1 \cap T_2 = \emptyset$. Create three copies $V \uplus V_1 \uplus V_2$ of the ground set and define three sets of variables $Y_i = \{y_{v,i} \mid v \in V\}$, $i = 1, 2, 3$. Let

$$A = \begin{matrix} & V & V_1 & T_1 & V_2 & T_2 \\ \begin{matrix} V \\ V_1 \\ T_1 \\ V_2 \\ T_2 \end{matrix} & \begin{pmatrix} O & B_1 & O & B_2 & O \\ -B_1 & A_1[V_1] & A_1[V_1, T_1] & B_3 & O \\ O & A_1[T_1, V_1] & A_1[T_1] & O & O \\ -B_2 & -B_3 & O & A_2[V_2] & A_2[V_2, T_2] \\ O & O & O & A_2[T_2, V_2] & A_2[T_2] \end{pmatrix} \end{matrix}$$

where B_i , $i = 1, 2, 3$ is a diagonal matrix whose v :th entry is $y_{v,i}$. Note that A can be seen as the sum of the representation of the disjoint union of D_1 and D_2 and the Tutte matrix of the graph consisting of a disjoint union of triangles on $\{v, v^1, v^2\}$ for every $v \in V$, where v^i is the copy of v in V_i , $i = 1, 2$. It now follows from the Pfaffian sum formula (Lemma 9) that $\mathbf{D}(A)/(V_1 \cup T_1 \cup V_2 \cup T_2)$ is an ε -approximate representation of $D_1 \Delta D_2$. ◀

Recall that a projected linear delta-matroid $D = (V, \mathcal{F})$ is a delta-matroid represented as $D = D'|X$ where $D' = (V \cup X, \mathcal{F}')$ is a linear delta-matroid. Projections of linear delta-matroids were studied by Geelen et al. [17] in the context of linear delta-matroids over fields of characteristic 2, and by Kakimura and Takamatsu [21] regarding generalizations of constrained matching problems. We observe that if D is linear, then the even (respectively odd) sets of $D|X$ form a linear delta-matroid, and that every projected linear delta-matroid $D|X$ can be represented via an elementary projection.

► **Lemma 17 (★).** *Let $D = (V \cup X, \mathcal{F})$ be a linear delta-matroid. Then the following delta-matroids are linear and approximate representations can be constructed efficiently.*

1. A linear delta-matroid $D' = (V \cup X', \mathcal{F}')$ such that $D|X = D'|X'$ and $|X'| \leq 1$
 2. The delta-matroid $D_e = (V, \mathcal{F}_e)$ where \mathcal{F}_e contains the sets of $\mathcal{F}|X$ of even cardinality
 3. The delta-matroid $D_o = (V, \mathcal{F}_o)$ where \mathcal{F}_o contains the sets of $\mathcal{F}|X$ of odd cardinality
- More precisely, let $D = \mathbf{D}(A)/T$ in contraction representation over a finite field \mathbb{F} and let $\varepsilon > 0$ be given. Let \mathbb{F}' be a field extension of \mathbb{F} with at least $n \cdot \lceil 1/\varepsilon \rceil$ elements. We can construct an ε -approximate contraction representation of each of the above delta-matroids in $O(n^2)$ operations over \mathbb{F}' .

4 Algorithms for fundamental delta-matroid problems

We now present the various algorithms over linear delta-matroids.

4.1 Max-weight feasible sets

We show an $O(n^\omega)$ -time algorithm for finding a max-weight feasible set in a linear delta-matroid. More precisely, let $D = (V, \mathcal{F})$ be a delta-matroid and $w(v) \in \mathbb{Q}$, $v \in V$ a set of element weights. Let $n = |V|$. The goal is to find a feasible set $F \in \mathcal{F}$ to maximize the

weight $w(F) = \sum_{v \in F} w(v)$. Note that since the feasible sets of D do not necessarily all have the same cardinality, the negative element weights cannot easily be removed by any simple transformation (as, e.g., shifting by a constant would affect different feasible sets differently). This problem can be solved via the “signed greedy” algorithm, which extends the normal greedy algorithm (in fact, like for matroids, the success of signed greedy can be taken as a definition of delta-matroids) [3, 1]. However, this requires $O(n)$ calls to a separation oracle. If D is linear, this algorithm thus runs in $O(n^{\omega+1})$ time. We show an improvement using the contraction representation. We begin with the following observation.

► **Lemma 18.** *Let $D = (V, \mathcal{F})$ be a delta-matroid and $w: V \rightarrow \mathbb{Q}$ a set of weights. Let $N \subseteq V$ be the set of elements $v \in V$ such that $w(v) < 0$. Let $D' = D\Delta N$, and define a set of weights w' by $w'(v) = |w(v)|$ for every $v \in V$. For any feasible set $F \in \mathcal{F}$, we have $w(F) = w'(F\Delta N) + w(N)$.*

Proof. The following hold: $w(F) = w'(F \setminus N) - w'(F \cap N)$ and $w'(F\Delta N) = w'(F \setminus N) + w'(N \setminus F)$. Thereby $w(F) - w'(F\Delta N) = -w'(N) = w(N)$ as promised. ◀

The problem of finding a max-weight feasible set in a linear delta-matroid in contraction representation now reduces to the problem of finding a max-weight basis of a linear matroid.

► **Theorem 2.** *Let $D = (V, \mathcal{F})$ be a linear or projected linear delta-matroid. In $O(n^\omega)$ field operations, we can find a max-weight feasible set in D .*

Proof. By Lemma 17, it suffices to prove the statement for linear delta-matroids.

Let $D = (V, \mathcal{F})$ and $w: V \rightarrow \mathbb{Q}$ be given as input, and as above define $N = \{v \in V \mid w(v) < 0\}$, $D' = D\Delta N$ and $w': V \rightarrow \mathbb{Q}$ where $w'(v) = |w(v)|$ for all $v \in V$. Let $\mathbf{D}(A)/T$ be a contraction representation of D' , which can be constructed using Lemma 12. Finally, order the columns of A to begin with T and thereafter elements $v \in V$ in order of non-increasing weight $w'(v)$. Let B be a lex-min column basis for A with respect to this ordering. Then B can be computed in $O(n^\omega)$ field operations over A (see e.g., [5]), and B is a max-weight column basis of A with respect to the weights w' . We claim that $F = (B \setminus T)\Delta N$ is a max-weight feasible set in D . For this, let F^* be a max-weight feasible set in D with respect to the weights w . Then by Lemma 18, $F^*\Delta N$ is a max-weight feasible set in D' with respect to the weights w' . Hence $B' = (F^*\Delta N) \cup T$ is feasible in $\mathbf{D}(A)$ and $w'(B \setminus T) \geq w'(B' \setminus T) = w(F^*) - w(N)$. On the other hand, let B be a lex-min basis of A in the above ordering. Then $T \subseteq B$ by construction. By Lemma 18, $F = (B \setminus T)\Delta N$ is a feasible set in D with $w(F) = w'(B \setminus T) + w(N) \leq w(F^*)$. Hence $w'(B \setminus T) = w(F^*) - w(N)$ by sandwiching and $w((B \setminus T)\Delta N) = w'(B \setminus T) + w(N) = w(F^*)$. ◀

4.2 Intersection, Parity, and Delta-Covering

Recall that the DM PARITY problem is defined as follows. Let $D = (V, \mathcal{F})$ be a delta-matroid with V partitioned into n pairs Π . The problem is to find a feasible set $F \in \mathcal{F}$ minimizing the number of *broken* pairs $\delta_\Pi(F) = |\{P \in \Pi : |F \cap P| = 1\}|$. Let $\delta(D, \Pi) = \min_{F \in \mathcal{F}} \delta_\Pi(F)$. We consider the equivalent DM DELTA-COVERING problem defined as follows. Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be two given delta-matroids. The problem is to find $F_1 \in \mathcal{F}_1$ and $F_2 \in \mathcal{F}_2$ maximizing $|F_1 \Delta F_2|$. Let $\tau(D_1, D_2) = \max_{F_i \in \mathcal{F}_i} |F_1 \Delta F_2|$. As described in Section 1, DM PARITY and DM DELTA-COVERING reduce to each other. Moreover, DM COVERING and DM INTERSECTION are special cases of DM PARITY and DM DELTA-COVERING.

We can compute $\tau(D_1, D_2)$ in $O(n^\omega)$ field operations as follows. Observe that $\tau(D_1, D_2)$ is the maximum feasible set size in the delta-sum $D_1 \Delta D_2$. By Lemma 16, we can find a linear representation $\mathbf{D}(A)/T$ of $D_1 \Delta D_2$ in $O(n^\omega)$ field operations. We then have $\tau(D_1, D_2) =$

rank $A - |T|$. Thus the decision variants of DM PARITY and DM DELTA-COVERING can be solved in $O(n^\omega)$ field operations. Via self-reducibility, we obtain an algorithm that finds a witness for DM PARITY and DM DELTA-COVERING in $O(n^{\omega+1})$ field operations, matching the result of Geelen et al. [17]. We present an improvement to $O(n^\omega)$, using the method of Harvey [18]. Specifically, we prove the following:

► **Lemma 19 (★).** *Let A be an $n \times n$ skew-symmetric, non-singular polynomial matrix with its row and column indexed by $V = \{v_1, \dots, v_n\}$. Suppose that $A = B + Y$, where (i) B is a matrix defined over a field \mathbb{F} containing $\Omega(n^3)$ elements, and (ii) Y is the Tutte matrix of a graph $G = (V, E)$ with variables $y_e, e \in E$. Suppose that G has connected components C_1, \dots, C_γ , with $|C_i| \in O(1)$ for every $i \in [\gamma]$. It is possible to find an inclusion-wise maximal set $S \subseteq E$ for which A remains non-singular when setting y_e to zero for all $e \in S$. This can be done with probability $1 - 1/\Omega(n)$ using $O(n^\omega)$ field operations over \mathbb{F} .*

Using Lemma 19, we prove Theorem 3, which yields algorithms for DM DELTA-COVERING and DM PARITY as well as DM COVERING and DM INTERSECTION using $O(n^\omega)$ field operations (Corollary 4).

► **Theorem 3.** *Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be (projected) linear delta-matroids over a common field with $\Omega(n^3)$ elements. For a feasible set $F \subseteq V$ in $D_1 \Delta D_2$, we can, with high probability, find feasible sets $F_1 \in \mathcal{F}_1$ and $F_2 \in \mathcal{F}_2$ with $F_1 \Delta F_2 = F$ in $O(n^\omega)$ field operations.*

► **Corollary 4.** *The following search problems can be solved in $O(n^\omega)$ field operations with high probability, given (projected) linear delta-matroids on ground sets of n elements represented over a common field with $\Omega(n^3)$ elements: DM COVERING; DM DELTA-COVERING; DM INTERSECTION; DM PARTITION; and DM PARITY.*

Proof sketch. We treat the problems in turn. Throughout, by Lemma 17 (and by exhaustively guessing the parities of the feasible sets involved in the solution), we assume that every delta-matroid D in the input is linear, and by Lemma 12 that it is provided in contraction form $D = \mathbf{D}(A)/T$ for some skew-symmetric matrix A . We omit details of field size and approximate representations.

DM Parity. Let the input be (D, Π) with $D = \mathbf{D}(A)/T$. Construct the matching delta-matroid D_Π of the graph with edge set Π and construct a contraction representation of $D' = D \Delta D_\Pi$ using Lemma 16. Let F be a maximum feasible set in D' and apply Theorem 3.

DM Intersection. Given input (D_1, D_2) , construct $D = D_1 \Delta D_2$. If \emptyset is feasible in D , then apply Theorem 3 to $F = \emptyset$; otherwise report a no-instance.

DM Partition. Given input (D_1, D_2) , construct $D = D_1 \Delta D_2$. If the full ground set V is feasible in D , apply Theorem 3; otherwise report a no-instance.

DM Covering. Given input (D_1, D_2) , construct $D = D_1 \cup D_2$ and let F be a maximal feasible set. Delete all elements of $V \setminus F$ to create the induced delta-matroids D'_1 and D'_2 with ground set F , and apply Theorem 3 to $D' = D'_1 \Delta D'_2$.

DM Delta-Covering. Given input (D_1, D_2) , construct $D = D_1 \Delta D_2$ and let F be a maximal feasible set. Apply Theorem 3 to F . ◀

Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be two linear delta-matroids with weights $w(v) \in \mathbb{N}, v \in V$. We consider the *intersection* problem, where the goal is to find a common feasible set $F \subseteq V$, i.e., $F \in \mathcal{F}_1$ and $F \in \mathcal{F}_2$. The decision problem, i.e., whether there exists $F \in \mathcal{F}_1 \cap \mathcal{F}_2$, can be solved in $O(n^\omega)$ by testing whether V is feasible in the delta-sum $D_1 \Delta D_2^*$. Moreover, we can find a common feasible set in $O(n^\omega)$ time using Theorem 3. We next give a (pseudo)polynomial-time randomized algorithm for the WEIGHTED DELTA-MATROID

INTERSECTION, where we are tasked with finding a common feasible set of maximum weight, answering an open question of Kakimura and Takamatsu [21]. Previously, there has been no polynomial-time algorithm even for the unweighted case.

► **Theorem 5.** *Let $D_1 = (V, \mathcal{F}_1)$ and $D_2 = (V, \mathcal{F}_2)$ be (projected) linear delta-matroids over a common field with $\Omega(n^2)$ elements and let $w: V \rightarrow \{1, \dots, W\}$ be element weights. In $O(Wn^\omega)$ field operations we can find with high probability the maximum weight of a common feasible set $F \in \mathcal{F}_1 \cap \mathcal{F}_2$. In particular, we can find the maximum cardinality of $|F|$ in $O(n^\omega)$ field operations with high probability.*

5 Conclusions

We have introduced a novel representation for linear delta-matroids, the *contraction* representation, which enable us to derive a range of new results, including linear representations of delta-matroid union and delta-sum, and faster algorithms for various problems including LINEAR DELTA-MATROID PARITY. We also show the first (pseudo)polynomial-time, randomized algorithms for the maximum cardinality and weighted versions of LINEAR DELTA-MATROID INTERSECTION, solving an open question of Kakimura and Takamatsu [21].

We note a few open questions. First, all our running times are stated purely in terms of the number of elements n . It would be interesting to explore faster algorithms for linear delta-matroids of bounded “rank”. But we also note two specific challenging questions.

1. Is there a strongly polynomial-time algorithm for WEIGHTED LINEAR DELTA-MATROID PARITY, extending the recent result for WEIGHTED LINEAR MATROID PARITY [20]?
2. Is there a $\tilde{O}(Wn^\omega)$ -time algorithm for SHORTEST DISJOINT \mathcal{S} -PATHS? This would extend results for graph matching [9].

Additionally, does there exist a good characterization of, and/or a deterministic algorithm for the maximum cardinality version of LINEAR DELTA-MATROID INTERSECTION?

References

- 1 André Bouchet. Greedy algorithm and symmetric matroids. *Math. Program.*, 38(2):147–159, 1987. doi:10.1007/BF02604639.
- 2 André Bouchet. Matchings and Δ -matroids. *Discret. Appl. Math.*, 24(1-3):55–62, 1989. doi:10.1016/0166-218X(92)90272-C.
- 3 André Bouchet. Coverings and delta-coverings. In *IPCO*, pages 228–243, 1995. doi:10.1007/3-540-59408-6_54.
- 4 André Bouchet and Werner Schwärzler. The delta-sum of matching delta-matroids. *Discret. Math.*, 181(1-3):53–63, 1998. doi:10.1016/S0012-365X(97)00044-7.
- 5 Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.
- 6 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Algebraic algorithms for linear matroid parity problems. *ACM Trans. Algorithms*, 10(3):10:1–10:26, 2014. doi:10.1145/2601066.
- 7 Gérard Cornuéjols. General factors of graphs. *J. Comb. Theory, Ser. B*, 45(2):185–198, 1988. doi:10.1016/0095-8956(88)90068-8.
- 8 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.
- 9 Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of Baur-Strassen’s theorem: Shortest cycles, diameter, and matchings. *J. ACM*, 62(4):28:1–28:30, 2015. doi:10.1145/2736283.

- 10 Szymon Dudycz and Katarzyna E. Paluch. Optimal general matchings. *CoRR*, abs/1706.07418, 2017. [arXiv:1706.07418](https://arxiv.org/abs/1706.07418).
- 11 Zdenek Dvorák and Martin Kupec. On planar Boolean CSP. In *ICALP (1)*, volume 9134 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2015. [doi:10.1007/978-3-662-47672-7_35](https://doi.org/10.1007/978-3-662-47672-7_35).
- 12 Eduard Eiben, Tomohiro Koana, and Magnus Wahlström. Determinantal sieving. In *SODA*, pages 377–423, 2024. [doi:10.1137/1.9781611977912.16](https://doi.org/10.1137/1.9781611977912.16).
- 13 Tomás Feder. Fanout limitations on constraint systems. *Theor. Comput. Sci.*, 255(1-2):281–293, 2001. [doi:10.1016/S0304-3975\(99\)00288-1](https://doi.org/10.1016/S0304-3975(99)00288-1).
- 14 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. [doi:10.1145/2886094](https://doi.org/10.1145/2886094).
- 15 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. [doi:10.1017/9781107415157](https://doi.org/10.1017/9781107415157).
- 16 Harold N. Gabow and Piotr Sankowski. Algorithms for weighted matching generalizations I: Bipartite graphs, b -matching, and unweighted f -factors. *SIAM J. Comput.*, 50(2):440–486, 2021. [doi:10.1137/16M1106195](https://doi.org/10.1137/16M1106195).
- 17 James F. Geelen, Satoru Iwata, and Kazuo Murota. The linear delta-matroid parity problem. *J. Comb. Theory, Ser. B*, 88(2):377–398, 2003. [doi:10.1016/S0095-8956\(03\)00039-X](https://doi.org/10.1016/S0095-8956(03)00039-X).
- 18 Nicholas J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput.*, 39(2):679–702, 2009. [doi:10.1137/070684008](https://doi.org/10.1137/070684008).
- 19 Masao Ishikawa and Masato Wakayama. Minor summation formula of Pfaffians. *Linear and Multilinear algebra*, 39(3):285–305, 1995.
- 20 Satoru Iwata and Yusuke Kobayashi. A weighted linear matroid parity algorithm. *SIAM J. Comput.*, 51(2):17–238, 2022. [doi:10.1137/17M1141709](https://doi.org/10.1137/17M1141709).
- 21 Naonori Kakimura and Mizuyo Takamatsu. Matching problems with delta-matroid constraints. *SIAM J. Discret. Math.*, 28(2):942–961, 2014. [doi:10.1137/110860070](https://doi.org/10.1137/110860070).
- 22 Alexandr Kazda, Vladimir Kolmogorov, and Michal Rolínek. Even delta-matroids and the complexity of planar Boolean CSPs. *ACM Trans. Algorithms*, 15(2):22:1–22:33, 2019. [doi:10.1145/3230649](https://doi.org/10.1145/3230649).
- 23 Yusuke Kobayashi. Optimal general factor problem and jump system intersection. In Alberto Del Pia and Volker Kaibel, editors, *IPCO*, volume 13904 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2023. [doi:10.1007/978-3-031-32726-1_21](https://doi.org/10.1007/978-3-031-32726-1_21).
- 24 Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Trans. Algorithms*, 10(4):20:1–20:15, 2014. [doi:10.1145/2635810](https://doi.org/10.1145/2635810).
- 25 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. [doi:10.1145/3390887](https://doi.org/10.1145/3390887).
- 26 László Lovász. Matroid matching and some applications. *J. Comb. Theory, Ser. B*, 28(2):208–236, 1980. [doi:10.1016/0095-8956\(80\)90066-0](https://doi.org/10.1016/0095-8956(80)90066-0).
- 27 Kazuki Matoya and Taihei Oki. Pfaffian pairs and parities: Counting on linear matroid intersection and parity problems. *SIAM J. Discret. Math.*, 36(3):2121–2158, 2022. [doi:10.1137/21M1421751](https://doi.org/10.1137/21M1421751).
- 28 Iain Moffatt. Delta-matroids for graph theorists. In Allan Lo, Richard Mycroft, Guillem Perarnau, and Andrew Treglown, editors, *Surveys in Combinatorics 2019*, London Mathematical Society Lecture Note Series, pages 167–220. Cambridge University Press, 2019. [doi:10.1017/9781108649094.007](https://doi.org/10.1017/9781108649094.007).
- 29 Kazuo Murota. *Matrices and matroids for systems analysis*, volume 20. Springer Science & Business Media, 1999.
- 30 James Oxley. *Matroid Theory*. Oxford University Press, 2011.

- 31 Michael O. Rabin and Vijay V. Vazirani. Maximum matchings in general graphs through randomization. *J. Algorithms*, 10(4):557–567, 1989. doi:10.1016/0196-6774(89)90005-9.
- 32 A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and combinatorics. Springer, 2003.
- 33 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.
- 34 Albert W Tucker. A combinatorial equivalence of matrices. *Combinatorial analysis*, pages 129–140, 1960.
- 35 Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.
- 36 Magnus Wahlström. Representative set statements for delta-matroids and the Mader delta-matroid. In *SODA*, pages 780–810, 2024. doi:10.1137/1.9781611977912.31.
- 37 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979. doi:10.1007/3-540-09519-5_73.

Approximation of Spanning Tree Congestion Using Hereditary Bisection

Petr Kolman   

Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Abstract

The *Spanning Tree Congestion* (STC) problem is the following NP-hard problem: given a graph G , construct a spanning tree T of G minimizing its maximum edge congestion where the congestion of an edge $e \in T$ is the number of edges uv in G such that the unique path between u and v in T passes through e ; the optimal value for a given graph G is denoted $\text{STC}(G)$.

It is known that every spanning tree is an $n/2$ -approximation for the STC problem. A long-standing problem is to design a better approximation algorithm. Our contribution towards this goal is an $\mathcal{O}(\Delta \cdot \log^{3/2} n)$ -approximation algorithm where Δ is the maximum degree in G and n the number of vertices. For graphs with a maximum degree bounded by a polylog of the number of vertices, this is an exponential improvement over the previous best approximation.

Our main tool for the algorithm is a new lower bound on the spanning tree congestion which is of independent interest. Denoting by $hb(G)$ the *hereditary bisection* of G which is the maximum bisection width over all subgraphs of G , we prove that for every graph G , $\text{STC}(G) \geq \Omega(hb(G)/\Delta)$.

2012 ACM Subject Classification Mathematics of computing \rightarrow Discrete mathematics; Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Spanning Tree Congestion, Bisection, Expansion, Divide and Conquer

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.63

1 Introduction

The spanning tree congestion problem has been studied from various viewpoints for more than twenty years, yet our ability to approximate it is still extremely limited. It has been shown that every spanning tree is an $n/2$ -approximation [9] but no $o(n)$ -approximation for *general* graphs is known. For graphs with $\omega(n \log^2 n)$ edges, Chandran et al. [2] described an algorithm that constructs in polynomial time a spanning tree with congestion at most $\mathcal{O}(\sqrt{mn})$; combined with the trivial lower bound $\Omega(m/n)$ on the spanning tree congestion, this yields an $\mathcal{O}(n/\log n)$ -approximation. There is also an $\tilde{\mathcal{O}}(n^{1-1/(\sqrt{\log n}+1)})$ -approximation¹ algorithm for graphs with maximum degree bounded by polylog of the number of vertices [5].

On the hardness side, the strongest known lower bound states that no c -approximation with c smaller than $6/5$ is possible unless $P = NP$ [8]. The $\Omega(n)$ gap between the best upper and lower bounds is highly unsatisfactory.

For a detailed overview of other related results, we refer to the survey paper by Otachi [9], to our recent paper [5], and to the new paper by Lampis et al. [7] that deals with the STC problem from the perspective of parameterized complexity.

1.1 Our Results

Our contribution in this paper is twofold. We describe an $\mathcal{O}(\Delta \cdot \log^{3/2} n)$ -approximation algorithm for the spanning tree congestion problem where Δ is the maximum degree in G and n the number of vertices. For graphs with maximum degree bounded by $\Delta = o(n/\log^{3/2} n)$,

¹ The Big-O-Tilde notation $\tilde{\mathcal{O}}$ ignores logarithmic factors.



© Petr Kolman;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 63; pp. 63:1–63:6



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



we get $o(n)$ -approximation; this significantly extends the class of graphs for which sublinear approximation is known, and provides a partial answer to the open problem P2 from our recent paper [5]. Moreover, for graphs with a stronger bound on the maximum degree, the approximation ratio is even better than $o(n)$. For example, for graphs with a maximum degree bounded by polylog of the number of vertices, the approximation is polylogarithmic which is an exponential improvement over the previous best approximation [5].

For graphs excluding any fixed graph as a minor (e.g., planar graphs or bounded genus graphs), we get a slightly better bound of $\mathcal{O}(\Delta \cdot \log n)$ on the approximation ratio.

Our key tool in the algorithm design is a new lower bound on $\text{STC}(G)$ which is our second contribution. In the recent paper [5], we proved that $\text{STC}(G) \geq \frac{b(G)}{\Delta \cdot \log n}$ where $b(G)$ is the bisection of G . We strengthen the bound and prove that $\text{STC}(G) \geq \Omega\left(\frac{hb(G)}{\Delta}\right)$ where $hb(G)$ is the *hereditary bisection* of G which is the maximum of $b(H)$ over all subgraphs H of G . This is a corollary of another new lower bound saying that for every subgraph H of G , $\text{STC}(G) \geq \frac{\beta(H) \cdot n'}{3 \cdot \Delta}$; here $\beta(H)$ is the expansion of H and n' is the number of vertices in H .

1.2 Sketch of the Algorithm

The algorithm uses the standard *Divide and Conquer* framework and is conceptually very simple: partition the graph by a $\frac{2}{3}$ -balanced cut into two or more connected components, solve the problem recursively for each of the components, and arbitrarily combine the spanning trees of the components into a spanning tree of the entire graph. The structure of the algorithm is the same as the structure of our recent $o(n)$ -approximation algorithm [5] for graphs with maximum degree bounded by *polylog*(n) - there is a minor difference in the tool used in the partitioning step and in the stopping condition for the recursion.

It is far from obvious that the *Divide and Conquer* approach works for the spanning tree congestion problem. The difficulty is that there is no apparent relation between $\text{STC}(G)$ and $\text{STC}(H)$ for a subgraph H of G . In the paper [5], we proved that $\text{STC}(G) \geq \frac{\text{STC}(H)}{e(H, G \setminus H)}$ where $e(H, G \setminus H)$ denotes the number of edges between the subgraph H and the rest of the graph G . Note that the bound is very weak when $e(H, G \setminus H)$ is large. Also, note that the bound is tight in the following sense: there exist graphs for which $\text{STC}(G)$ and $\frac{\text{STC}(H)}{e(H, G \setminus H)}$ are equal, up to a small multiplicative constant. For example, let G be a graph obtained from a 3-regular expander H on n vertices by adding a new vertex r and connecting it by an edge to every vertex of H . Then $\text{STC}(H) = \Omega(n)$ (cf. Lemma 4) while $\text{STC}(G) = O(1)$ (consider the spanning tree of G consisting only of all the edges adjacent to the new vertex r).

The main reason for the significant improvement of the bound on the approximation ratio of the algorithm is the new lower bound $\text{STC}(G) \geq \Omega\left(\frac{hb(G)}{\Delta}\right)$ that connects $\text{STC}(G)$ and properties of subgraphs of G in a much tighter way. This connection yields a simpler algorithm with better approximation, broader applicability and simpler analysis.

1.3 Preliminaries

For an undirected graph $G = (V, E)$ and a subset of vertices $S \subset V$, we denote by $E(S, V \setminus S)$ the set of edges between S and $V \setminus S$ in G , and by $e(S, V \setminus S) = |E(S, V \setminus S)|$ the number of these edges. An edge $\{u, v\} \in E$ is also denoted by uv for notational simplicity. For a subset of vertices $S \subseteq V$, $G[S]$ is the subgraph induced by S . By $V(G)$, we mean the vertex set of the graph G and by $E(G)$ its edge set. Given a graph $G = (V, E)$ and an edge $e \in E$, $G \setminus e$ is the graph $(V, E \setminus \{e\})$.

Let $G = (V, E)$ be a connected graph and $T = (V, E_T)$ be a spanning tree of G . For an edge $uv \in E_T$, we denote by $S_u, S_v \subset V$ the vertex sets of the two connected components of $T \setminus uv$ containing u and v , resp. The *congestion* $c(uv)$ of the edge uv with respect to G and T , is the number of edges in G between S_u and S_v . The *congestion* $c(G, T)$ of the spanning tree T of G is defined as $\max_{e \in E_T} c(e)$, and the *spanning tree congestion* $\text{STC}(G)$ of G is defined as the minimum value of $c(G, T)$ over all spanning trees T of G .

A *bisection* of a graph with n vertices is a partition of its vertices into two sets, S and $V \setminus S$, each of size at most $\lceil n/2 \rceil$. The *width* of a bisection $(S, V \setminus S)$ is $e(S, V \setminus S)$. The minimum width of a bisection of a graph G is denoted $b(G)$. The *hereditary bisection width* $hb(G)$ is the maximum of $b(H)$ over all subgraphs H of G . In approximation algorithms, the requirement that each of the two parts in a partition of V is of size at most $\lceil n/2 \rceil$ is sometimes relaxed to $2n/3$, or to some other fraction, and then we talk about balanced cuts. In particular, a c -balanced cut is a partition of the graph vertices into two sets, each of size at most $c \cdot n$. The *edge expansion* of G is

$$\beta(G) = \min_{A \subseteq V} \frac{e(A, V \setminus A)}{\min\{|A|, |V \setminus A|\}}. \quad (1)$$

There are several approximation and pseudo-approximation algorithms for bisection and balanced cuts. In our algorithm, we will employ the algorithm by Arora, Rao and Vazirani [1], and for graphs excluding any fixed graph as a minor (e.g., planar graphs), a slightly stronger algorithm by Klein, Protkin and Rao [4].

► **Theorem 1** ([1, 4]). *A $2/3$ -balanced cut of cost within a ratio of $\mathcal{O}(\sqrt{\log n})$ of the optimum bisection can be computed in polynomial time. For graphs excluding any fixed graph as a minor, even $\mathcal{O}(1)$ ratio is possible.*

We conclude this section with two more statements that will be used later.

► **Theorem 2** (Jordan [3]). *Given a tree on n vertices, there exists a vertex whose removal partitions the tree into components, each with at most $n/2$ vertices.*

► **Lemma 3** (Kolman and Matoušek [6]). *Every graph G on n vertices contains a subgraph on at least $2 \cdot n/3$ vertices with edge expansion at least $b(G)/n$.*

2 New Lower Bound

The main result of this section is captured in the following lemma and its corollary.

► **Lemma 4.** *For every graph $G = (V, E)$ on n vertices with maximum degree Δ and every subgraph H of G on n' vertices, we have*

$$\text{STC}(G) \geq \frac{\beta(H) \cdot n'}{3 \cdot \Delta}. \quad (2)$$

► **Corollary 5.** *For every graph $G = (V, E)$ with maximum degree Δ ,*

$$\text{STC}(G) \geq \frac{2 \cdot hb(G)}{9 \cdot \Delta}. \quad (3)$$

Before proving the lemma and its corollary, we state a slight generalization of Theorem 2; for the sake of completeness, we also provide proof of it, though it is a straightforward extension of the standard proof of Theorem 2.

► **Lemma 6.** *Given a tree T on n vertices with $n' \leq n$ vertices marked, there exists a vertex (marked or unmarked) whose removal partitions the tree into components, each with at most $n'/2$ marked vertices.*

Proof. Start with an arbitrary vertex $v_0 \in T$ and set $i = 0$. We proceed as follows. If the removal of v_i partitions the tree into components such that each contains at most $n'/2$ marked vertices, we are done. Otherwise, one of the components, say a component C , has strictly more than $n'/2$ marked vertices. Let v_{i+1} be the neighbour of v_i that belongs to the component C . Note that for every $i > 0$, v_i is different from all the vertices v_0, v_1, \dots, v_{i-1} . As the number of vertices in the tree is bounded, eventually, this process has to stop, and we get to a vertex with the desired properties. ◀

Proof of Lemma 4. Let T be the spanning tree of G with the minimum congestion. By Lemma 6, there exists a vertex $z \in T$ whose removal partitions the tree T into components, each with at most $n'/2$ vertices from H . We organize the components of $T \setminus z$ into two parts so that the total number of vertices from H in the smaller part is at least $n'/3$; such a partition can be found greedily. Let $C \subseteq V(H)$ be the vertices from H in the smaller part. Then, by the definition of expansion (1), $e(C, V(H) \setminus C) \geq \beta(H) \cdot n'/3$. As for each edge $uv \in E(C, V(H) \setminus C)$, the path connecting u and v in T uses at least one edge adjacent to z , we conclude that

$$\text{STC}(G) \geq \frac{e(C, V(H) \setminus C)}{\Delta} \geq \frac{\beta(H) \cdot n'}{3 \cdot \Delta}. \quad \blacktriangleleft$$

Proof of Corollary 5. Consider a subgraph H' of G such that $b(H') = hb(G)$. By Lemma 3, there is a subgraph H of H' , such that $|V(H)| \geq 2 \cdot |V(H')|/3$ and $\beta(H) \geq b(H')/|V(H')|$. Since H is a subgraph of G , by Lemma 4,

$$\text{STC}(G) \geq \frac{\beta(H) \cdot |V(H)|}{3 \cdot \Delta} \geq \frac{2 \cdot b(H')}{3 \cdot 3 \cdot \Delta} = \frac{2 \cdot hb(G)}{9 \cdot \Delta}. \quad \blacktriangleleft$$

3 Approximation Algorithm

Given a connected graph $G = (V, E)$, we construct the spanning tree of G by the recursive algorithm CONGSPANTREE called on the graph G . In step 3, one of the algorithms of Theorem 1 is used: for general graphs, the algorithm by Arora, Rao and Vazirani [1], for graphs excluding any fixed graph as a minor, the algorithm by Klein, Protkin and Rao; by $\alpha(n)$ we denote the respective pseudo-approximation factor.

■ **Algorithm 1** CONGSPANTREE(H).

```

1: if  $|V(H)| = 1$  then
2:   return  $H$ 
3: construct a  $\frac{2}{3}$ -balanced cut  $(S, V(H) \setminus S)$  of  $H$ 
4:  $F \leftarrow E(S, V(H) \setminus S)$ 
5: for each connected component  $C$  of  $H \setminus F$  do
6:    $T_C \leftarrow \text{CONGSPANTREE}(C)$ 
7: arbitrarily connect all the trees  $T_C$  by edges from  $F$  to form a spanning tree  $T$  of  $H$ 
8: return  $T$ 

```

Let τ denote the tree representing the recursive decomposition of G (implicitly) constructed by the algorithm CONGSPANTREE: The root r of τ corresponds to the graph G , and the children of a non-leaf node $t \in \tau$ associated with a set V_t correspond to the connected

components of $G[V_i] \setminus F$ where F is the set of edges of the $\frac{2}{3}$ -balanced cut of $G[V_i]$ from step 4; by Theorem 1, $|F| \leq \alpha(n) \cdot b(G[V_i])$. We denote by $G_t = G[V_i]$ the subgraph of G induced by the vertex set V_i , by T_t the spanning tree constructed for G_t by the algorithm CONGSPANTREE. The *height* $h(t)$ of a tree node $t \in \tau$ is the number of edges on the longest path from t to a leaf in its subtree (i.e., to a leaf that is a descendant of t).

► **Lemma 7.** *Let $t \in \tau$ be a node of the decomposition tree and t_1, \dots, t_k its children. Then*

$$c(G_t, T_t) \leq \max_i c(G_{t_i}, T_{t_i}) + \alpha(n) \cdot b(G_t) . \quad (4)$$

Proof. Let F be the set of edges of the $\frac{2}{3}$ -balanced cut of G_t from step 4. We will show that for every edge $e \in E(T_t)$, its congestion $c(e)$ with respect to G_t and T_t is at most $\max_i c(G_{t_i}, T_{t_i}) + |F|$; as $|F| \leq \alpha(n) \cdot b(G[V_i])$, this will prove the lemma. Recall that $E(T_t) \subseteq \bigcup_{i=1}^k E(T_{t_i}) \cup F$, as the spanning tree T_t is constructed (step 7) from the spanning trees T_{t_1}, \dots, T_{t_k} and the set F .

Consider first an edge $e \in E(T_t)$ that belongs to a tree T_{t_i} , for some i . The only edges from $E(G)$ that may contribute to the congestion $c(e)$ of e with respect to G_t and T_t are the edges in $E(G_{t_i}) \cup F$; the contribution of the edges in $E(G_{t_i})$ is at most $c(G_{t_i}, T_{t_i})$, the contribution of the edges in F is at most $|F|$. Thus, the congestion $c(e)$ of the edge e with respect to G_t and T_t is at most $c(G_{t_i}, T_{t_i}) + |F|$.

Consider now an edge $e \in F \cap E(T_t)$. As the only edges from $E(G)$ that may contribute to the congestion $c(e)$ of e with respect to G_t and T_t are the edges in F , its congestion is at most $|F|$.

Thus, for every edge $e \in E(T_t)$, its congestion with respect to G_t and T_t is at most $\max_i c(G_{t_i}, T_{t_i}) + |F|$, and the proof of the lemma is completed. ◀

► **Lemma 8.** *Let $T = \text{CONGSPANTREE}(G)$. Then*

$$c(G, T) \leq \mathcal{O}(\alpha(n) \cdot \log n) \cdot hb(G) . \quad (5)$$

Proof. For technical reasons, we extend the notion of the spanning tree congestion also to the trivial graph $H = (\{v\}, \emptyset)$ consisting of a single vertex and no edge (and having a single spanning tree $T_H = H$) by defining $c(H, T_H) = 0$.

By induction on the height of vertices in the decomposition tree τ , we prove the following auxiliary claim: for every $t \in \tau$,

$$c(G_t, T_t) \leq h(t) \cdot \alpha(n) \cdot hb(G) . \quad (6)$$

Consider first a node $t \in \tau$ of height zero, that is, a node t that is a leaf. Then both sides of (6) are zero and the inequality holds.

Consider now a node $t \in \tau$ such that for all his children the inequality (6) holds. Let t' be the child of the node t for which $c(G_{t'}, T_{t'})$ is the largest among the children of t . Then, as $b(G_t) \leq hb(G)$ by the definition of hb , by Lemma 7 we get

$$c(G_t, T_t) \leq c(G_{t'}, T_{t'}) + \alpha(n) \cdot hb(G) .$$

By the inductive assumption applied on the node t' ,

$$c(G_{t'}, T_{t'}) \leq h(t') \cdot \alpha(n) \cdot hb(G) .$$

Because $h(t') + 1 \leq h(t)$, the proof of the auxiliary claim is completed.

Observing that the height of the root of the decomposition tree τ is at most $\mathcal{O}(\log n)$, as all cuts used by the algorithm are balanced, the proof is completed. ◀

► **Theorem 9.** *Given a graph G with maximum degree Δ , the algorithm `CONGSPANTREE` constructs an $\mathcal{O}(\Delta \cdot \log^{3/2} n)$ -approximation of the minimum congestion spanning tree; for graphs excluding any fixed graph as a minor, the approximation is $\mathcal{O}(\Delta \cdot \log n)$.*

Proof. By Corollary 5, for every graph G , $\Omega(\text{hb}(G)/\Delta)$ is a lower bound on $\text{STC}(G)$. By Lemma 8, the algorithm `CONGSPANTREE`(G) constructs a spanning tree T of congestion at most $\mathcal{O}(\alpha(n) \cdot \log n) \cdot \text{hb}(G)$. Combining these two results yields the theorem: $c(G, T) \leq \mathcal{O}(\alpha(n) \cdot \log n) \cdot \text{hb}(G) \leq \mathcal{O}(\alpha(n) \cdot \log n \cdot \Delta) \cdot \text{STC}(G)$. Plugging in the bounds on $\alpha(n)$ from Theorem 1 yields the theorem. ◀

4 Open Problems

The inevitable question is whether it is possible to eliminate the dependency of the approximation ratio of the algorithm on the largest degree Δ in the graph and obtain an $o(n)$ -approximation algorithm for the STC problem for all graphs.


References

- 1 Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5:1–5:37, 2009. Preliminary version in *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2004. doi:10.1145/1502793.1502794.
- 2 L. Sunil Chandran, Yun Kuen Cheung, and Davis Issac. Spanning tree congestion and computation of generalized Györi-Lovász partition. In *Proc. of 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPICs*, pages 32:1–32:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.32.
- 3 Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869.
- 4 Philip N. Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proc. of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 682–690, 1993. doi:10.1145/167088.167261.
- 5 Petr Kolman. Approximating spanning tree congestion on graphs with polylog degree. In *Proc. of International Workshop on Combinatorial Algorithms (IWOCA)*, pages 497–508, 2024. doi:10.1007/978-3-031-63021-7_38.
- 6 Petr Kolman and Jiří Matoušek. Crossing number, pair-crossing number, and expansion. *Journal of Combinatorial Theory, Series B*, 92(1):99–113, 2004. doi:10.1016/j.jctb.2003.09.002.
- 7 Michael Lampis, Valia Mitsou, Edouard Nemery, Yota Otachi, Manolis Vasilakis, and Daniel Vaz. Parameterized spanning tree congestion, 2024. doi:10.48550/arXiv.2410.08314.
- 8 Huong Luu and Marek Chrobak. Better hardness results for the minimum spanning tree congestion problem. *Algorithmica*, pages 1–18, 2024. Preliminary version in *Proc. of 17th International Conference and Workshops on Algorithms and Computation (WALCOM)*, 2023. doi:10.1007/s00453-024-01278-5.
- 9 Yota Otachi. A survey on spanning tree congestion. In *Treewidth, Kernels, and Algorithms: Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 165–172, 2020. doi:10.1007/978-3-030-42071-0_12.


Cluster Editing on Cographs and Related Classes

Manuel Lafond ✉ 

Department of Computer Science, Université de Sherbrooke, Canada

Alitzel López Sánchez ✉ 

Department of Computer Science, Université de Sherbrooke, Canada

Weidong Luo ✉ 

Department of Computer Science, Université de Sherbrooke, Canada

Abstract

In the CLUSTER EDITING problem, sometimes known as (unweighted) CORRELATION CLUSTERING, we must insert and delete a minimum number of edges to achieve a graph in which every connected component is a clique. Owing to its applications in computational biology, social network analysis, machine learning, and others, this problem has been widely studied for decades and is still undergoing active research. There exist several parameterized algorithms for general graphs, but little is known about the complexity of the problem on specific classes of graphs.

Among the few important results in this direction, if only deletions are allowed, the problem can be solved in polynomial time on cographs, which are the P_4 -free graphs. However, the complexity of the broader editing problem on cographs is still open. We show that even on a very restricted subclass of cographs, the problem is NP-hard, W[1]-hard when parameterized by the number p of desired clusters, and that time $n^{o(p/\log p)}$ is forbidden under the ETH. This shows that the editing variant is substantially harder than the deletion-only case, and that hardness holds for the many superclasses of cographs (including graphs of clique-width at most 2, perfect graphs, circle graphs, permutation graphs). On the other hand, we provide an almost tight upper bound of time $n^{O(p)}$, which is a consequence of a more general $n^{O(cw \cdot p)}$ time algorithm, where cw is the clique-width. Given that forbidding P_4 s maintains NP-hardness, we look at $\{P_4, C_4\}$ -free graphs, also known as trivially perfect graphs, and provide a cubic-time algorithm for this class.

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

Keywords and phrases Cluster editing, cographs, parameterized algorithms, clique-width, trivially perfect graphs

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.64

Related Version *Full Version*: <https://arxiv.org/abs/2412.12454>

Acknowledgements We thank the anonymous reviewers for their valuable comments.

1 Introduction

Clustering objects into groups of similarity is a ubiquitous task in computer science, with applications in computational biology [45, 40], social network analysis [46, 1, 2], machine learning [16, 18], and many others [4]. There are many interpretations of what a “good” clustering is, with one of the most simple, elegant, and useful being the CLUSTER EDITING formulation – sometimes also known as (unweighted) CORRELATION CLUSTERING [3]. In this graph-theoretical view, pairs of objects that are believed to be similar are linked by an edge, and non-edges correspond to dissimilar objects. If groups are perfectly separable, this graph should be a *cluster graph*, that is, a graph in which each connected component is a clique. However, due to noise and errors, this is almost never observed in practice. To remove such errors, CLUSTER EDITING asks for a minimum number of edges to correct to obtain a cluster graph, where “correcting” means adding a non-existing edge or deleting an edge.



© Manuel Lafond, Alitzel López Sánchez, and Weidong Luo;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 64; pp. 64:1–64:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Owing to its importance, this APX-hard [12], of course also NP-hard [3, 37], problem has been widely studied in the parameterized complexity community. Let k be the number of required edge modifications. After a series of works, the problem now can be solved in time $O^*(1.62^k)$ [6, 7, 8, 24, 25] and admits a $2k$ kernel [11, 13, 26]. In addition, if we require that the solution contains exactly p clusters, then the problem is NP-hard for every $p \geq 2$ [45], but admits a PTAS [23], a $(p+2)k + p$ kernel [26], and can be solved in $2^{O(\sqrt{pk})}n^{O(1)}$ time. This is shown to be tight assuming the ETH, under which $2^{o(\sqrt{pk})}n^{O(1)}$ is not possible [19].

Another angle, which we study in this paper, is to focus on specific classes of graphs. For example, restricting the input to bounded-degree graphs does not help, as CLUSTER EDITING is NP-hard even on planar unit disk graphs with maximum degree 4 [35, 43]. In [5], the authors circumvent the APX-hardness of the problem by proposing a PTAS on planar graphs. A polynomial-time algorithm is provided for the problem on unit interval graphs [41], a subclass of unit disk. The CLUSTER DELETION problem, in which only edge deletions are allowed, has received much more attention on restricted classes. It is polynomial-time solvable on graphs of maximum degree 3 and NP-hard for larger degrees [35]. Various results were also obtained on interval and split graphs [36], other subclasses of chordal graphs [9], and unit disk graphs [43]. In [31], graphs with bounded structural parameters are studied, with the weighted variant being paraNP-hard in twin cover, but FPT in the unweighted case.

If we forbid specific induced subgraphs, the reduction in [35] implies that CLUSTER DELETION is NP-hard on C_4 -free graphs (as observed in [21]). If instead we forbid P_4 , i.e., induced paths on four vertices, we obtain the class of *cographs*, on which the deletion problem is remarkably shown to be polynomial-time solvable in [21] using a greedy max-clique strategy. However, the complexity of CLUSTER EDITING on cographs has remained open. In addition, to our knowledge, there are no known non-trivial polynomial-time algorithms for CLUSTER EDITING on specific graph classes with a finite set of forbidden induced subgraphs. It is not hard to obtain a polynomial-time algorithm for the problem on the *threshold graphs*, i.e. the $\{P_4, C_4, 2K_2\}$ -free graphs, using standard dynamic programming on its co-tree. However, it appears to be unknown whether removing any of the three induced subgraphs from this set leads to NP-hardness.

In this paper, we focus on open complexity questions for the CLUSTER EDITING problem on cographs and related classes. It is worth mentioning that the cograph restriction is more than a mere complexity classification endeavor – it can be useful to determine how well an equivalence relation (i.e., a cluster graph) can approximate a different type of relation (see for example [47]). In the case of cograph-like relations, our motivations have roots in phylogenetics. In this field, gene pairs are classified into *orthologs* and *paralogs*, with orthology graphs known to correspond exactly to cographs [29, 38, 30]. However, as argued in [44], most orthology prediction software use clustering libraries and infer a cluster graph of orthologs. The question that arises is then “how much information is lost by predicting a cluster graph, knowing that the true graph is a cograph”? This requires finding a cluster graph that is the closest to a given cograph G , leading to CLUSTER EDITING on cographs. Furthermore, researchers argue that social communities should sometimes be modeled as cographs [33] or trivially perfect graphs [42], as opposed to cluster graphs as is done in most community detection approaches. This leads to CLUSTER EDITING on cographs or trivially perfect graphs. Additionally, an algorithm for the NP-hard TRIVIAL PERFECT GRAPH EDITING, which can practical scalability to large real-world graphs, is provided in [10].

Our contributions. We first settle the complexity of CLUSTER EDITING on cographs by showing that it is not only NP-hard, but also $W[1]$ -hard when a parameter p is specified, which represents the number of desired clusters. We use the UNARY BIN PACKING hardness

results from [32], which also implies an Exponential Time Hypothesis (ETH) lower bound that forbids time $n^{o(p/\log p)}$ under this parameter. In fact, our hardness holds for very restricted classes of cographs, namely graphs obtained by taking two cluster graphs, and adding all possible edges between them (this also correspond to cographs that admit a cotree of height 3). Moreover, because cographs have *clique-width* (cw) at most 2, this also means that the problem is para-NP-hard in clique-width, and that a complexity of the form $n^{g(cw)}$ is unlikely, for any function g (the same actually holds for the *modular-width* parameter and generalizations, see [20, 39]). In fact, the ETH forbids time $f(p)n^{g(cw)\cdot o(p/\log p)}$ for any functions f and g , which contrasts with the aforementioned subexponential bounds in pk [19].

The hardness also extends to all superclasses of cographs, such as circle graphs, perfect graphs, and permutation graphs. On the other hand we show that time $n^{O(p)}$ can be achieved on any cograph, which is almost tight. This contrasts with the general CLUSTER EDITING problem which is NP-hard when $p = 2$. In fact, this complexity follows from a more general algorithm for arbitrary graphs that runs in time $n^{O(cw\cdot p)}$, which shows that CLUSTER EDITING is XP in parameter $cw + p$. Note that our hardness results imply that XP membership in either parameter individually is unlikely, and so under standard assumptions both cw and p must contribute in the exponent of n .

Finally, we aim to find the largest subclass of cographs on which CLUSTER EDITING is polynomial-time solvable. The literature mentioned above implies that such a class lies somewhere between P_4 -free and $\{P_4, C_4, 2K_2\}$ -free graphs. We improve the latter by showing that CLUSTER EDITING can be solved in time $O(n^3)$ on $\{P_4, C_4\}$ -free graphs, also known as trivially perfect graphs (TPG). This result is achieved by a characterization of optimal clusterings on TPGs, which says that as we build a clustering going up in the cotree, only the largest cluster is allowed to become larger as we proceed.

Our results are summarized in the following, where n is the number of vertices of the graphs, and p -CLUSTER EDITING is the variant in which the edge modifications must result in p connected components that are cliques. We treat p as a parameter specified in the input.

► **Theorem 1.** *The following results hold:*

- CLUSTER EDITING is NP-complete on cographs, and solvable in time $O(n^3)$ on trivially perfect graphs.
- p -CLUSTER EDITING admits an $n^{O(cw\cdot p)}$ time algorithm if a cw -expression is given, but admits no $f(p)n^{g(cw)\cdot o(p/\log p)}$ time algorithm for any functions f and g unless ETH fails.
- p -CLUSTER EDITING on cographs is NP-complete, and $W[1]$ -hard parameterized by p .
- p -CLUSTER EDITING on cographs admits an $n^{O(p)}$ time algorithm, but admits no $f(p)n^{o(p/\log p)}$ time algorithm for any function f unless ETH fails.

2 Preliminaries

We use the notation $[n] = \{1, \dots, n\}$. For two sets A and B , $A \triangle B$ is the symmetric difference between A and B . For a graph G , $V(G)$ and $E(G)$ are the vertex and edge sets of G , respectively, and $G[S]$ is the subgraph induced by $S \subseteq V(G)$. The complement of G is denoted \overline{G} . Given two graphs G and H , the *disjoint union* $G \cup H$ is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. The *join* $G \vee H$ of two graphs G and H is the graph obtained from $G \cup H$ by adding every possible edge uv with $u \in V(G)$ and $v \in V(H)$.

It will be useful to consider two equivalent views on the CLUSTER EDITING problem, in terms of the edge operations to perform to achieve a cluster graph, and in terms of the resulting partition into clusters. A graph is a *cluster graph* if it is a disjoint union of complete

graphs. Let $G = (V, E)$ be a graph and $F \subseteq V \times V$. If $G' = (V, E \triangle F)$ is a cluster graph, then F is called a *cluster editing set*. The edges of F can be divided into two types: $F \cap E(G)$ are called *deleted edges*, and $F \setminus E(G)$ are called *inserted edges*, where the deleted edges *disconnect* some adjacent vertices in G and the inserted edges *connect* some non-adjacent vertices in G to transform G into G' .

Note that the clusters of G' result in a partition of $V(G)$. Conversely, given any partition \mathcal{C} of $V(G)$, we can easily infer the editing set F that yields the clusters \mathcal{C} : it consists of non-edges within the clusters, and of edges with endpoints in different clusters. To be precise, a *clustering* of G is a partition $\mathcal{C} = \{C_1, \dots, C_l\}$ of $V(G)$. The *cluster editing set* of \mathcal{C} is

$$\text{edit}(\mathcal{C}) := \{uv \in E(G) : u \in C_i, v \in C_j, i \neq j\} \cup \bigcup_{i \in [l]} E(\overline{G[C_i]}).$$

We define $\text{cost}_G(\mathcal{C}) = |\text{edit}(\mathcal{C})|$. An element of $\mathcal{C} \in \mathcal{C}$ is called a *cluster*, and the cardinality of \mathcal{C} is called *cluster number*. An *optimal cluster editing set* for G is a cluster editing set for G of minimum size. An *optimal clustering* is a partition \mathcal{C} of $V(G)$ of minimum cost.

A formal definition of CLUSTER EDITING problem is as follows.

CLUSTER EDITING

Input: A graph G and an integer k .

Question: Is there a clustering of G with cost at most k ?

A clustering with exactly p clusters is called a *p-clustering*. In p -CLUSTER EDITING, the problem is the same, but we must find a p -clustering of cost at most k .

We will sometimes use the fact that twins, which are pairs of vertices that have the same closed neighborhood, can be assumed to be in the same cluster. More generally, a *critical clique* of a graph G is a clique K such that all $v \in K$ have the same neighbor vertices in $V(G) \setminus K$, and K is maximal under this property.

► **Proposition 2** ([13, 26]). *Let K be a critical clique of G . For any optimal clustering \mathcal{C} of G , there is a $C \in \mathcal{C}$ such that $K \subseteq C$.*

Note that Proposition 2 is not always true if we require a p -clustering instead of any optimal clustering. For example if p is imposed and G is a complete graph with p vertices, then G consists of a critical clique which be broken. We will ensure that this is not problematic in the results that follow.

Cographs and cotrees. A *cograph* is a graph that can be constructed using the three following rules: (1) a single vertex is a cograph; (2) the disjoint union $G \cup H$ of cographs G, H is a cograph; (3) the join $G \vee H$ of cographs G, H is a cograph. Cographs are exactly the P_4 -free graphs, i.e., that do not contain a path on four vertices as an induced subgraph.

Cographs are also known for their tree representation. For a graph G , a *cotree* for G is a rooted tree T whose set of leaves, denoted $L(T)$, satisfies $L(T) = V(G)$. Moreover, every internal node $v \in V(T) \setminus L(T)$ is one of two types, either a 0-node or a 1-node, such that $uv \in E(G)$ if and only if the lowest common ancestor of u and v in T is a 1-node¹. It is well-known that G is a cograph if and only if there exists a cotree T for G . This can be seen from the intuition that leaves represent applications of Rule (1) above, 0-node represents disjoint unions, and 1-node represents joins.

¹ To emphasize the distinction between general graphs and trees, we will refer to vertices of a tree as *nodes*.

A *trivially perfect graph* (TPG), among several characterizations, is a cograph G that has no induced cycle on four vertices, i.e., a $\{P_4, C_4\}$ -free graph. TPGs are also the chordal cographs. For our purposes, a TPG is a cograph that admits a cotree T in which every 1-node has at most one child that is not a leaf (see [28, Lemma 4.1]).

Clique-width and NLC-width. Our clique-width (cw) algorithm does not use the notion of cw directly, but instead the analogous measure of *NLC-width* [28]. We only provide the definition of the latter. Let $G = (V, E, lab)$ be a graph in which every vertex has one of k node labels (k -NL), where lab is a function from V to $[k]$. A single-vertex graph labeled t is denoted by \bullet_t .

► **Definition 3.** A k -node labeled controlled (k -NLC) graph is a k -NL graph defined recursively as follows.

1. \bullet_t is a k -NLC graph for every $t \in [k]$.
2. Let $G_1 = (V_1, E_1, lab_1), G_2 = (V_2, E_2, lab_2)$ be two k -NLC graphs, relation $S \subseteq [k]^2$, and $E_{add} = \{uv : u \in V_1 \wedge v \in V_2 \wedge (lab_1(u), lab_2(v)) \in S\}$.
The k -NL graph $G = (V, E, lab)$ denoted $G_1 \times_S G_2$ is a k -NLC graph defined as: $V = V_1 \cup V_2$, $E = E_1 \cup E_2 \cup E_{add}$, and $lab(u) = lab_1(u)$, $lab(v) = lab_2(v)$ for $u \in V_1$, $v \in V_2$.
3. Let $G = (V, E, lab)$ be a k -NLC graph and R be a function from $[k]$ to $[k]$. Then $\circ_R(G) = (V, E, lab')$ is a k -NLC graph, where $lab'(v) = R(lab(v))$ for all $v \in V$.

Intuitively, operation 2 denoted by \times_S takes the disjoint union of the graphs G_1, G_2 , then adds all possible edges between labeled vertices of G_1 and labeled vertices of G_2 as controlled by the pairs of S . Operation 3 denoted by \circ_R relabels the vertices of a graph controlled by R . NLC_k denotes all k -NLC graphs. The *NLC-width* of a labeled graph G is the smallest k such that G is a k -NLC graph. Furthermore, for a labeled graph $G = (V, E, lab)$, the *NLC-width* of a graph $G' = (V, E)$, obtained from G with all labels removed, equals the *NLC-width* of G .

A well-parenthesized expression X built with operations 1, 2, 3 is called a k -expression. The graph constructed by X is denoted by G_X . We associate the k -expression tree T of G_X with X in a natural way, that is, leaves of T correspond to all vertices of G_X and the internal nodes of T correspond to the operations 2, 3 of X . For each node u of T , the sub-tree rooted at u corresponds to a *sub k -expression* of X denoted by X_u , and the graph G_{X_u} constructed by X_u is also denoted by G^u . Moreover, we say G^u is the *related graph* of u in T .

Let us briefly mention that a graph has clique-width at most k if it can be built using what we will call a $cw(k)$ -expression, which has four operations instead: creating a single labeled vertex \bullet_t ; taking the disjoint union; adding all edges between vertices of a specified label pair (i, i') ; relabeling all vertices with label i to another label j . We do not detail further, since the following allows us to use NLC-width instead of clique-width.

► **Proposition 4** ([27, 34]). *For every graph G , the clique-width of G is at least the NLC-width of G , and at most two times the NLC-width of G . Moreover, any given $cw(k)$ -expression can be transformed into an equivalent k -expression in polynomial time (i.e., yielding the same graph).*

We will assume that our k -expressions are derived from a given $cw(k)$ -expression. Reference [34] is often cited for this transformation, but seems to have vanished from nature. The transformation can be done using the normal form of a $cw(k)$ -expression described in [17].

3 Cluster editing on cographs

We first prove our hardness results for CLUSTER EDITING and p -CLUSTER EDITING on cographs, using a reduction from UNARY BIN PACKING. An instance of UNARY BIN PACKING consists of a unary-encoded integer multiset $A = [a_1, \dots, a_n]$, which represent the sizes of n items, and integers b, k . We must decide whether the items can be packed into at most k bins that each have capacity b . Thus, we must partition A into k multisets A_1, \dots, A_k , some possibly empty, such that $\sum_{a \in A_i} a \leq b$ for each $i \in [k]$.

For our purpose, we introduce a variant of this problem called UNARY PERFECT BIN PACKING. The problem is identical, except that the partition of A into A_1, \dots, A_k must satisfy $\sum_{a \in A_i} a = b$ for each $i \in [k]$. That is, we must fill all k bins to their maximum capacity b . Note that for this to be possible, $\sum_{i=1}^n a_i = kb$ must hold. Note that these packing problems can be solved in polynomial time for any fixed k [32]. We assume henceforth that $k, n \geq 10$, $\max_{i=1}^n a_i \leq b$, and $\sum_{i=1}^n a_i \leq kb$, otherwise, they can be decided in polynomial time. It is known that UNARY BIN PACKING is NP-complete [22], W[1]-hard parameterized by the number of bins k [32], and does not admit an $f(k)|I|^{o(k/\log k)}$ time algorithm for any function f unless Exponential Time Hypothesis (ETH) fails [32], where $|I|$ is the size of the instance (in unary). The results easily extend to the perfect version.

► **Proposition 5.** *UNARY PERFECT BIN PACKING is NP-complete, W[1]-hard parameterized by k , and has no $f(k)|I|^{o(k/\log k)}$ time algorithm for any function f unless ETH fails.*

Proof. Clearly, UNARY PERFECT BIN PACKING is in NP. Let (A, b, k) be an instance of UNARY BIN PACKING, where $A = [a_1, \dots, a_n]$. We assume that k is even, as otherwise we increase k by 1 and add to A an item of value b . If $\sum_{i=1}^n a_i \leq 0.1kb$, we argue that we have a YES instance: we can pack the largest $k/2$ of the n items into $k/2$ bins, and pack the other $n - k/2$ items in the remaining bins, as follows. Assume w.l.o.g. $b \geq a_1 \geq \dots \geq a_{k/2} \geq a_{k/2+1} \geq \dots \geq a_n$. We have $k/2 \cdot a_{k/2} \leq \sum_{i=1}^{k/2} a_i \leq 0.1kb$, so $a_{k/2} \leq 0.2b$. This means that $a_{k/2+1}, \dots, a_n \leq 0.2b$ and we can always select a batch of items with these sizes such that the total size of a batch is in the interval $[0.8b, b]$ until there are no items left (the last batch may have a size less than $0.8b$). Thus, we can pack the items with sizes $a_{k/2+1}, \dots, a_n$ using at most $\frac{0.1kb}{0.8b} + 1 = 0.125k + 1 < k/2$ bins.

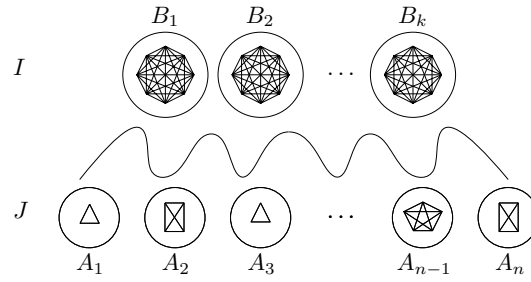
So assume henceforth that $\sum_{i=1}^n a_i > 0.1kb$. Construct an instance (B, b, k) for UNARY PERFECT BIN PACKING, where $B = [a_1, \dots, a_n, 1, \dots, 1]$ consists of all integers of A and $kb - \sum_{i=1}^n a_i$ 1s. Since $0.1kb < \sum_{i=1}^n a_i \leq kb$, the new instance size is bounded by a linear function of the original instance size. Moreover, the new instance can be obtained in polynomial time. It is easy to verify that (A, b, k) is a YES instance of UNARY BIN PACKING if and only if (B, b, k) is a YES instance of UNARY PERFECT BIN PACKING. ◀

► **Lemma 6.** *Given a unary-encoded integer multiset $A = [a_1, \dots, a_n]$ for the sizes of n items, and integers b, k satisfying $kb = \sum_{i=1}^n a_i$, there is a polynomial-time algorithm which outputs a cograph G and an integer t such that,*

1. *for any optimal clustering \mathcal{C} of G , $|\mathcal{C}| = k$ and the cost of \mathcal{C} is at least t ;*
2. *the n items can be perfectly packed by k bins with capacity b if and only if there is a clustering of G with cost at most t .*

Moreover, G is obtained by taking a join of two cluster graphs.

Proof. For the remainder, let us denote $a := \sum_{i=1}^n a_i$ and $s := \sum_{i=1}^n a_i^2$. Construct an instance (G, t) for CLUSTER EDITING as follows. First, add two cluster graphs $I = B_1 \cup \dots \cup B_k$ and $J = A_1 \cup \dots \cup A_n$ into G , where B_i is a complete graph with $h := (nka)^{10}$ vertices for



■ **Figure 1** An illustration of the construction. In the subgraph I , each B_i is a “large enough” complete graph, and in the subgraph J , each A_j is a complete graph with a_j vertices. The wiggly line indicates that all possible edges between I and J are present (there are no edges between two B_i ’s, and no edge between two A_j ’s).

every $i \in [k]$, and A_j is a complete graph with a_j vertices for every $j \in [n]$. Then, connect each $v \in V(I)$ to all vertices of $V(J)$. See Figure 1. One can easily verify that G is a cograph obtained from the join of two cluster graphs. In addition, define $t := (k - 1)ah + \frac{1}{2}(kb^2 - s)$. Clearly, t is an integer since $s \equiv a \equiv kb \equiv kb^2 \pmod{2}$. Let $\mathcal{I} = \{V(B_1), \dots, V(B_k)\}$ and $\mathcal{J} = \{V(A_1), \dots, V(A_n)\}$. Clearly, every element in $\mathcal{I} \cup \mathcal{J}$ is a critical clique of G .

▷ **Claim 7.** Let \mathcal{C} be an optimal clustering of G , and F be the cluster editing set for this solution. Then, $|\mathcal{C}| = k$, and each element of \mathcal{C} is a superset of exactly one element from \mathcal{I} . Moreover, $|F| \geq t$, and $|F| = t$ if and only if all elements of \mathcal{C} have the same cardinality.

Proof. Let $\mathcal{C} = \{P_1, \dots, P_l\}$. Since each element in $\mathcal{I} \cup \mathcal{J}$ is a critical clique of G , each such element is a subset of some cluster in \mathcal{C} by Proposition 2. Note that each cluster of \mathcal{C} could contain 0, 1, or more cliques of \mathcal{I} . We split the possibilities into three cases and provide bounds on $|F|$ for each case. First, if there exists a cluster of \mathcal{C} that includes at least two critical cliques from \mathcal{I} , then F contains h^2 inserted edges to connect two critical cliques from \mathcal{I} , and thus $|F| \geq h^2$. Assume instead that every cluster of \mathcal{C} includes at most one critical cluster from \mathcal{I} . Then, we have $l \geq k$. Suppose $l \geq k + 1$. Then, there are $l - k$ clusters in \mathcal{C} that do not contain any critical cliques from \mathcal{I} . Let $\mathcal{C}' \subseteq \mathcal{C}$ consist of these $l - k$ clusters and $U = \bigcup_{P \in \mathcal{C}'} P$. Then, for every $v \in U$, kh deleted edges are required in F to disconnect v from all vertices of $V(I)$, and for every $u \in V(J) \setminus U$, $(k - 1)h$ deleted edges are required in F to disconnect u from all vertices of $V(I) \setminus P$, where P is the clique of \mathcal{I} contained in the same cluster as u . Therefore,

$$\begin{aligned} |F| &\geq kh|U| + (k - 1)h|V(J) \setminus U| \\ &= h|U| + (k - 1)h|U| + (k - 1)h|V(J) \setminus U| \\ &= h|U| + (k - 1)ha \\ &\geq (k - 1)ah + h. \end{aligned}$$

Now assume that $l = k$. Then, every element of \mathcal{C} includes exactly one critical clique from \mathcal{I} . Consider each $i \in [k]$ and assume w.l.o.g. that $V(B_i) \subseteq P_i$. Let $W_i = P_i \setminus V(B_i)$ and let $\{\mathcal{J}_1, \dots, \mathcal{J}_k\}$ be a partition of \mathcal{J} such that, for each $i \in [k]$, the union of all elements of \mathcal{J}_i is W_i (such a partition exists because each element of \mathcal{J} is entirely contained in some W_i). Firstly, $(k - 1)h$ deleted edges are required in F to disconnect each $v \in W_i$ from all vertices of $V(I) \setminus V(B_i)$. Secondly, for each $i \in [k]$, $\frac{1}{2} \sum_{S, S' \in \mathcal{J}_i} |S||S'|$ inserted edges are required in F to connect all vertices of W_i . One can easily check that for each $i \in [k]$, F accounts for every edge with an endpoint in P_i and an endpoint outside, and accounts for all non-edges within P_i . Therefore, all the possible edges of F are counted. Thus,

$$\begin{aligned}
 |F| &= \sum_{i=1}^k \left(|W_i|(k-1)h + \frac{1}{2} \sum_{S, S' \in \mathcal{J}_i} |S||S'| \right) \\
 &= |V(J)|(k-1)h + \frac{1}{2} \sum_{i=1}^k \left(\left(\sum_{S \in \mathcal{J}_i} |S| \right)^2 - \sum_{S \in \mathcal{J}_i} |S|^2 \right) \\
 &= (k-1)ah + \frac{1}{2} \sum_{i=1}^k |W_i|^2 - \frac{1}{2} \sum_{S \in \mathcal{J}} |S|^2 \\
 &= (k-1)ah + \frac{1}{2} \sum_{i=1}^k |W_i|^2 - \frac{s}{2}.
 \end{aligned}$$

We can now compare $|F|$ from the lower bounds obtained in the previous two cases, as follows.

$$\begin{aligned}
 |F| &= (k-1)ah + \frac{1}{2} \sum_{i=1}^k |W_i|^2 - \frac{s}{2} \\
 &< (k-1)ah + \sum_{i=1}^k |W_i|^2 + \sum_{1 \leq i, j \leq k} |W_i||W_j| \\
 &= (k-1)ah + \left(\sum_{i=1}^k |W_i| \right)^2 \\
 &= (k-1)ah + a^2 \\
 &< (k-1)ah + h < h^2.
 \end{aligned}$$

The last line implies that having $|\mathcal{C}| = k$, with each element of \mathcal{C} a superset of exactly one element from \mathcal{I} , always achieves a lower cost than the other possibilities.

Next, consider the lower bound of $|F| \geq t$ and the conditions on equality. Using the same starting point,

$$\begin{aligned}
 |F| &= (k-1)ah + \frac{1}{2} \sum_{i=1}^k |W_i|^2 - \frac{s}{2} \\
 &= (k-1)ah + \frac{1}{2k} \left(\sum_{i=1}^k |W_i|^2 \right) \left(\sum_{i=1}^k 1^2 \right) - \frac{s}{2} \\
 &\geq (k-1)ah + \frac{1}{2k} \left(\sum_{i=1}^k |W_i| \right)^2 - \frac{s}{2} \tag{1} \\
 &= (k-1)ah + \frac{1}{2k} a^2 - \frac{s}{2} = t.
 \end{aligned}$$

In (1), we used the Cauchy-Schwarz inequality, and the two sides are equal if and only if $|W_1| = \dots = |W_k|$. In addition, $|W_1| = \dots = |W_k|$ if and only if $|P_1| = \dots = |P_k|$ (recall that $W_i = P_i \setminus V(B_i)$ and $|V(B_i)| = h$ for each $i \in [k]$). This proves every statement of the claim.

◁

Armed with the above claim, we immediately deduce the first statement of the theorem. We now show the second part, i.e., the n items of A can fit perfectly into k bins of capacity b if and only if we can make G a cluster graph with at most t edge modifications.

(\Leftarrow): Assume there is a clustering \mathcal{C} of G with cost at most t . By Claim 7, the size of any optimal cluster editing set for G is at least t . Thus, the cost of \mathcal{C} must be equal to t , and \mathcal{C} must be optimal. Claim 7 then implies that $\mathcal{C} = \{P_1, \dots, P_k\}$, such that $|P_1| = \dots = |P_k|$ and $V(B_i) \subseteq P_i$ for each $i \in [k]$ (w.l.o.g.). Moreover, each critical clique of \mathcal{J} is a subset of some element of \mathcal{C} by Proposition 2. So there is a partition $\mathcal{J}_1, \dots, \mathcal{J}_k$ of \mathcal{J} such that $P_i \setminus V(B_i) = \bigcup_{S \in \mathcal{J}_i} S$ for every $i \in [k]$. This means that \mathcal{J} can be partitioned into k groups such that the number of vertices in each group equals b .

(\Rightarrow): Assume the n items can be perfectly packed into k bins with capacity b . We construct a cluster editing set F of size at most t . Consider each $i \in [k]$. Let $S_i \subseteq A$ consist of the sizes of the items packed in the i -th bin, and define a corresponding cluster graph $B'_i = \bigcup_{a_j \in S_i} A_j$. We put in F a set of $(k-1)h$ deleted edges, by disconnecting each $v \in V(B'_i)$ from every vertex in $V(I) \setminus V(B'_i)$ in G . We then add to F the set of $\frac{1}{2} \sum_{V(A_j), V(A_r) \subseteq V(B'_i)} a_j a_r$ of inserted edges into F to make B'_i a complete graph, where $j, r \in [n]$. Applying the modifications in F results in a clustering $\mathcal{C} = \{P_1, \dots, P_k\}$ such that $P_i = V(B_i) \cup V(B'_i)$ for each $i \in [k]$. Moreover, the cardinality of the cluster editing set is

$$\begin{aligned}
|F| &= \sum_{i=1}^k \left((k-1)h|V(B'_i)| + \frac{1}{2} \sum_{V(A_j), V(A_r) \subseteq V(B'_i)} a_j a_r \right) \\
&= (k-1)h \sum_{i=1}^k |V(A_i)| + \frac{1}{2} \sum_{i=1}^k \left(\left(\sum_{V(A_j) \subseteq V(B'_i)} a_j \right)^2 - \sum_{V(A_j) \subseteq V(B'_i)} a_j^2 \right) \\
&= (k-1)h|V(\mathcal{J})| + \frac{1}{2} \sum_{i=1}^k \left(|V(B'_i)|^2 - \sum_{V(A_j) \subseteq V(B'_i)} a_j^2 \right) \\
&= (k-1)ah + \frac{1}{2} \sum_{i=1}^k b^2 - \frac{1}{2} \sum_{V(A_j) \subseteq V(\mathcal{J})} a_j^2 \\
&= (k-1)ah + \frac{1}{2}kb^2 - \frac{1}{2}s = t.
\end{aligned}$$

Thus, G has a clustering with cost at most t . ◀

Since the reduction given in Lemma 6 is a parameter-preserving Karp reduction from Unary Perfect Bin Packing with parameter k to p -Cluster Editing in cographs with parameter $p = k$, we can make the following series of deductions.

► **Theorem 8.** *The following hardness results hold:*

- *CLUSTER EDITING on cographs and p -CLUSTER EDITING on cographs are NP-complete.*
- *p -CLUSTER EDITING on cographs is $W[1]$ -hard parameterized by p .*
- *p -CLUSTER EDITING on cographs admits no $f(p)n^{o(p/\log p)}$ time algorithm for any function f unless ETH fails, where n is the number of vertices of the input graph.*

Since cographs have a constant clique-width [15], we have the following.

► **Corollary 9.** *p -CLUSTER EDITING admits no $f(p)n^{g(cw) \cdot o(p/\log p)}$ time algorithm for any functions f and g unless ETH fails.*

An $n^{O(p \cdot cw)}$ time algorithm

We propose a dynamic programming algorithm over the k -expression tree T of G as described in the preliminaries, where k is at most twice the clique-width of G . The idea is that, for each node u of T and every way of distributing the vertices of $V(G^u)$ among p clusters and k labels, we compute the minimum-cost clustering conditioned on such a distribution. The latter is described as a matrix of vertex counts per cluster per label, and the cost can be computed by combining the appropriate cost for matrices of the children of u .

The entries of all matrices discussed here are natural numbers between 0 and n . $M_{i,:}$ and $M_{:,j}$ represent the i -th row and j -th column of matrix M , respectively. We write $m_{i,j}$ for the entry of M in row i and column j . The sum of all entries in $M_{i,:}$ and $M_{:,j}$ are denoted by $\text{sum}\{M_{i,:}\}$ and $\text{sum}\{M_{:,j}\}$, respectively. We use $\{S_i\}_{i=1}^n$ to denote a sequence (S_1, \dots, S_n) .

Let M be a $k \times p$ matrix and G be a k -NL graph. An M -sequencing of G is a sequence $\{C_i\}_{i=1}^p$ of subsets of $V(G)$ such that

1. the non-empty subsets of this sequence form a partition \mathcal{C} of $V(G)$;
2. the number of vertices in C_j labeled i is equal to $m_{i,j}$ for every $i \in [k], j \in [p]$.

The partition \mathcal{C} obtained from $\{C_i\}_{i=1}^p$ is called an M -clustering, and the *cost* of the M -sequencing is the cost of the clustering \mathcal{C} of G , which is $\text{cost}_G(\mathcal{C})$.

Clearly, an M -sequencing of G exists if and only if the sum of all entries of M equals the number of vertices of G and the sum of all entries of the i -th row of M equals the number of vertices in G labeled i for every $i \in [k]$. The cost of M -sequencing is defined as ∞ if it does not exist. In addition, we say M is a *well-defined matrix* for G if an M -sequencing of G exists. An *optimal M -sequencing* of G is an M -sequencing of G of minimum cost.

► **Theorem 10.** *p -CLUSTER EDITING has an $O(n^{2p \cdot cw + 4})$ time algorithm if a k -expression is given, where $k = cw$ and n is the number of vertices of the input graph.*

Proof. Let T be a k -expression tree of k -NLC graph $G = (V, E, \text{lab})$. For every $u \in V(T)$, let $G^u = (V^u, E^u, \text{lab}^u)$ be the related graph of u , and let V_i^u denote the vertices of V^u labeled i for each $i \in [k]$. Let $M^u = (m_{i,j}^z)$ be a well-defined matrix of G^u . Assume u is a node of T corresponding to operation \times_S , and v, w are the two children of u . We define $h(M^v, M^w, S)$, for later use, which equals

$$\sum_{j \in [p]} \text{sum}\{M_{:,j}^v\} \cdot \text{sum}\{M_{:,j}^w\} + \sum_{(i,i') \in S} \text{sum}\{M_{i,:}^v\} \cdot \text{sum}\{M_{i',:}^w\} - 2 \sum_{(i,i') \in S} \sum_{j \in [p]} m_{i,j}^v m_{i',j}^w.$$

Next, we will first provide the dynamic programming algorithm for this problem, and then prove its correctness.

Let dynamic programming table $D(u, M^u)$ denote the cost of an optimal M^u -sequencing of G^u . For a leaf u of T and all well-defined M^u of u , $D(u, M^u) := 0$. For an internal vertex u of T with corresponding operation \times_S , and children v, w , we have

$$D(u, M^u) := \min_{M^u = M^v + M^w} \{D(v, M^v) + D(w, M^w) + h(M^v, M^w, S)\}.$$

For an internal node u of T with corresponding operation \circ_R , and child v , we have

$$D(u, M^u) := \min_{M^v} D(v, M^v),$$

where the minimization is taken over every well-defined matrix M^v for G^v that satisfies $M_{i',:}^u = \sum_{(i,i') \in R} M_{i,:}^v$ for every $i' \in [k]$.

Since every entry of the $k \times p$ matrix M^u is at most n , each $u \in V(T)$ has at most n^{pk} tables. For the \times_S vertices, one entry $D(u, M^u)$ can be computed in time $O(n^{pk+3})$ by enumerating all the possible $O(n^{pk})$ matrices M^v , from which M^w can be deduced from

$M^u - M^v$ in time $O(pk) = O(n^2)$, and computing $h(M^v, M^w, S)$ in time $O(n^3)$ (because S has at most $k^2 \leq n^2$ entries, and we treat p and k as upper-bounded by n for simplicity). Thus the set of all entries for u can be computed in $O(n^{2pk+3})$ time if all tables of its children are given. For the \circ_R vertices, each entry can be computed in time $O(n^{pk+3})$ similarly by enumerating the M^v matrices and checking the sum conditions, for a total time of $O(n^{2pk+3})$ as well. Since T has $O(n)$ nodes, we can compute all tables for all nodes, from leaves to root, of T in $O(n^{2pk+4})$ time, which is $O(n^{2p \cdot cw+4})$ if we assume $cw = k$. Let r be the root of T . The output of our algorithm is the minimum $D(r, M^r)$ such that M^r has no zero columns (note that if we require *at most* p clusters, we can simply tolerate zero columns).

We now prove that the recurrences are correct. The leaf case is easy to verify, so we assume inductively that the table is filled correctly for the children of an internal node u . Suppose that u corresponds to operation \times_S and consider the value we assign to an entry $D(u, M^u)$. Assume the two children of u are v, w . Suppose $\{C_i^u\}_{i=1}^p$ is an optimal M^u -sequencing for G^u with M^u -clustering \mathcal{C}^u . Let $\mathcal{C}^v = \{C \cap V^v : C \in \mathcal{C}^u\} \setminus \{\emptyset\}$ and $\mathcal{C}^w = \{C \cap V^w : C \in \mathcal{C}^u\} \setminus \{\emptyset\}$. Since G^u is a union of G^v and G^w by adding some edges between them controlled by S , we claim that the cost of the optimal M^u -sequencing $\text{cost}_{G^u}(\mathcal{C}^u)$ equals

$$\begin{aligned} \sum_{y \in \{v, w\}} \text{cost}_{G^y}(\mathcal{C}^y) + \sum_{C \in \mathcal{C}^u} |C \cap V^v| |C \cap V^w| + \sum_{(i, i') \in S} |V_i^v| |V_{i'}^w| \\ - 2 \sum_{(i, i') \in S} \sum_{C \in \mathcal{C}^u} |C \cap V_i^v| |C \cap V_{i'}^w|, \end{aligned}$$

justified as follows. First, any inserted or deleted edge of \mathcal{C}^u whose endpoints are both in V^v , or both in V^w , is counted exactly once, namely in the first summation $\sum_{y \in \{v, w\}} \text{cost}_{G^y}(\mathcal{C}^y)$. Then, consider the inserted/deleted edges with one endpoint in V^v and the other in V^w . Observe that a non-edge between V^v and V^w is counted once in the second summation if and only if it is an inserted edge. That non-edge is not counted in the third summation, because the latter only counts edges of G^u , and it is not subtracted in the last term for the same reason. Thus the expression counts inserted edges between V^v, V^w in \mathcal{C}^u exactly once, and no other such non-edge. Now, consider an edge e between V^v and V^w , which exists because of S . If e is a deleted edge, its endpoints are in different clusters and it is counted exactly once, in the third summation. If e is not deleted, its endpoints are in the same cluster and it is counted in both the second and third summation. On the other hand, that edge is subtracted twice in the last term, and so overall it is not counted. We deduce that the expression correctly represents the cost of \mathcal{C}^u in G^u .

For each $y \in \{v, w\}$, we further define $M^y = (m_{i,j}^y)$ such that $m_{i,j}^y = |C_j^y \cap V_i^y|$ for all i, j . Clearly, we have $M^v + M^w = M^u$. Now, we claim that $\{C_i^u \cap V^y\}_{i=1}^p$ is an *optimal* M^y -sequencing of G^y with M^y -clustering \mathcal{C}^y for every $y \in \{v, w\}$. Roughly speaking, this can be seen by observing that merging any M^v and M^w -clustering will result, in the cost expression given above, in the same values for the three last summations. Since the choice of M^y clusterings only affects the first summation, they should be chosen to minimize it. To see this in more detail, assume for contradiction that $\{D_i^v\}_{i=1}^p$ is an M^v -sequencing of G^v with M^v -clustering \mathcal{D}^v and $\{D_i^w\}_{i=1}^p$ is an M^w -sequencing of G^w with M^w -clustering \mathcal{D}^w such that $\sum_{y \in \{v, w\}} \text{cost}_{G^y}(\mathcal{D}^y) < \sum_{y \in \{v, w\}} \text{cost}_{G^y}(\mathcal{C}^y)$. Then, $\{D_i^v \cup D_i^w\}_{i=1}^p$ is a M^u -sequencing for G^u since $M^v + M^w = M^u$ and $V^v + V^w = V^u$. Furthermore, the cost of the M^u -sequencing $\{D_i^v \cup D_i^w\}_{i=1}^p$ is $\sum_{y \in \{v, w\}} \text{cost}_{G^y}(\mathcal{D}^y) + h(M^v, M^w, S)$, where $h(M^v, M^w, S)$ also equals

$$\sum_{C \in \mathcal{C}^u} |C \cap V^v| |C \cap V^w| + \sum_{(i, i') \in S} |V_i^v| |V_{i'}^w| - 2 \sum_{(i, i') \in S} \sum_{C \in \mathcal{C}^u} |C \cap V_i^v| |C \cap V_{i'}^w|$$

based on the definitions of M^v and M^w . As a result, $\sum_{y \in \{v,w\}} \text{cost}_{G^y}(\mathcal{D}^y) + h(M^v, M^w, S) < \sum_{y \in \{v,w\}} \text{cost}_{G^y}(\mathcal{C}^y) + h(M^v, M^w, S) = \text{cost}_{G^u}(\mathcal{C}^u)$, a contradiction. Therefore, \mathcal{C}^u is obtained by merging optimal M^v and M^w -clusterings. Since our value of $D(u, M^u)$ eventually considers $D(v, M^v) + D(w, M^w)$, which by induction contain the optimal costs, we have that $D(u, M^u)$ is at most the cost of \mathcal{C}^u . It is also easy to see that each possible entry considered in the minimization corresponds to an M^u -clustering of G^u , which cannot be better than \mathcal{C}^u , and so it follows that $D(u, M^u)$ is also at least the cost of \mathcal{C}^u . Thus our value of $D(u, M^u)$ is correct.

Consider an internal node u with corresponding operation \circ_R . Let v be the child of u . Suppose $\{C_i^u\}_{i=1}^p$ is an optimal M^u -sequencing for G^u with M^u -clustering \mathcal{C}^u . We define $M^v = (m_{i,j}^v)$ such that $m_{i,j}^v = |V_i^v \cap C_j^u|$. Then, $M_{i',j}^v = \sum_{(i,i') \in R} M_{i,j}^u$ for each $i \in [k]$. Now, we claim that $\{C_i^u\}_{i=1}^p$ is also an optimal M^v -sequencing of G^v with M^v -clustering \mathcal{C}^u . Otherwise, assume for contradiction that $\{C_i^v\}_{i=1}^p$ is an M^v -sequencing of G^v with M^v -clustering \mathcal{C}^v such that $\text{cost}_{G^v}(\mathcal{C}^v) < \text{cost}_{G^v}(\mathcal{C}^u)$. Then, $\{C_i^v\}_{i=1}^p$ is also a M^u -sequencing of G^u with M^u -clustering \mathcal{C}^v , because (1) the non-empty sets of $\{C_i^v\}_{i=1}^p$ define a partition of V^v , thus also define a partition of V^u since $V^v = V^u$. (2) In C_j^v of V^v , the number of vertices labeled i is $m_{i,j}^v$. In addition, G^u is obtained from G^v by relabeling vertices controlled by function R . Hence, in C_j^v of V^u , the number of vertices labeled i' is $\sum_{(i,i') \in R} m_{i,j}^v$, which equals $m_{i',j}^u$ for each $i' \in [k]$. Thus, the cost of the M^u -sequencing $\{C_i^v\}_{i=1}^p$ is $\text{cost}_{G^u}(\mathcal{C}^v) = \text{cost}_{G^v}(\mathcal{C}^v) < \text{cost}_{G^v}(\mathcal{C}^u) = \text{cost}_{G^u}(\mathcal{C}^u)$, a contradiction. As before, this shows that our value of $D(u, M^u)$ is both an upper and lower bound on the cost of \mathcal{C}^u , and is therefore correct. \blacktriangleleft

Recall that cographs have clique-width at most 2. Moreover, a $cw(2)$ -expression and then a 2-expression can easily be derived from a binary cotree, since 0-nodes are simply join operations that add no edges, and 1-nodes can be expressed by joining graphs with two different colors and adding every edge between them (in fact, dynamic programming over the cotree directly could be more efficient). We therefore have the following consequence, which we note is not conditioned on receiving a k -expression.

► **Corollary 11.** *p -CLUSTER EDITING on cographs admits an $O(n^{4p+4})$ time algorithm.*

4 Cluster Editing on Trivially Perfect Graphs

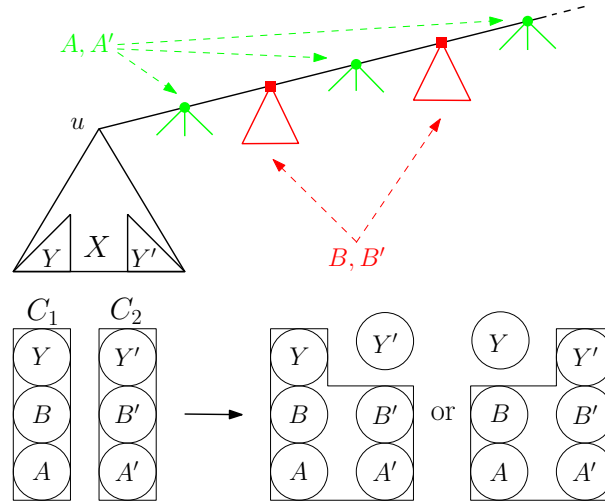
We now show that CLUSTER EDITING can be solved in cubic time on TPGs, first providing some intuitions. Consider a TPG G , and recall that it admits a cotree T in which every 1-node has at most one child that is not a leaf. Such a cotree can be found in linear time [14]. Let \mathcal{C} be an optimal clustering of G . Let $v \in V(T)$ and let $X \subseteq V(G)$ be the set of leaves that descend from v , which we call a *clade*. Suppose that there are two clusters $C_1, C_2 \in \mathcal{C}$ that intersect with X , but that also satisfy $C_1 \setminus X \neq \emptyset, C_2 \setminus X \neq \emptyset$ (we will say that X “grows” in C_1 and C_2). We can show that there is an alternate optimal clustering obtainable by moving $C_1 \cap X$ into a new cluster by itself, and merging $(C_1 \setminus X) \cup C_2$ into a single cluster². In this manner, X “grows” in one less cluster, and we can repeat this until it grows in at most one cluster. This property allows dynamic programming over the clades of the cotree, as we only need to memorize optimal solutions with respect to the size of the only cluster that can grow in the subsequent clades.

² Note that this is where the properties of TPGs are used: this works because if we consider the neighbors of X outside of X , they are all children of 1-nodes on the path from v to the root. They thus form a clique, which makes the merging $(C_1 \setminus X) \cup C_2$ advantageous as it saves the deletions of edges from that clique. Also, we may need to exchange the roles of C_1 and C_2 .

For two vertex-disjoint subsets of vertices X, Y , we denote $E_G(X, Y) = \{uv \in E(G) : u \in X, v \in Y\}$ and $e_G(X, Y) = |E_G(X, Y)|$. We further denote $\bar{e}_G(X, Y) = |X||Y| - e_G(X, Y)$, which is the number of non-edges between X and Y . We drop the subscript G from these notations if it is clear from the context. Two disjoint subsets $X, Y \subseteq V(G)$ are *neighbors* if $e(X, Y) = |X||Y|$, and they are *non-neighbors* if $\bar{e}(X, Y) = |X||Y|$. That is, every possible edge is present, and absent, respectively. Note that using this notation, for a given clustering $\mathcal{C} = \{C_1, \dots, C_l\}$, the set of inserted edges is $\bigcup_{C \in \mathcal{C}} E(\overline{G[\mathcal{C}]})$ and the set of deleted edges is $\bigcup_{1 \leq i < j \leq l} E(C_i, C_j)$.

Let G be a cograph with cotree T . For $v \in V(T)$, we denote by $L(v)$ the set of leaves that descend from v in T (note that $L(v) \subseteq V(G)$). We call $L(v)$ the *clade* of v . We say that $X \subseteq V(G)$ is a *clade* of T if X is a clade of some $v \in V(T)$. In this case, we say that X is *rooted at* v . The set of clades of T is defined as $\text{clades}(T) = \{L(v) : v \in V(T)\}$.

We first show a technical property of optimal solutions that will be useful. The lemma statement is illustrated in Figure 2.



■ **Figure 2** Illustration of Lemma 12. The tree shown is the cotree of G , with disc vertices being 1-nodes and square vertices 0-nodes. Here, $Y = X \cap C_1$ and $Y' = X \cap C_2$. If $|A| \geq |B|$ and $|A'| \geq |B'|$, then rearranging C_1 and C_2 in one of the two ways shown also gives an optimal clustering.

► **Lemma 12.** *Let G be a TPG with cotree T , let $u \in V(T)$, and let $X = L(u)$. Let \mathcal{C} be an optimal clustering of G and let $C_1, C_2 \in \mathcal{C}$ be distinct clusters that both intersect with X . Let U be the set of strict ancestors of u in T . Let A (resp. A') be the set of vertices of C_1 (resp. C_2) that are children of 1-nodes in U , and let $B = C_1 \setminus (X \cup A)$ (resp. $B' = C_2 \setminus (X \cup A')$).*

If $|A| \geq |B|$ and $|A'| \geq |B'|$, then at least one of the alternate clusterings $(\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cup A' \cup B', C_2 \cap X\}$ or $(\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cap X, C_2 \cup A \cup B\}$ has cost at most $\text{cost}_G(\mathcal{C})$.

Let G be a TPG with cotree T . Let $\mathcal{C} = \{C_1, \dots, C_l\}$ be a clustering of G . Let X be a clade of T . For $C_i \in \mathcal{C}$, we say that X *grows in* C_i if $C_i \cap X \neq \emptyset$ and $C_i \setminus X \neq \emptyset$. Note that this differs from the notion of overlapping, since $X \subset C_i$ is possible. We then say that X has a *single-growth in* \mathcal{C} if X grows in at most one element of \mathcal{C} . In other words, at most one cluster of \mathcal{C} containing elements of X also has elements outside of X , and the rest of X is split into clusters that are subsets of X . Note that X could grow in zero elements of \mathcal{C} . Furthermore, we will say that X has a *heritable single-growth in* \mathcal{C} if, for all clades Y of T

such that $Y \subseteq X$, Y has a single-growth in \mathcal{C} . For $v \in V(T)$, we may also say that v grows in C_i if $L(v)$ grows in C_i , or that v has a single-growth in \mathcal{C} if $L(v)$ does. For brevity, we may write SG for single-growth, and HSG for heritable single-growth.

► **Lemma 13.** *Suppose that G is a TPG with cotree T . Then there exists an optimal clustering \mathcal{C} of G such that every clade X of T has an SG in \mathcal{C} .*

Proof. Consider an optimal clustering $\mathcal{C} = \{C_1, \dots, C_l\}$ of G , chosen such that the number of clades of T that have an HSG in \mathcal{C} is maximum, among all optimal clusterings³. If every clade of T has the HSG property, then we are done (since HSG implies SG), so we assume that not every clade has the HSG property. Clearly, in \mathcal{C} , every leaf of T has an HSG, the root of T does not have an HSG, and there exists at least one internal node of T that does not have an SG. Choose $u \in V(T)$ with the minimum number of descendants such that u does not have an SG in \mathcal{C} . Then, every descendant of u has an SG, thus, also has an HSG in \mathcal{C} . Notice that u cannot be the root of T , because the root trivially has an SG in \mathcal{C} .

Denote $X = L(u)$. Since u does not have the SG property, X grows in at least two clusters of \mathcal{C} , say C_1 and C_2 . Thus $X \cap C_1, C_1 \setminus X, X \cap C_2, C_2 \setminus X$ are all non-empty. Denote $Y = X \cap C_1, Y' = X \cap C_2$. We show that we can transform \mathcal{C} into another optimal clustering \mathcal{C}' in which one of Y or Y' is a cluster by itself, and such that the number of clades that have an HSG in \mathcal{C}' is no less than in \mathcal{C} .

As in Lemma 12, let U be the set of strict ancestors of u in T . Let A and A' be the sets of vertices of C_1 and C_2 , respectively, that are children of 1-nodes in U . Then let $B = C_1 \setminus (X \cup A)$ and $B' = C_2 \setminus (X \cup A')$. Note that because $C_1 \setminus X \neq \emptyset$ and X does not intersect A or B , we have that $A \cup B$ is non-empty. Likewise, $A' \cup B'$ is non-empty.

We argue that $|A| \geq |B|$. Suppose instead that $|A| < |B|$. Consider the clustering $\mathcal{C}' = (\mathcal{C} \setminus \{C_1\}) \cup \{Y, A \cup B\}$. Then \mathcal{C}' has $|A||Y|$ edge deletions that are not in \mathcal{C} (and no other edge modification is in \mathcal{C}' but not in \mathcal{C} , since Y and B share no edge, as the lowest common ancestor of any vertex in X and a vertex of B is a 0-node in the cotree). On the other hand, \mathcal{C} has $|B||Y| > |A||Y|$ edge additions that are not in \mathcal{C}' . Hence, the cost of \mathcal{C}' is strictly lower than that of \mathcal{C} , a contradiction. By the same argument, we get that $|A'| \geq |B'|$.

We see that C_1 and C_2 satisfy all the requirements of Lemma 12, and so we may get an alternate optimal clustering \mathcal{C}' or \mathcal{C}'' , where \mathcal{C}' is obtained from \mathcal{C} by replacing C_1, C_2 by $Y, C_2 \cup A \cup B$, and \mathcal{C}'' is obtained by replacing C_1, C_2 by $Y', C_1 \cup A' \cup B'$. Notice that X grows in fewer clusters in \mathcal{C}' than in \mathcal{C} , and the same is true for \mathcal{C}'' . Before proceeding, we need to argue that T has as many clades with an HSG in \mathcal{C}' , and in \mathcal{C}'' than in \mathcal{C} .

So, let $w \in V(T)$ be such that w has an HSG in \mathcal{C} . Observe that w cannot be in $U \cup \{u\}$, since these have u as a descendant and u does not have an SG in \mathcal{C} . Therefore, w must either be: (1) a leaf child of a 1-node in U ; (2) a descendant of u ; or (3) a node whose first ancestor in U is a 0-node. Let v be a descendant of w , with $v = w$ possible. By the definition of HSG, v has an SG in \mathcal{C} . In all cases, we argue that v still has an SG in \mathcal{C}' and \mathcal{C}'' .

1. If w is the child of a 1-node of U , then $w = v$ is a leaf and it trivially has an SG in \mathcal{C}' and \mathcal{C}'' .
2. Suppose that w , and thus v , is a descendant of u . If $L(v)$ does not intersect with C_1 nor C_2 , then the clusters of \mathcal{C} that intersect with $L(v)$ are unaltered in \mathcal{C}' and \mathcal{C}'' , and thus v also has an SG in \mathcal{C}' , and in \mathcal{C}'' . Thus suppose that $L(v)$ intersects with $C_1 \cup C_2$.

³ We could attempt to choose \mathcal{C} to maximize the clades with an SG instead, but we will modify \mathcal{C} later on, and keeping track of changes in HSG clades is much easier than SGs.

In that case, since v descends from u , $L(v) \subseteq X$ and it must thus intersect with $Y \cup Y'$. If $L(v) \cap Y \neq \emptyset$, then v grows in C_1 because $A \cup B \neq \emptyset$. Likewise, if $L(v) \cap Y' \neq \emptyset$, then v grows in C_2 . It follows that $L(v)$ intersects exactly one of Y or Y' , and grows in exactly one of C_1 or C_2 . If $L(v)$ intersects Y , then in \mathcal{C}' , $L(v)$ may grow in the cluster Y , but it does not grow in $C_2 \cup A \cup B$ since it does intersect with it, and does not grow in other clusters of \mathcal{C}' since these were unaltered. Thus $L(v)$ has an SG in \mathcal{C}' . In \mathcal{C}'' , $L(v)$ grows in $C_1 \cup A' \cup B'$ but no other cluster for the same reason. Thus $L(v)$ has an SG in \mathcal{C}'' as well. If $L(v)$ intersects Y' , the same arguments can be used to deduce that $L(v)$ has an SG in \mathcal{C}' and \mathcal{C}'' .

3. Finally, suppose that w is such that its first ancestor in U is a 0-node. Again we may assume that $L(v)$ intersects $C_1 \cup C_2$. This implies that $L(v)$ intersects with $B \cup B'$, and does not intersect with $Y \cup Y' \cup A \cup A'$. If $L(v)$ intersects B , then it grows in C_1 since $|A| \geq |B|$. Then in \mathcal{C}' , v grows in $C_2 \cup A \cup B$, but not Y nor any other unaltered cluster. In \mathcal{C}'' it grows in $C_1 \cup A' \cup B'$, but not Y' nor any other unaltered cluster. If $L(v)$ intersects B' , the same argument applies. Either way, $L(v)$ has an SG in \mathcal{C}' and \mathcal{C}'' .

Since any descendant of w has an SG in either \mathcal{C}' and \mathcal{C}'' , we deduce that w has an HSG in both alternate clusterings.

We have thus found an optimal clustering $\mathcal{C}^* \in \{\mathcal{C}', \mathcal{C}''\}$ such that every $w \in V(T)$ that has an HSG in \mathcal{C} also has an HSG in \mathcal{C}^* . Moreover, since \mathcal{C}^* “extracts” either Y or Y' from its cluster, X grows in one less cluster of \mathcal{C}^* . If X grows in only one such cluster, then u has an SG in \mathcal{C}^* and therefore also an HSG in \mathcal{C}^* , by the choice of u . In this case, T has more clades that have an HSG in \mathcal{C}^* than with \mathcal{C} , which contradicts our choice of \mathcal{C} . If X still grows in at least two clusters of \mathcal{C}^* , we may repeat the above modification as many times as needed until X grows in a single cluster, yielding the same contradiction. We deduce that every node of T has an HSG, and therefore an SG in \mathcal{C} . ◀

Our goal is to use the above to perform dynamic programming over the cotree. For a node v of this cotree, we will store the value of a solution for the subgraph induced by $L(v)$, and will need to determine which cluster of such a partial solution should grow. As it turns out, we should choose the largest cluster to grow.

► **Lemma 14.** *Let G be a TPG with cotree T . Let \mathcal{C} be an optimal clustering of G such that every clade of T has an SG in \mathcal{C} and, among all such possible optimal clusterings, such that $|\mathcal{C}|$ is maximum. Then for every clade X of T , one of the following holds:*

- X does not grow in any $C_i \in \mathcal{C}$; or
- X grows in one $C_i \in \mathcal{C}$, and $|X \cap C_i| = \max_{C_j \in \mathcal{C}} |X \cap C_j|$.

Proof. Let X be a clade of T and suppose that X grows in some $C_i \in \mathcal{C}$. Note that vertices of X share the same neighborhood outside of X . Thus C_i can be partitioned into $\{X \cap C_i, A, B\}$ such that X, A are neighbors and X, B are non-neighbors. Note that $|A| \geq |B|$ as otherwise, we could obtain an alternate clustering \mathcal{C}' by replacing C_i by $\{X \cap C_i, A \cup B\}$ and save a cost of $|X \cap C_i|(|B| - |A|) > 0$.

We also argue that $|A| > |B|$. This is because if $|A| = |B|$, the same clustering \mathcal{C}' has $\text{cost}_G(\mathcal{C}') = \text{cost}_G(\mathcal{C})$, but has one more cluster. We also argue that every clade of T has an SG in \mathcal{C}' , contradicting our choice of \mathcal{C} . Let $Y = X \cap C_i$ and consider some $v \in V(T)$. By assumption v has an SG in \mathcal{C} . If $C_i \cap L(v) = \emptyset$, then the clusters of \mathcal{C} that intersect with $L(v)$ are unaltered in \mathcal{C}' and v also has an SG in \mathcal{C}' . Suppose that $C_i \subseteq L(v)$. Apart from $C_i = Y \cup A \cup B$, the clusters of \mathcal{C} that intersect with $L(v)$ are unaltered in \mathcal{C}' . Moreover, $L(v)$ does not grow in $Y \cup A \cup B$, and neither does it grow in Y or $A \cup B$. Thus v also has

an SG \mathcal{C}' . The remaining case is when $L(v)$ grows in C_i (but no other cluster of \mathcal{C}). Let $u \in V(T)$ be such that $L(u) = X$. If $v = u$, then $L(v)$ does not grow in any cluster of \mathcal{C}' . If v is a strict descendant of u , then $L(v)$ can only grow in Y of \mathcal{C}' . If v is a strict ancestor of u , then $Y \subseteq L(v)$ and $L(v)$ can only grow in $A \cup B$ of \mathcal{C}' . If v is in the rest of $V(T)$, then $Y \cap L(v) = \emptyset$ and $L(v)$ grows only in $A \cup B$ of \mathcal{C}' . As a result, $L(v)$ has an SG in \mathcal{C}' . Since this holds for any v , every node has an SG in \mathcal{C}' , which is a contradiction since $|\mathcal{C}| < |\mathcal{C}'|$. Therefore, $|A| > |B|$.

Now suppose that there is some $C_j \neq C_i$ such that $|X \cap C_j| > |X \cap C_i|$. Because X already grows in C_i , the SG property implies that $C_j \subseteq X$. Consider the alternate clustering \mathcal{C}^* obtained from \mathcal{C} by replacing C_i, C_j by $C_j \cup A \cup B, X \cap C_i$. The number of modifications in \mathcal{C}^* but not in \mathcal{C} is $|X \cap C_i||A| + |C_j||B|$, but the number of modifications in \mathcal{C} not in \mathcal{C}^* is $|X \cap C_i||B| + |C_j||A|$. The difference between the latter and the former is $(|C_j| - |X \cap C_i|)(|A| - |B|)$. Since $|A| > |B|$ and $|C_j| = |X \cap C_j| > |X \cap C_i|$, this is greater than 0, and thus \mathcal{C}^* is a clustering of cost lower than \mathcal{C} , a contradiction. \blacktriangleleft

Our algorithm will search for an optimal clustering that satisfies all the requirements of Lemma 14. That is, for a TPG G with cotree T , a clustering \mathcal{C} is *well-behaved* if it is optimal, every clade of T has an SG in \mathcal{C} , and among all such possible clusterings $|\mathcal{C}|$ is maximum. As we know that such a \mathcal{C} exists, we will search for one using dynamic programming.

In the remainder, for an arbitrary clustering \mathcal{C} of G and $X \subseteq V(G)$, define $\mathcal{C}|_X = \{C_i \cap X \mid C_i \in \mathcal{C}\} \setminus \{\emptyset\}$, i.e., the restriction of \mathcal{C} to X . Note that $\mathcal{C}|_X$ is a clustering of $G[X]$, and we refer to $\text{cost}_{G[X]}(\mathcal{C}|_X)$ as the *cost of \mathcal{C} in $G[X]$* . Although $\mathcal{C}|_X$ is not necessarily an optimal clustering of $G[X]$, we can deduce from the above that it has minimum cost among those clusterings with the same largest cluster.

► **Corollary 15.** *Let G be a TPG with cotree T , and let \mathcal{C} be a well-behaved clustering of G . Let $u \in V(T)$ with clade $X = L(u)$, and let $k_u^* = \max_{C_j \in \mathcal{C}} |X \cap C_j|$ denote the size of the largest intersection of \mathcal{C} with X .*

Then $\mathcal{C}|_X$ is a clustering of $G[X]$ such that $\text{cost}_{G[X]}(\mathcal{C}|_X)$ is minimum, among all clusterings of $G[X]$ whose largest cluster has size k_u^ .*

We define a 2-dimensional dynamic programming table D indexed by $V(T) \times [n]$, with the intent that $D[u, k]$ has the cost of an optimal clustering of $G[L(u)]$ in which the largest cluster has size k . Notice that this intent is mostly for intuition purposes, since we will not be able to guarantee that $D[u, k]$ stores the correct value for each u, k . Indeed, if we require a clustering of $G[L(u)]$ with largest cluster size k , such a clustering may not be optimal and the properties of the above lemmas may not hold. However, we will argue that when k is the size of a largest cluster in an optimal clustering, then the entries are correct, as we prove that they combine information from optimal clusterings at the children of u which may also be assumed to be correct.

We assume that the cotree T of G is binary (note that such a cotree always exists and, since the previous lemmas make no assumption on the structure of the cotree, this is without loss of generality). If u is a leaf, put $D[u, 1] = 0$ and $D[u, k] = \infty$ for every $k \neq 1$. For internal node u , let u_1, u_2 be the two children of u in T . We put

$$D[u, k] = \min \begin{cases} D[u_1, k] + \min_{j \in [k]} D[u_2, j] + \mathbb{I}_u \cdot |L(u_1)||L(u_2)| \\ D[u_2, k] + \min_{j \in [k]} D[u_1, j] + \mathbb{I}_u \cdot |L(u_1)||L(u_2)| \\ \min_{j \in [k-1]} (D[u_1, j] + D[u_2, k-j] + \alpha(u)), \end{cases}$$

where $\mathbb{I}_u = 0$ if u is a 0-node and $\mathbb{I}_u = 1$ if it is a 1-node, and

$$\alpha(u) = \min \begin{cases} |L(u_1)||L(u_2)| - j(k-j) & \text{if } u \text{ is a 1-node} \\ j(k-j) & \text{otherwise.} \end{cases}$$

The recurrence for $D[u, k]$ mainly says that there are three ways to obtain a solution with a largest cluster of size k : either that cluster is taken directly from the solution at u_1 , from the solution at u_2 , or we take a cluster of size j from u_1 and size $k-j$ from u_2 , and merge them together.

► **Lemma 16.** *Let \mathcal{C} be a well-behaved clustering of G . Let $u \in V(T)$, and denote by k_u^* the size of the largest cluster of $\mathcal{C}|_{L(u)}$. Then both of the following hold:*

- for each k such that $D[u, k] \neq \infty$, there exists a clustering of $G[L(u)]$ of cost at most $D[u, k]$ whose largest cluster has size k ;
- $D[u, k_u^*]$ is equal to $\text{cost}_{G[L(u)]}(\mathcal{C}|_{L(u)})$, the cost of \mathcal{C} restricted to $G[L(u)]$.

Proof. We prove the statement by induction on the cotree T . For a leaf u , it is clear that both statements hold with $D[u, 1] = 0$. Let u be an internal node of T and let u_1, u_2 be its children. Denote $X = L(u)$, $X_1 = L(u_1)$, $X_2 = L(u_2)$.

We focus on the first part and assume that $D[u, k] \neq \infty$. The value of $D[u, k]$ is the minimum among three cases. If $D[u, k] = D[u_1, k] + \min_{j \in [k]} D[u_2, j] + \mathbb{I}_u |L(u_1)||L(u_2)|$, then because $D[u, k] \neq \infty$, $D[u_1, k] \neq \infty$ and $D[u_2, j] \neq \infty$ for the chosen j in the minimum expression. We can take the disjoint union of a clustering of $G[L(u_1)]$ whose largest cluster has size k (one exists of cost at most $D[u_1, k]$ by induction), and a clustering of $G[L(u_2)]$ with largest cluster of size $j \leq k$ (one exists of cost at most $D[u_2, j]$ by induction). If u is a 1-node, all edges between $L(u_1)$ and $L(u_2)$ are present and $\mathbb{I}_u |X_1||X_2|$ must be added to the cost (whereas if u is a 0-node, no additional cost is required). This confirms that, if the first case of the recurrence is the minimum, there exists a clustering with cost at most $D[u, k]$ whose largest cluster has size k . The same argument applies to the second case of the recurrence.

So assume that $D[u, k] = \min_{j \in [k-1]} (D[u_1, j] + D[u_2, k-j] + \alpha(u))$. This case corresponds to taking, by induction, a clustering of $G[X_1]$ and of $G[X_2]$ with the largest cluster of size j and $k-j$, respectively, and merging their largest cluster into one cluster of size k (leaving the other clusters intact). If u is a 1-node, the number of deleted edges between the resulting clusters is $|L(u_1)||L(u_2)| - j(k-j)$, and if u is a 0-node we must pay $j(k-j)$ edge insertions. This shows the first part of the statement.

For the second part, suppose that $k = k_u^*$. Here, \mathcal{C} is the optimal clustering stated in the lemma. As we just argued, there is a clustering of $G[X]$ of cost at most $D[u, k_u^*]$ with the largest cluster size k_u^* . By Corollary 15, $\mathcal{C}|_X$ has minimum cost among such clusterings, and so $D[u, k_u^*]$ is at least $\text{cost}_{G[X]}(\mathcal{C}|_X)$. We argue that the latter is also an upper bound on $D[u, k_u^*]$. Let $C_1 \in \mathcal{C}|_{X_1}$, and note that if $C_1 \notin \mathcal{C}|_X$, then C_1 was “merged” with some cluster of $\mathcal{C}|_{X_2}$ to obtain $\mathcal{C}|_X$. In this case, u_1 grows in some cluster of $\mathcal{C}|_X$, and therefore also grows in some cluster of \mathcal{C} . By the SG property, this means that there is at most one C_1 of $\mathcal{C}|_{X_1}$ such that $C_1 \notin \mathcal{C}|_X$. For the same reason, there is at most one C_2 of $\mathcal{C}|_{X_2}$ that is not in $\mathcal{C}|_X$. This in turn implies that either $\mathcal{C}|_X = \mathcal{C}|_{X_1} \cup \mathcal{C}|_{X_2}$, or that there is a unique $C_1 \in \mathcal{C}|_{X_1}$ and a unique $C_2 \in \mathcal{C}|_{X_2}$ such that $C_1 \cup C_2$ is in $\mathcal{C}|_X$. We now consider both cases.

- If $\mathcal{C}|_X = \mathcal{C}|_{X_1} \cup \mathcal{C}|_{X_2}$, since $\mathcal{C}|_X$ has its largest cluster of size $k = k_u^*$, one of $\mathcal{C}|_{X_1}$ or $\mathcal{C}|_{X_2}$ must have a largest cluster of size k , and the other a largest cluster of some size $j \in [k]$. Using induction on the second part of the lemma, the corresponding entries

$D[u_i, k]$ and $D[u_{i'}, j]$ for $\{i, i'\} = \{1, 2\}$ store the costs of $\mathcal{C}|_{X_1}$ and $\mathcal{C}|_{X_2}$, and since all the possibilities are considered by the $D[u, k_u^*]$ recurrence, it is clear that $D[u, k_u^*]$ is at most $\text{cost}_{G[X]}(\mathcal{C}|_X)$.

- If $C_1 \cup C_2 \in \mathcal{C}|_X$, we have $\mathcal{C}|_X = ((\mathcal{C}|_{X_1} \cup \mathcal{C}|_{X_2}) \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$. Observe that X_1 grows in $C_1 \cup C_2$. This means that X_1 also grows in the cluster of $C' \in \mathcal{C}$ that contains $C_1 \cup C_2$. By Lemma 14, $|X_1 \cap C'| \geq |X_1 \cap C''|$ for each $C'' \in \mathcal{C}$. Since C' contains C_1 , we get that $|X_1 \cap C'| = |C_1|$, and since $\{X \cap C'' : C'' \in \mathcal{C}\}$ contains all the clusters of $\mathcal{C}|_{X_1}$, it must be that C_1 is the largest cluster of $\mathcal{C}|_{X_1}$. By the same argument, C_2 is the largest cluster of $\mathcal{C}|_{X_2}$. Let $j = |C_1|$. We have that $\mathcal{C}|_X$ has the largest cluster size k , and is obtained by taking a clustering of $G[L(u_1)]$ with the largest cluster of size j , and a clustering of $G[L(u_2)]$ with largest cluster of size $k - j$, and merging these two clusters. If u is a 1-node, the cost of this is the cost of \mathcal{C} in $G[L(u_1)]$, which is $D[u_1, j]$ by induction, plus the cost of \mathcal{C} in $G[L(u_2)]$, which is $D[u_2, k - j]$ by induction, plus the cost for all the edges between $L(u_1)$ and $L(u_2)$, excluding those between C_1 and C_2 . If u is a 0-node, the cost is the same, except that we pay $j(k - j)$ for the non-edges between C_1 and C_2 . Either way, this case is considered by our recurrence, and we get $D[u, k] \leq \text{cost}_{G[X]}(\mathcal{C}|_X)$. Having shown both the lower and upper bounds, we get that $D[u, k_u^*] = \text{cost}_{G[X]}(\mathcal{C}|_X)$. ◀

► **Theorem 17.** *The CLUSTER EDITING problem can be solved in time $O(n^3)$ on trivially perfect graphs.*

Open problems. We observe that the structural properties shown on TPGs only work if the number of desired clusters is unrestricted. The complexity of p -CLUSTER EDITING on TPGs is open (note that if p is constant, our $n^{O(p)}$ time algorithm on cographs provides a polynomial-time algorithm). Regarding our clique-width (or rather NLC-width), it might be possible to improve the complexity, for example by achieving $n^{O(p+cw)}$ instead of $n^{O(p \cdot cw)}$.

We proved the problem in P on $\{P_4, C_4\}$ -free graphs, but we do not know the complexity on $\{P_4, 2K_2\}$ -graphs. The complement of such graphs are $\{P_4, C_4\}$ -free and may be in P as well, but it is unclear whether the editing problem on the complement can be solved using our techniques. More generally, it would be ideal to aim for a dichotomy theorem for forbidden induced subgraphs, that is, to characterize the forbidden induced subgraphs that make CLUSTER EDITING in P, and the ones that make it NP-hard.

References

- 1 Faisal N Abu-Khzam, Joseph R Barr, Amin Fakhredine, and Peter Shaw. A greedy heuristic for cluster editing with vertex splitting. In *2021 4th International conference on artificial intelligence for industries (AI4I)*, pages 38–41. IEEE, 2021. doi:10.1109/AI4I51902.2021.00017.
- 2 Emmanuel Arrighi, Matthias Bentert, Pål Grønås Drange, Blair D Sullivan, and Petra Wolf. Cluster editing with overlapping communities. In *18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.
- 3 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56:89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.
- 4 Hila Becker. A survey of correlation clustering. *Advanced Topics in Computational Learning Theory*, pages 1–10, 2005.
- 5 André Berger, Alexander Grigoriev, and Andrej Winokurow. A PTAS for the cluster editing problem on planar graphs. In *International Workshop on Approximation and Online Algorithms*, pages 27–39. Springer, 2016. doi:10.1007/978-3-319-51741-4_3.

- 6 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *J. Discrete Algorithms*, 16:79–89, 2012. doi:10.1016/J.JDA.2012.04.005.
- 7 Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. Going weighted: Parameterized algorithms for cluster editing. *Theor. Comput. Sci.*, 410(52):5467–5480, 2009. doi:10.1016/J.TCS.2009.05.006.
- 8 Sebastian Böcker and Peter Damaschke. Even faster parameterized cluster deletion and cluster editing. *Inf. Process. Lett.*, 111(14):717–721, 2011. doi:10.1016/J.IPL.2011.05.003.
- 9 Flavia Bonomo, Guillermo Duran, and Mario Valencia-Pabon. Complexity of the cluster deletion problem on subclasses of chordal graphs. *Theoretical Computer Science*, 600:59–69, 2015. doi:10.1016/J.TCS.2015.07.001.
- 10 Ulrik Brandes, Michael Hamann, Ben Strasser, and Dorothea Wagner. Fast quasi-threshold editing. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 251–262. Springer, 2015. doi:10.1007/978-3-662-48350-3_22.
- 11 Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012. doi:10.1007/S00453-011-9595-1.
- 12 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. In *FOCS 2003, 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 524–533. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238225.
- 13 Jianer Chen and Jie Meng. A $2k$ kernel for the cluster editing problem. *J. Comput. Syst. Sci.*, 78(1):211–220, 2012. doi:10.1016/J.JCSS.2011.04.001.
- 14 Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926–934, 1985. doi:10.1137/0214065.
- 15 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.
- 16 Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.
- 17 Wolfgang Espelage, Frank Gurski, and Egon Wanke. Deciding clique-width for graphs of bounded tree-width. *Journal of Graph Algorithms and Applications*, 7(2):141–180, 2003. doi:10.7155/JGAA.00065.
- 18 Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022. doi:10.1016/J.ENGAPPAI.2022.104743.
- 19 Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *J. Comput. Syst. Sci.*, 80(7):1430–1447, 2014. doi:10.1016/J.JCSS.2014.04.015.
- 20 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers 8*, pages 163–176. Springer, 2013.
- 21 Yong Gao, Donovan R. Hare, and James Nastos. The cluster deletion problem for cographs. *Discret. Math.*, 313(23):2763–2771, 2013. doi:10.1016/J.DISC.2013.08.017.
- 22 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 23 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory Comput.*, 2(13):249–266, 2006. doi:10.4086/TOC.2006.V002A013.
- 24 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In Rossella Petreschi, Giuseppe Persiano, and Riccardo Silvestri, editors, *CIAC 2003, Rome, Italy, May 28-30, 2003, Proceedings*, volume 2653 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2003. doi:10.1007/3-540-44849-7_17.

- 25 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica*, 39(4):321–347, 2004. doi:10.1007/S00453-004-1090-5.
- 26 Jiong Guo. A more effective linear kernelization for cluster editing. *Theor. Comput. Sci.*, 410(8-10):718–726, 2009. doi:10.1016/J.TCS.2008.10.021.
- 27 Frank Gurski, Egon Wanke, and Eda Yilmaz. Directed NLC-width. *Theor. Comput. Sci.*, 616:1–17, 2016. doi:10.1016/J.TCS.2015.11.003.
- 28 P Heggenes and D Kratsch. Linear-time certifying algorithms for recognizing trivially perfect graphs. *Reports In Informatics ISSN*, pages 0333–3590, 2006.
- 29 Marc Hellmuth, Maribel Hernandez-Rosales, Katharina T Huber, Vincent Moulton, Peter F Stadler, and Nicolas Wieseke. Orthology relations, symbolic ultrametrics, and cographs. *Journal of mathematical biology*, 66:399–420, 2013.
- 30 Marc Hellmuth, Nicolas Wieseke, Marcus Lechner, Hans-Peter Lenhof, Martin Middendorf, and Peter F Stadler. Phylogenomics with paralogs. *Proceedings of the National Academy of Sciences*, 112(7):2058–2063, 2015.
- 31 Giuseppe F Italiano, Athanasios L Konstantinidis, and Charis Papadopoulos. Structural parameterization of cluster deletion. In *International Conference and Workshops on Algorithms and Computation*, pages 371–383. Springer, 2023. doi:10.1007/978-3-031-27051-2_31.
- 32 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/J.JCSS.2012.04.004.
- 33 Songwei Jia, Lin Gao, Yong Gao, James Nastos, Yijie Wang, Xindong Zhang, and Haiyang Wang. Defining and identifying cograph communities in complex networks. *New Journal of Physics*, 17(1):013044, 2015.
- 34 Ojvind Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition-relationships and results for random graphs. *Congressus Numerantium*, pages 39–60, 1998.
- 35 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012. doi:10.1016/J.DAM.2012.05.019.
- 36 Athanasios L Konstantinidis and Charis Papadopoulos. Cluster deletion on interval graphs and split related graphs. *Algorithmica*, 83(7):2018–2046, 2021. doi:10.1007/S00453-021-00817-8.
- 37 Mirko Krivánek and Jaroslav Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986. doi:10.1007/BF00289116.
- 38 Manuel Lafond and Nadia El-Mabrouk. Orthology and paralogy constraints: satisfiability and consistency. *BMC genomics*, 15:1–10, 2014.
- 39 Manuel Lafond and Weidong Luo. Parameterized complexity of domination problems using restricted modular partitions. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *MFCS 2023*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.61.
- 40 Manuel Lafond, Mona Meghdari Miardan, and David Sankoff. Accurate prediction of orthologs in the presence of divergence after duplication. *Bioinformatics*, 34(13):i366–i375, 2018. doi:10.1093/BIOINFORMATICS/BTY242.
- 41 Bassel Manna. Cluster editing problem for points on the real line: A polynomial time algorithm. *Information processing letters*, 110(21):961–965, 2010. doi:10.1016/J.IPL.2010.08.002.
- 42 James Nastos and Yong Gao. Familial groups in social networks. *Soc. Networks*, 35(3):439–450, 2013. doi:10.1016/J.SOCNET.2013.05.001.
- 43 Sebastian Ochs. *Cluster Deletion on Unit Disk Graphs*. Master’s thesis, Philipps-Universität Marburg, 2023.
- 44 Alitzel López Sánchez and Manuel Lafond. Colorful orthology clustering in bounded-degree similarity graphs. *Journal of Bioinformatics and Computational Biology*, 19(06):2140010, 2021.
- 45 Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discret. Appl. Math.*, 144(1-2):173–182, 2004. doi:10.1016/J.DAM.2004.01.007.

- 46 Nate Veldt, David F Gleich, and Anthony Wirth. A correlation clustering framework for community detection. In *Proceedings of the 2018 World Wide Web Conference*, pages 439–448, 2018. doi:10.1145/3178876.3186110.
- 47 Charles T Zahn, Jr. Approximating symmetric relations by equivalence relations. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):840–847, 1964.

On Average Baby PIH and Its Applications

Yuwei Liu  

Shanghai Jiao Tong University, China

Yijia Chen  


Shanghai Jiao Tong University, China

Shuangle Li  

Nanjing University, China

Bingkai Lin  

Nanjing University, China

Xin Zheng  

Nanjing University, China

Abstract

The *Parameterized Inapproximability Hypothesis* (PIH) asserts that no FPT algorithm can decide whether a given 2CSP instance parameterized by the number of variables is satisfiable, or at most a constant fraction of its constraints can be satisfied simultaneously. In a recent breakthrough, Guruswami, Lin, Ren, Sun, and Wu (STOC 2024) proved the PIH under the Exponential Time Hypothesis (ETH). However, it remains a major open problem whether the PIH can be established assuming only $W[1] \neq \text{FPT}$. Towards this goal, Guruswami, Ren, and Sandeep (CCC 2024) showed a weaker version of the PIH called the *Baby PIH* under $W[1] \neq \text{FPT}$. In addition, they proposed one more intermediate assumption known as the *Average Baby PIH*, which might lead to further progress on the PIH. As the main contribution of this paper, we prove that the Average Baby PIH holds assuming $W[1] \neq \text{FPT}$.

Given a 2CSP instance where the number of its variables is the parameter, the Average Baby PIH states that no FPT algorithm can decide whether (i) it is satisfiable or (ii) any *multi-assignment* that satisfies all constraints must assign each variable more than r values on *average* for any fixed constant $r > 1$. So there is a *gap* between (i) and (ii) on the average number of values that are assigned to a variable, i.e., 1 vs. r . If this gap occurs in *each* variable instead of on average, we get the original Baby PIH. So central to our paper is an FPT self-reduction for 2CSP instances that turns the above gap for each variable into a gap on average. By the known $W[1]$ -hardness for the Baby PIH, this proves that the Average Baby PIH holds under $W[1] \neq \text{FPT}$.

As applications, we obtain (i) for the first time, the $W[1]$ -hardness of constant approximating k -EXACTCOVER, and (ii) a tight relationship between running time lower bounds in the Average Baby PIH and approximating the parameterized Nearest Codeword Problem (k -NCP).

2012 ACM Subject Classification Theory of computation \rightarrow Problems, reductions and completeness; Theory of computation \rightarrow Parameterized complexity and exact algorithms

Keywords and phrases Average Baby PIH, Parameterized Inapproximability, Constraint Satisfaction Problem, Exact Set Cover, $W[1]$ -hardness

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.65

Funding Yuwei Liu and Yijia Chen are supported by the National Natural Science Foundation of China (Project 62372291).

Acknowledgements The authors want to thank Guohang Liu, Mingjun Liu, Yangluo Zheng for discussions in the early stage of this work. The comments and suggestions from anonymous reviewers also help to improve the paper significantly. In particular, Theorem 2 is due to one of them.



© Yuwei Liu, Yijia Chen, Shuangle Li, Bingkai Lin, and Xin Zheng;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 65; pp. 65:1–65:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 Introduction

In classical complexity theory, the PCP theorem [3, 2, 8] serves as an essential tool for proving most of the existing results in the hardness of approximation. As a variant, the Multi-Assignment PCP theorem [1, Lemma 11] states that, for any constant $r > 1$ and $0 < \varepsilon < 1$, it is even NP-hard to decide whether a CSP instance is satisfiable, or any multi-assignment (see Definition 7.) that assigns each variable no more than r values cannot satisfy a $(1 - \varepsilon)$ -fraction of constraints. Among others, the Multi-Assignment PCP theorem was used to show the NP-hardness of approximating SETCOVER [1]. It turns out that the Multi-Assignment PCP theorem is a simple consequence of the PCP theorem by a straightforward probabilistic argument. Nevertheless, Barto and Kozik [4] gave a direct and purely combinatorial proof for the simple case of $\varepsilon = 0$. Observe that it means that the CSP instances under consideration are either satisfiable or cannot be satisfied by a desired multi-assignment. This restricted version of the Multi-Assignment PCP theorem is termed the *Baby PCP Theorem* in [4].

As an analog of the PCP theorem in parameterized complexity theory, the Parameterized Inapproximability Hypothesis [21], PIH for short, is an important assumption from which we can prove many FPT inapproximability results, including the inapproximability of k -CLIQUE, k -EXACTCOVER [13], and DIRECT ODD CYCLE TRANSVERSAL [21], etc. It claims that for some constant $0 < \varepsilon < 1$, no $f(k) \cdot n^{O(1)}$ -time (i.e., FPT) algorithm can distinguish a satisfiable 2CSP instance with k variables from one where less than $(1 - \varepsilon)$ -fraction of constraints can be satisfied simultaneously [21]. Unlike the PCP theorem, the PIH is still a major open problem in parameterized complexity. The current state of the art is that the PIH holds under the Exponential Time Hypothesis (ETH) [12, 11], and a proof of the PIH under the minimum assumption $W[1] \neq \text{FPT}$ remains elusive. Toward this goal, studying some consequences of the PIH and proving them under $W[1] \neq \text{FPT}$ might provide new insights and valuable lessons.

Recently, Guruswami, Ren, and Sandeep [13] proved a parameterized version of the Baby PCP Theorem, appropriately coined as the *Baby PIH*, under $W[1] \neq \text{FPT}$. The Baby PIH states that for any constant $r > 1$, no FPT algorithm can distinguish a satisfiable 2CSP instance from one with no satisfying multi-assignment which assigns each variable no more than r values. Just like the relationship between the PCP theorem and the Baby PCP theorem, the Baby PIH is a direct consequence of the PIH. As a next step, a further complexity assumption is suggested, i.e., the *Average Baby PIH* [13, Conjecture 3], which seems to be sandwiched between the PIH and the Baby PIH. It postulates the $W[1]$ -hardness of the problem known as AVG- r -GAP-2CSP (see Definition 8) which asks for distinguishing a satisfiable 2CSP instance from one without satisfying multi-assignment which assigns each variable no more than r values *on average*. The authors of [13] also demonstrated the difference between the Baby PIH and the Average Baby PIH. In fact, for all $r > 1$ and $\delta > 0$, they constructed 2CSP instances with variable set X that cannot be satisfied by any multi-assignment assigning each variable in X no more than r values, but can be satisfied by a multi-assignment that assigns in total $(1 + \delta)|X|$ values to all the variables in X , that is, every variable is assigned $1 + \delta$ values on average. Compared to proving $W[1]$ -hardness¹ of the PIH, it is apparently easier to show the $W[1]$ -hardness of the Average Baby PIH, and

¹ Strictly speaking, the PIH is not a computational problem and we cannot directly define its hardness. The formal statement should be “proving $W[1]$ -hardness of the problem described in the PIH”, and we use “ $W[1]$ -hardness for the PIH” for short in the introduction.

studying the Average Baby PIH might bring us further closer to a proof of the $W[1]$ -hardness for the PIH. Moreover, the Average Baby PIH is already sufficient for proving some non-trivial inapproximability results such as k -EXACTCOVER [13].

1.1 Main Results

Let $\Pi = (X, \Sigma, \Phi)$ be a 2CSP instance with a set X of variables, an alphabet Σ , and a set Φ of constraints. A multi-assignment $\hat{\sigma} : X \rightarrow 2^\Sigma$ relaxes the standard notion of assignments by assigning each variable $x \in X$ a set of values in Σ , i.e., $\hat{\sigma}(x) \subseteq \Sigma$. Thereby, Π is said to be *satisfied* by $\hat{\sigma}$ if for every constraint $\varphi \in \Phi$, one can pick for each variable x of φ a value from the set $\hat{\sigma}(x)$ assigned to this variable to satisfy φ . We say that $\hat{\sigma}$ assigns $\sum_{x \in X} |\hat{\sigma}(x)|$ values to X in total, or equivalently, each variable in X is assigned $\frac{\sum_{x \in X} |\hat{\sigma}(x)|}{|X|}$ values *on average* (see Definition 7). Our main result is the following theorem stating that the Average Baby PIH holds under $W[1] \neq \text{FPT}$.

► **Theorem 1** (Informal, see Theorem 16). *Assume $W[1] \neq \text{FPT}$. Then for any constant $r > 1$, given a 2CSP instance $\Pi = (X, \Sigma, \Phi)$ parameterized by $|X|$, no FPT time algorithm can distinguish between:*

- Π is satisfiable.
- Any multi-assignment assigning no more than $r|X|$ values to X does not satisfy Π .

Clearly any standard assignment $\sigma : X \rightarrow \Sigma$ can be identified with a multi-assignment $\hat{\sigma}$ that assigns each variable $x \in X$ a set of a single value, i.e., $\hat{\sigma}(x) = \{\sigma(x)\}$. Hence, there is a constant r gap in Theorem 1 between YES and NO instances on the average number of values assigned to each variable, which gives us the aforementioned AVG- r -GAP-2CSP problem. On the other hand, the constant gap for the PIH is on the fraction of constraints that can be satisfied by an assignment. That is, a YES instance is a 2CSP instance whose *all* constraints can be satisfied by an assignment, while any assignment can only satisfy at most a *constant fraction* of constraints in a NO instance. So, perhaps surprisingly, the difference between the Average Baby PIH and the PIH can be pinpointed within the AVG- r -GAP-2CSP problem precisely in terms of whether a given instance contains a “dense” or “sparse” set of constraints.² More precisely:

► **Theorem 2.**

- Under $W[1] \neq \text{FPT}$, the Average Baby PIH holds for AVG- r -GAP-2CSP instances $\Pi = (X, \Sigma, \Phi)$ with $|\Phi| = \omega(|X|)$.
- If the Average Baby PIH holds for AVG- r -GAP-2CSP instances $\Pi = (X, \Sigma, \Phi)$ with $|\Phi| = O(|X|)$, then the PIH holds as well.

As a first application of the Average Baby PIH under $W[1] \neq \text{FPT}$, using a reduction in [13], we obtain the $W[1]$ -hardness of constant approximating the k -EXACTCOVER problem (see Definition 9), improving its previous approximation lower bound under a stronger assumption, i.e., the Gap-ETH [23].

► **Theorem 3** (Theorem 27 restated). *For any constant $r > 1$, r -approximating k -EXACTCOVER is $W[1]$ -hard.*

² This is pointed out by an anonymous reviewer.

We remark that the $W[1]$ -hardness of approximating k -EXACTCOVER has been a long-standing open problem in parameterized complexity. Although the $W[1]$, $W[2]$, ETH-hardness of approximating the k -SETCOVER problem has been established in [6, 15, 18, 20], as a special case of k -SETCOVER, the hardness of approximating k -EXACTCOVER was only known under the PIH [23] prior to our work.

The second application is a close relationship between running time lower bounds for constant approximating the parameterized Nearest Codeword Problem γ -GAP- k -NCP _{p} (see Definition 10) and AVG- r -GAP-2CSP. Its proof is a straightforward composition of two known reductions in [23, 13].

► **Theorem 4.** *For any prime p , computable function g , and constant r , if no $f(k) \cdot n^{o(g(k))}$ -time algorithm can decide AVG- r -GAP-2CSP with k variables for any computable function f , then r -GAP- k -NCP _{p} cannot be solved in time $f(k) \cdot n^{o(g(k))}$ for any computable function f .*

Proof Sketch. Theorem 4 follows from the gap-preserving reduction [13] from AVG- r -GAP-2CSP to k -EXACTCOVER (see also Section A), and the gap-preserving reduction from k -EXACTCOVER to r -GAP- k -NCP _{p} [23, Theorem 28]. Note that in both reductions the parameter k is preserved. ◀

1.2 Technical Overview: Local-to-Global Reduction For 2CSP

To prove Theorem 1, we show that the $W[1]$ -hardness for the Baby PIH implies the $W[1]$ -hardness for the Average Baby PIH. Here, the “ $W[1]$ -hardness for the Baby PIH” means that, for all $r > 1$, it’s $W[1]$ -hard to decide whether (i) a 2CSP instance is satisfiable, or (ii) it cannot be satisfied by any multi-assignment assigning each variable no more than r values. Thus there is a constant r gap between (i) and (ii) in the number of values assigned to *each* variable. Thereby, the gap is “local.” As already mentioned, for the Average Baby PIH, the gap is on the average number of values, or equivalently, the *total* number of values assigned to *all* the variables. Hence, the gap is “global.” Our reduction from the Baby PIH to the Average Baby PIH is thus said to be “local-to-global.”

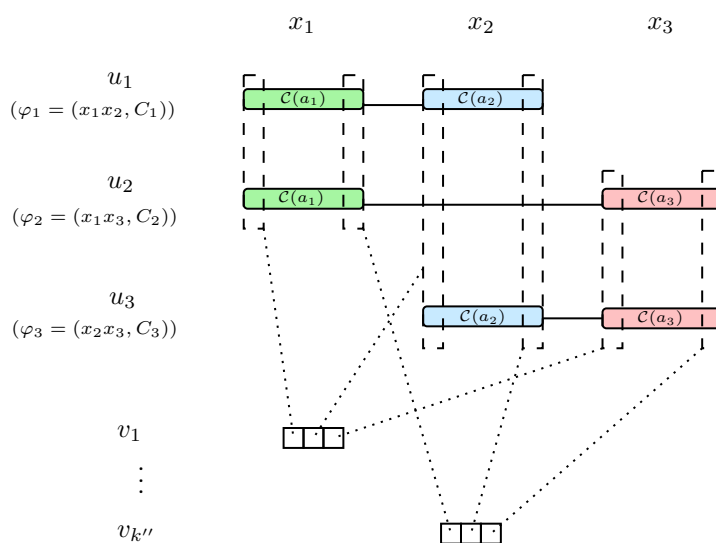
Technically, our reduction relies on a simple but crucial property of high-distance error-correcting codes (ECC) shown in [16, 20]. In particular, we need an ECC $\mathcal{C} \subseteq \mathbb{F}_p^m$ with relative distance $1 - \delta$ such that any two distinct codewords $x, y \in \mathcal{C}$ can agree on at most δm entries. So if we have a set S of codewords and an ε -fraction of entries (denoted by $I \subseteq [m]$) with $\varepsilon \gg \delta$ such that for each $i \in I$, we can find distinct $x, y \in S$ that agree on their i -th entry, then the size of S must be large. The lower bound of $|S|$ is called the *collision number* of \mathcal{C} , denote by $\text{Col}_\varepsilon(\mathcal{C})$. A simple counting argument in [20] shows that $\text{Col}_\varepsilon(\mathcal{C}) \geq \sqrt{2\varepsilon/\delta}$.

Now given a 2CSP instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$, let $n = |\Pi_0| > |\Sigma_0|$, parameter $k = |X_0|$, and $k' = |\Phi_0|$. We fix some ECC with very large relative distance (e.g., a Reed-Solomon code) $\mathcal{C} : \mathbb{F}_p^k \rightarrow \mathbb{F}_p^{k''}$ for prime $n^{1/k} \leq p < 2n^{1/k}$ and $k'' = \Theta(k^5)$. Then we have $\text{Col}_\varepsilon(\mathcal{C}) > 2rk$. We construct a new 2CSP instance $\Pi = (X_1 \dot{\cup} X_2, \Sigma, \Phi)$ as:

- $X_1 = \{u_1, \dots, u_{k'}\}$, $X_2 = \{v_1, \dots, v_{k''}\}$.
- Each u_j takes value from the (encoding of) satisfying assignments of $\varphi_j = (x_{i_1} x_{i_2}, C_j) \in \Phi_0$, i.e., $\{(\mathcal{C}(a_1), \mathcal{C}(a_2)) : (a_1, a_2) \in C_j\}$. Each v_ℓ takes value from \mathbb{F}_p^k .
- For each $u_j \in X_1$ and $v_\ell \in X_2$, there is a constraint that checks whether u_j ’s assigned value (w_1, w_2) and v_ℓ ’s assigned value s satisfy $w_1[\ell] = s[i_1]$ and $w_2[\ell] = s[i_2]$.

See Figure 1 for an illustration. Finally, we duplicate X_1 and X_2 each an appropriate number of times to make them equal in size, finishing our reduction.

In general, an assignment to X_1 should correspond to a satisfying multi-assignment to the original instance Π_0 , and the value assigned to each $v_\ell \in X_2$ is a “guess” of the ℓ -th entry of the encoding of every x_i . It is easy to see that if the input instance Π_0 is satisfiable, then so



■ **Figure 1** An illustration of our construction for an input instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$ with $|X_0| = |\Phi_0| = 3$.

does the new 2CSP instance Π , since for each $u_j \in X_1$ corresponding to $\varphi_j = (x_{i_1}x_{i_2}, C_j)$, we can simply assign it the value $(\mathcal{C}(\sigma(x_{i_1})), \mathcal{C}(\sigma(x_{i_2})))$, where σ is a satisfying assignment for Π_0 . At the same time, the value assigned to each $v_\ell \in X_2$ is $(\mathcal{C}(\sigma(x_1))[\ell], \dots, \mathcal{C}(\sigma(x_k))[\ell]) \in \mathbb{F}_p^k$.

For soundness, suppose Π_0 has no satisfying multi-assignment assigning at most r values to each variable, we need to argue that Π has no satisfying multi-assignment that assigns $r(1 - \varepsilon)(|X_1| + |X_2|)/2$ values in total. To that end, we exploit the collision number of the code \mathcal{C} . Fix any satisfying multi-assignment $\hat{\sigma}$ to Π . Recall that each variable $u_j \in X_1$ is assigned some value from a satisfying partial assignment to $\varphi_j \in \Phi_0$. Then, $\hat{\sigma}(X_1)$ naturally gives a satisfying multi-assignment to Π_0 , which implies that there exists a variable $x_i \in X_0$ such that more than r different values are assigned by $\hat{\sigma}$ to u_j , for which the corresponding constraint φ_j contains x_i .

Now we have two possible cases for $\hat{\sigma}$. In the first case, for $(1 - \varepsilon)$ -fraction of variables in X_2 , the multi-assignment $\hat{\sigma}$ assigns each of them more than r values. We are done, since this implies that the total number of assigned values by $\hat{\sigma}$ is more than $(1 - \varepsilon)r|X_2|$. For the second case, there exists an ε -fraction of variables in X_2 , each of which is assigned by $\hat{\sigma}$ at most r values. Given such a variable v_ℓ , we have more than r different *codewords* assigned to X_1 in x_i 's position which has at most r possible values in the ℓ -th entry. This entails the existence of two different codewords with the same ℓ -th entry. Since there are εm such entries, the assignment to X_1 must contain at least $\text{Col}_\varepsilon(\mathcal{C})$ different codewords. Each assignment to $u_j \in X_1$ contributes two codewords, so the total number of assigned values by $\hat{\sigma}$ is at least $\text{Col}_\varepsilon(\mathcal{C})/2 > rk$. In summary, both cases guarantee a constant gap on either X_1 or X_2 , showing that any satisfying multi-assignment to Π must assign $r(1 - \varepsilon)(|X_1| + |X_2|)/2$ values to $X_1 \cup X_2$ in total.

More details are referred to Section 3.

1.3 Discussions

For minimization problems, the technique of constructing two parts of variables and arguing that at least one part has a large gap seems quite general, as exhibited by the previous works showing the W[1]-hardness of approximating k -SETCOVER [6, 20] and k -NCP [17].

The use of the collision number of error-correcting codes in the context of parameterized inapproximability was first introduced in [16] and further developed in [20, 17]. We ask whether these techniques can be unified.

► **Question 1.** *Is there a general framework for proving parameterized inapproximability of minimization problems?*

We also suggest two new variants of the Average Baby PIH, which might serve as a next step towards proving the PIH under $W[1] \neq FPT$. On closer inspection of our construction, particularly Case 1 in the proof of Lemma 20, it only guarantees the total number of values assigned to X_1 (i.e., $\sum_{x \in X_1} |\hat{\sigma}(x)|$) is large. This could happen if an $o(1)$ -fraction of x 's in X_1 is assigned super-constant number of values. We ask if a larger gap can be achieved. So the first one asks whether the Average Baby PIH can be strengthened by requiring a constant fraction of variables to be assigned multiple values.

► **Question 2.** *Under $W[1] \neq FPT$, can we prove that for any constant $r > 2$, there exists a constant $c > 0$ such that no FPT algorithm can decide whether a 2CSP instance is satisfiable, or any satisfying multi-assignment must have at least a c -fraction of variables assigned r values?*

We remark that the case of $r = 2$ follows from the inapproximability of k -CLIQUE [19, 14, 5].

The second variant is already contained in Theorem 2, thus equivalent to PIH.

► **Question 3.** *Under $W[1] \neq FPT$, can we prove that the Average PIH holds even for AVG- r -GAP-2CSP instances with the number of constraints being linear in the number of variables?*

It is also interesting to consider whether the inapproximability factor in the Average Baby PIH can be improved to $\omega(1)$, since this would directly lead to better lower bounds for approximating k -EXACTCOVER. The current obstacle is that, although the running time of our reduction does not depend on the approximation factor, our reduction relies on the gap created in the Baby PIH [13]. In order to achieve an r -gap in the Baby PIH, the reduction in [13] runs in time $\Omega(n^{(2r)^r})$, consequently, the existing FPT reduction cannot create a super-constant $r = \omega(1)$ gap.

► **Question 4.** *Under $W[1] \neq FPT$, can we prove that the Average PIH (Theorem 1) holds for inapproximability factor $r = \omega(1)$, hence giving better inapproximability result for k -EXACTCOVER?*

1.4 Organization

In Section 2, we introduce the main computational problems and complexity assumptions studied in this paper. As the central contribution, Section 3 explains our reduction from the Baby PIH to the Average Baby PIH. This, in fact, establishes the Average Baby PIH under $W[1] \neq FPT$. In Section A, we present a reduction from AVG- r -GAP-2CSP to the constant approximation of k -EXACTCOVER, which slightly differs from the construction in [13].

2 Preliminaries

For a positive integer n , we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. $S_1 \dot{\cup} \dots \dot{\cup} S_k$ is the *disjoint union* of sets S_1, \dots, S_k , where we tacitly assume that S_1, \dots, S_k are pairwise disjoint. We use \log (without subscript) to denote the logarithm number with base 2. For any prime

number p , we write \mathbb{F}_p for the (unique) finite field of size p . The asymptotic notations, i.e., O, Ω, ω , and Θ , are used following the general convention. The reader is assumed to be familiar with basic notions in parameterized complexity theory, in particular FPT and W[1]. Otherwise, the standard references are, e.g., [10, 9, 7].

2.1 Problems

► **Definition 5** (Parameterized 2CSP). *A 2CSP instance is defined as a triple $\Pi = (X, \Sigma, \Phi)$ where:*

- X is a set of variable.
- $\Sigma = \bigcup_{x \in X} \Sigma_x$, where each Σ_x contains values that the variable $x \in X$ can be assigned. Often, we assume that there exists an $n \in \mathbb{N}$ such that $|\Sigma_x| \leq n$ for all $x \in X$.
- $\Phi = \{\varphi_1, \dots, \varphi_{k'}\}$, where each $\varphi_j = (x_{i_1}x_{i_2}, C_j)$ for some $x_{i_1}, x_{i_2} \in X$, and C_i is a subset of $\Sigma_{x_{i_1}} \times \Sigma_{x_{i_2}}$.

The problem is to decide whether there exists an assignment $\sigma : X \rightarrow \Sigma$ that satisfies:

- For all $x \in X$, $\sigma(x) \in \Sigma_x$.
- For all $\varphi_j = (x_{i_1}x_{i_2}, C_j) \in \Phi$, $(\sigma(x_{i_1}), \sigma(x_{i_2})) \in C_j$.

The parameter for this problem is $k = |X|$, the number of variables. Each pair of variables has at most one constraint, so $|\Phi| \leq \binom{k}{2}$. Without loss of generality, each variable is related to some constraint in Φ . The size of instance Π is defined as $|\Pi| = |\Sigma| + |\Phi|$, where the size of each φ_j is defined as $|\varphi_j| = |C_j|$.

The approximation of parameterized 2CSP refers to the following problem.

► **Definition 6** (ε -GAP-2CSP). *Given a 2CSP instance $\Pi = (X, \Sigma, \Phi)$ with parameter $k = |X|$, we want to distinguish between:*

- Π is satisfiable;
- any assignment can satisfy at most an ε -fraction of constraints in Φ .

As already mentioned, the notion of multi-assignment extends the usual assignment in such a way that each variable can be assigned multiple values.

► **Definition 7** (Multi-assignment). *A multi-assignment of a 2CSP instance $\Pi = (X, \Sigma, \Phi)$ is a function $\hat{\sigma} : X \rightarrow 2^\Sigma$,³ such that for all $x \in X$ we have $\hat{\sigma}(x) \subseteq \Sigma_x$. Furthermore, we say that $\hat{\sigma}$ satisfies Π if:*

- For all $\varphi_j = (x_{i_1}x_{i_2}, C_j) \in \Phi$, there exist $c_1 \in \hat{\sigma}(x_{i_1})$ and $c_2 \in \hat{\sigma}(x_{i_2})$ with $(c_1, c_2) \in C_j$.

The individual size of $\hat{\sigma}$ is defined as $\max_{x \in X} |\hat{\sigma}(x)|$, and the total size of $\hat{\sigma}$ is $\sum_{x \in X} |\hat{\sigma}(x)|$.

Let $r \geq 1$. We say that a 2CSP instance $\Pi = (X, \Sigma, \Phi)$ is *r-list satisfiable* if there exists a multi-assignment $\hat{\sigma}$ with individual size no more than r which satisfies Π , and Π is *r-average list satisfiable* if there exists a multi-assignment $\hat{\sigma}$ with total size no more than $r|X|$ which satisfies Π .

► **Definition 8** (AVG- r -GAP-2CSP). *Given a 2CSP instance Π , the goal is to distinguish between the following two cases:*

- Π is satisfiable.
- Π is not r -average list satisfiable.

³ Here we use 2^Σ to denote the power set of Σ .

We also consider the k -EXACTCOVER problem (aka, the k -UNIQUESETCOVER problem) and the k -NCP problem (aka, k -MLD, for the parameterized Maximum Likelihood Decoding problem) as defined below.

► **Definition 9** (k -EXACTCOVER). *Given a set U (which we call universe) and a collection of U 's subsets \mathcal{S} , the goal is to distinguish between the following two cases:*

- *there exist at most k disjoint sets in \mathcal{S} that form a partition of U ,*
- *or U is not the union of any k sets in \mathcal{S} .*

► **Definition 10** (k -NCP). *For prime p , integer $d > 0$, given a (multi-)set V of vectors in \mathbb{F}_p^d , and a target vector $\vec{t} \in \mathbb{F}_p^d$, the k -NCP $_p$ problem asks for distinguishing between:*

- *the Hamming distance between \vec{t} and the vector space spanned by V is at most k ,*
- *or the Hamming distance between \vec{t} and the vector space spanned by V is at least $k + 1$.*

2.2 Hypotheses

► **Hypothesis 11** (PIH [21]). *For every constant $0 < \varepsilon < 1$, there is no FPT algorithm solving ε -GAP-2CSP.*

The Baby PIH, a hypothesis implied by PIH, asserts the hardness of approximating individual size of a satisfying multi-assignment. Formally,

► **Hypothesis 12** (Baby PIH [13]). *For any constant $r > 0$, no FPT algorithm can on input a 2CSP instance, distinguish whether it is satisfiable, or cannot be satisfied by any multi-assignment with individual size at most r .*

We emphasize that the Baby PIH is a hardness hypothesis with a **local** condition, i.e., the individual size of satisfying assignments. It is shown that the standard assumption $W[1] \neq \text{FPT}$ implies the Baby PIH:

► **Theorem 13** ([13]). *The Baby PIH holds under $W[1] \neq \text{FPT}$.*

In contrast, the Average Baby PIH is defined on a **global** condition concerning the total size of satisfying assignments. The precise statement of this complexity assumption contains a technical property on the “shape” of the constraints in a 2CSP instance.

► **Definition 14** (Rectangular relation). *A 2CSP instance $\Pi = (X, \Sigma, \Phi)$ is said to have rectangular relations if for each $\varphi_j = (x_{i_1} x_{i_2}, C_j) \in \Phi$, there exist a set Q_j and mappings $\pi_j, \rho_j : \Sigma \rightarrow Q_j$, such that $(a, b) \in C_j$ iff $\pi_j(a) = \rho_j(b)$. We call Q_j the underlying set of φ_j .*

Some explanation for “rectangular” might be in order. Recall that a subset $S \subseteq \Sigma^2$ is a (combinatorial) *rectangle* if and only if there exist $A, B \subseteq \Sigma$ such that $S = A \times B$. It is easy to verify that $R \subseteq \Sigma^2$ is rectangular if and only if R is the union of a set of pairwise disjoint rectangles.

► **Hypothesis 15** (Average Baby PIH). *For any constant $r > 0$, there exists no FPT algorithm solving the AVG- r -GAP-2CSP problem, even when the instance contains only rectangular relations.*

3 Average Baby PIH from Baby PIH

In this section, we show that the Average Baby PIH even for instances with only rectangular relations, is implied by the Baby PIH.

3.1 Proofs of Main Results

We employ a local-to-global reduction developed in [17] to amplify the local gap for one variable (Theorem 13) into a global gap for all variables, thus proving the Average Baby PIH from the Baby PIH.

► **Theorem 16.** *Under $W[1] \neq \text{FPT}$, for any constant $r > 0$, no FPT algorithm can distinguish a given 2CSP instance with rectangular relation is satisfiable, or cannot be satisfied by any multi-assignment with total size no more than r .*

To show Theorem 16, we first introduce some tools from coding theory. The *collision number* of an error-correcting code characterizes the number of codewords needed to find “collision” on a constant fraction of coordinates. We use the definition in [17]:

► **Definition 17** (ε -Collision Number). *Let $m \geq 1$ and $x, y \in \Sigma^m$ with $x \neq y$. For every $i \in [m]$ we say that x and y collide on position i if $x[i] = y[i]$. Furthermore, a subset $S \subseteq \Sigma^m$ collides on position i if there exist distinct $x, y \in S$ with $x[i] = y[i]$. We define the collision set of S as*

$$\text{ColSet}(S) = \{i \in [m] \mid S \text{ collides on position } i\}.$$

Observe that if $|S| \leq 1$, then $\text{ColSet}(S) = \emptyset$.

Now for every $C \subseteq \Sigma^m$ and $0 < \varepsilon < 1$ the ε -collision number of C , denoted by $\text{Col}_\varepsilon(C)$, is the maximum $s \leq |C| + 1$ such that for all $S \in \binom{C}{s-1}$ we have

$$|\text{ColSet}(S)| \leq \varepsilon m.$$

For Reed-Solomon codes, we have the following lower bounds on their collision number.

► **Theorem 18** (Theorem 10 in [20], see also [16]). *For any $0 < \varepsilon < 1$, any Reed-Solomon code $\mathcal{C}^{RS} : \mathbb{F}_p^k \rightarrow \mathbb{F}_p^m$ with sufficiently large $k < m \leq p$, $\text{Col}_\varepsilon(\mathcal{C}^{RS}) \geq \sqrt{\frac{2\varepsilon m}{k}}$.*

Our proof of Theorem 16 consists of two reductions. The first one (Lemma 20) reduces 2CSP instances from the Baby PIH, i.e., with a “local” gap as explained in Section 1.2, to a new instance whose constraints are between two disjoint groups of variables. The new instance has a different “global” gap on each group of variables. As the sizes of the two groups might not be balanced, we do not necessarily have a “global” gap on all the variables. But this is easily remedied by the second reduction (Lemma 22) which makes an appropriate number of copies of the two groups.

► **Definition 19** ((r, s) -Average Multi-Assignment). *Let $\Pi = (X, \Sigma, \Phi)$ be a bipartite 2CSP instance, in particular $X = X_1 \dot{\cup} X_2$ and every $\varphi = (x_1 x_2, C) \in \Phi$ has $x_1 \in X_1$ and $x_2 \in X_2$. Then for $r_1, r_2 \geq 1$ an (r_1, r_2) -average multi-assignment of Π is a multi-assignment $\hat{\sigma} : X \rightarrow 2^\Sigma$ such that*

$$\frac{\sum_{x \in X_1} |\hat{\sigma}(x)|}{|X_1|} \leq r_1 \quad \text{and} \quad \frac{\sum_{x \in X_2} |\hat{\sigma}(x)|}{|X_2|} \leq r_2.$$

That is, the total size of $\hat{\sigma}$ restricted to X_1 is at most $r_1 |X_1|$, and the total size of $\hat{\sigma}$ restricted to X_2 is at most $r_2 |X_2|$ (cf. Definition 7). We say Π is (r_1, r_2) -average list satisfiable if there is an (r_1, r_2) -average multi-assignment which satisfies Π .

► **Lemma 20.** *There is an algorithm \mathcal{A} which on input a 2CSP instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$, $\varepsilon > 0$, and $r \geq 1$ computes a bipartite 2CSP instance $\Pi = (X_1 \dot{\cup} X_2, \Sigma, \Phi)$ with the following properties.*

Completeness. *If Π_0 is satisfiable, then so is Π ,*

65:10 On Average Baby PIH and Its Applications

Soundness. For every $r \geq 1$ if Π_0 is not $2r$ -list satisfiable, then Π is not (r_1, r_2) -average list satisfiable for every $r_1, r_2 \in \mathbb{N}$ with

$$r_1 + r_2 \leq 2(1 - \varepsilon)r.$$

Rectangularity. All constraints in Φ are rectangular.

In addition, there exists a computable function f upper bounding the running time of \mathcal{A} as

$$f(|X_0| + |\Phi_0| + 1/\varepsilon + r)|\Sigma_0|^{O(1)}. \quad (1)$$

And the number of variables $|X_1| + |X_2|$ and the number of constraints $|\Pi|$ in Π can also be upper bounded by $f(|X_0| + |\Phi_0| + 1/\varepsilon + r)$.

Proof. For the given 2CSP instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$ we let

$$k = |X_0| \quad \text{and} \quad k' = |\Phi_0|.$$

Thereby we fix some enumerations of the variables in X_0 and the constraints in Φ_0 as

$$X_0 = \{x_1, \dots, x_k\} \quad \text{and} \quad \Phi_0 = \{\varphi_1, \dots, \varphi_{k'}\}.$$

Let $\mathcal{C} : \mathbb{F}_p^k \rightarrow \mathbb{F}_p^{k''}$ be a Reed-Solomon code with

$$2|\Sigma_0|^{1/k} > p \geq |\Sigma_0|^{1/k} \quad \text{and} \quad k'' = \left\lfloor \frac{8(1 - \varepsilon)^2 r^2}{\varepsilon} k(k')^2 \right\rfloor + 1.$$

Clearly $|\Sigma_0| \leq p^k$, and therefore we can assume without loss of generality

$$\Sigma_0 \subseteq \mathbb{F}_p^k.$$

Moreover, we only consider the case that

$$k'' \leq p \left(= |\mathbb{F}_p| \right) < 2|\Sigma_0|^{1/k},$$

i.e., Σ_0 is sufficiently larger than k and k' .⁴ Hence we can invoke Theorem 18 on $\Sigma \leftarrow \mathbb{F}_p$, $k \leftarrow k$, $m \leftarrow k''$, and $\varepsilon \leftarrow \varepsilon$ to obtain

$$\text{Col}_\varepsilon(\mathcal{C}(\mathbb{F}_p^k)) \geq \sqrt{\frac{2\varepsilon k''}{k}} > 4(1 - \varepsilon)rk', \quad (2)$$

where the second inequality is by our choice of k'' .

Now the algorithm \mathcal{A} constructs the following bipartite 2CSP instance $\Pi = (X, \Sigma, \Phi)$.

Variables. $X = X_1 \dot{\cup} X_2$ with

$$X_1 = \{u_1, \dots, u_{k'}\} \quad \text{and} \quad X_2 = \{v_1, \dots, v_{k''}\}.$$

Alphabets. $\Sigma = \bigcup_{u \in X_1} \Sigma_u \cup \bigcup_{v \in X_2} \Sigma_v$ where:

- For every $j \in [k']$ the alphabet of the variable $u_j \in X_1$ is

$$\Sigma_{u_j} = \left\{ (\mathcal{C}(a_1), \mathcal{C}(a_2)) \mid \varphi_j = (x_{i_1} x_{i_2}, C) \text{ and } (a_1, a_2) \in C \right\} \subseteq (\mathcal{C}(\mathbb{F}_p^k))^2 \subseteq (\mathbb{F}_p^{k''})^2. \quad (3)$$

That is, Σ_{u_j} contains all the (partial) satisfying assignments of φ_j encoded by $\mathcal{C} : \mathbb{F}_p^k \rightarrow \mathbb{F}_p^{k''}$ as pairs of vectors in $\mathbb{F}_p^{k''}$. (Recall $\Sigma_0 \subseteq \mathbb{F}_p^k$.)

- For every $\ell \in [k'']$ we have $\Sigma_{v_\ell} = \mathbb{F}_p^k$. Since $p < 2|\Sigma_0|^{1/k}$, we have $|\Sigma_{v_\ell}| \leq 2^k |\Sigma_0|$.

⁴ Otherwise, the original instance Π_0 can be solved in time of the form (1), and we can then output some predetermined Π depending on whether Π_0 is satisfiable.

Constraints. Let $j \in [k']$ and $\varphi_j = (x_{i_1}x_{i_2}, C)$. Then for every $\ell \in [k'']$ we have a constraint between the variable $u_j \in X_1$ and $v_\ell \in X_2$ which checks whether u_j is assigned to $(w_1, w_2) \in (\mathbb{F}_p^{k''})^2$ and v_ℓ to $s \in \mathbb{F}_q^\ell$ such that

$$w_1[\ell] = s[i_1] \quad \text{and} \quad w_2[\ell] = s[i_2]. \quad (4)$$

Consequently (4) implies that the constraint is rectangular.⁵ Moreover, the number of constraints in Π is

$$k'k'' = k' \left\lceil \frac{2(1-\varepsilon)^2 r^2}{\varepsilon} k(k')^2 \right\rceil + k'.$$

The completeness of our reduction is straightforward. So we turn to the soundness. In particular, we assume that the given 2CSP instance Π_0 is *not* $2r$ -list satisfiable. Furthermore, let $\hat{\sigma} : X \rightarrow 2^\Sigma$ be a satisfying multi-assignment for Π . We need to show that, for any $r_1, r_2 \in \mathbb{N}$ if there is a satisfying (r_1, r_2) -average multi-assignment $\hat{\sigma}$, then

$$r_1 + r_2 > 2(1-\varepsilon)r. \quad (5)$$

To that end, let

$$\text{Word}_{\hat{\sigma}} = \bigcup_{u_j \in X_1} \bigcup_{(w_1, w_2) \in \hat{\sigma}(u_j)} \{w_1, w_2\} \subseteq \mathbb{F}_p^{k''}. \quad (6)$$

That is, $\text{Word}_{\hat{\sigma}}$ is the set of all codewords in $\mathbb{F}_p^{k''}$ that $\hat{\sigma}$ uses for the variables in X_1 .

▷ **Claim 21.** Let $\ell \in [k'']$ with $|\hat{\sigma}(v_\ell)| \leq 2r$. Then $\text{Word}_{\hat{\sigma}}$ collides on position ℓ .

Proof of Claim 21. Let $\ell \in [k'']$ be fixed with $|\hat{\sigma}(v_\ell)| \leq 2r$.

Consider an arbitrary constraint $\varphi_j = (x_{i_1}x_{i_2}, C) \in \Phi_0$ (i.e., $j \in [k']$). Since $\hat{\sigma}$ is a satisfying multi-assignment for Π , there exist

$$(w_1, w_2) \in \hat{\sigma}(u_j) \subseteq \Sigma_{u_j} \subseteq (\mathbb{F}_p^{k''})^2 \quad \text{and} \quad s \in \hat{\sigma}(v_\ell) \subseteq \mathbb{F}_p^k$$

such that $u_j = (w_1, w_2)$ and $v_\ell = s$ satisfy the constraint between u_j and v_ℓ in Π . By $(w_1, w_2) \in \Sigma_{u_j}$ and (3) there are $a_1, a_2 \in \Sigma_0$ with $w_1 = \mathcal{C}(a_1)$ and $w_2 = \mathcal{C}(a_2)$ such that

$$x_{i_1} = a_1 \quad \text{and} \quad x_{i_2} = a_2 \quad \text{satisfy} \quad \varphi_j. \quad (7)$$

Then we say that a_1 is $(\hat{\sigma}, \varphi_j)$ -suitable for x_{i_1} with respect to s , and similarly a_2 is $(\hat{\sigma}, \varphi_j)$ -suitable for x_{i_2} with respect to s .

In addition, by (4)

$$\mathcal{C}(a_1)[\ell] = s[i_1] \quad \text{and} \quad \mathcal{C}(a_2)[\ell] = s[i_2]. \quad (8)$$

Now we define a *multi-assignment* $\hat{\sigma}_0 : X_0 \rightarrow 2^{\Sigma_0}$ for the original instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$ as follows. For every $x \in X_0$ let

$$\hat{\sigma}_0(x) = \bigcup_{s \in \hat{\sigma}(v_\ell)} \{a \in \Sigma_0 \mid j \in [k'] \text{ and } a \text{ is } (\hat{\sigma}, \varphi_j)\text{-suitable for } x \text{ with respect to } s\}. \quad (9)$$

(Recall that we have fixed an $\ell \in [k'']$ and hence $\hat{\sigma}(v_\ell)$.) Since every variable x must appear in at least one constraint $\varphi_j \in \Phi_0$ (cf. Definition 5), it is easy to see that $\hat{\sigma}_0$ is a satisfying multi-assignment for Π_0 by (7).

⁵ To see this, we take $\pi(u_j) = \pi(w_1, w_2) = (w_1[\ell], w_2[\ell])$ and $\rho(v_\ell) = (v_\ell[i_1], v_\ell[i_2])$. Then equation (4) is precisely $\pi(u_j) = \rho(v_\ell)$ as in Definition 14.

65:12 On Average Baby PIH and Its Applications

As Π_0 is *not* $2r$ -list satisfiable, there exists an $x_{i^*} \in X_0$ (i.e., $i^* \in [k]$) with

$$|\hat{\sigma}_0(x_{i^*})| \geq 2r + 1.$$

We have assumed that

$$|\hat{\sigma}(v_\ell)| \leq 2r,$$

so by (9) there is an $s \in \hat{\sigma}(v_\ell)$ such that

$$\left| \{a \in \Sigma_0 \mid j \in [k'] \text{ and } a \text{ is } (\hat{\sigma}, \varphi_j)\text{-suitable for } x_{i^*} \text{ with respect to } s\} \right| \geq 2.$$

Hence there are $a_1, a_2 \in \Sigma_0$ with $a_1 \neq a_2$ and $j_1, j_2 \in [k']$ such that

- a_1 is $(\hat{\sigma}, \varphi_{j_1})$ -suitable for x_{i^*} with respect to s ,
- and a_2 is $(\hat{\sigma}, \varphi_{j_2})$ -suitable for x_{i^*} with respect to s .

Then (8) implies that

$$\mathcal{C}(a_1)[\ell] = s[i^*] = \mathcal{C}(a_2)[\ell].$$

In other words, $\mathcal{C}(a_1)$ and $\mathcal{C}(a_2)$ collide on position ℓ . Clearly $\mathcal{C}(a_1), \mathcal{C}(a_2) \in \text{Word}_{\hat{\sigma}}$, so this finishes the proof of the claim. \triangleleft

Let

$$r_1 = \frac{\sum_{x \in X_1} |\hat{\sigma}(x)|}{|X_1|} = \frac{\sum_{j \in [k']} |\hat{\sigma}(u_j)|}{k'} \quad \text{and} \quad r_2 = \frac{\sum_{x \in X_2} |\hat{\sigma}(x)|}{|X_2|} = \frac{\sum_{\ell \in [k'']} |\hat{\sigma}(v_\ell)|}{k''}.$$

Now we distinguish two cases.

1. There are more than ε fraction of $\ell \in [k'']$ such that $|\hat{\sigma}(v_\ell)| \leq 2r$, then Claim 21 implies that $\text{Word}_{\hat{\sigma}}$ collides on more than ε fraction of positions $\ell \in [k'']$. Recall (2), i.e.,

$$\text{Col}_\varepsilon(\mathcal{C}(\mathbb{F}_p^k)) \geq \sqrt{\frac{2\varepsilon k''}{k}} > 4(1 - \varepsilon)rk'.$$

Hence,

$$|\text{Word}_{\hat{\sigma}}| \geq \text{Col}_\varepsilon(\mathcal{C}(\mathbb{F}_p^k)) > 4(1 - \varepsilon)rk'.$$

By the definition (6) of $\text{Word}_{\hat{\sigma}}$ we deduce

$$\begin{aligned} |\text{Word}_{\hat{\sigma}}| &= \left| \bigcup_{u_j \in X_1} \bigcup_{(w_1, w_2) \in \hat{\sigma}(u_j)} \{w_1, w_2\} \right| \\ &\leq \sum_{u_j \in X_1} \left| \bigcup_{(w_1, w_2) \in \hat{\sigma}(u_j)} \{w_1, w_2\} \right| \leq \sum_{u_j \in X_1} 2|\hat{\sigma}(u_j)| \end{aligned}$$

It follows that

$$r_1 = \frac{\sum_{j \in [k']} |\hat{\sigma}(u_j)|}{k'} > \frac{4(1 - \varepsilon)rk'}{2k'} = 2(1 - \varepsilon)r.$$

2. There are at most ε fraction of $\ell \in [k'']$ with $|\hat{\sigma}(v_\ell)| \leq 2r$. Or equivalently, there are at least $(1 - \varepsilon)$ fraction of $\ell \in [k'']$ with $|\hat{\sigma}(v_\ell)| \geq 2r + 1$. Then

$$r_2 = \frac{\sum_{\ell \in [k'']} |\hat{\sigma}(v_\ell)|}{k''} \geq \frac{(1 - \varepsilon)k''(2r + 1) + \varepsilon k''}{k''} > 2(1 - \varepsilon)r.$$

So both cases lead to (5) as desired. \blacktriangleleft

With some proper replication, the unbalanced (r_1, r_2) -gap can be turned into a balanced one, and yield the desired r -average list unsatisfiability.

► **Lemma 22.** *For any bipartite 2CSP instance $\Pi = (X_1 \dot{\cup} X_2, \Sigma, \Phi)$ and $r > 1$ we can compute in polynomial time a 2CSP instance $\Pi' = (X', \Sigma', \Phi')$ with*

$$|X| = 2|X_1||X_2|$$

such that

Completeness. *If Π is satisfiable, then so is Π' ,*

Soundness. *Let $r \geq 1$. If Π is not (r_1, r_2) -average list satisfiable for every $r_1, r_2 \geq 1$ with $r_1 + r_2 \leq 2r$, then Π' is not r -average list satisfiable. Or equivalently, if Π' is r -average list satisfiable, then for some $r_1, r_2 \in \mathbb{N}$ with $r_1 + r_2 \leq 2r$ the bipartite Π is (r_1, r_2) -average list satisfiable.*

Furthermore, if Π is rectangular, then so is Π' .

Proof. Let

$$k_1 = |X_1| \quad \text{and} \quad k_2 = |X_2|.$$

The desired $\Pi' = (X', \Sigma', \Phi')$ is constructed as below.

Variables. X' consists of k_2 copies of X_1 and k_1 copies of X_2 , i.e., $X' = X'_1 \dot{\cup} X'_2$ where

$$X'_1 = \{x^{(i)} \mid x \in X_1 \text{ and } i \in [k_2]\} \quad \text{and} \quad X'_2 = \{x^{(i)} \mid x \in X_2 \text{ and } i \in [k_1]\}.$$

Note, $|X'_1| = |X'_2| = k_1 k_2$, therefore Π' contains $2k_1 k_2$ many variables.

Alphabets. $\Sigma' = \bigcup_{x \in X'} \Sigma'_x$ where:

- For every $x \in X_1$ and $i \in [k_2]$, let $\Sigma'_{x^{(i)}} = \Sigma_x$. Recall that $\Sigma_x \subseteq \Sigma$ is the alphabet for the variable x in the original 2CSP instance Π .
- Similarly, for every $x \in X_2$ and $i \in [k_1]$ let $\Sigma'_{x^{(i)}} = \Sigma_x$.

Constraints. For every constraint $\varphi = (x_1 x_2, C) \in \Phi$ with $x_1 \in X_1$ and $x_2 \in X_2$, $i_1 \in [k_2]$, and $i_2 \in [k_1]$ we have a constraint

$$\varphi^{i_1, i_2} = (x_1^{(i_1)} x_2^{(i_2)}, C) \in \Phi'.$$

That is, φ^{i_1, i_2} is a copy of φ where the variable x_1 is replaced by its i_1 -th copy $x_1^{(i_1)}$ and x_2 by its i_2 -th copy $x_2^{(i_2)}$. It immediately implies that if Π is rectangular, then Π' is rectangular too.

Again the completeness is immediate. Towards the soundness, let $\hat{\sigma}' : X' \rightarrow 2^{\Sigma'}$ be a satisfying r -average multi-assignment for Π' . In particular,

$$r = \frac{\sum_{x \in X'} |\hat{\sigma}'(x)|}{|X'|} = \frac{\sum_{x \in X'_1} |\hat{\sigma}'(x)| + \sum_{x \in X'_2} |\hat{\sigma}'(x)|}{|X'_1| + |X'_2|} = \frac{\sum_{x \in X'_1} |\hat{\sigma}'(x)| + \sum_{x \in X'_2} |\hat{\sigma}'(x)|}{2k_1 k_2}.$$

We set

$$r_1 = \frac{\sum_{x \in X'_1} |\hat{\sigma}'(x)|}{|X'_1|} = \frac{\sum_{x \in X'_1} |\hat{\sigma}'(x)|}{k_1 k_2} \quad \text{and} \quad r_2 = \frac{\sum_{x \in X'_2} |\hat{\sigma}'(x)|}{|X'_2|} = \frac{\sum_{x \in X'_2} |\hat{\sigma}'(x)|}{k_1 k_2} \quad (10)$$

It follows that

$$r_1 + r_2 = \frac{\sum_{x \in X'_1} |\hat{\sigma}'(x)| + \sum_{x \in X'_2} |\hat{\sigma}'(x)|}{k_1 k_2} = 2r.$$

65:14 On Average Baby PIH and Its Applications

Note that

$$X'_1 = \bigcup_{i \in [k_2]} \{x^{(i)} \mid x \in X_1\}. \quad (11)$$

Therefore,

$$\begin{aligned} r_1 k_1 k_2 &= r_1 |X'_1| && \text{(by } |X'_1| = k_1 k_2) \\ &= \sum_{x \in X'_1} |\hat{\sigma}'(x)| && \text{(by (10))} \\ &= \sum_{i \in [k_2]} \sum_{x \in X_1} |\hat{\sigma}'(x^{(i)})|. && \text{(by (11))} \end{aligned}$$

Hence, there exists an $i_1 \in [k_2]$ such that

$$\sum_{x \in X_1} |\hat{\sigma}'(x^{(i_1)})| \leq r_1 k_1, \quad \text{or equivalently} \quad \frac{\sum_{x \in X_1} |\hat{\sigma}'(x^{(i_1)})|}{|X_1|} \leq r_1$$

by $|X_1| = k_1$. Arguing similarly for X_2 we get an $i_2 \in [k_1]$ such that

$$\frac{\sum_{x \in X_2} |\hat{\sigma}'(x^{(i_2)})|}{|X_2|} \leq r_2$$

Finally we define a multi-assignment $\hat{\sigma}$ for the original instance Π by

$$\hat{\sigma}(x) = \begin{cases} \hat{\sigma}'(x^{(i_1)}) & \text{if } x \in X_1 \\ \hat{\sigma}'(x^{(i_2)}) & \text{if } x \in X_2. \end{cases}$$

By the above argument, $\hat{\sigma}$ is (r_1, r_2) -average. Moreover, it satisfies Π , since $\hat{\sigma}'$ satisfies Π' . ◀

Putting all pieces together, we have Theorem 16.

Proof of Theorem 16. We give an FPT reduction from instances in the Baby PIH (Theorem 13) to $\text{AVG-}r\text{-GAP-2CSP}$. Then, since the Baby PIH holds under $\text{W}[1] \neq \text{FPT}$, we deduce that the Average Baby PIH also holds under $\text{W}[1] \neq \text{FPT}$.

For any 2CSP instance $\Pi_0 = (X_0, \Sigma_0, \Phi_0)$, we can construct a bipartite 2CSP instance $\Pi_1 = (X_1, \Sigma_1, \Phi_1)$ by Lemma 20, and then construct an $\text{AVG-}r\text{-GAP-2CSP}$ instance $\Pi = (X, \Sigma, \Phi)$ from Π_1 by Lemma 22. Trivially, Π is satisfiable when Π_0 is satisfiable. When Π_0 is not r -list satisfiable, Π_1 is not (r_1, r_2) -average list satisfiable for all constants r_1, r_2 with $r_1 + r_2 \geq 2(1 - \varepsilon)r$, and thus Π is not $(1 - \varepsilon)r$ -average list satisfiable. Furthermore, Π has rectangular relations because Π_1 has rectangular relations.

Moreover, the running time of this reduction can be bounded by

$$f(|X_0| + |\Phi_0| + 1/\varepsilon + r)|\Sigma_0|^{O(1)}$$

for a computable function f , and

$$|X| + |\Phi| \leq f(|X_0| + |\Phi_0| + 1/\varepsilon + r)|\Sigma_0|^{O(1)}$$

as well, so the reduction is an FPT reduction. ◀

3.2 Average Baby PIH on Dense and Sparse Instances

In this section we prove Theorem 2, which is divided into two separate lemmas to help with readability. For our purposes, a 2CSP instance $\Pi = (X, \Sigma, \Phi)$ is *dense* if $|\Phi| = \omega(|X|)$; or it is *sparse*, if $|\Phi| = O(|X|)$.

► **Lemma 23.** *Under $W[1] \neq \text{FPT}$, the Average Baby PIH holds for all AVG- r -GAP-2CSP instances that are dense.*

Proof. The reduction in the proof of Lemma 20 yields *complete* bipartite 2CSP instances $\Pi_1 = (X_1 \dot{\cup} X_2, \Sigma_1, \Phi_1)$, i.e., for each $x_1 \in X_1$ and $x_2 \in X_2$, there exists a constraint $\varphi = (x_1 x_2, C) \in \Phi$. Then the reduction in the proof of Lemma 22 makes $|X_2|$ copies of X_1 and $|X_1|$ copies of $|X_2|$, while keeping the constraints in each pair of copies. So in the final instance $\Pi = (X, \Sigma, \Phi)$ from the proof of Theorem 16, the number of constraints is

$$|\Phi| = |X_1|^2 |X_2|^2 = \frac{|X|^2}{4}.$$

Now consider any function $h \in \omega(1)$. We produce a new instance $\Pi' = (X', \Sigma', \Phi')$ by simply copying Π for t times, where t is chosen as the minimum number satisfying

$$h(t|X|) \geq \frac{|X|}{4}.$$

Note that there is no constraint between different copies. Then, the new parameter is $|X'| = t|X|$, and

$$|\Phi'| = t|\Phi| = \frac{t|X|^2}{4} = \frac{|X|}{4}|X'| \leq h(|X'|)|X'|.$$

It's clear that this reduction runs in FPT time. Also, if Π is satisfiable, then Π' is satisfiable. If any satisfying multi-assignment to Π must have total size more than $r|X|$, then any satisfying multi-assignment to Π' must assign each copy of Π more than $r|X|$ values, so in total more than $r|X'|$ values, preserving the gap. ◀

► **Lemma 24.** *Let $r > 1$. If there exists a constant $c > 0$ such that no FPT algorithm can solve AVG- r -GAP-2CSP on instance $\Pi = (X, \Sigma, \Phi)$ with $|\Phi| \leq c \cdot r|X|$, i.e., Π is sparse, then the PIH holds.*

Proof Sketch. Let $\Pi = (X, \Sigma, \Phi)$ be a NO instance of AVG- r -GAP-2CSP. For any (standard) assignment $\sigma : X \rightarrow \Sigma$, assume that σ violates t constraints, then one can simply add at most $2t$ values to σ and obtain a satisfying multi-assignment $\hat{\sigma}$ with total size $|X| + 2t$. Since Π is a NO instance, we have $|X| + 2t > r|X|$. Thus,

$$t > \frac{r-1}{2}|X| = \frac{r-1}{2c \cdot r} \cdot c \cdot r|X| \geq \frac{r-1}{2c \cdot r}|\Phi|.$$

In other words, any assignment to Π must violate a constant fraction of the constraints in Π . This gives a reduction from AVG- r -GAP-2CSP to PIH. ◀

Putting Lemma 23 and Lemma 24 together, we obtain Theorem 2. As already mentioned in the introduction, this result indicates that the current barrier to the $W[1]$ -hardness of the PIH is the lack of reduction for AVG- r -GAP-2CSP on sparse instances, i.e., instances with linearly many constraints.

References

- 1 Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k -restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006. doi:10.1145/1150334.1150336.
- 2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.
- 3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.
- 4 Libor Barto and Marcin Kozik. Combinatorial gap theorem and reductions between promise CSPs. 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 1204–1220. SIAM, 2022. doi:10.1137/1.9781611977073.50.
- 5 Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. Simple combinatorial construction of the $k^{o(1)}$ -lower bound for approximating the parameterized k -Clique. In *2025 Symposium on Simplicity in Algorithms (SOSA)*, pages 263–280. Society for Industrial and Applied Mathematics, 2025. doi:10.1137/1.9781611978315.21.
- 6 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized Dominating Set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. doi:10.1137/17M1127211.
- 7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 8 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.
- 9 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.
- 10 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.
- 11 Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Almost optimal time lower bound for approximating parameterized Clique, CSP, and more, under ETH. *CoRR*, abs/2404.08870, 2024. doi:10.48550/arXiv.2404.08870.
- 12 Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized Inapproximability Hypothesis under Exponential Time Hypothesis. In Bojan Mohar, Igor Shinkar, and Ryan O’Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 24–35. ACM, 2024. doi:10.1145/3618260.3649771.
- 13 Venkatesan Guruswami, Xuandi Ren, and Sai Sandeep. Baby PIH: parameterized inapproximability of min CSP. In Rahul Santhanam, editor, *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPICs*, pages 27:1–27:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.CCC.2024.27.
- 14 Karthik C. S. and Subhash Khot. Almost polynomial factor inapproximability for parameterized k -Clique. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPICs*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CCC.2022.6.
- 15 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating Dominating Set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.
- 16 Karthik C. S. and Inbal Livni Navon. On hardness of approximation of parameterized Set Cover and Label Cover: Threshold graphs from error correcting codes. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 210–223. SIAM, 2021. doi:10.1137/1.9781611976496.24.

- 17 Shuangle Li, Bingkai Lin, and Yuwei Liu. Improved lower bounds for approximating parameterized nearest codeword and related problems under ETH. 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 107:1–107:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICs.ICALP.2024.107.
- 18 Bingkai Lin. A simple gap-producing reduction for the parameterized Set Cover problem. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPICs*, pages 81:1–81:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.81.
- 19 Bingkai Lin. Constant approximating k -Clique is W[1]-hard. 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 1749–1756. ACM, 2021. doi:10.1145/3406325.3451016.
- 20 Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Constant approximating parameterized k -SetCover is W[2]-hard. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3305–3316. SIAM, 2023. doi:10.1137/1.9781611977554.CH126.
- 21 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of Directed Odd Cycle Transversal. 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 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.
- 22 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994. doi:10.1145/185675.306789.
- 23 Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k -Coverage, Unique Set Cover and related problems (via t -wise agreement testing theorem). In 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 62–81. SIAM, 2020. doi:10.1137/1.9781611975994.5.

A From Average Baby PIH to Inapproximability of k -ExactCover

We present a proof relying on a construction that slightly differs from the one in [13]. Their proof makes use of the (T, m) -set gadget [22, 1] that was previously used to show the hardness of approximating SETCOVER problem. On the other hand, our proof develops a novel composition of the collision number of ECCs (recall Definition 17) with the following well-known combinatorial object.

► **Definition 25** (Hypercube Partition System). *Let A, B be two sets. Then the (A, B) -hypercube partition system is defined by*

- *the universe $\mathcal{M} = A^B$ ($= \{z \mid \text{a function } z : B \rightarrow A\}$), and*
- *a collection of subsets $\{P_{x,y}\}_{x \in B, y \in A}$ where each $P_{x,y} = \{z \in \mathcal{M} \mid z(x) = y\}$.*

► **Theorem 26** (cf. Theorem 21 in [13]). *Assume that the Average Baby PIH holds on all 2CSP instances with rectangular relations. Then k -EXACTCOVER cannot be approximated in FPT time within any constant factor. More precisely, for every constant $r > 1$ no FPT algorithm, on a given k -SETCOVER instance $\Pi = (S, U)$ with size n and $k \geq 1$, can distinguish between the following two cases:*

- *We can choose k disjoint sets in S whose union is U .*
- *U is not the union of any rk sets in S .*

65:18 On Average Baby PIH and Its Applications

Proof. Let $\Pi = (X, \Sigma, \Phi)$ be an AVG- r -GAP-2CSP instance with rectangular relations. We set $k = |X|$. Moreover, for each rectangular constraint $\varphi_j = (x_{i_1} x_{i_2}, C_j) \in \Phi$ we use Q_j to denote the underlying set and $\pi_j, \rho_j : \Sigma \rightarrow Q_j$ the associated mappings as in Definition 14. That is, for every $a, b \in \Sigma$, it holds that $(a, b) \in C_j$ if and only if $\pi_j(a) = \rho_j(b)$. Then we set

$$t = \max_{\varphi_j \in \Phi} |Q_j|. \quad (12)$$

Clearly, we can assume without loss of generality

$$t \leq |\Pi|.$$

Now we reduce Π to a k -EXACTCOVER instance. To that end, we choose a further alphabet Δ whose size is a prime and satisfies

$$\max \left\{ \lceil \log t \rceil, 2^{2r^2 k^2} \right\} \leq |\Delta| \leq 2 \max \left\{ \lceil \log t \rceil, 2^{2r^2 k^2} \right\}.$$

Moreover, let

$$d = \left\lceil \frac{2r^2 k^2 \log t}{\log |\Delta|} \right\rceil.$$

This leads to the following code with very large distance (here we simply use Reed-Solomon code again)

$$\text{Enc} : \Delta^{\left\lceil \frac{\log t}{\log |\Delta|} \right\rceil} \rightarrow \Delta^d.$$

Plugging

$$k \leftarrow \left\lfloor \frac{\log t}{\log |\Delta|} \right\rfloor, \quad m \leftarrow d, \quad p \leftarrow |\Delta|,^6 \quad \text{and} \quad \varepsilon \leftarrow 1/2$$

in Theorem 18 we conclude that the $1/2$ -collision number of Enc is

$$\text{Col}_{1/2}(\text{Enc}) \geq \sqrt{\frac{d}{\log t / \log |\Delta|}} > rk.$$

Observe that (12) implies that every tuple in C_i can be identified with a string in $\Delta^{\left\lceil \frac{\log m}{\log |\Delta|} \right\rceil}$, i.e., the domain of Enc.

Then, for each variable $x \in X$ and every its possible value $a \in \Sigma$, we define a set $S_{x,a}$ as follows. For each constraint $\varphi_j = (x_{i_1} x_{i_2}, C_j) \in \Phi$ with associated set T_j and mappings $\pi_j, \rho_j : \Sigma \rightarrow Q_j$, and for each $\ell \in [d]$, we construct a $([2], \Delta)$ -hypercube partition system

$$\left(\mathcal{M}^{(j,\ell)}, \{P_{u,v}^{(j,\ell)}\}_{u \in \Delta, v \in [2]} \right). \quad (13)$$

⁶ Observe that

$$\left\lfloor \frac{\log t}{\log |\Delta|} \right\rfloor < \left\lceil \frac{2r^2 k^2 \log t}{\log |\Delta|} \right\rceil \leq \lceil \log t \rceil \leq |\Delta|,$$

hence, the condition $k < m \leq p$ in Theorem 18 is satisfied.

Then for each $(a, b) \in C_j$ we add $P_{\text{Enc}(\pi_j(a))[\ell],1}^{(j,\ell)}$ to $S_{x_{i_1},a}$ and similarly $P_{\text{Enc}(\rho_j(b))[\ell],2}^{(j,\ell)}$ to $S_{x_{i_2},b}$. Finally, let the universe be

$$U = \bigcup_{\varphi_j \in \Phi, \ell \in [d]} \mathcal{M}^{(j,\ell)}, \quad \text{and} \quad \mathcal{S} = \{S_{x,a} \mid x \in X \text{ and } a \in \Sigma\}.$$

For the completeness, let $\sigma : X \rightarrow \Sigma$ be a satisfying assignment of Π , it is routine to check that $\{S_{x,\sigma(x)}\}_{x \in X}$ is a partition of U .

For the soundness, assume that every satisfying multi-assignment of Π has total size at least rk (cf. Definition 7). Let $\mathcal{S}' \subseteq \mathcal{S}$ be a cover of U . Consider the multi-assignment that maps every variable $x \in X$ to $\{a \in \Sigma \mid S_{x,a} \in \mathcal{S}'\}$. If this multi-assignment satisfies Π , then our assumption implies $|\mathcal{S}'| \geq rk$. Otherwise, assume that there exists some constraint $\varphi_j = (x_{i_1}x_{i_2}, C_j) \in \Phi$ which is not satisfied. Note that the above multi-assignment assigns x_{i_1} to $E_1 = \{a \in \Sigma \mid S_{x_{i_1},a} \in \mathcal{S}'\}$ and x_{i_2} to $E_2 = \{b \in \Sigma \mid S_{x_{i_2},b} \in \mathcal{S}'\}$. Since φ_j is not satisfied, for all $(a, b) \in E_1 \times E_2$ we have $\text{Enc}(\pi_j(a)) \neq \text{Enc}(\rho_j(b))$. However, for each $\ell \in [d]$, since $\mathcal{M}^{(j,\ell)}$ is covered by \mathcal{S}' , there must exist $a \in E_1$ and $b \in E_2$ with $\text{Enc}(\pi_j(a))[\ell] = \text{Enc}(\rho_j(b))[\ell]$. Therefore, the set $\{\pi_j(a)\}_{a \in E_1} \cup \{\rho_j(b)\}_{b \in E_2}$ collides on all coordinates $\ell \in [d]$, hence it must have size at least $\text{Col}_{1/2}(\text{Enc})$. We deduce

$$|\mathcal{S}'| \geq |E_1| + |E_2| \geq |\{\pi_j(a)\}_{a \in E_1} \cup \{\rho_j(b)\}_{b \in E_2}| \geq \text{Col}_{1/2}(\text{Enc}) > rk.$$

Finally, in each hypercube partition system (13) it holds that

$$|\mathcal{M}^{(j,\ell)}| = 2^{|\Delta|} \leq 4^{\lceil \log t \rceil} + 4^{2^{2r^2k^2}} \leq |\Pi|^2 + 4^{2^{2r^2k^2}},$$

and there are at most $\binom{k}{2}d \leq k^2r^2k^2 \log t \leq r^2k^4 \log |\Pi|$ such systems. The size of the universe U is thus at most $g(r, k)|\Pi|^3$ for some appropriate computable function $g : \mathbb{N}^2 \rightarrow \mathbb{N}$, while the parameter of the k -EXACTCOVER instance remains $k = |X|$. It follows easily that the running time of this reduction is FPT. \blacktriangleleft

Combining Theorem 26 and Theorem 16, we obtain:

\blacktriangleright **Theorem 27.** *For any constant $r > 1$, r -approximating k -EXACTCOVER is $W[1]$ -hard.*

The Hardness of Decision Tree Complexity

Bruno Loff  

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Alexey Milovanov  

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract

Let f be a Boolean function given as either a truth table or a circuit. How difficult is it to find the decision tree complexity, also known as deterministic query complexity, of f in both cases? We prove that this problem is NC^1 -hard and PSPACE-hard, respectively. The second bound is tight, and the first bound is close to being tight.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Decision tree, Log-depth circuits

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.66

Funding This work was funded by the European Union (ERC, HOFGA, 101041696). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. It was also supported by FCT through the LASIGE Research Unit, ref. UIDB/00408/2025 and ref. UIDP/00408/2025, and by CMAFcIO, FCT Project UIDB/04561/2020, <https://doi.org/10.54499/UIDB/04561/2020>.

Acknowledgements The authors would like to thank Wei Zhan for his answer at StackOverflow.

1 Introduction

The decision tree is one of the most important computational models in theoretical computer science. Decision trees were invented in the 50s with the purpose of analyzing data. In this context, at each node in the tree we *query* some feature of the data, which partitions the data points depending on the value of this chosen feature. The resulting partition at the leaves should allow us to better understand the data. Starting in the 1980s, several learning algorithms were developed that would process data and produce a *classifier*. Meaning, we assume the existence of some function f of which we know some sampled pairs $(x, f(x))$ (the data), and we wish to produce a decision tree that would be able to predict $f(x)$, even on a previously-unseen input x (some famous algorithms are CHAID, CART, ID3, and C4.5, see ([9, 4, 14, 15])). The goal here is to produce the smallest possible decision tree, while making the fewest possible mistakes. This task, of *decision-tree synthesis*, is used in applications (e.g. [19]).

But one can also consider a more *algorithmic problem*, which is of theoretical interest. Here, the function f is completely known (i.e., we know $(x, f(x))$ for all x), and we wish to produce a decision tree which computes f (e.g., without any mistakes). As far as we are aware it was in the 1970s (e.g. [20]) when people first studied, for specific functions f , and for specific ways of querying the input x , how small can be the depth of a decision tree that computes $f(x)$ when given x as input. In the meantime, decision trees have become a ubiquitous computational model, useful in the study of various kinds of computation. To give a few examples, decision trees are relevant to the study of data structures (the cell-probe model [13, 11]), cryptographic reductions (in black-box reductions [17]), and communication



© Bruno Loff and Alexey Milovanov;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 66; pp. 66:1–66:13



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



complexity¹. There is a meta-complexity problem underlying this algorithmic problem: how does one determine the decision-tree complexity of a given function f ? This question has been studied before, usually for the learning problem [10], but also for the algorithmic problem [1, 18].

Let us restrict our attention to the simplest of all decision-tree models, where the known function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a Boolean function, and the computational model is deterministic decision tree that must compute $f(x)$ by querying the bits of x . Here, we can consider two scenarios, with respect to how f is given:

(tt-DT) We are given f as a truth table, meaning a binary string of length $N = 2^n$ so that $f(x)$ appears at the x -th position.

(circuit-DT) We are given f as a Boolean circuit, which potentially allows for a more succinct encoding of f .

The meta-complexity problem is: we are given f as input, either as a truth-table (tt-DT) or as a circuit (circuit-DT) and we wish to find the deterministic *query complexity* complexity of f , namely, the smallest depth of a deterministic decision tree that computes $f(x)$ by querying the bits of x . How hard is this problem? In this paper, we give a satisfactory answer for both scenarios, essentially via the same technique.

It is not difficult to see that circuit-DT belongs to PSPACE: see Proposition 7 below. One could immediately conjecture that the problem is PSPACE-hard, but one soon comes across various difficulties in proving such a statement. (After the required definitions are in place, in Section 3.2, we discuss precisely where the difficulty lies.) Our first main result, Theorem 10, is showing that this problem is indeed PSPACE-hard.

It is also well-known, and appears as Proposition 6 below, that tt-DT belongs to P. Meaning, denoting the input length for tt-DT as $N = 2^n$, which is the length of the truth-table of a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, Proposition 6 states that tt-DT can be solved in $\text{poly}(N)$ time. In Proposition 9, we observe that tt-DT can also be computed in parallel, namely, by a Boolean circuit of depth $O(\log N \log \log N)$ and size $\text{poly}(N)$. This result is simple enough that it should be considered folklore: easy to prove for anyone who would care to do so. However, despite being a natural and fundamental problem, no matching lower-bound was known. Our second main result, Theorem 11, shows that the above bound is close to tight: tt-DT is NC^1 -hard (under uniform NC^0 reductions).

2 Definitions and upper bounds

► **Definition 1.** We let $\{0, 1\}^n$ denote the set of all binary strings of length n , sometimes called the Boolean cube. We let $\{0, 1, *\}^n$ denote the set of $(n\text{-bit})$ partial assignments. The elements $\rho \in \{0, 1, *\}^n$ are in bijection with the Boolean subcubes:

$$\{x \in \{0, 1\}^n \mid \forall i \in [n] \ \rho_i \neq * \implies x_i = \rho_i\}.$$

We will denote this set also by ρ , by abuse of notation.

We let $[i \leftarrow b] = *^{i-1}b*^{n-i-1}$ assign b to the i -th coordinate. Two partial assignment ρ and ρ' are called compatible if $\rho_i \neq *$ and $\rho'_i \neq *$ implies $\rho_i = \rho'_i$.

If $\rho, \rho' \in \{0, 1, *\}^n$ are compatible, then $\rho \cdot \rho'$ is the partial assignment such that $(\rho \cdot \rho')_i$ equals to ρ_i if $\rho_i \neq *$, equals ρ'_i if $\rho'_i \neq *$, and equals $*$ if $\rho_i = \rho'_i = *$.

¹ In lifting theorems. Some examples of lifting theorems that use *deterministic* decision trees are: [16, 7, 6, 5, 12].

If $|\rho^{-1}(*)| = \ell$ and $y \in \{0, 1\}^\ell$, we let $\rho(y) \in \{0, 1\}^n$ be the binary string which equals ρ where $\rho_i \neq *$, and equals y in the remaining coordinates (which are filled in order).

Given a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and a partial assignment $\rho \in \{0, 1, *\}^n$ with $\ell = |\rho^{-1}(*)|$ the number of $*$, we let $f|_\rho : \{0, 1\}^\ell \rightarrow \{0, 1\}$ be given by

$$f|_\rho(y) = f(\rho(y)).$$

I.e. it is the restriction of f to the Boolean subcube ρ .

► **Definition 2.** A deterministic decision tree over $\{0, 1\}^n$ is a rooted, labelled, ordered binary tree T :

- Each non-leaf node v is labelled by an index $i_v \in [n]$.
- Each non-leaf node v has two children v_0 and v_1 .

Associated with each node v of T is a partial assignment ρ_v

- The root is associated with $*^n$.
- For a non-leaf node v , and for both $b \in \{0, 1\}$, $\rho_{v_b} = \rho_v \cdot [x_{i_v} = b]$.

► **Definition 3.** The computation of T on input $x \in \{0, 1\}^n$ is a path in T , which begins at the root, and proceeds at each node v by going to the child $v_{x_{i_v}}$, until it reaches a leaf.

It is easy to see that ρ_v is the set of inputs whose computation goes through v . We have $(\rho_v)_i = *$ if and only if the coordinate x_i has not yet been queried in the computation between the root and node v , i.e., at or before v . If x_i has been queried at or before v , then $(\rho_v)_i = x_i$ for every x whose computation goes through v .

► **Definition 4.** We say that T computes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ if, for every leaf v of T , f is constant on ρ_v . The deterministic query complexity of a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, which will be denoted $D(f)$, is the smallest depth of any decision tree that computes f .

► **Definition 5.** We let tt-DT be the computational problem where we are given the truth-table of a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and wish to compute $D(f)$. We let circuit-DT be the computational problem where we are given a Boolean circuit C computing a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and we wish to compute $D(f)$.

The simplest observation that one can make is that tt-DT has a polynomial-time algorithm.

► **Proposition 6** ([8, 1]). *tt-DT belongs to P. More precisely, there is an algorithm that computes the DT-complexity of an n -ary Boolean function in time $O(3^n \cdot n) = O(N^{1.585\dots} \log N)$, where $N = 2^n$.*

This algorithm should be considered folklore, but we include it here because it is simple and insightful.

Proof. The main, crucial observation is that the best decision tree for f must first choose a coordinate i , query x_i , and then run the best decision tree for $f|_{[i \leftarrow x_i]}$. In general, for any partial assignment $\rho \in \{0, 1, *\}^n$, $D(f|_\rho) = 0$ if f is constant on ρ , and otherwise

$$D(f|_\rho) = \min_{i \in \rho^{-1}(*)} \{1 + \max_{b \in \{0, 1\}} D(f|_{\rho \cdot [i \leftarrow b]})\}. \quad (*)$$

This gives us a dynamic programming algorithm: knowing $D(f|_\rho)$ for all partial assignments $\rho \in \{0, 1, *\}^n$ with $|\rho^{-1}(*)| = \ell$ free variables, we can use the above formula to compute $D(f|_\rho)$ for all $\rho \in \{0, 1, *\}^n$ with $|\rho^{-1}(*)| = \ell + 1$ free variables. Finally $f = f|_{*^n}$, so we learned $D(f)$. There are 3^n partial assignments in total, and each computation $D(f|_\rho)$ takes time $O(n)$ in a random-access machine. ◀

Some more insight will come from the following *game reformulation* of the statement “ $D(f) \leq k$ ”. Consider the following game between two players, Alice and Bob. The game lasts for k steps. At every step, Alice chooses a variable x_i , and Bob sets a Boolean value to the corresponding variable, either $x_i = 0$ or $x_i = 1$. After k steps, Alice wins if $f|_\rho$ is constant on the partial assignment ρ corresponding to Alice and Bob’s moves; otherwise, Bob wins. It follows that Alice has a winning strategy in this game if and only if $D(f) \leq k$. Indeed, if $D(f) \leq k$, then Alice can make moves according to the corresponding tree, and if $D(f) > k$, then Bob wins because this inequality means that for every i , the decision tree complexity of $f|_{[i \leftarrow 1]}$ or $f|_{[i \leftarrow 0]}$ is at least k . Bob’s strategy is then to repeatedly choose the value $b \in \{0, 1\}$ that maximizes $D(f|_{[i \leftarrow b]})$. One can algorithmically find the winner in this game by a simple recursive algorithm. It is easy to see that, if a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is given as a Boolean circuit C , an algorithm can decide which of the two players has a winning strategy, using $\text{poly}(n, |C|)$ memory. So, we get the following:

► **Proposition 7.** *circuit-DT belongs to PSPACE.*

One may now ask whether Proposition 6 can be at all improved. Indeed, the algorithm can be parallelized. First, a definition:

► **Definition 8.** *For $i \in \mathbb{N}$, we let NC^i denote the class of functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ computable by Boolean circuits with binary AND and OR gates, and unary NOT gates, in depth $O((\log n)^i)$ and size $\text{poly}(n, m)$. We let $\widetilde{\text{NC}}^1$ denote the class of functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ computable by such circuits in depth $O(\log n \cdot \log \log n)$ and size $\text{poly}(n, m)$.*

We now claim the following.

► **Proposition 9.** *tt-DT can be computed by a Boolean circuit of size $O(3^n \cdot \text{poly}(n)) = O(N^{1.583\dots} \text{polylog } N)$ and depth $O(n \log n) = O(\log N \log \log N)$. Hence, tt-DT is in $\widetilde{\text{NC}}^1$.*

Proof. The equation (*) directly gives us a circuit that uses $O(3^n)$ min, max, increment, and all-equal gates: for each partial assignment ρ we check if $f|_\rho$ is constant using an all-equal gate, otherwise we compute the formula given by (*). This circuit has depth $O(n)$ using such gates. Implementing such gates using Boolean gates will result in a circuit of depth $O(n \log n)$. ◀

3 Lower bounds

Our two main theorems are the following.

► **Theorem 10.** *circuit-DT is PSPACE-hard under polynomial-time reductions.*

► **Theorem 11.** *tt-DT is NC^1 -hard under NC^0 -reduction.*

The first theorem is well understood, but the second requires some clarification regarding reductions and uniformity. Recall the definition of DLOGTIME-uniform circuits from [3].

► **Definition 12.** *Let $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ be a family of Boolean circuits, so that C_n computes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$.*

- *The “direct connection language” of \mathcal{C} is the set of tuples $\langle 0^n, t, a, b \rangle$, where $a \in [|C_n|]$ and $b \in \{0\} \cup [|C_n|]$ are gate numbers in C_n , t is the type of gate a (AND, OR, NOT, or input gate x_i), and gate b is a child of gate a (and equals 0 if a is a leaf).*
- *The circuit family \mathcal{C} is DLOGTIME-uniform if its direct connection language can be recognized in $O(\log(n))$ time by a deterministic multi-tape Turing machine with an index tape for random access to the input.*

We say that language $A \subseteq \{0, 1\}^*$ is NC^0 -reduced to language $B \subseteq \{0, 1\}^*$, which we write $A \leq_{\text{NC}^0} B$, if there is a DLOGTIME -uniform family of NC^0 -circuits $\mathcal{C} = \{C_n\}$ such that, for every $x \in \{0, 1\}^n$, $x \in A$ iff $C_n(x) \in B$.

It is not difficult to see that this type of reduction satisfies the natural properties like transitivity and *closure*: If $A \leq_{\text{NC}^0} B$ and $B \in \text{NC}^1$ then $A \in \text{NC}^1$ (this holds for both uniform and non-uniform variants of NC^1).

3.1 TQBF

The proof of theorems 10 and 11 are similar: we prove that TQBF reduces to DT. Recall that TQBF (True Quantified Boolean Formula) is the problem of determining the truth of a formula

$$\exists y_1 \forall x_1 \exists y_2 \forall x_2 \dots \exists y_n \forall x_n h(y_1, x_1, y_2, x_2, \dots, y_n, x_n), \quad (\dagger)$$

where h is some Boolean function. As we did for decision-tree complexity, let us consider two variants of the TQBF problem:

tt-TQBF We are given as input a Boolean function $h : \{0, 1\}^{2n} \rightarrow \{0, 1\}$ as a truth table, and wish to know whether (\dagger) holds.

circuit-TQBF We are given h as a circuit, and wish to know whether (\dagger) holds.

It is well-known that circuit-TQBF is PSPACE-complete. It turns out that tt-TQBF is NC^1 -complete:

► **Theorem 13.** *tt-TQBF is NC^1 -complete under \leq_{NC^0} reductions.*

To prove Theorem 13 we use the following result of Barrington.

► **Theorem 14** ([2, 3]). *The S_5 identity problem, $S_5\text{IP}$, is the problem of deciding if the product of given permutations from S_5 is equal to the identity. Then $S_5\text{IP}$ is NC^1 -complete under \leq_{NC^0} reductions.*

This theorem was proved in [2], and in [3] the authors verified that the reasoning proves the desired statement with DLOGTIME -uniform NC^0 reductions.

Proof of Theorem 13. It is easy to see that tt-TQBF is in NC^1 : the formula (\dagger) is a Boolean formula of depth n whose leaves are entries in the truth-table of h . To prove that tt-TQBF is NC^1 -hard, the idea is to consider $S_5\text{IP}$ as a game that can be interpreted as a TQBF formula.

Consider the input of $S_5\text{IP}$: permutations π_1, \dots, π_N , with $N = 2^n$. Imagine two players Alice and Bob; Alice wants to prove that the product is equal to the identity permutation and Bob does not trust her.

They play in the following game. Bob asks: what is the product of the first half of the permutations, i.e. $\pi_1 \cdot \dots \cdot \pi_{\lfloor \frac{N}{2} \rfloor}$. Alice states that this product is some permutation σ_1 . Additionally, since Alice is trying to prove to Bob that all the product of all permutations is equal to the identity, she is also implicitly stating that the product $\pi_{\lfloor \frac{N}{2} \rfloor + 1} \cdot \dots \cdot \pi_N$ is equal to σ_1^{-1} . Bob does not trust Alice, and so he chooses $b_1 \in \{0, 1\}$ to mean that he believes one of Alice's statements to be false. I.e. he chooses $b_1 = 0$ if he believes that the first part is actually not σ_1 , and he sets $b_1 = 1$ if he believes that the second part is not σ_1^{-1} . After this, a similar procedure repeats: Alice states that the value of the product of the first half of Bob's chosen part is σ_2 (this is one quarter of all permutations), which implies that the second half of Bob's chosen part is $\sigma_2^{-1} \sigma_1^{(-1)^{b_1}}$. Then Bob chooses one of these two quarters $b_2 \in \{0, 1\}$ where he believes Alice's statement is false, and so on.

This game produces a sequence $\sigma_1, b_1, \dots, \sigma_n, b_n$. At the end of the game, the winner is identified by whether $\alpha = \pi_i$ or not, where α and i are inferred from the sequence. This game can be interpreted as a TQBF: each statement σ_i of Alice can be encoded as 7 bits (since $5! < 2^7$), and each choice b_i of Bob can be encoded as 1 bit. Every value of the truth table of h for tt-TQBF can be constructed from Alice and Bob's moves and the corresponding value of some π_i .

Indeed, $h(\sigma_1, b_1, \dots, \sigma_n, b_n) = 1$ if and only if $\alpha = \pi_i$ for the appropriate $\alpha \in S_5$ and $i \in [n]$. Hence, this reduction is in NC^0 , since every value in the truth-table of h (i.e. the winner in the corresponding game) depends only on one permutation π_i of the input, which is encoded using a constant number of bits.

We further claim that the corresponding NC^0 reduction can be made DLOGTIME uniform. By logarithmic time we mean $O(\log N) = O(n)$. We first describe an algorithm which, when given $\sigma_1, b_1, \dots, \sigma_n, b_n$ (i.e. the moves in Alice-Bob game), outputs the index i and the permutation α required to compute the bit $h(\sigma_1, b_1, \dots, \sigma_n, b_n) = [\alpha = \pi_i?]$ of h 's truth table.

The algorithm looks at every pair of moves $(\sigma_1, b_1), \dots, (\sigma_n, b_n)$ one time and maintains in memory a permutation α , an index $k \in [n]$, and two indexes $1 \leq s \leq t \leq N$. These values in memory have the following meaning: after round k , Alice has stated that $\pi_s \dots \pi_t = \alpha$. Initially α is the identity, $k = 0$, $i = 1$, and $j = N$.

The move of Alice in the k -th round is some permutation $\sigma_k \in S_5$, and Alice states that $\pi_s \dots \pi_m = \sigma_k$, where $m = \frac{s+t}{2}$. The move of Bob (some bit b_k) is a choice "left" or "right". If Bob's answer is "left", then we set $s := s$, $t := m$, $\alpha := \sigma$ and if Bob's choice is "right" then we set $s := m + 1$, $t := t$, and $\alpha := \sigma_k^{-1} \cdot \alpha$. Each such calculation can be done in constant time, so the entire computation can be done in linear time.

The above circuit is very simple, and the above algorithm shows how to compute the connections between the various gates. Since this algorithm runs in $O(\log N)$ time, this circuit is DLOGTIME-uniform. ◀

3.2 The crucial difficulty: TQBF vs DT

We have now laid out enough definitions that we can discuss the crucial difficulty in proving our main result (Theorems 10 and 11). We would like to prove that circuit-DT is PSPACE-hard, and we know that circuit-TQBF is PSPACE-hard. We would like to prove that tt-DT is NC^1 -hard, and we know that tt-TQBF is NC^1 -hard. The fundamental difference between the two problems can be understood by looking them as a game.

In TQBF, we have two disjoint sets of variables: Alice sets the y_i variables and Bob sets the x_i variables. In DT, we have a single set of variables: Alice chooses a variable, and Bob sets the variable. Now, what we would like to do, is to simulate the TQBF game using the DT game. The difficulty is that it is not at all obvious how such a simulation should proceed.

To simulate a TQBF game over variables $y_1, x_1, \dots, y_n, x_n$, we use a DT game over variables $y_1, y'_1, x_1, x'_1, \dots, y_n, y'_n, x_n, x'_n$, and some other auxiliary variables. The hope is that the DT instance works in such a way that Alice must play by choosing first y_i and then y'_i , or first y'_i and then y_i . Whichever order she chooses, Bob must play by setting the first chosen y_i or y'_i variable to 1, and by setting the second chosen to 0. Then Alice must play by choosing one of the two x_i or x'_i variables, and Bob can set it anyway he wants. This way we simulate the TQBF game using the DT game. The difficulty is now in ensuring that Alice and Bob are indeed forced to play by the above "standard" strategies. To enforce this, additional gadgets will be put in place, so that any deviation from the above standard strategies will cause the deviating player to lose the DT game.

So let us begin.

3.3 First auxiliary function

In the reduction we will need an example, due to Wei Zhan, of a Boolean function W such that for some variable p it holds that $D(W) = D(W|_{p=1}) \gg D(W|_{p=0})$. This function will be a kind of product of the following function $w : \{0, 1\}^4 \rightarrow \{0, 1\}$:

$$w(p, a_0, a_1, r) = \begin{cases} 0 & \text{if } p = 1, \text{ and } a_0 = a_1, \\ a_r & \text{if } p = 0, \text{ or } a_0 \neq a_1. \end{cases}$$

► **Lemma 15** ([21]). *The function w is such that*

- (i) $D(w) = 3$,
- (ii) $D(w|_{p=1}) = 3$ and
- (iii) $D(w|_{p=0}) = 2$.

Proof of Lemma 15.

- (i) To determine w one can ask the values of a_0 and a_1 . If $a_0 \neq a_1$ then it is enough to know r to determine the value of the function. If $a_0 = a_1$ then it is enough to know p (if $p = 0$ then $w = a_0 = a_1$, if $p = 1$ then $w = 0$). The proof of the lower bound follows from the second item.
- (ii) The upper-bound follows from (i). The lower-bound is proven by brute force. We have

$$w(1, a_0, a_1, r) = \begin{cases} 0 & \text{if } a_0 = a_1, \\ a_r & \text{if } a_0 \neq a_1. \end{cases}$$

If we choose to query a_0 and a_1 first, and they differ, we still need to query r . If we choose to query a_i and r first, and $r = 1 - i$, we still need to query a_{1-i} .

- (iii) To determine $w(0, a_0, a_1, r)$ we may ask r then a_r . It is easy to see that one question is not enough. ◀

Denote by $W_k : \{0, 1\}^{1+3k}$ the Boolean function given by:

$$W_k(p, a_0^1, a_1^1, r^1, \dots, a_0^k, a_1^k, r^k) = w(p, a_0^1, a_1^1, r^1) \oplus \dots \oplus w(p, a_0^k, a_1^k, r^k).$$

► **Lemma 16.** *The function W_k has the following properties:*

1. $D(W_k) = 3k$; Moreover, there is a strategy for the second player (who sets the values) such that if the first player (Alice) chooses variable p then she loses.
2. $D(W_k|_{p=1}) = 3k$;
3. $D(W_k|_{p=0}) = 2k$.

Proof of Lemma 16. It is easy to see that if functions f and g have disjoint variables then $D(f \oplus g) = D(f) + D(g)$. By these reasons the second and the third items are direct corollaries of the same items in Lemma 15. To prove the first item just consider the following obvious inequalities:

$$3k = D(W_k|_{p=1}) \leq D(W_k) \leq k \cdot D(w) = 3k. \quad \blacktriangleleft$$

3.4 Second auxiliary function

Another tool in the reduction is the following function:

$$F_n(y_1, y'_1, x_1, x'_1, \dots, y_n, y'_n, x_n, x'_n) := f_1 \oplus g_1 \oplus \dots \oplus f_n \oplus g_n, \text{ where:}$$

- every f_i is defined as $f_i(y_i, y'_i) = y_i \wedge y'_i$;

66:8 The Hardness of Decision Tree Complexity

■ every g_i is defined as

$$g_i(y_1, y'_1, x_1, x'_1, \dots, y_i, y'_i, x_i, x'_i) = \begin{cases} x_i & \text{if } f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i = 1 \\ x'_i & \text{otherwise.} \end{cases}$$

We claim that $D(F_n) = 3n$. Moreover, we want to claim some properties of the corresponding DT-game between Alice (who chooses variables) and Bob (who sets values). We would like to say that Alice and Bob must follow certain *standard* strategies, or else they will lose the DT game. Standard strategies are as follows:

Standard strategies for Alice. In a standard strategy for Alice, she spends $2n$ questions to ask about the variables y_i and y'_i for all i . She can ask these questions in an arbitrary order.

She also spends n questions to ask about exactly one of the two variables x_i and x'_i , for each i . Here the order is crucial. The correct variable to choose depends on the value of $f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i$: she chooses x_i if this is 1, and x'_i if this is 0, as in the definition of g_i above. So, before asking about x_i or x'_i , Alice must first ask about $y_1, y'_1, \dots, y_{i-1}, y'_{i-1}$ and about the appropriate variable in every couple (x_j, x'_j) for all $j < i$. This defines f_j , for all $j \leq i$, and g_j , for all $j < i$.

Standard strategies for Bob. A standard strategy for Bob is any strategy such that, the first time Alice asks about one of the two variables y_i or y'_i , Bob answers 1.

We now formulate the following lemma about standard strategies.

► **Lemma 17.** *It holds that $D(F_n) = 3n$ and, furthermore,*

1. *If Alice plays according to a standard strategy, she wins the DT game within $3n$ rounds.*
2. *If Alice does not play according to a standard strategy then Bob can play in such a way that Alice does not win within $3n$ rounds.*
3. *If, while Alice is playing according to a standard strategy, Bob does not answer according to one of his standard strategies, then Alice can win the DT game in strictly fewer than $3n$ rounds.*

This auxiliary function is one of the central pieces in the reduction. It will allow us to replace the TQBF game, where Alice and Bob choose values for some variables, by a DT game, where Alice chooses some variables and Bob sets their value. For the reduction to go through, however, we need to put the pieces together in just the right way.

Proof of Lemma 17. The first observation follows from Items 1 and 2. Item 1 follows because, in a standard strategy, Alice has asked about all variables in the functions f_i , and because she asked about the relevant variable among x_i and x'_i , she also learned all the g_i .

Now we prove Item 2. Assume that Alice does not play according to a standard strategy. It means that either (a) Alice does not ask about some y_i or y'_i , or (b) she did not ask about one of the x_i or x'_i , or (c.i) she asks about x_i or x'_i before seeing all the variables of $f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i$, or (d) for some i , she saw all the variables of $f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i$, but asked about the wrong x_i or x'_i .

In case (a), Alice did not ask about some y_i or y'_i , so that the value of f_i is not defined and independent of the remaining values, and hence the value of F_n is also not defined, so if Bob plays according to any standard strategy, they end in a non-monochromatic subcube, and Alice loses. Likewise, in cases (b) and (d), g_i is undefined and independent of the remaining values, and hence F_n is also undefined, and Alice also loses.

Now suppose we are in case (c.i), but not (a), (b), or (d), or (c.j), for any $j < i$. Then, Alice has asked about one of the variables x_i or x'_i , but she did so before asking every variable of $f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i$. We then claim that Bob can answer Alice's questions in such a way that Alice will need to ask at least $3n + 1$ questions in total to know the value of F . Indeed, Alice has asked about x_i or x'_i before $f_1 \oplus g_1 \oplus \dots \oplus g_{i-1} \oplus f_i$ became defined. So after choosing some value for the variable x_i or x'_i , Bob can still answer Alice's questions by a standard strategy such that g_i equals the variable x'_i or x_i which Alice did not choose. The remaining questions Bob answers according to an arbitrary standard strategy. Since Alice wasted (at least) one question by asking an irrelevant variable, it follows that she needs $3n$ other questions to fix the value of F_n .

Finally, to prove Item 3, suppose that the first time Alice asks one of the two variables y_i or y'_i , Bob answers 0. Then the function $f_i = y_i \wedge y'_i$ becomes fixed and equal to 0, so Alice can fix the value of F without ever asking about the other variable. And so Alice wins within $3n - 1$ moves. ◀

3.5 The reduction

Let us here sketch of proofs of theorems 10 and 11. We reduce the TQBF instance

$$\exists y_1 \forall x_1 \dots \forall x_n h(y_1, x_1, \dots, y_n, x_n) \tag{1}$$

to computing the query complexity $D(D)$ of the function:

$$D = \begin{cases} q, & \text{if } G_n \vee p = 0 \\ W_{10n} \oplus F_n, & \text{otherwise.} \end{cases}$$

where $D = D(x_1, x'_1, y_1, y'_1, \dots, x_n, x'_n, y_n, y'_n, p, q, a_0^1, a_1^1, r^1, \dots, a_0^{10n}, a_1^{10n}, r^{10n})$

is a Boolean function on $4n + 2 + 30n$ variables.

$$\begin{aligned} W_{10n} &= W_{10n}(p, a_0^1, a_1^1, r^1, \dots, a_0^{10n}, a_1^{10n}, r^{10n}) \text{ was defined in Section 3.3,} \\ F_n &= F_n(y_1, y'_1, x_1, x'_1, \dots, y_n, y'_n, x_n, x'_n) \text{ was defined in Section 3.4. And} \\ G_n &= G_n(y_1, y'_1, x_1, x'_1, \dots, y_n, y'_n, x_n, x'_n) \end{aligned}$$

is a (previously undefined) Boolean function on $2n$ variables, given by:

$$G_n = h(y_1, x_1, \dots, y_n, x_n) \vee \bigvee_{i=1}^n (y_i \wedge y'_i) \vee \bigvee_{i=1}^n x_i \oplus x'_i$$

In the next subsection we prove that this reduction is correct. But let us sketch here how the above pieces (p, q, W_{10n}, F_n and G_n) fit together:

- If Equation (1) holds, Alice can query the variables of F_n using a standard strategy for F_n : if she wants to set y_i to 1, she asks about y_i and then y'_i ; to set y_i to 0, she asks first about y'_i and then y_i . If Bob plays according to a standard strategy, he will be forced to set the variables y_i as Alice wants, and ultimately h will be 1, so G will be 1, and then Alice and Bob must play the game for W_{10n} . So Alice wins with $3n + 30n$ queries in total.
- If Equation (1) does not hold, and Alice plays according to a standard strategy on the $3n$ variables of F_n , then Bob can force h and hence G_n to be 0. Now Alice is in trouble: if she doesn't query p , she won't know whether q or W_{10n} is relevant, so the function won't be fixed. But as soon as she queries p , Bob can answer $p = 1$, and now she must query further $30n$ variables to learn W_{10n} . That's $3n + 1 + 30n$ queries in total.

66:10 The Hardness of Decision Tree Complexity

- If at any point one of the players does not use a standard strategy, then the other player will have a way of winning .

Completion of the proofs of Theorems 10 and 11. First we note that this reduction can be computed in polynomial time if h is given as a circuit and by a NC^0 -circuit if h is given a truth-table: every value of D depends only on one value of h . To see DLOGTIME -uniformity one can check that, given values for the input variables of D , one can calculate the output, as a function of h , in time $O(n)$. We simply calculate every value in the above expression for D in time $O(n)$, and as a result, the output will be either 1, 0, $h_n(\dots)$ or $\neg h_n(\dots)$. So the truth table of D can be computed from the truth table of h by a DLOGTIME -uniform NC^0 -reduction.

We now claim that (1) is true if and only if $D(D) \leq 33n$.

When (1) holds

First we prove if (1) is true, i.e., Alice has a winning strategy in the TQBF-game, then $D(D) \leq 33n$, i.e. Alice has a winning strategy in the DT-game, which allows her to win within $33n$ steps.

In the TQBF-game Alice first chooses y_1 , then y_2 as a function of y_1 and (Bob's choice for) x_1 then y_3 as a function from y_1, x_1, y_2, x_2 , and so on. We wish to translate such a strategy for the TQBF-game, into a strategy for the DT-game. The translation is the following.

Alice begins by fixing the value of F_n to a constant. She will do so, by using the following standard DT-strategy (*standard* as in the proof of Lemma 17). If the TQBF-strategy of Alice sets $y_1 = 1$, then the standard DT-strategy of Alice first asks about the variable y_1 , and then asks about the variable y'_1 . If the TQBF-strategy of Alice sets $y_1 = 0$, the standard DT-strategy of Alice swaps the order: first asking about y'_1 and then about y_1 . The standard DT-strategy of Alice then asks about the appropriate relevant variable x_1 or x'_1 (according to her standard strategy). She considers the value of the chosen variable (x_1 or x'_1) to be the value Bob has chosen for x_1 in TQBF-game. Then, the DT-strategy of Alice asks about y_2 and y'_2 in some order, that again depends on the corresponding value for y_2 in the TQBF-strategy, and so on.

Now, either Bob follows a standard strategy (as in Lemma 17), or not. If he does, then (because the TQBF-strategy is a winning strategy for Alice) the value of h is equal to 1 and hence G_n is also equal to 1 and therefore D is equal to $F_n \oplus W_{10n}$. Note, that the value of F_n is already defined. By Lemma 16 Alice can define the value of W_{10n} in remaining $30n$ moves, so she wins.

If Bob does not use a standard strategy, Alice can define the value of F_n in fewer than $3n$ moves. She can do it by Lemma 17. Then Alice asks about p . Then, either:

- *Bob answers $p = 1$.* In this case, D is equal to $F_n \oplus W_{10n}$, F_n is already defined and Alice can ask at least $30n$ questions, which is enough to determine the value of W_{10n} .
- *Bob answers $p = 0$.* In this case by Lemma 16 Alice can define the value of $W_{10n|p=0}$ in $20n$ questions, so she has at least $10n$ questions left. She uses these questions to ask about all the variables $q, x_1, x'_1, y_1, y'_1, \dots, x_n, x'_n, y_n$ and y'_n . Therefore, now Alice knows the values of W_{10n} , q, p, G_n , and F_n , so Alice knows the value of D .

When (1) is false

Now we prove that if (1) is false then Alice cannot win in $33n$ moves. First we prove the following

► **Lemma 18.** *Assume that (1) is false. Then, Bob can answer the questions of Alice about the variables of F_n in such a way that*

- F_n will not be defined as long as Alice has asked fewer than $3n$ questions.
- If Alice has asked exactly $3n$ questions, then either (i) the value of F_n is not defined, or (ii) the value of F_n is defined, but there exists an assignment of variables not asked by Alice such that $G_n = 0$.

Proof of Lemma 18. From Lemma 17 we know that, if Alice does not play according to a standard strategy, she will need more than $3n$ questions to define F_n . So we can assume that Alice plays according to a standard strategy. Bob will also play according to a standard strategy, with the following additional constraint: for every pair $\{y_i, y'_i\}$ Bob will answer 0 to the second requested variable. (Recall that, a standard strategy for Bob is one where he answers 1 for the first requested variable in the pair.)

We claim that if (1) is false, and Alice uses a standard strategy, then Bob can make $x_i = x'_i$ for every i , and $h = 0$, thus forcing $G_n = 0$. Indeed, in a standard strategy Alice asks variables in the right order, so that Bob can set x_i and x'_i to the same value, according to his winning strategy in the TQBF-game. This makes the value of h equal to 0. ◀

Now we are ready to describe the strategy for Bob. To recall: we are assuming that (1) is false, and we will devise a strategy for Bob that will leave D undefined unless Alice asks more than $33n$ queries.

Bob's strategy.

- For the variables that define F_n , Bob uses the strategy from Lemma 18.
- If Alice asks about p then Bob answers 1.
- For the variables that define W_{10n} , Bob uses some best strategy for $W_{10n}|_{p=1}$.
- For q the answer is arbitrary.

We argue that if Bob uses this strategy then Alice cannot define the value of D in $33n$ questions. Indeed, assume that Alice asks about p . Then $p = 1$ and the value of D is equal to the value of $F_n \oplus W_{10n}$. Now either Alice asked $< 3n$ questions about F_n or $< 30n$ questions about $W_{10n}|_{p=1}$. Either way, one of these functions is not defined and hence D is also not defined.

Now assume that Alice does not ask about p . Then by setting $p = 1$, we can always force the value of D to be equal to $F_n \oplus W_{10n}$, so this value must be defined. But to define F_n and W_{10n} , Alice must spend all her $33n$ questions, and so she cannot ask about q . Moreover, to learn the value of F_n , she must spend exactly $3n$ questions about F_n , but she cannot spend any more. From Lemma 18, there must then exist some setting of the variables Alice didn't ask, such that $G_n = 0$. Then, by way of this assignment together with $p = 0$, D becomes equal to q , which Alice did not ask and hence is not defined. ◀

4 Open questions

1. What is the exact time-complexity of tt-DT? Is it possible to improve $O(3^n n)$ -algorithm of Proposition 6? Is it possible to prove any non-trivial bounds (for example, under the Exponential Time Hypothesis)?
2. Is it possible to improve the $O(\log N \log \log N)$ -depth bound of Proposition 9?
3. What is the exact time, space, and circuit complexity of the problem of finding the minimum *size* of a decision tree that computes a given Boolean function? It is known that this problem belongs to P[1], but the best depth upper-bound we know is $O((\log N)^2)$. It seemed to us that our reduction cannot be adapted to this case without a significantly new idea.

4. What can we say about the problem of *approximating* DT complexity? One can consider, in the reduction above, the function $D_n^1 \oplus \dots \oplus D_n^k$ instead of D_n for some k , where all D_n^i are the same functions as D_n with fresh variables. It allows to prove that the problem of the approximation of DT with constant term has the same complexity as exact calculation of DT. Is it possible to improve on this result?

References

- 1 Scott Aaronson. Algorithms for boolean function query properties. *SIAM Journal on Computing*, 32(5):1140–1157, 2003. doi:10.1137/S0097539700379644.
- 2 David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1 . In *Proceedings STOC*, 1986. doi:10.1145/12130.12131.
- 3 David A Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1 . *Journal of Computer and System Sciences*, 41(3):274–306, 1990. doi:10.1016/0022-0000(90)90022-D.
- 4 L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.
- 5 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *Computational Complexity*, 28:617–659, 2019. doi:10.1007/S00037-019-00190-7.
- 6 Mika Göös. Lower bounds for clique vs. independent set. In *Proceedings of FOCS*, 2015. doi:10.1109/FOCS.2015.69.
- 7 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th FOCS*, 2015. doi:10.1109/FOCS.2015.70.
- 8 David Guijarro, Victor Lavin, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Information Processing Letters*, 70(5):233–239, 1999. doi:10.1016/S0020-0190(99)00063-0.
- 9 G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119–127, 1980.
- 10 C. Koch, C. Strassle, and L. Tan. Properly learning decision trees with queries is NP-hard. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2383–2407, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi:10.1109/FOCS57990.2023.00146.
- 11 Kasper Green Larsen. *Models and techniques for proving data structure lower bounds*. PhD thesis, Aarhus University, 2013.
- 12 Bruno Loff and Sagnik Mukhopadhyay. Lifting theorems for equality. In *Proceedings of STACS*, 2019. doi:10.4230/LIPIcs.STACS.2019.50.
- 13 Mihai Pătraşcu. *Lower bound techniques for data structures*. PhD thesis, Massachusetts Institute of Technology, 2008.
- 14 J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi:10.1023/A:1022643204877.
- 15 J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
- 16 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. doi:10.1007/S004930050062.
- 17 Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography Conference*, pages 1–20. Springer, 2004. doi:10.1007/978-3-540-24638-1_1.
- 18 Detlef Sieling. Minimization of decision trees is hard to approximate. *J. Comput. Syst. Sci.*, 74(3):394–403, 2008. doi:10.1016/J.JCSS.2007.06.014.
- 19 Shihao Xuanyuan, Shiang Xuanyuan, and Ye Yue. Application of c4. 5 algorithm in insurance and financial services using data mining methods. *Mobile Information Systems*, 2022(1), 2022.

- 20 Andrew Chi-Chih Yao. On the complexity of comparison problems using linear functions. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 85–89. IEEE Computer Society, 1975.
- 21 Wei Zhan. URL: <https://cstheory.stackexchange.com/questions/52546/a-question-about-decision-tree-complexity>.

Commutative \mathbb{N} -Rational Series of Polynomial Growth

Aliaume Lopez  

University of Warsaw, Poland

Abstract

This paper studies which functions computed by \mathbb{Z} -weighted automata can be realised by \mathbb{N} -weighted automata, under two extra assumptions: commutativity (the order of letters in the input does not matter) and polynomial growth (the output of the function is bounded by a polynomial in the size of the input). We leverage this effective characterization to decide whether a function computed by a commutative \mathbb{N} -weighted automaton of polynomial growth is star-free, a notion borrowed from the theory of regular languages that has been the subject of many investigations in the context of string-to-string functions during the last decade.

2012 ACM Subject Classification Theory of computation \rightarrow Quantitative automata; Theory of computation \rightarrow Transducers

Keywords and phrases Rational series, weighted automata, polyregular function, commutative

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.67

Related Version *Full Version:* <https://arxiv.org/abs/2404.02232> [11]

1 Introduction

Given a semiring \mathbb{S} , and a finite alphabet Σ , the class of (*noncommutative*) \mathbb{S} -rational series is defined as functions from Σ^* to \mathbb{S} that are computed by \mathbb{S} -weighted automata [1]. This computational model is a generalization of the classical notion of non-deterministic finite automata to the weighted setting, where transitions are labeled with elements of \mathbb{S} . The semantics of \mathbb{S} -weighted automata on a given word w is defined by the sum over all accepting runs reading w , of the product of the weights of the transitions taken along this run. In this paper, we are interested in the case where \mathbb{S} equals \mathbb{N} or \mathbb{Z} , hence, in \mathbb{N} -rational series (\mathbb{N} Series) and \mathbb{Z} -rational series (\mathbb{Z} Series). It is clear that \mathbb{N} Series is a proper subclass of \mathbb{Z} Series, and a longstanding open problem is to provide an algorithm that decides whether a given \mathbb{Z} Series is in \mathbb{N} Series [10].

► **Problem 1.** *Input:* A \mathbb{Z} Series f . *Output:* Is f in \mathbb{N} Series?

Problem 1 recently received attention in the context of polyregular functions (Poly), a computational model that aims to generalize the theory of regular languages to the setting of string-to-string functions [2]. In the case of regular languages, *star-free languages* form a robust subclass of regular languages described equivalently in terms of first order logic [13], counter-free automata [13], or aperiodic monoids [16]. Analogously, there exists a *star-free* fragment of polyregular functions called star-free polyregular functions (SF) [2]. One open question in this area is to decide whether a given polyregular function is star-free.

► **Problem 2.** *Input:* A polyregular function f . *Output:* Is f star-free?

In order to approach decision problems on polyregular functions, restricting the output alphabet to a single letter has proven to be a fruitful method [6, 7]. Because words over a unary alphabet are canonically identified with natural numbers, unary output polyregular functions are often called \mathbb{N} -polyregular functions (NPoly), and their *star-free* counterpart star-free \mathbb{N} -polyregular functions (NSF). Coincidentally, polyregular functions with unary



© Aliaume Lopez;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

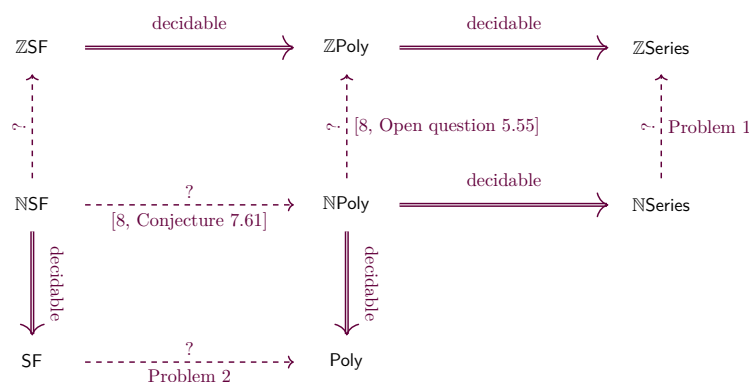
Article No. 67; pp. 67:1–67:16



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** Decidability and inclusions of classes of functions, arranged along two axes. The first one is the complexity of the output alphabet (\mathbb{Z} , \mathbb{N} , Σ). The second one is the allowed computational power (star-free polyregular functions, polyregular functions, rational series). Arrows denote strict inclusions, and effectiveness (both in terms of decidability and of effective representation) is represented by thick double arrows. Inclusions that are suspected to be effective are represented using a dashed arrow together with a question mark.

output forms a subclass of \mathbb{N} -rational series, namely the class of \mathbb{N} -rational series of *polynomial growth*, i.e. the output of the function is bounded by a polynomial in the size of the input. In [5], the authors introduced the class of \mathbb{Z} -polyregular functions ($\mathbb{Z}\text{Poly}$) as a subclass of \mathbb{Z} -rational series that generalizes \mathbb{N} -polyregular functions by allowing negative outputs, and showed that membership in the *star-free* subclass $\mathbb{Z}\text{SF}$ inside $\mathbb{Z}\text{Poly}$ is decidable [5, Theorem V.8]. Although this could not be immediately leveraged to decide $\mathbb{N}\text{SF}$ inside $\mathbb{N}\text{Poly}$, it was conjectured that $\mathbb{N}\text{Poly} \cap \mathbb{Z}\text{SF} = \mathbb{N}\text{SF}$ [8, Conjecture 7.61]. It was believed that understanding the membership problem of $\mathbb{N}\text{Poly}$ inside $\mathbb{Z}\text{Poly}$, that is, a restricted version of Problem 1, would be a key step towards proving $\mathbb{N}\text{Poly} \cap \mathbb{Z}\text{SF} = \mathbb{N}\text{SF}$, which itself would give hope in designing an algorithm for Problem 2. We illustrate in Figure 1 the known inclusions and related open problems between the discussed classes of functions.

Contributions. In this paper, we work under the extra assumption of *commutativity*, that is, assuming that the function is invariant under the permutation of its input. In this setting, we prove that $\mathbb{N}\text{Poly} \cap \mathbb{Z}\text{SF} = \mathbb{N}\text{SF}$ [8, Conjecture 7.61] and design an algorithm that decides whether a function in $\mathbb{Z}\text{Poly}$ is in $\mathbb{N}\text{Poly}$ [8, Open question 5.55]. As a consequence, the upper left square of Figure 1 has all of its arrows decidable and with effective conversion procedures under this extra assumption. Because \mathbb{Z} -rational series with *polynomial growth* are exactly \mathbb{Z} -polyregular functions [5], this can be seen as decision procedure for Problem 1 under the extra assumption of *commutativity* and *polynomial growth*. Similarly, our results provide an algorithm for Problem 2 under the extra assumption of *commutativity* and *unary output alphabet*.

As an intermediate step, we provide a complete and decidable characterization of polynomials in $\mathbb{Q}[\vec{X}]$ that can be computed using $\mathbb{N}\text{Series}$ (resp. $\mathbb{Z}\text{Series}$). These characterizations uncover a fatal flaw in the proof of a former characterization of such polynomials [10, Theorem 3.3, page 4]. We also prove that this previous results holds for polynomials with at most two indeterminates (Lemma 29), which may explain why it was not detected earlier. Furthermore, these characterizations provide effective descriptions of polynomials that can be expressed in $\mathbb{Z}\text{Series}$ as those obtained using integer combinations of products of *binomial coefficients* (called integer binomial polynomials, defined page 11) and similarly for $\mathbb{N}\text{Series}$

by introducing the notion of strongly natural binomial polynomials (defined page 12), which we believe has its own interest. Finally, these characterizations demonstrate that polynomials expressible by $\mathbb{Z}\text{Series}$ (resp. $\mathbb{N}\text{Series}$) are exactly those expressible by $\mathbb{Z}\text{SF}$ (resp. $\mathbb{N}\text{SF}$), that is, polynomials are inherently *star free* functions.

Outline of the paper. In Section 2, we provide a combinatorial definition of \mathbb{N} -polyregular functions (resp. \mathbb{Z} -polyregular functions), show that one can decide if a function $f \in \mathbb{Z}\text{Poly}$ is commutative (Lemma 8). In Section 3, we provide a counterexample to the flawed result of [10, Theorem 3.3, page 4] (Lemma 15), and correct it by providing effective characterizations of polynomials computed by $\mathbb{Z}\text{Series}$ (Theorem 31) and $\mathbb{N}\text{Series}$ (Theorem 34). Finally, in Section 4, we answer positively to [8, Open question 5.55] (Theorem 37) and to [8, Conjecture 7.61] (Theorem 40), both under the extra assumption of *commutativity*.

2 Preliminaries

The capital letters Σ, Γ denote fixed alphabets, i.e. finite set of letters, and Σ^*, Γ^* (resp. Σ^+, Γ^+) are the set of words (resp. non-empty words) over Σ, Γ . The empty word is written $\varepsilon \in \Sigma^*$. When $w \in \Sigma^*$ and $a \in \Sigma$, we let $|w| \in \mathbb{N}$ be the length of w , and $|w|_a$ be the number of occurrences of a in w .

We assume that the reader is familiar with the basics of automata theory, in particular the notions of monoid morphisms, idempotents in monoids, monadic second-order (MSO) logic and first-order (FO) logic over finite words (see e.g. [17]). As aperiodicity will be a central notion of this paper, let us recall that a monoid M is *aperiodic* whenever for all $x \in M$, there exists $n \in \mathbb{N}$ such that $x^{n+1} = x^n$. If the monoid M is finite, this n can be uniformly chosen for all elements in M .

We use the notation $\{\{\cdot\}\}: \Sigma^* \rightarrow \mathbb{N}^\Sigma$ for the map that counts occurrences of every letter in the input word (that is, computes the Parikh vector) namely: $\{\{w\}\} := (a \mapsto |w|_a)_{a \in \Sigma}$. Given a set X , a function $f: \Sigma^* \rightarrow X$ is *commutative* whenever for all $u \in \Sigma^*$, for all permutations σ of $\{1, \dots, |w|\}$, $f(\sigma(u)) = f(u)$. Equivalently, it is *commutative* whenever there exists a map $g: \mathbb{N}^\Sigma \rightarrow X$ such that $g \circ \{\{\cdot\}\} = f$.

Let $k \in \mathbb{N}$, and let Σ be a finite alphabet. Given a function $\eta: \{1, \dots, k\} \rightarrow \Sigma$, we define the $\eta^\dagger: \mathbb{N}^k \rightarrow \Sigma^*$ as $\eta^\dagger(\vec{x}) := \eta(1)^{x_1} \dots \eta(k)^{x_k}$. A function $f: \mathbb{N}^k \rightarrow X$ is *represented* by a commutative function $g: \Sigma^* \rightarrow X$ if there exists a map $\eta: \{1, \dots, k\} \rightarrow \Sigma$ such that $g \circ \eta^\dagger = f$. This notion will be useful to formally state that a polynomial “is” a commutative polyregular function. For instance, the polynomial function $P(X, Y) = X \times Y$ is represented by the commutative function $g: \{a, b\}^* \rightarrow \mathbb{Z}$ defined by $g(w) := |w|_a \times |w|_b$.

2.1 Polynomials

A polynomial $P \in \mathbb{Z}[X_1, \dots, X_k]$ is *non-negative* when for all non-negative integer inputs $n_1, \dots, n_k \geq 0$, the output $P(n_1, \dots, n_k)$ of the polynomial is non-negative. In the case of at most three indeterminates, we use variables X, Y, Z instead of X_1, X_2, X_3 to lighten the notation. Beware that we do not consider negative values as input, as the numbers n_i will ultimately count the number of occurrences of a letter in a word. As an example, the polynomial $(X - Y)^2$ is non-negative, and so is the polynomial X^3 , but the polynomial $X^2 - 2X$ is not.

A *monomial* is a product of indeterminates and integers. For instance, XY is a monomial, $3X$ is a monomial, $-Y$ is a monomial, but $X + Y$ and $2X^2 + XY$ are not. Every polynomial $P \in \mathbb{Z}[X_1, \dots, X_n]$ decomposes uniquely into a sum of monomials. A monomial S *divides*

a monomial T , when S divides T seen as polynomials in \mathbb{Q} . For instance, $2X$ divides XY , $-YZ$ divides X^2YZ^3 , and Y does not divide X . In the decomposition of $P \in \mathbb{Z}[X_1, \dots, X_k]$, a monomial is a *maximal monomial* if it is a maximal element for the divisibility preordering of monomials. In the polynomial $P(X, Y) := X^2 - 2XY + Y^2 + X + Y$, the set of *maximal monomials* is $\{X^2, -2XY, Y^2\}$. For instance, the non-negative monomials of $P(X, Y) := (X - Y)^2$ are X^2 and Y^2 .

2.2 Polyregular Functions

Because the functions of interest in this paper have output in \mathbb{N} or \mathbb{Z} , we will only provide the definition of *polyregular functions* for these two output semigroups, and we refer the reader to [3] for the general definition of polyregular functions and their aperiodic counterpart, the *star-free polyregular functions*. We chose in this paper to provide a combinatorial description of polyregular functions with commutative outputs because it will play nicely with our analysis on polynomials. This description is very similar in shape to the *finite counting automata* introduced by [15].

► **Definition 3** (\mathbb{Z} -polyregular functions [5]). *Let $d \in \mathbb{N}$. The set $\mathbb{Z}\text{Poly}_d$ of polyregular \mathbb{Z} -polyregular functions of degree at most d , is the set of functions $f: \Sigma^* \rightarrow \mathbb{Z}$ such that there exists a finite monoid M , a morphism $\mu: \Sigma^* \rightarrow M$, and a function $\pi: M^{d+1} \rightarrow \mathbb{Z}$ satisfying for all $w \in \Sigma^*$:*

$$f(w) = \pi^\dagger(w) := \sum_{w=u_1 \cdots u_{d+1}} \pi(\mu(u_1), \dots, \mu(u_{d+1})) \quad .$$

We call π the *production function* of f . If the function π has codomain \mathbb{N} , then f is \mathbb{N} -polyregular of degree at most d , i.e., $f \in \mathbb{N}\text{Poly}_d$. If the monoid M is aperiodic then the function f is *star-free \mathbb{Z} -polyregular* ($\mathbb{Z}\text{SF}_d$), resp. *star-free \mathbb{N} -polyregular* ($\mathbb{N}\text{SF}_d$).

We complete Definition 3 by letting $\mathbb{N}\text{Poly} := \bigcup_{d \in \mathbb{N}} \mathbb{N}\text{Poly}_d$, and similarly for $\mathbb{Z}\text{Poly}$, $\mathbb{N}\text{SF}$, and $\mathbb{Z}\text{SF}$. In order to illustrate these definitions, let us provide an example of an \mathbb{N} -polyregular function computed using a finite monoid in Example 4. Let us also introduce in Example 5 a function that serves as an example of *division* computed by a \mathbb{N} -polyregular function.

► **Example 4.** The map $f: w \mapsto |w| + 1$ belongs to $\mathbb{N}\text{SF}_1$.

Proof. Let us define $M := (\{1\}, \times)$ which is a finite aperiodic monoid, $\mu: \Sigma^* \rightarrow M$ defined by $\mu(w) := 1$, and $\pi: M^2 \rightarrow \mathbb{N}$ that is the constant function equal to 1. We check that for all $w \in \Sigma^*$: $\pi^\dagger(w) = \sum_{uv=w} 1 = |w| + 1 = f(w)$. ◀

► **Example 5.** Let $f: \Sigma^* \rightarrow \mathbb{N}$ be the function that maps a word w to the number of distinct pairs of positions in w , i.e., $f(w) = \binom{|w|}{2} = |w|(|w| - 1)/2$. Then, $f \in \mathbb{N}\text{SF}_2$.

Proof. Let us remark that the set P_w of distinct pairs of positions $i < j$ in a word w is in bijection with the set D_w of decompositions of the form $w = xyz$, where x and y are non-empty, via the map $(i, j) \mapsto (w_{1,i}, w_{i+1,j}, w_{j+1,|w|})$. Let us write $M := (\{0, 1\}, \max)$ which is a finite aperiodic monoid, and $\mu: \Sigma^* \rightarrow M$ that maps the empty word ε to 0 and the other words to 1. Then, let us define $\pi: M^3 \rightarrow \mathbb{N}$ via $\pi(x, y, z) = x \times y$. We conclude because:

$$\pi^\dagger(w) := \sum_{xyz=w} \pi(\mu(x), \mu(y), \mu(z)) = \sum_{xyz=w \wedge x \neq \varepsilon \wedge y \neq \varepsilon} 1 = |D_w| = |P_w| = f(w) \quad . \quad \blacktriangleleft$$

One of the appeals of $\mathbb{N}\text{Poly}$ and $\mathbb{Z}\text{Poly}$ are the numerous characterizations of these classes in terms of logic, weighted automata, and the larger class of polyregular functions [5, 8]. In this paper, the main focus will be the connection to weighted automata, which is based on the notion of *growth rate*. The *growth rate* of a function $f: \Sigma^* \rightarrow \mathbb{Z}$ is defined as the minimal d such that $|f(w)| = \mathcal{O}(|w|^d)$. If such a d exists, we say that the function f has *polynomial growth*. It turns out that for all $k \in \mathbb{N}$, $\mathbb{Z}\text{Poly}_d$ (resp. $\mathbb{N}\text{Poly}_d$) are precisely functions in $\mathbb{Z}\text{Series}$ (resp. in $\mathbb{N}\text{Series}$) that have growth rate at most d .

► **Lemma 6** ([8, Theorem 5.22]). *Let $f \in \mathbb{Z}\text{Series}$. The following are equivalent:*

1. $f \in \mathbb{Z}\text{Poly}_d$.
2. f has polynomial growth of degree at most d .

And similarly for $\mathbb{N}\text{Poly}$ and $\mathbb{N}\text{Series}$.

Let us introduce some compositional properties of \mathbb{Z} -polyregular functions that will be used in this paper to construct \mathbb{Z} -polyregular functions.

► **Lemma 7** ([5, Theorem II.20]). *Let $d \geq 1$, $f, g \in \mathbb{N}\text{Poly}_d$ (resp. $\mathbb{Z}\text{Poly}_d, \mathbb{N}\text{SF}_d, \mathbb{Z}\text{SF}_d$), L be a star-free language over Σ^* , and $h: \Sigma^* \rightarrow \Gamma^*$ be a polyregular function (resp. a star-free polyregular function). Then, the following are also in $\mathbb{N}\text{Poly}_d$ (resp. $\mathbb{Z}\text{Poly}_d, \mathbb{N}\text{SF}_d, \mathbb{Z}\text{SF}_d$): $f \circ h$, $f + g := w \mapsto f(w) + g(w)$, $f \times g := w \mapsto f(w) \times g(w)$, $\mathbf{1}_L \times f$. Furthermore, the above constructions preserve commutativity.*

Let us briefly state that commutativity is a decidable property of \mathbb{Z} -rational series, hence of \mathbb{Z} -polyregular functions. As a consequence, we are working inside a relatively robust and decidable subclass of \mathbb{Z} -rational series.

► **Lemma 8.** *Let $f \in \mathbb{Z}\text{Series}$. One can decide if f is commutative.*

Proof. Remark that the group of permutations of $\{1, \dots, n\}$ is generated by the cycle $c := (n, 1, \dots, n-1)$ and the transposition $t := (1, 2)$. As a consequence, a function f is commutative if and only if $f \circ c = f = f \circ t$. When f is a rational series, $f \circ c$ and $f \circ t$ are both rational series that can be effectively computed from f ,¹ and since equivalence of rational series is decidable [1, Corollary 3.6], we have obtained a decision procedure. ◀

3 \mathbb{N} -rational Polynomials

In this section, we will completely characterize which polynomials in $\mathbb{Q}[\vec{X}]$ are represented by \mathbb{N} -rational series (resp. \mathbb{Z} -rational series). To that end, we start by characterizing these classes for polynomials in $\mathbb{Z}[\vec{X}]$. We say that a polynomial $P \in \mathbb{Z}[X_1, \dots, X_n]$ is an *\mathbb{N} -rational polynomial* if it is represented by a \mathbb{N} -rational series. It is an easy check that polynomials with coefficients in \mathbb{N} are \mathbb{N} -rational polynomials (Lemma 9). However, Example 10 provides a polynomial with negative coefficients that is an \mathbb{N} -rational polynomial. The problem of characterizing \mathbb{N} -rational polynomials was claimed to be solved in [10], using the Definition 11 to characterize \mathbb{N} -rational polynomials, as restated in Flawed Theorem 12.

► **Lemma 9.** *Let $P \in \mathbb{N}[\vec{X}]$. Then, P is an \mathbb{N} -rational polynomial.*

¹ This can be done by guessing the second (resp. last) letter of the input word, remembering the first letter in a state, and then running the original automaton for f on the modified input, checking at the second position (resp. the end of the word) if the guess was correct.

► **Example 10.** The polynomials X , $X^2 + 3$, and $X^2 - 2X + 2$ are \mathbb{N} -rational polynomials, but $-X$ is not an \mathbb{N} -rational polynomial.

► **Definition 11** ([10, Section 3, page 3]). *The class $\text{PolyNNeg}[\vec{X}]$ is the class of polynomials $P \in \mathbb{Z}[\vec{X}]$ that are non-negative and such that every maximal monomial is non-negative. When the indeterminates are clear from the context, we write this class PolyNNeg .*

► **Flawed Theorem 12** ([10, Theorem 3.3, page 4]). *Let $P \in \mathbb{Z}[\vec{X}]$ be a polynomial. Then, P is an \mathbb{N} -rational polynomial if and only if $P \in \text{PolyNNeg}$.*

Before giving a counterexample to the above statement, let us first exhibit in Example 14 some non-negative polynomial that is not an \mathbb{N} -rational polynomial. While the example will not be in PolyNNeg , it illustrates the key difference between non-negative polynomials and \mathbb{N} -rational polynomials. In order to derive this example, we will need the following fundamental result about the pre-image of regular languages by polyregular functions.² Before that, let us remark that if a polynomial P is represented by a \mathbb{N} -rational series, then it is in fact represented by a \mathbb{N} -polyregular function thanks to Lemma 6.

► **Theorem 13** ([2, Theorem 1.7]). *The pre-image of a regular language by a (string-to-string) polyregular function is a regular language.*

► **Example 14.** Let $P(X, Y) := (X - Y)^2$. Then P is non-negative, but is not an \mathbb{N} -rational polynomial. Indeed, assume by contradiction that $f \in \mathbb{N}\text{Poly}$ represents P over the alphabet $\Sigma := \{a, b\}$. Then, $f^{-1}(\{0\})$ is a regular language (Theorem 13), but $f^{-1}(\{0\}) = \{w \in \Sigma^* \mid |w|_a = |w|_b\}$ is not.

Please note that the same argument cannot be leveraged for proving that P is not represented by a \mathbb{Z} -rational series: Theorem 13 only holds for *string-to-string* functions, and is applied to the specific case where the output alphabet is $\{1\}$, i.e., where the output of the function belongs to $\{1\}^*$ which is isomorphic to \mathbb{N} .

Let us now design a counterexample to Flawed Theorem 12 by suitably tweaking Example 14 to ensure that the polynomial not only is non-negative, but also belongs to PolyNNeg . We define $P_{\text{bad}}(X, Y, Z) := Z(X + Y)^2 + 2(X - Y)^2$.

► **Lemma 15.** *The polynomial P_{bad} belongs to PolyNNeg , but is not an \mathbb{N} -rational polynomial. As a corollary, [10, Theorem 3.3], restated in Flawed Theorem 12, is false when allowing at least 3 indeterminates.*

Proof. It is clear that P_{bad} is non-negative. We can expand the expression of P_{bad} to obtain $P_{\text{bad}} = ZX^2 + ZY^2 + 2ZXY + 2X^2 - 4XY + 2Y^2$. The maximal monomials of P are ZX^2 , ZY^2 , and $2ZXY$, all of which are non-negative.

Assume by contradiction that P_{bad} is an \mathbb{N} -rational polynomial. Let $\Sigma := \{a, b, c\}$ be a finite alphabet. There exists a commutative \mathbb{N} -polyregular function $f: \Sigma^* \rightarrow \mathbb{N}$ such that for all $w \in \Sigma^*$, $P_{\text{bad}}(|w|_a, |w|_b, |w|_c) = f(w)$. Remark that for all $x, y, z \geq 0$, $P_{\text{bad}}(x, y, z) = 0$ if and only if $z(x + y)^2 = -2(x - y)^2$. Hence, $P_{\text{bad}}(x, y, z) = 0$ if and only if $z = 0$ and $x = y$, or $z \neq 0$, and $x = y = 0$. Now, let us consider the language $L := \{w \mid f(w) = 0\}$. By the above computation, we conclude that $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\} \cup \{c\}^*$. Because $L \cap \{a, b\}^*$ is not a regular language, we conclude that L is not a regular language. However, $L = f^{-1}(\{0\})$ is a regular language (Theorem 13). ◀

² In this particular case, one could have considered more generally \mathbb{N} -rational series, and replaced regular languages over a unary alphabet by semi-linear sets.

We will discuss at the end of Section 3.2 why Lemma 15 is minimal in the number of indeterminates, which first requires us to provide a *correct* analogue of Flawed Theorem 12. Our counterexample relies on the fact that PolyNNeg is not stable under fixing indeterminates, while \mathbb{N} -rational polynomials are. Indeed, the polynomial P_{bad} satisfies $P_{\text{bad}}(X, Y, 1) = 3X^2 + 3Y^2 - 2XY$, which has a negative coefficient for a maximal monomial. Let us now prove that closing PolyNNeg under variable assignments is enough to recover from Flawed Theorem 12. We use the following notation to fix the value of some indeterminate, if $P(X, Y)$ is a polynomial in $\mathbb{Z}[X, Y]$, then $[P(X, Y)]_{X=1}$ is the polynomial $P(1, Y) \in \mathbb{Z}[Y]$. More generally, if ν is a partial function from \vec{X} to \mathbb{N} , written $\nu: \vec{X} \rightarrow \mathbb{N}$, the restriction $[P(\vec{X})]_{\nu}$ is the polynomial with indeterminates $\vec{Y} := \vec{X} - \text{dom}(\nu)$ obtained by fixing the variables of the domain of ν .

► **Definition 16.** *The class $\text{PolyStrNNeg}[\vec{X}]$ is the collection of polynomials $P \in \mathbb{Z}[\vec{X}]$ such that, for every partial function $\nu: \vec{X} \rightarrow \mathbb{N}$, every maximal monomial of $[P]_{\nu}$ is non-negative.*

First, let us remark that $\text{PolyStrNNeg} \subseteq \text{PolyNNeg}$, because polynomials in PolyStrNNeg are non-negative. We also remarked at the beginning of this section that our counterexample P_{bad} provided in Lemma 15 is not in PolyStrNNeg . The rest of the section is mainly concerned with proving the following corrected version of Flawed Theorem 12.

► **Theorem 17.** *Let $P \in \mathbb{Z}[\vec{X}]$. The following are equivalent:*

1. $P \in \text{PolyStrNNeg}$,
2. P is represented by a \mathbb{N} -rational series,
3. P is represented by a \mathbb{N} -polyregular function,
4. P is represented by a star-free \mathbb{N} -polyregular function,

Furthermore, the properties are decidable, and conversions effective.

Theorem 17 is surprising given the fact that it is not possible to decide whether a polynomial $P \in \mathbb{Z}[\vec{X}]$ is non-negative or if a polynomial P belongs to PolyNNeg (Remark 18), by reduction to the undecidability of Hilbert's Tenth Problem [9, 12]. That is, PolyStrNNeg is a decidable class that strictly contains $\mathbb{N}[\vec{X}]$, and is contained in the undecidable classes PolyNNeg and the class of non-negative polynomials.

► **Remark 18.** Checking whether a polynomial $P \in \mathbb{Z}[\vec{X}]$ is non-negative is undecidable. Similarly, checking whether a polynomial $P \in \mathbb{Z}[\vec{X}]$ belongs to PolyNNeg is undecidable.

The proof of Theorem 17 is divided into two parts. First, we provide in Section 3.1 a fine combinatorial understanding of what functions can be computed in $\mathbb{N}\text{Poly}$ and $\mathbb{Z}\text{Poly}$. This allows us to prove that \mathbb{N} -polyregular functions are in PolyStrNNeg (Corollary 22). Then, in Section 3.2 we will show how to compute polynomials in PolyStrNNeg using $\mathbb{N}\text{Poly}$ (Lemma 26). Finally, we will next generalize Theorem 17 to polynomials in $\mathbb{Q}[\vec{X}]$ in Section 3.3.

3.1 From \mathbb{N} -polyregular functions to polynomials

Let us prove that \mathbb{N} -rational polynomials are in PolyStrNNeg . This fact follows from the correct implication in the statement of Flawed Theorem 12, but we provide a self-contained proof using a refinement of the classical combinatorial *pumping arguments* for $\mathbb{Z}\text{Poly}$ [5, Lemma 4.16] and $\mathbb{N}\text{Poly}$ [8, Lemma 5.37]. We take extra care to reprove in our upcoming Lemma 20 a strong statement that has two main goals. Our first goal is to highlight the role of commutative polyregular functions in the broader study of polyregular functions, which is done by reformulating the traditional pumping argument as a composition property involving said functions, which will be reused in the upcoming Definitions 35 and 39 of Section 4. Our second goal is to give a precise shape of the functions that arise from such *pumping arguments*, which was lacking in former similar statements.

To address our first goal, let us define that a function q is a *pumping pattern* from \mathbb{N}^p to Σ^* whenever there exists words $\alpha_0, \dots, \alpha_p \in \Sigma^*$, and words $u_1, \dots, u_p \in \Sigma^*$, such that $q(X_1, \dots, X_p) = \alpha_0 \prod_{i=1}^p u_i^{X_i} \alpha_i$. That is, q is syntactically defined by a non-commutative monomial over the monoid Σ^* . Pumping patterns are commutative polyregular functions.

Our second goal is achieved by understanding that \mathbb{N} -polyregular functions essentially compute binomial coefficients, as illustrated by the polynomial $X(X-1)/2 = \binom{X}{2}$ of Example 5. A *simple binomial function* is a function of the form $\binom{X-\ell}{k}$, where ℓ and k are natural numbers. We extend this to *natural binomial functions* that are obtained by considering \mathbb{N} -linear combinations of products of simple binomial functions, that is, we consider functions that have the following shape: $f(x_1, \dots, x_k) = \sum_{i=1}^n n_i \prod_{j=1}^k \binom{x_j - p_{i,j}}{k_{i,j}}$. Beware that $\binom{X-\ell}{k}$ is defined to be 0 when $X \leq \ell$, and is therefore not a polynomial. Let us immediately prove that simple binomial functions can be represented in NSF, generalizing Example 5. Conversely, we prove in Lemma 20 that, when suitably pumping a \mathbb{N} -polyregular function, one always obtains natural binomial functions.

► **Lemma 19.** *Let F be a simple binomial function from \mathbb{N}^k to \mathbb{N} . Then it is represented by a star-free polyregular function.*

Proof. Because of the stability properties of NSF (Lemma 7), we only need to check that given $r, s \in \mathbb{N}$, the function $x \mapsto \binom{x-r}{s}$ is represented by a function $f_{r,s} \in \text{NSF}$. Let us prove it when $r = 0$, since the other functions can be obtained by translating $f_{r,s}$. By definition, $\binom{x}{s} = |P_{x,s}|$, where $P_{x,s} := \{(x_1, \dots, x_s) \in \mathbb{N}^s \mid 1 \leq x_1 < \dots < x_s \leq x\}$. Let us proceed as in Example 5 and define $D_{w,s} := \{(u_1, \dots, u_s, u_{s+1}) \in (\Sigma^+)^s \times \Sigma^* \mid w = u_1 u_2 \dots u_s u_{s+1}\}$. It is clear that $D_{a^x, s}$ is in bijection with $P_{x,s}$ for all $x \in \mathbb{N}$ using the map $(x_1, \dots, x_s) \mapsto (a^{x_1}, \dots, a^{x_s})$. Now, using the monoid $M := (\{0, 1\}, \max)$ and the morphism $\mu(\varepsilon) := 0$ and $\mu(a) := 1$, one can compute $|D_{w,s}|$ as $\pi^\dagger(w)$ where $\pi: M^{s+1} \rightarrow \mathbb{N}$ is defined by $\pi(m_1, \dots, m_s, m_{s+1}) := m_1 \times \dots \times m_s$. We conclude that $f_{0,s} \in \text{NSF}_s$ is a star-free polyregular function. ◀

► **Lemma 20.** *Let f be an \mathbb{N} -polyregular function. There exists a computable $\omega \in \mathbb{N}_{\geq 1}$ such that for all pumping patterns $q: \mathbb{N}^p \rightarrow \Sigma^*$, there exists a computable natural binomial function F such that:*

$$f \circ q(\omega X_1, \dots, \omega X_p) = F \quad \text{over } (\mathbb{N}_{\geq 1})^p \quad .$$

The multiplicative factor ω is necessary in Lemma 20. Indeed, the function $f: \{a\}^* \rightarrow \mathbb{N}$ defined as 0 when the input is of odd length and 1 when the input is of even length is \mathbb{N} -polyregular, but $f(a^X)$ is not a polynomial. We can trade off this multiplicative factor for a constant term addition under the extra assumption that the function is star-free polyregular, as described in the following Lemma 21. This lemma is not immediately of use, but is crucial for the upcoming characterization of \mathbb{N} -rational polynomials in Theorem 34, which in turn is a key ingredient of our main Theorem 40.

► **Lemma 21.** *Let f be a star-free \mathbb{N} -polyregular function. There exists a computable $s \in \mathbb{N}_{\geq 1}$ such that for all pumping patterns $q: \mathbb{N}^p \rightarrow \Sigma^*$, there exists a computable natural binomial function F such that:*

$$f \circ q(X_1 + s, \dots, X_p + s) = F \quad \text{over } \mathbb{N}^p \quad .$$

Because natural binomial functions behave as polynomials with non-negative maximal monomials on large enough inputs, we can conclude from Lemma 20 that \mathbb{N} -rational polynomials are in PolyStrNNeg.

► **Corollary 22.** *Let $P \in \mathbb{Z}[X_1, \dots, X_p]$ be an \mathbb{N} -rational polynomial. Then, $P \in \text{PolyStrNNeg}$.*

Proof. Let f be a commutative \mathbb{N} -rational series with domain defined as $\Sigma := \{a_1, \dots, a_p\}$ that represents P . Because f has polynomial growth, $f \in \text{NPoly}$ (Lemma 6). Using Lemma 20, there exists a number $\omega \in \mathbb{N}_{\geq 1}$ and natural binomial function Q such that for all $n_1, \dots, n_p \geq 1$:

$$f\left(\prod_{i=1}^p a_i^{\omega n_i}\right) = Q(n_1, \dots, n_p) = P(\omega n_1, \dots, \omega n_p) \quad .$$

For large enough values of X , the simple binomial function $\binom{X-p}{k}$ coincides with a polynomial whose leading coefficient is $1/k!$ which is non-negative. We conclude that the maximal monomials of $P(\omega X_1, \dots, \omega X_p)$ are non-negative, and since $\omega \geq 1$, we conclude that the maximal monomials of P have non-negative coefficients.

For every partial valuation $\nu: \vec{X} \rightarrow \mathbb{N}$, the polynomial $[P]_\nu$ continues to be represented by a \mathbb{N} -polyregular function, namely $f_u: w \mapsto f(uw)$ where w belongs to a restricted alphabet. As a consequence, the maximal monomials of $[P]_\nu$ are also non-negative, and we have proven that $P \in \text{PolyStrNNeg}$. ◀

3.2 From polynomials to \mathbb{N} -polyregular functions

This section is devoted to proving that polynomials in PolyStrNNeg can be represented by star-free \mathbb{N} -polyregular functions. The key lemma of this section is Lemma 26, which is proved by induction on the number of indeterminates of a given polynomial P . In order to prove that result, we use the combinatorial Lemma 25 that allows us to transform a polynomial $P \in \text{PolyStrNNeg}$ into a polynomial in $\mathbb{N}[\vec{X}]$ through a well-chosen translation of the indeterminates. This argument is based on the notion of *discrete derivative* which is built by translating the domain of the polynomial. To that end, let us write τ_K for the *translation function* that maps a polynomial $P \in \mathbb{Z}[X_1, \dots, X_k]$ to the polynomial $P(X_1 + K, \dots, X_k + K)$.

► **Definition 23.** *Let $K \in \mathbb{N}$, and $P \in \mathbb{Z}[\vec{X}]$ be a polynomial, then $\Delta_K(P) := \tau_K(P) - P$.*

► **Lemma 24.** *Let $P \in \mathbb{N}[\vec{X}]$ that is non-constant, and $K \in \mathbb{N}$, then $\Delta_K(P) \in \mathbb{N}[\vec{X}]$ and all of its coefficients are (positive) multiples of K . Furthermore, every monomial that strictly divides some monomial of P appears in $\Delta_K(P)$.*

Proof. We prove the result for monomials, as it extends to \mathbb{N} -linear combinations by linearity. Let $P = \prod_{i=1}^k X_i^{\alpha_i}$ be a monomial. Notice that $\tau_K(P) = \prod_{i=1}^k (X_i + K)^{\alpha_i}$, and using a binomial expansion we list all the possible divisors of P , all of which with coefficients that are positive integers and multiples of K except the coefficient of the maximal monomial (equal to P itself) which is 1. As a consequence, $\tau_K(P) - P$ is simply obtained by removing this maximal monomial, which concludes the proof. ◀

► **Lemma 25.** *Let $P \in \text{PolyStrNNeg}$, P_1 be the sum of maximal monomials of P , and $P_2 := P - P_1$ be the sum of non-maximal monomials of P . There exists a computable number $K \in \mathbb{N}$, such that $Q := (\Delta_K(P_1) + \tau_K(P_2)) \in \mathbb{N}[\vec{X}]$.*

Proof. Let us first tackle the specific case where P is a constant polynomial. In this case, $P_1 = P$ and $P_2 = 0$. Furthermore, $\Delta_K(P_1) = 0$ for all $K \in \mathbb{N}$. We conclude that $\Delta_K(P_1) + \tau_K(P_2) = 0$ for all $K \in \mathbb{N}$, hence belongs to $\mathbb{N}[\vec{X}]$. Selecting $K = 0$ we conclude. Assume now that P is not a constant polynomial. We will use Lemma 24 on a well-selected

value of K . Let us write α to be the maximal absolute value of a coefficient in P . Let D be the number of unitary monomials that divide some monomial appearing in P . We can now define $K := D \times \alpha$, and let $Q := (\Delta_K(P_1) + \tau_K(P_2))$. Remark that $\Delta_K(P_1)$ is already in $\mathbb{N}[\vec{X}]$, and the constant coefficient of $\tau_K(P_2)$ is also in \mathbb{N} . For any other monomial of P_2 , by the maximality of P_1 , it strictly divides some monomial of P_1 , and equals some monomial of $\Delta_K(P_1)$ up to a multiplication by a factor in \mathbb{Q} . Because every monomial of $\Delta_K(P_1)$ has a coefficient that is a multiple of $K = \alpha \times D$, we can compensate every monomial of P_2 by a monomial of $\Delta_K(P_1)$. Therefore, $Q \in \mathbb{N}[\vec{X}]$. \blacktriangleleft

► **Lemma 26.** *Let $P \in \mathbb{Z}[\vec{X}]$. If $P \in \text{PolyStrNNeg}$, then P is represented by a star-free \mathbb{N} -polyregular function, which is computable given P .*

Proof. We prove the result by induction on the number of indeterminates of P . In the proof, we write \vec{X} for the indeterminates appearing in P , that is, we assume without loss of generality that all indeterminates are used.

Base case: If the (unique) maximal monomial of P is a constant term. Since $P \in \text{PolyStrNNeg}$, $P = n \in \mathbb{N}$, and therefore P is represented by a constant star-free \mathbb{N} -polyregular function.

Induction: Assume that P is not a constant polynomial, and let us write $P = P_1 + P_2$ where P_1 is the sum of the maximal monomials of P . We compute a bound K such that $Q := (\Delta_K(P_1) + \tau_K(P_2)) \in \mathbb{N}[\vec{X}]$ (Lemma 25). In particular, Q is represented by a star-free \mathbb{N} -polyregular function using Lemma 9, the latter being effectively computable from Q . Let us now remark that $P_1 \in \mathbb{N}[\vec{X}]$, and is therefore (effectively) represented by a star-free \mathbb{N} -polyregular function (using again Lemma 9). As a consequence, $\tau_K(P) = P_1 + Q$ is (effectively) represented by a function f_Δ .

For all partial valuations $\nu: \vec{X} \rightarrow \{0, \dots, K\}$ fixing at least one indeterminate, one can use the induction hypothesis to compute a star-free \mathbb{N} -polyregular function f_ν that represents $[P]_\nu$. This is possible because we assumed that all indeterminates in \vec{X} are used in P .

Let us assume that the alphabet over which the (commutative) functions f_Δ and f_ν are defined is $\{a_1, \dots, a_k\}$, with a_i representing the indeterminate X_i of the polynomials. Now, let us define by case analysis the following commutative star-free \mathbb{N} -polyregular function, defined on words w of the form $w := a_1^{x_1} \dots a_k^{x_k}$, with $x_1, \dots, x_k \geq 0$.

$$f(w) := \begin{cases} f_{[X_i \mapsto x_i]}(w) & \text{if } \exists i \in \{1, \dots, k\}, x_i \leq K \\ f_\Delta(a_1^{x_1-K} \dots a_k^{x_k-K}) & \text{otherwise} \end{cases} .$$

Remark that f is a commutative star-free \mathbb{N} -polyregular function that represents P . \blacktriangleleft

While Lemma 26 provides an effective conversion procedure, it does not explicitly state that the membership is decidable to keep the proof clearer. A similar proof scheme can be followed to conclude that membership is decidable, and even show that elements in PolyStrNNeg are, up to suitable translations, polynomials in $\mathbb{N}[\vec{X}]$ (Lemma 27). Beware that partial applications are still needed in this characterization, as Example 28 illustrates.

► **Lemma 27.** *Let $P \in \mathbb{Z}[\vec{X}]$. There exists a computable number $K \in \mathbb{N}$ such that the following are equivalent:*

1. $P \in \text{PolyStrNNeg}$,
2. for all partial functions $\nu: \vec{X} \rightarrow \mathbb{N}$, $\tau_K([P]_\nu) \in \mathbb{N}[\vec{X}]$,
3. for all partial functions $\nu: \vec{X} \rightarrow \{0, \dots, K\}$, $\tau_K([P]_\nu) \in \mathbb{N}[\vec{X}]$.

In particular, the above properties are decidable.

► **Example 28.** The polynomial P_{bad} is not a \mathbb{N} -rational polynomial, but is non-negative and satisfies $\tau_{10}(P_{\text{bad}}) \in \mathbb{N}[\vec{X}]$.

We now have all the tools to prove the corrected version of Flawed Theorem 12.

Proof of Theorem 17 on page 7. The implications Item 4 \implies Item 3 \implies Item 2 are obvious. Lemma 26 proves Item 1 \implies Item 4, while Corollary 22 proves Item 2 \implies Item 1. Note that the lemmas provide effective conversion procedures, and that Lemma 27 also provides a decision procedure. ◀

For completeness, let us remark that the counterexample of Lemma 15 uses three indeterminates, and this is not a coincidence: in the particular cases of one or two indeterminates, the classes PolyStrNNeg and PolyNNeg coincide. In particular, the examples appearing in [10] are not invalidated, as they all use at most two indeterminates. Note that the equivalence is clear for the univariate case, where being non-negative and having non-negative maximal coefficient clearly imply being an \mathbb{N} -rational polynomial.

► **Lemma 29.** $\text{PolyStrNNeg}[X, Y] = \text{PolyNNeg}[X, Y]$.

Proof. It is clear that $\text{PolyStrNNeg}[X, Y] \subseteq \text{PolyNNeg}[X, Y]$, by considering the empty valuation $\nu: \{X, Y\} \rightarrow \mathbb{N}$. For the converse inclusion, let us consider $P(X, Y)$ that is non-negative, such that the maximal monomials are non-negative.

If we fix none of the variables, then the maximal monomials are non-negative by assumption. If we fix one of the variables, we can assume without loss of generality that we fix $X = k$ for some $k \in \mathbb{N}$. Then $P(k, Y)$ is a non-negative *univariate* polynomial, and therefore must either have a positive leading coefficient (which is the unique maximal monomial in this case) or be constant equal to 0. In both cases, the maximal monomials have positive coefficient. The same reasoning applies *a fortiori* in the case where we fix the two indeterminates, leading to a constant polynomial. ◀

3.3 From \mathbb{Z} to \mathbb{Q}

Let us complete our analysis of polynomials represented by $\mathbb{N}\text{Series}$ or $\mathbb{Z}\text{Series}$ by considering polynomials with coefficients in \mathbb{Q} , and justify that all the combinatorial work has already happened in \mathbb{Z} and \mathbb{N} . From Lemma 21, we know that the polynomials that can be computed by star-free \mathbb{N} -polyregular functions are going to coincide (on large enough inputs) with natural binomial functions. For that reason, we introduce the following “polynomial counterpart” of a binomial coefficient: given two numbers $\ell, k \in \mathbb{N}$, $\binom{X-\ell}{k}$ defined as $(X-\ell) \cdots (X-\ell-k)/k!$,³ that we call a *binomial monomial*, and we introduce *natural binomial polynomials* as \mathbb{N} -linear combinations of products of binomial monomials, i.e., of the shape: $P(X_1, \dots, X_k) = \sum_{i=1}^n n_i \prod_{j=1}^k \binom{X_j - p_{i,j}}{k_{i,j}}$. Similarly, we introduce the class of *integer binomial polynomials*, which are obtained by \mathbb{Z} -linear combinations of products of binomial monomials.

These definitions are justified by the classical result of Pólya that characterizes polynomials P in $\mathbb{Q}[X]$ that are *integer-valued* (i.e., are such that $P(\mathbb{Z}) \subseteq \mathbb{Z}$) as integer binomial polynomials [14, 4]. Note that this result straightforwardly extends to multiple indeterminates as we prove in Lemma 30.

³ In particular, $\binom{X-\ell}{k}$ is defined to be 1 when $k = 0$, and $X - \ell$ when $k = 1$.

67:12 Commutative \mathbb{N} -Rational Series of Polynomial Growth

► **Lemma 30.** *Let $P \in \mathbb{Q}[X_1, \dots, X_k]$ be a polynomial. Then, P is an integer binomial polynomial if and only if $P(\mathbb{Z}^k) \subseteq \mathbb{Z}$, if and only if $P(\mathbb{N}^k) \subseteq \mathbb{Z}$.*

As an immediate corollary, we completely characterize the class of polynomials in $\mathbb{Q}[\vec{X}]$ that are represented by $\mathbb{Z}\text{Poly}$ as the integer binomial polynomials.

► **Theorem 31.** *Let $P \in \mathbb{Q}[\vec{X}]$. Then, the following properties are equivalent:*

1. P is integer-valued,
2. P is represented by a \mathbb{Z} -rational series,
3. P is represented by a \mathbb{Z} -polyregular function,
4. P is represented by a star-free \mathbb{Z} -polyregular function,
5. P is an integer binomial polynomial.

These properties are furthermore decidable.

Proof. The implications Item 4 \implies Item 3 \implies Item 2 \implies Item 1 are obvious. Now, Item 1 \implies Item 5 is a direct consequence of Lemma 30. Finally, Item 5 \implies Item 4 follows from the fact that $\binom{X-p}{k}$ is represented by a star-free \mathbb{Z} -polyregular function defined by hardcoding the output values (in \mathbb{Z}) when $0 \leq X \leq p$, and using a star-free \mathbb{N} -polyregular function when $X > p$ (Lemma 19). Because $\mathbb{Z}\text{SF}$ is closed under products and \mathbb{Z} -linear combinations, we conclude. ◀

Obtaining an analogue of Theorem 31 for \mathbb{N} -polyregular functions requires a bit more work, as polynomials in $\mathbb{Q}[\vec{X}]$ that are represented by $\mathbb{N}\text{Poly}$ are not exactly natural binomial polynomials (see Example 32). To address the issues raised by the former example, we introduce the notion of *strongly natural binomial polynomials*, as the polynomials $P \in \mathbb{Q}[X]$ such that for all partial valuation $\nu: \rightarrow \mathbb{N}$, $[P]_\nu$ is a natural binomial polynomial, which characterizes the class of polynomials that are represented by $\mathbb{N}\text{Poly}$ (Theorem 34).

► **Example 32.** The polynomial $Q(X, Y, Z) := \binom{X-4}{1} \bullet \binom{Y}{1} \bullet \binom{Z}{1} \bullet + 8 \bullet \binom{Y}{2} \bullet + 8 \bullet \binom{Z}{2} \bullet + 4$ is a non-negative natural binomial polynomial in $\mathbb{Z}[X, Y, Z]$, but cannot be computed by a star-free \mathbb{N} -polyregular function. Indeed, $Q(0, Y, Z)$ has a negative maximal monomial, hence $Q \notin \text{PolyStrNNeg}$, and we conclude using Theorem 17.

► **Lemma 33.** *Let $P \in \mathbb{Q}[\vec{X}]$ be an integer-valued polynomial, and $n \in \mathbb{N}_{\geq 1}$ be such that $nP \in \text{PolyStrNNeg}$. Then, P is a strongly natural binomial polynomial.*

► **Theorem 34.** *Let $P \in \mathbb{Q}[\vec{X}]$ be a polynomial with rational coefficients and let α be the smallest number in $\mathbb{N}_{\geq 1}$ such that $\alpha P \in \mathbb{Z}[\vec{X}]$. Then, the following are equivalent:*

1. $\alpha P \in \text{PolyStrNNeg}$ and P is integer-valued,
2. P is represented by a \mathbb{N} -rational series,
3. P is represented by a \mathbb{N} -polyregular function,
4. P is represented by a star-free \mathbb{N} -polyregular function,
5. P is a strongly natural binomial polynomial.

In particular, the properties are decidable.

Proof. Let us first remark that $\mathbb{N}\text{Poly} \subseteq \mathbb{N}\text{Series}$, and that if P is represented by a function $f \in \mathbb{N}\text{Series}$, then said function has polynomial growth, and in particular $f \in \mathbb{N}\text{Poly}$ thanks to Lemma 6. As a consequence, Item 2 \iff Item 3. For the implication Item 3 \implies Item 1, we obtain $\alpha P \in \text{PolyStrNNeg}$ via Theorem 17 by remarking that \mathbb{N} -polyregular functions have output in \mathbb{N} and are closed under multiplication by a constant $\alpha \in \mathbb{N}$. The fact that P is integer-valued follows from Theorem 31 and the fact that $\mathbb{N}\text{SF} \subseteq \mathbb{Z}\text{Poly}$. The implication Item 1 \implies Item 5 is obtained thanks to Lemma 33.

Let us now prove by induction on the number of indeterminates that Item 5 \implies Item 4. Note that by construction, there exists a number $K \in \mathbb{N}$ such that when the input values of P are all greater than K , P coincides with a natural binomial function, which is itself represented by a star-free \mathbb{N} -polyregular function. If some input value X_i is set to a number $x_i \leq K$, then one can leverage the fact that $[P]_{X_i=x_i}$ remains a strongly natural binomial polynomial to conclude by induction that $[P]_{X_i=x_i}$ is represented by a star-free \mathbb{N} -polyregular function. Combining these, we obtain a star-free \mathbb{N} -polyregular function representing P .

Finally, the implication Item 4 \implies Item 3 is immediate as $\text{NSF} \subseteq \text{NPoly}$. \blacktriangleleft

Let us remark that Theorem 34 shows that the class of polynomials represented by NPoly is the same as the class of polynomials represented by NSF , which is a non-trivial statement that will be reused in the study of more general commutative functions in $\mathbb{Z}\text{Poly}$.

4 Beyond Polynomials

In this section, we leverage the decidability results of Section 3 to decide membership in NPoly inside $\mathbb{Z}\text{Poly}$ and membership in NSF inside NPoly , both under the extra assumption of commutativity. To characterize NPoly inside $\mathbb{Z}\text{Poly}$ we introduce the notion of (k, \mathbb{N}) -combinatorial function (Definition 35), following the spirit of previous characterizations of subclasses of $\mathbb{Z}\text{Poly}$ in terms of *polynomial pumping arguments* [6, 7, 5].

► **Definition 35.** Let $k \in \mathbb{N}$, and $f: \Sigma^* \rightarrow \mathbb{Z}$ be a \mathbb{Z} -polyregular function. The function f is (k, \mathbb{N}) -combinatorial if there exists $\omega \in \mathbb{N}$, such that for all pumping patterns $q: \mathbb{N}^k \rightarrow \Sigma^*$, there exists a strongly natural binomial polynomial P satisfying:

$$f \circ q(\omega X_1, \dots, \omega X_k) = P \quad \text{over } (\mathbb{N}_{\geq 1})^k \quad .$$

Let us now introduce a decomposition of commutative \mathbb{Z} -polyregular functions into integer binomial polynomials. Given a number $\omega \in \mathbb{N}$, let us write ωTypes^k for the collection of pairs (S, \vec{r}) where $S \subseteq \{1, \dots, k\}$ and $r \in \{0, \dots, \omega - 1\}^k$. To a tuple $\vec{x} \in \mathbb{N}^k$, one can associate its ω -type, written $\omega\text{type}(\vec{x})$, which is the pair (S, \vec{r}) where $S = \{i \in \{1, \dots, k\} \mid x_i \geq \omega\}$ and $\vec{r} = (x_i \bmod \omega)_{i \in \{1, \dots, k\}}$.

► **Lemma 36.** Let $f: \Sigma^* \rightarrow \mathbb{Z}$ be a commutative \mathbb{Z} -polyregular function, where we fix the alphabet $\Sigma = \{a_1, \dots, a_k\}$. There exists a computable $\omega \in \mathbb{N}_{\geq 1}$, and computable integer binomial polynomials $P_{(S, \vec{r})} \in \mathbb{Q}[(X_i)_{i \in S}]$ for $(S, \vec{r}) \in \omega\text{Types}^k$, such that for all $\vec{x} \in \mathbb{N}^k$,

$$f \left(\prod_{i=1}^k a_i^{x_i} \right) = P_{(S, \vec{r})}((\lfloor x_i / \omega \rfloor)_{i \in S}) \quad \text{where } (S, \vec{r}) = \omega\text{type}(\vec{x}) \quad .$$

► **Theorem 37.** Let $k, d \in \mathbb{N}$, and $f \in \mathbb{Z}\text{Poly}_d$ be commutative over an alphabet of size k . Then, the following are equivalent:

1. f is (k, \mathbb{N}) -combinatorial,
2. $f \in \text{NPoly}_d$,

Furthermore, the properties are decidable, and conversions effective.

Proof. Let $f \in \mathbb{Z}\text{Poly}_d$ be commutative over an alphabet of size k . We apply Lemma 36 to compute an $\omega \in \mathbb{N}$ and integer binomial polynomials $(P_{(S, \vec{r})})_{(S, \vec{r}) \in \omega\text{Types}^k}$ such that for all $\vec{x} \in \mathbb{N}^k$, $f \left(\prod_{i=1}^k a_i^{x_i} \right) = P_{(S, \vec{r})}((\lfloor x_i / \omega \rfloor)_{i \in S})$, where $(S, \vec{r}) = \omega\text{type}(\vec{x})$. We are first going to prove that $f \in \text{NPoly}_d$ if and only if $P_{(S, \vec{r})}$ is a strongly natural binomial polynomial for all $(S, \vec{r}) \in \omega\text{Types}^k$. This will also provide decidability of Item 2, since one can decide whether a polynomial is strongly natural binomial polynomial using Theorem 34.

Assume that $f \in \mathbb{NPoly}$, then by definition, the polynomials $P_{(S, \vec{r})}$ are represented by an \mathbb{N} -polyregular function, hence are strongly natural binomial polynomials (Theorem 34). Conversely, if $P_{(S, \vec{r})}$ is a strongly natural binomial polynomial for all $(S, \vec{r}) \in \omega \mathbf{Types}^k$, then $f \in \mathbb{NPoly}$ because one can compute the ω -type of the input using a polyregular function, and then compute the suitable strongly natural binomial polynomial $P_{(S, \vec{r})}$ which is possible in \mathbb{NPoly} thanks to Theorem 34. The resulting composition belongs to \mathbb{NPoly} thanks to Lemma 7, and we conclude that $f \in \mathbb{NPoly}_d$ because it has growth rate at most d (Lemma 6).

Note that the same proof scheme can be used to conclude that Item 2 implies Item 1. For the converse implication, we are going to introduce ω_2 associated to the fact that f is (k, \mathbb{N}) -combinatorial. Because polynomials represented by \mathbb{N} -polyregular functions and integer binomial polynomials are both closed under multiplication of their input by a constant factor, we can assume that $\omega = \omega_2$ in the decomposition of f . Now, consider $(S, \vec{r}) \in \omega \mathbf{Types}^k$. Notice that for all vectors $\vec{x} \in (\mathbb{N}_{\geq 1})^k$, the vector $(x_1 \omega \mathbf{1}_S(1) + r_1, \dots, x_k \omega \mathbf{1}_S(k) + r_k)$ has ω -type (S, \vec{r}) . In particular, the following equality holds:

$$f \left(\prod_{i=1}^k a_i^{x_i \omega \mathbf{1}_S(i) + r_i} \right) = P_{(S, \vec{r})}((x_i)_{i \in S}) \quad \forall \vec{x} \in (\mathbb{N}_{\geq 1})^k \quad .$$

Let us therefore consider the pumping pattern $q: \mathbb{N}^k \rightarrow \Sigma^*$ that is simply defined as $q(X_1, \dots, X_k) := \prod_{i=1}^k a_i^{X_i \mathbf{1}_S(i) + r_i}$. Because f is (k, \mathbb{N}) -combinatorial with parameter ω , there exists a strongly natural binomial polynomial $P \in \mathbb{Q}[X_1, \dots, X_k]$ such that $f \circ q(\omega X_1, \dots, \omega X_k) = P(X_1, \dots, X_k)$ over $(\mathbb{N}_{\geq 1})^k$. This proves that $P_{(S, \vec{r})}((X_i)_{i \in S})$ equals $P(X_1, \dots, X_k)$ as polynomials, hence, that $P_{(S, \vec{r})}$ is a strongly natural binomial polynomial for all $(S, \vec{r}) \in \omega \mathbf{Types}^k$. We have proven that $f \in \mathbb{NPoly}_d$. ◀

It was already known that \mathbb{Z} -polyregular functions with unary input that are non-negative are \mathbb{N} -polyregular [1, Proposition 2.1 p 137]. Let us derive this fact from our Theorem 37.

► **Corollary 38.** *Let $f: \{a\}^* \rightarrow \mathbb{Z}$ be a non-negative \mathbb{Z} -polyregular function, then $f \in \mathbb{NPoly}$.*

Proof. Since f has unary input, it is commutative. Furthermore, f is $(1, \mathbb{N})$ -combinatorial because for all $q: \mathbb{N} \rightarrow \{a\}$ and all $\omega \geq 1$, $f(q(\omega X))$ is non-negative. When it is a polynomial function, it therefore belongs to $\mathbf{PolyStrNNeg}$, hence is a strongly natural binomial polynomial. We conclude using Theorem 37. ◀

Let us now prove that the above characterizations of commutative \mathbb{N} -polyregular functions can be combined with the recent advances in the study of \mathbb{Z} -polyregular functions [5] allowing to decide the membership of \mathbb{ZSF} inside \mathbb{ZPoly} . The key ingredient of this study is the use of a semantic characterization of star-free \mathbb{Z} -polyregular functions among \mathbb{Z} -rational series that generalizes the notion of aperiodicity for languages to functions.

► **Definition 39** (Ultimately polynomial). *Let Σ be a finite alphabet. A function $f: \Sigma^* \rightarrow \mathbb{Z}$ is ultimately polynomial when there exists $N_0 \in \mathbb{N}$ such that for all $k \in \mathbb{N}$, for all pumping pattern $q: \mathbb{N}^k \rightarrow \Sigma^*$, there exists a polynomial $P \in \mathbb{Q}[X_1, \dots, X_k]$ such that:*

$$f \circ q = P \quad \text{over } (\mathbb{N}_{\geq N_0})^k \quad .$$

It was observed in [5, Claim V.6], and in [8, Claim 7.45, Lemma 7.53] that a regular language L is *star-free* if and only if its indicator function $\mathbf{1}_L$ is ultimately polynomial. We can now answer [8, Conjecture 7.61] positively, by proving that $\mathbb{NPoly} \cap \mathbb{ZSF} = \mathbf{NSF}$.

► **Theorem 40.** *Let Σ be a finite alphabet, and $f: \Sigma^* \rightarrow \mathbb{Z}$ be a commutative \mathbb{N} -polyregular function. Then, the following are equivalent:*

1. f is ultimately polynomial,

2. $f \in \mathbb{ZSF}$,
3. $f \in \mathbb{NSF}$.

Furthermore, membership is decidable and conversions are effective.

Proof. The implication Item 3 \Rightarrow Item 2 is immediate since $\mathbb{NSF} \subseteq \mathbb{ZSF}$. Furthermore, Item 2 implies Item 1 following previous results for star-free \mathbb{Z} -polyregular functions [5, Theorem V.13].

For the implication Item 1 \Rightarrow Item 3, let us assume that f is ultimately polynomial. We prove the result by induction on the size of the alphabet Σ . By definition, there exists $N_0 \in \mathbb{N}$, and $P \in \mathbb{Q}[(X_a)_{a \in \Sigma}]$ such that:

$$f \left(\prod_{a \in \Sigma} a^{x_a} \right) = P((x_a)_{a \in \Sigma}) \quad \forall \vec{x} \in (\mathbb{N}_{\geq N_0})^\Sigma .$$

It is clear that $\tau_{N_0}(P)$ is represented by an \mathbb{N} -polyregular function, namely, $f_u: w \mapsto f(uw)$ where $u := \prod_{a \in \Sigma} a^{N_0}$, and is therefore represented by a star-free \mathbb{N} -polyregular function thanks to Theorem 34. For every letter $a \in \Sigma$ and number $0 \leq n \leq N_0$, there exists, by induction hypothesis, a star-free \mathbb{N} -polyregular function g_{a^n} that represents the function $f_{a^n}: (\Sigma \setminus \{a\})^* \rightarrow \mathbb{Z}$ that maps $w \in (\Sigma \setminus \{a\})^*$ to $f(a^n w)$.

Let us conclude by computing f using the following star-free \mathbb{N} -polyregular function $g: \Sigma^* \rightarrow \mathbb{Z}$:

$$g: w \mapsto \begin{cases} g_{a^n}(w) & \text{if } |w|_a = n \text{ for some } a \in \Sigma \text{ and } n \leq N_0 \\ \tau_{N_0}(P)((|w|_a - N_0)_{a \in \Sigma}) & \text{otherwise} \end{cases} \quad \blacktriangleleft$$

5 Outlook

Let us end on a more general discussion regarding the status of commutative input functions in the study of unary output polyregular functions. A *quantitative pumping argument* for polyregular function $f: \Sigma^* \rightarrow \mathbb{Z}$ states that f has property X if and only if for all pumping pattern $q: \mathbb{N}^k \rightarrow \Sigma^*$, $f \circ q$ has property X . Let us formalize such a statement for growth rate and aperiodicity respectively in Lemmas 41 and 42. Note that we generalized pumping patterns to commutative star-free polyregular functions to simplify the statements.

► **Lemma 41.** *Let $f \in \mathbb{ZSeries}$, and $d \in \mathbb{N}$. Then, $f \in \mathbb{ZPoly}_d$ if and only if for every commutative star-free polyregular function h of growth rate $l \in \mathbb{N}$, $(f \circ h) \in \mathbb{ZPoly}_{d \times l}$.*

► **Lemma 42.** *Let $f \in \mathbb{ZPoly}$. Then, $f \in \mathbb{ZSF}$, if and only if for every commutative star-free polyregular function h , $(f \circ h) \in \mathbb{ZSF}$.*


Remark that if Lemma 42 were to hold for \mathbb{N} -polyregular functions, then the decidability of \mathbb{NPoly} inside \mathbb{ZPoly} , and the decidability of \mathbb{NSF} inside \mathbb{NPoly} would immediately follow. On the one hand, one can guess a candidate function and check for equivalence, on the other hand, one can guess a commutative star-free polyregular function and check membership (which is decidable thanks to this paper). This is restated in our concluding conjecture.

► **Conjecture 43.** *Let $f \in \mathbb{ZPoly}$. Then, $f \in \mathbb{NPoly}$ if and only if for every commutative star-free polyregular function h , $(f \circ h) \in \mathbb{NPoly}$.*

References

- 1 Jean Berstel and Christophe Reutenauer. *Noncommutative rational series with applications*, volume 137 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2010. doi:10.1017/CB09780511760860.
- 2 Mikołaj Bojańczyk. Polyregular functions, 2018. arXiv:1810.08760.
- 3 Mikołaj Bojańczyk, Sandra Kiefer, and Nathan Lhote. String-to-String Interpretations With Polynomial-Size Output. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 106:1–106:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.106.
- 4 Paul-Jean Cahen and Jean-Luc Chabert. *Integer-Valued Polynomials*. American Mathematical Society, December 1996. doi:10.1090/surv/048.
- 5 Thomas Colcombet, Gaëtan Douéneau-Tabot, and Aliaume Lopez. Z-polyregular functions. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, Los Alamitos, CA, USA, June 2023. IEEE Computer Society. doi:10.1109/LICS56636.2023.10175685.
- 6 Gaëtan Douéneau-Tabot. Pebble Transducers with Unary Output. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2021.40.
- 7 Gaëtan Douéneau-Tabot. Hiding Pebbles When the Output Alphabet Is Unary. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 120:1–120:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.120.
- 8 Gaëtan Douéneau-Tabot. *Optimization of string transducers*. PhD thesis, Université Paris Cité, 2023. URL: https://gdoueneau.github.io/pages/DOUENEAU-TABOT_Optimization-transducers_v2.pdf.
- 9 David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1902. doi:10.1090/s0002-9904-1902-00923-3.
- 10 Juhani Karhumäki. Remarks on commutative \mathbb{N} -rational series. *Theoretical Computer Science*, 5(2):211–217, 1977. doi:10.1016/0304-3975(77)90008-1.
- 11 Aliaume Lopez. Commutative n -polyregular functions, 2024. doi:10.48550/arXiv.2404.02232.
- 12 Yuri Vladimirovich Matiyasevich. The diophantineness of enumerable sets. *Doklady Akademii Nauk SSSR*, 191:279–282, 1970. in Russian.
- 13 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. The MIT Press, 1971. doi:10.5555/1097043.
- 14 G. Pólya. Über ganzwertige ganze Funktionen. *Rend. Circ. Mat. Palermo*, 40:1–16, 1915. doi:10.1007/BF03014836.
- 15 Marcel P. Schützenberger. Finite counting automata. *Information and control*, 5(2):91–107, 1962. doi:10.1016/S0019-9958(62)90244-9.
- 16 Marcel P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.
- 17 Wolfgang Thomas. Languages, automata, and logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of formal languages*, pages 389–455. Springer, 1997. doi:10.1007/978-3-642-59136-5.

Slightly Non-Linear Higher-Order Tree Transducers

Lê Thành Dũng (Tito) Nguyễn ✉ 🏠 

CNRS & Aix-Marseille University, France

Gabriele Vanoni ✉ 🏠 

IRIF, Université Paris Cité, France

Abstract

We investigate the tree-to-tree functions computed by “affine λ -transducers”: tree automata whose memory consists of an affine λ -term instead of a finite state. They can be seen as variations on Gallot, Lemay and Salvati’s Linear High-Order Deterministic Tree Transducers.

When the memory is almost purely affine (*à la* Kanazawa), we show that these machines can be translated to tree-walking transducers (and with a purely affine memory, we get a reversible tree-walking transducer). This leads to a proof of an inexpressivity conjecture of Nguyễn and Pradic on “implicit automata” in an affine λ -calculus. We also prove that a more powerful variant, extended with preprocessing by an MSO relabeling and allowing a limited amount of non-linearity, is equivalent in expressive power to Engelfriet, Hoogeboom and Samwel’s invisible pebble tree transducers.

The key technical tool in our proofs is the Interaction Abstract Machine (IAM), an operational avatar of Girard’s geometry of interaction, a semantics of linear logic. We work with ad-hoc specializations to λ -terms of low exponential depth of a tree-generating version of the IAM.

2012 ACM Subject Classification Theory of computation \rightarrow Linear logic; Theory of computation \rightarrow Transducers

Keywords and phrases Almost affine lambda-calculus, geometry of interaction, reversibility, tree transducers, tree-walking automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.68

Related Version *Technical report with appendix*: <https://arxiv.org/abs/2402.05854> [38]

Funding *Lê Thành Dũng (Tito) Nguyễn*: Supported by the DyVerSe project (ANR-19-CE48-0010). *Gabriele Vanoni*: Supported by the ANR PPS Project (ANR-19-CE48-0014) and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101034255.

Acknowledgements We thank Damiano Mazza and Cécilia Pradic for discussions on the possible applications of the Geometry of Interaction to implicit complexity and automata theory.

1 Introduction

This paper investigates the expressive power of various kinds of *tree transducers*: automata computing tree-to-tree functions. This is a topic with a long history, and many equivalences between machine models are already known. For instance, the class of monadic second-order transductions¹ (MSOTs), whose name refers to a definition by logic, is also captured by various tree transducer models (e.g. [19]) or by a system of primitive functions and combinators [10]. This class is closed under composition, and includes functions such as:

mirror and add a d above each b : $a(b(c), c) \mapsto a(c, d(b(c)))$
relabel each c by parity of its depth: $a(b(c), c) \mapsto a(b(0), 1)$
count number of non- a nodes in unary: $a(b(c), c) \mapsto S(S(S(0)))$

¹ We consider only tree-to-tree transductions here, but there is a rich theory of MSOTs between graphs, or between arbitrary relational structures, cf. [12]. See also [8] for a history of MSO transductions. In the well-studied special case of string functions, MSOTs are called “regular functions”, cf. [33].

Some machines for MSOTs, such as the restricted macro tree transducers² of [20], involve *tree contexts* as data structures used in their computation. A tree context is a tree with “holes” at some leaves – for example $b(a(c, a(\cdot, c)))$ – and these holes are meant to be substituted by other trees. Thus, it represents a simple function taking trees to trees: these transducers use functions as data. From this point of view, in the spirit of functional programming, it also makes sense to consider transducers manipulating *higher-order data*, that is, functions that may take functions as arguments (a function of order $k + 1$ takes arguments of order at most k). This idea goes back to the 1980s [21] (cf. Remark 2.9), and a recent variant by Gallot, Lemay and Salvati [22, 23] computes exactly the tree-to-tree MSOTs.

Linear/affine λ -calculus for MSOTs. Gallot et al. [22, 23] use the λ -calculus to represent higher-order data, and in order to control the expressive power, they impose a *linearity* restriction on λ -terms. In programming language theory, a *linear* function uses its argument *exactly once*, while an *affine* function must use its argument *at most once* – affineness thus mirrors the “single-use restrictions” that appear in various tree transducer models [7, 20, 4, 10].

An independent but similar characterization of MSOTs appeared around the same time in Nguyễn and Pradic’s work on “implicit automata” [37, 34]. In an analogous fashion to how the untyped λ -calculus can be used as a Turing-complete programming language, they consider typed λ -terms seen as standalone programs. For a well-chosen type system – which enforces a linearity restriction – and input/output convention, it turns out that the functions computed by these λ -terms are exactly the MSO tree transductions [34, Theorem 1.2.3].

In fact, the proof of [34, Theorem 1.2.3] introduces, as an intermediate step, a machine model – the “single-state \mathfrak{L} -BRTTs” of [34, Section 6.4] – that uses λ -terms as memory to capture the class of MSOTs. Clearly, this device is very close to the tree transducer model of Gallot et al. Yet there are also important differences between the two: they can be understood as two distinct ways of extending what we call “purely linear λ -transducers”, as we will explain. Actually, to avoid some uninteresting pathologies (cf. [34, Theorem 7.0.2]), we shall prefer to work with affine λ -transducers instead.

► **Example 1.1.** Purely affine λ -transducers will be properly defined later, but for now, for the sake of concreteness, let us exhibit one such transducer. It takes input trees with binary a -labeled nodes, unary b -labeled nodes and c -labeled leaves, and is specified by the λ -terms:

$$t_a = \lambda \ell. \lambda r. \lambda x. \ell (r x) \quad t_b = \lambda f. \lambda x. S (f x) \quad t_c = S \quad u = \lambda f. f 0$$

where S and 0 are constants from the output alphabet. The input $a(b(c), c)$ is then mapped to $u (t_a (t_b t_c) t_c)$ which evaluates to the normal form $S (S (S 0))$. So this λ -transducer computes the aforementioned “number of non- a nodes written in unary” function. (It actually is purely linear: each bound variable has exactly one occurrence.)

The issue with this transducer model is its lack of expressiveness, as shown by the following consequence of our results, which settles an equivalent³ conjecture on “implicit automata” that had been put forth by Nguyễn and Pradic in [37, Section 5.3]. (We will come back in Remark 7.1 to how they overcome this to characterize tree-to-tree MSO transductions.)

² Which are very similar to some other transducer models for MSOTs: the bottom-up ranked tree transducers of [4, Sections 3.7–3.8] and the register tree transducers of [10, Section 4].

³ A routine syntactic analysis akin to [37, Lemma 3.6] shows that it is indeed equivalent. Note that the results of [37] are indeed about affine, not linear, λ -calculi.

► **Corollary 1.2** (of Theorem 1.4 below). *There exists a regular tree⁴ language whose indicator function cannot be computed by any purely affine λ -transducer.*

Gallot et al. avoid this limitation by extending the transducer model with common automata-theoretic features. They then show [23, Chapter 7] that the usual linear λ -terms lead to the previously mentioned characterization of MSOTs, while relaxing the linearity condition to “almost linearity” yields the larger class of “MSOTs with sharing”⁵ (MSOT-S).

This notion of *almost linear λ -term* was introduced by Kanazawa [28], who also studied the almost affine case in [28, 27]. In fact, the aforementioned characterization of MSOT-S was first claimed by Kanazawa in a talk more than 15 years ago [26], although he never published a proof to the best of our knowledge.

Flavors of affine types. We shall work with an affine type system that allows some data to be marked as duplicable via the exponential modality “!”. The grammar of types is thus $A, B ::= o \mid A \multimap B \mid !A$ – the connective \multimap is the affine function arrow.

The point is to allow us to restrict duplication by means of syntactic constraints on “!”.

► **Definition 1.3.** *A purely affine type does not contain any “!”, i.e. is built from o and \multimap . A type is said to be almost purely affine when the only occurrences of “!” are applied to o . In particular, every purely affine type is almost purely affine.*

For example, $(o \multimap !o) \multimap !o$ is almost purely affine but not $!(o \multimap o)$.

The latter definition is motivated by the aforementioned almost affine λ -terms [27], which allow the variables of base type o to be used multiple times, e.g. $\lambda y. \lambda f. \lambda g. (\lambda x. f x x) (g y y)$ is almost affine (and even almost linear) for $x : o$ and $y : o$. Inconveniently, almost linear/affine λ -terms are not closed under β -reduction, as remarked in [27, §4]. For instance, the previous term reduces to $\lambda y. \lambda f. \lambda g. f (g y y) (g y y)$ which uses $g : o \multimap o \multimap o$ twice. The “!” modality provides a convenient way to realize a similar idea while avoiding this drawback.

Each λ -transducer has in its definition a *memory type*, which is for instance $o \multimap o$ for Example 1.1. We extend Definition 1.3 to λ -transducers: a λ -transducer is purely (resp. almost purely) affine when its memory type is purely (resp. almost purely) affine.

Contributions. First, we study (almost) purely affine λ -transducers, relating them to yet another machine model: *tree-walking tree transducers*, see e.g. [19]. Those are devices with a finite-state control and a reading head moving around the nodes of the input tree; in one step, the head can move to the parent or one of the children of the current node.

► **Theorem 1.4.** *Writing \subseteq for a comparison in expressive power, we have:*

$$\begin{aligned} \text{purely affine } \lambda\text{-transducer} &\subseteq \text{reversible tree-walking transducer} \\ \text{almost purely affine } \lambda\text{-transducer} &\subseteq \text{tree-walking transducer} \end{aligned}$$

Corollary 1.2 then follows immediately from a result of Bojańczyk and Colcombet [9]: there exists a regular tree language not recognized by any tree-walking automaton.

⁴ This phenomenon depends on using trees as inputs. Over strings, purely affine λ -terms can be used to recognize any regular language [37, Theorem 5.1].

⁵ An MSOT-S can be decomposed as an MSO tree-to-graph transduction (Footnote 1), that produces a rooted directed acyclic graph (DAG), followed by the unfolding of this graph into a tree. The rooted DAG is a compressed representation of the output tree with some “sharing” of subtrees.

We also obtain characterizations of MSOTs and of MSOT-Ss, by preprocessing the input with an *MSO relabeling* – a special kind of MSOT that can only change the node labels in a tree, but keeps its structure as it is. Morally, this preprocessing amounts to the same thing as the automata-theoretic features – finite states and regular look-ahead – used in [22, 23].

► **Theorem 1.5.** *Using \equiv to denote an equivalence in expressive power, we have:*

$$\begin{aligned} & \text{purely affine } \lambda\text{-transducer} \circ \text{MSO relabeling} \equiv \text{MSOT} \\ & \text{almost purely affine } \lambda\text{-transducer} \circ \text{MSO relabeling} \equiv \text{MSOT-S} \end{aligned}$$

This should be informally understood as a mere rephrasing of the results of Gallot et al. and of Kanazawa, except with affine rather than linear λ -terms. On a technical level, we derive our right-to-left inclusions from their results; there turns out to be a minor mismatch, and working with affineness rather than linearity proves convenient to overcome it. As for the left-to-right inclusions, we get them as corollaries of Theorem 1.4.

Finally, we give a genuinely new characterization of the class MSOT-S² of functions that can be written as *compositions of two MSO transductions with sharing*.

► **Definition 1.6.** *A type is almost !-depth 1 when the only occurrences of ‘!’ are applied to almost purely affine types (e.g. !(lo \multimap o) is almost !-depth 1, but not !(o \multimap o)).*

► **Theorem 1.7.** *Almost !-depth 1 λ -transducer \circ MSO relabeling \equiv MSOT-S².*

To prove the left-to-right inclusion, we compile these λ -transducers to *invisible pebble tree transducers* [18] – an extension of tree-walking transducers known to capture MSOT-S². For the converse, we rely on a simple composition procedure for λ -transducers, which gives us:

► **Proposition 1.8.** *Suppose that the tree-to-tree functions f and g are computed by λ -transducers with respective memory types A and B . Then $g \circ f$ is computed by a λ -transducer with memory type $A\{o := B\}$.*

Key tool: the Interaction Abstract Machine (IAM). To translate λ -transducers into tree-walking or invisible pebble tree transducers, we use a mechanism that evaluates a λ -term using a pointer to its syntax tree that is moved by local steps – just like a tree-walking reading head. This mechanism, called the Interaction Abstract Machine, was derived from a family of semantics of linear λ -calculi known as “geometry of interaction” (GoI) – for more on its history, see [46, Chapter 3]. Our approach thus differs from both the proofs of Gallot et al. in [23], which go through a syntactic procedure for lowering the order of λ -terms, and those of Nguyễn and Pradic in [34], which rely on another kind of denotational semantics.

The IAM satisfies a well-known *reversibility* property, see e.g. [46, Proposition 3.3.4] – it even appears in the title of the seminal paper [14]. In the purely affine case, this gives us reversible tree-walking transducers in Theorem 1.4 – these have not appeared explicitly in the literature, but we define them similarly to the existing reversible graph-walking automata [39] and reversible two-way string transducers [15]. In the almost purely affine and almost !-depth 1 cases, we use an ad-hoc optimization of the IAM that breaks reversibility.

Preliminaries on trees. A *ranked alphabet* Σ is a finite set with a “rank” (or “arity”) $\text{rk}(c) \in \mathbb{N} = \{0, 1, \dots\}$ associated to each “letter” $c \in \Sigma$. It can be seen as a first-order signature of function symbols, and by a *tree* over Σ we mean a closed first-order term over this signature. For instance, a tree over $\{a : 2, b : 1, c : 0\}$ may have binary a -labeled nodes, unary b -labeled nodes and c -labeled leaves; examples include $a(b(c), c)$ or $b(a(a(c, c), c))$.

We write $\text{Tree}(\Sigma)$ for the set of trees over the ranked alphabet Σ . It will also be convenient to work with the sets $\text{Tree}(\Sigma, X) = \text{Tree}(\Sigma \cup \{x : 0 \mid x \in X\})$ of trees over Σ with the additional possibility of having X -labeled leaves.

2 Affine λ -terms and tree-to-tree λ -transducers

The grammar of our λ -terms, where a ranges over constants and x over variables, is

$$t, u ::= a \mid x \mid \lambda x. t \mid t u \mid !t \mid \mathbf{let} !x = u \mathbf{in} t$$

These λ -terms are considered up to renaming of bound variables ($\lambda x. t$ binds x in t , while $\mathbf{let} !x = u \mathbf{in} t$ binds x in t but not in u). We will also use *contexts*, which are λ -terms containing one occurrence of a special symbol, the hole $\langle \cdot \rangle$:

$$C, D, E ::= \langle \cdot \rangle \mid \lambda x. C \mid t C \mid C u \mid !C \mid \mathbf{let} !x = C \mathbf{in} t \mid \mathbf{let} !x = u \mathbf{in} C$$

Plugging, i.e. substituting the hole of a context C for a term t , potentially capturing free variables, is written $C\langle t \rangle$. For example, if $C = \lambda x. t \langle \cdot \rangle$, then $C\langle u v \rangle = \lambda x. t (u v)$. The linear logic tradition calls a term of the form $!t$ a *box*, and the number of nested boxes surrounding a subterm of some term is called its *depth* within this term. Similarly, the *depth of a context* C is defined to be the number of boxes surrounding the hole $\langle \cdot \rangle$. In this paper, however, we are counting only boxes which do *not* surround terms of the base type o .

Typing rules. Our type system is a variant of Dual Intuitionistic Linear Logic [6], with weakening to make it affine. As already said, our grammar of types is $A, B ::= o \mid A \multimap B \mid !A$. As usual, \multimap is right-associative: the parentheses in $A \multimap (B \multimap C)$ can be dropped.

The typing contexts are of the form (unrestricted variables) \mid (affine variables). In the rules below, a comma between two sets of typed affine variables denotes a disjoint union; this corresponds to prohibiting the use of the same affine free variable in two distinct subterms.

$$\frac{}{\Theta \mid \Phi, x : A \vdash x : A} \quad \frac{\Theta \mid \Phi, x : A \vdash t : B}{\Theta \mid \Phi \vdash \lambda x. t : A \multimap B} \quad \frac{\Theta \mid \Phi \vdash t : A \multimap B \quad \Theta \mid \Phi' \vdash u : A}{\Theta \mid \Phi, \Phi' \vdash t u : B}$$

$$\frac{}{\Theta, x : A \mid \Phi \vdash x : A} \quad \frac{\Theta \mid \emptyset \vdash t : A}{\Theta \mid \emptyset \vdash !t : !A} \quad \frac{\Theta \mid \Phi \vdash u : !A \quad \Theta, x : A \mid \Phi' \vdash t : B}{\Theta \mid \Phi, \Phi' \vdash \mathbf{let} !x = u \mathbf{in} t : B}$$

We also work with constants whose types are fixed in advance. Fixing $a : A$ means that we have the typing rule $\frac{}{\Theta \mid \Phi \vdash a : A}$. We shall also abbreviate $\emptyset \mid \emptyset \vdash t : A$ as $t : A$.

▷ **Claim 2.1 (Affineness).** If $\lambda x. t$ is well-typed, the variable x occurs at most once in t , at depth 0. (But let-bound variables are not subject to any such restriction).

Normalization. Our β -reduction rules, which can be applied in any context C , are as follows, where the context L consists of a succession of let-binders (i.e. $L ::= \langle \cdot \rangle \mid \mathbf{let} !x' = t' \mathbf{in} L$):

$$L\langle \lambda x. t \rangle u \longrightarrow_{\beta} L\langle t\{x := u\} \rangle \quad \mathbf{let} !x = L\langle !u \rangle \mathbf{in} t \longrightarrow_{\beta} L\langle t\{x := u\} \rangle$$

For instance, the following is a valid β -reduction:

$$(\mathbf{let} !x = u \mathbf{in} \mathbf{let} !y = v \mathbf{in} \lambda z. z x y) t \longrightarrow_{\beta}^* \mathbf{let} !x = u \mathbf{in} \mathbf{let} !y = v \mathbf{in} t x y$$

This “reduction at a distance” – an idea of Accattoli & Kesner [3] – is a way to get the desirable Proposition 2.4 below without having to introduce cumbersome “commuting conversions”. For an extended discussion in the context of a system very close to ours, see [32, §1.2.1].

► **Proposition 2.2** (Normalization and subject reduction). *Any well-typed term has a β -normal form. Furthermore, if $\Theta \mid \Phi \vdash t : A$ then $\Theta \mid \Phi \vdash t' : A$ for any β -normal form t' of t .*

► **Definition 2.3.** *We say that a term $\Theta \mid \Phi \vdash t : A$ is purely affine when all of its subterms have purely affine types (cf. Definition 1.3) and $\Theta = \emptyset$, which implies that it contains no $!$ -box or let-binding. We also call almost purely affine (resp. almost $!$ -depth 1) the terms $\Theta \mid \Phi \vdash t : A$ in which, for every subterm u of t ,*

- *the type of u is almost purely affine (resp. almost $!$ -depth 1 – cf. Definition 1.6),*
- *if $u = !r$ then r is purely (resp. almost purely) affine,*
- *$\Theta = x_1 : o, \dots, x_n : o$ (resp. $\Theta = x_1 : A_1, \dots, x_n : A_n$ where A_1, \dots, A_n are almost purely affine).*

► **Proposition 2.4.** *Assume that we work with purely affine constants, as will always be the case in this paper. Let $\Theta \mid \Phi \vdash t : A$ and suppose t is in normal form. If A and the types in Θ and Φ are purely affine (resp. almost purely affine, almost $!$ -depth 1), then so is t .*

For example, $\text{let } !z = !(\lambda x. x) \text{ in } z : o \multimap o$ is not purely affine but its normal form $\lambda x. x$ is.

Encoding of trees in our affine λ -calculus. Fix a ranked alphabet Σ . We consider λ -terms built over the constants $c : o^{\text{rk}(c)} \multimap o$ for $c \in \Sigma$. There is a canonical encoding $\widetilde{(\cdot)}$ of trees as closed terms of type o ; for instance, $\tau = a(b(c), c)$ is encoded as $\tilde{\tau} = a(b\ c)\ c$.

► **Proposition 2.5.** *Every closed term of type o using these constants admits a unique normal form. Furthermore, $\widetilde{(\cdot)}$ is a bijection between $\text{Tree}(\Sigma)$ and these normal forms.*

Given a type A and a family of λ -terms $\vec{t} = (t_c)_{c \in \Sigma}$ such that $t_c : A^{\text{rk}(c)} \multimap A$ for each letter $c \in \Sigma$, we write $\widehat{\tau}(\vec{t})$ for the result of replacing each constant c in $\tilde{\tau}$ by t_c . It is always well typed, with type A . For the example $\tau = a(b(c), c)$, we have $\widehat{\tau}((t_x)_{x \in \{a,b,c\}}) = t_a(t_b\ t_c)\ t_c$.

Higher-order transducers (or λ -transducers). Let us fix an input ranked alphabet Γ .

► **Definition 2.6.** *An (affine) λ -transducer $\text{Tree}(\Gamma) \rightarrow \text{Tree}(\Sigma)$ is specified by a memory type A and a family of terms (that can use the aforementioned constants from Σ):*

$$\underbrace{t_a : A^{\text{rk}(a)} \multimap A}_{\text{“transition terms”}} \text{ for each letter } a \in \Gamma \quad \text{and} \quad \underbrace{u : A \multimap o}_{\text{“output term”}}$$

The λ -transducer defines the function

$$\tau \in \text{Tree}(\Gamma) \mapsto \underbrace{\sigma \in \text{Tree}(\Sigma) \text{ such that } \tilde{\sigma} \text{ is the normal form of } u \widehat{\tau}((t_a)_{a \in \Gamma})}_{\text{well-defined and unique thanks to Proposition 2.5}}$$

This amounts to specifying a structurally recursive function over $\text{Tree}(\Gamma)$ with return type A , followed by some post-processing that produces an output tree. Alternatively, a λ -transducer can be seen as a kind of tree automaton whose memory consists of affine λ -terms of some type A (with constants from Σ) and whose bottom-up transitions are also defined by λ -terms.

In addition to the purely affine Example 1.1, we exhibit two other λ -transducers.

► **Example 2.7.** The following almost affine λ -transducer maps $S^n(0) = S(\dots(S(0)))$ to the list $[1, 2, \dots, n]$, encoded as the tree $\text{cons}(S(0), \dots(\text{cons}(S^n(0), \text{nil}) \dots))$.

$$\begin{aligned} t_0 &= \lambda x. \text{nil} : !o \multimap o \text{ (memory type)} & u &= \lambda g. g\ !(S\ 0) \\ t_S &= \lambda g. \lambda x. \text{let } !y = x \text{ in cons } y\ (g\ !(S\ y)) \end{aligned}$$

► **Example 2.8.** The following λ -transducer takes as input the binary encoding of a natural number n and returns a complete binary tree of height n , e.g. $0(0(1(0(\varepsilon)))) \mapsto a(a(c, c), a(c, c))$. Thus, its growth is doubly exponential. Its memory type $!(o \multimap o) \multimap o$ is almost $!$ -depth 1.

$$t_0 = \lambda g. \lambda x. \text{let } !f = x \text{ in } g!(\lambda y. f(f y)) \quad t_\varepsilon = \lambda x. \text{let } !f = x \text{ in let } !z = f !c \text{ in } z \\ t_1 = \lambda g. \lambda x. \text{let } !f = x \text{ in } g!(\lambda y. \text{let } !z = f(f y) \text{ in } !(a z z)) \quad u = \lambda g. g!(\lambda y. y)$$

Moreover, composing Examples 1.1 and 2.7 according to Proposition 1.8 gives another almost purely affine example.

► **Remark 2.9.** The original impetus for Engelfriet and Vogler’s higher-order transducers [21] was that their transducers of order k are equivalent to compositions of k unrestricted macro tree transducers. A major motivation of Gallot et al.’s machine model using linear λ -terms was also function composition, for which they give efficient constructions [23, Chapter 6] (which are non-trivial). And in Nguyễn and Pradic’s “implicit automata”, composition is just a matter of plugging two λ -terms together [37, Lemma 2.8].

3 Tree-walking transducers (last definitions needed for Theorem 1.4)

Generalities. In this paper, we shall encounter several machine models that generate some output tree in a top-down fashion, starting from the root. (This is not the case of λ -transducers.) They follow a common pattern, which we abstract as a lightweight formalism here: essentially, a deterministic regular tree grammar with infinitely many non-terminals.

► **Remark 3.1.** Engelfriet’s tree grammars with storage (see for instance [17]) are more complex formalisms that also attempt to unify several definitions of tree transducer models.

► **Definition 3.2.** A tree-generating machine over the ranked alphabet Σ consists of:

- a (possibly infinite) set \mathcal{K} of configurations;
- an initial configuration $\kappa_0 \in \mathcal{K}$ – in concrete instances, κ_0 will be defined as a simple function of some input tree (for tree transducers) or some given λ -term (for the IAM);
- a computation-step (partial) function $\mathcal{K} \rightarrow \text{Tree}(\Sigma, \mathcal{K})$.

► **Example 3.3.** To motivate the formal semantics for these machines that we will soon define, we give a tree-generating machine that is meant to produce the list $[1, 2, \dots, n]$ (for an arbitrarily chosen $n \in \mathbb{N}$), encoded as in Example 2.7.

- The set of configurations is $\mathcal{K} = \{\text{spine}, \text{num}\} \times \mathbb{N}$ where **spine** and **num** are formal symbols.
- The initial configuration is (spine, n) – let us write this pair as $\langle \text{spine}, n \rangle$.

- The computation-step function is
$$\left\{ \begin{array}{l} \langle \text{spine}, 0 \rangle \mapsto \text{nil} \\ \langle \text{spine}, m+1 \rangle \mapsto \text{cons}(\langle \text{num}, n-m \rangle, \langle \text{spine}, m \rangle) \\ \langle \text{num}, 0 \rangle \mapsto 0 \\ \langle \text{num}, m+1 \rangle \mapsto S(\langle \text{num}, m \rangle) \end{array} \right.$$

For $n = 3$, one possible run is

$$\langle \text{spine}, 3 \rangle \rightsquigarrow \text{cons}(\langle \text{num}, 1 \rangle, \langle \text{spine}, 2 \rangle) \rightsquigarrow \text{cons}(S(\langle \text{num}, 0 \rangle), \langle \text{spine}, 2 \rangle) \rightsquigarrow \dots$$

All runs starting from $\langle \text{spine}, 3 \rangle$ eventually reach the tree that encodes $[1, 2, 3]$.

Let us now discuss the general case. Intuitively, the execution of the machine involves spawning several independent concurrent processes, outputting disjoint subtrees. We formalize this parallel computation as a rewriting system \rightsquigarrow on $\text{Tree}(\Sigma, \mathcal{K})$: we have $\tau_1 \rightsquigarrow \tau_2$ whenever

τ_2 is obtained from τ_1 by substituting one of its configuration leaves by its image by the computation-step function. This rewriting system is orthogonal, and therefore confluent, which means that the initial configuration has at most one normal form. If this normal form exists and belongs to $\text{Tree}(\Sigma)$, it is the output of the machine; we then say that the machine converges. Otherwise, the output is undefined; the machine diverges.

Tree-walking transducers. Before giving the definition, let us see a concrete example.

► **Example 3.4.** According to Theorem 1.4, since the λ -transducer of Example 1.1 is purely affine, the function “count non- a nodes” that it defines can also be computed by some (reversible) tree-walking transducer. We show the run of such a transducer on the input $a_1(b_2(c_3), c_4)$ – the indices are not part of the node labels, they serve to distinguish positions:

$$\begin{aligned} (q, \circ, a_1) &\rightsquigarrow (q, \downarrow_\bullet, b_2) \rightsquigarrow S((q, \downarrow_\bullet, c_3)) \rightsquigarrow S(S((q, \uparrow_1^\bullet, b_2))) \rightsquigarrow S(S((q, \uparrow_1^\bullet, a_1))) \\ &\rightsquigarrow S(S((q, \downarrow_\bullet, c_4))) \rightsquigarrow S(S(S((q, \uparrow_2^\bullet, a_1)))) \rightsquigarrow S(S(S(0))) \end{aligned}$$

where q is the single state of the transducer. The second component records the “provenance”, i.e. the previous position of the tree-walking transducer relatively to the current node (stored in the third component): \downarrow_\bullet refers to its parent, \circ to itself, and \uparrow_i^\bullet to its i -th child.

► **Definition 3.5.** A tree-walking transducer (TWT) $\text{Tree}(\Gamma) \rightarrow \text{Tree}(\Sigma)$ consists of:

- a finite set of states Q with an initial state $q_0 \in Q$
- a family of (partial) transition functions for $a \in \Gamma$

$$\delta_a: Q \times \{\downarrow_\bullet, \circ, \uparrow_1^\bullet, \dots, \uparrow_k^\bullet\} \rightarrow \text{Tree}(\Sigma, Q \times \{\uparrow_\bullet, \circ, \downarrow_1^\bullet, \dots, \downarrow_k^\bullet\}) \quad \text{where } k = \text{rk}(a)$$

- a family of (partial) transition functions at the root for $a \in \Gamma$

$$\delta_a^{\text{root}}: Q \times \{\circ, \uparrow_1^\bullet, \dots, \uparrow_k^\bullet\} \rightarrow \text{Tree}(\Sigma, Q \times \{\circ, \downarrow_1^\bullet, \dots, \downarrow_k^\bullet\}) \quad \text{where } k = \text{rk}(a)$$

The TWT associates to each input tree τ a tree-generating machine whose output is the image of τ . Its set of configurations is $Q \times \{\downarrow_\bullet, \circ, \uparrow_1^\bullet, \dots\} \times \{\text{nodes of } \tau\}$ and its initial configuration of is $(q_0, \circ, \text{root of } \tau)$.

To define the image of (q, p, v) by the computation-step function (it is undefined if one of the following steps is undefined), we start with either $\delta_a^{\text{root}}(q, p)$ if v is the root or $\delta_a(q, p)$ otherwise – where a is the label of the node v – then replace each $(q', p) \in Q \times \{\uparrow_\bullet, \dots\}$ by

$$\begin{cases} (q', \downarrow_\bullet, i\text{-th child of } v) & \text{if } p = \downarrow_i^\bullet & (q', \circ, v) & \text{if } p = \circ \\ (q', \uparrow_j^\bullet, \text{parent of } v) & \text{if } p = \uparrow_\bullet \text{ and } v \text{ is the } j\text{-th child of its parent} \end{cases}$$

► **Example 3.6.** Using the idea of Example 3.3, we define a tree-walking transducer that computes the function “ $n \mapsto [1, \dots, n]$ modulo encodings” of Example 2.7. Its set of states is $Q = \{\text{spine}, \text{num}\}$, its initial state is spine and its transitions are

$$\begin{aligned} \delta_S^{\text{root}}(\text{spine}, \circ) &= \delta_S(\text{spine}, \downarrow_\bullet) = \text{cons}((\text{num}, \circ), (\text{spine}, \downarrow_1^\bullet)) \\ \delta_S(\text{num}, \circ) &= \delta_S(\text{num}, \uparrow_1^\bullet) = S((\text{num}, \uparrow_\bullet)) \\ \delta_S^{\text{root}}(\text{num}, \circ) &= \delta_S^{\text{root}}(\text{num}, \uparrow_1^\bullet) = S(0) & \delta_0^{\text{root}}(\text{spine}, \circ) &= \delta_0(\text{spine}, \downarrow_\bullet) = \text{nil} \end{aligned}$$

► **Remark 3.7.** In the above example, the provenance information in $\{\downarrow_\bullet, \dots\}$ plays no role. On the contrary, it is crucial in the former Example 3.4, where we have $\delta_a^{\text{root}}(q, \uparrow_2^\bullet) = 0$ and

$$\delta_a^{\text{root}}(q, \circ) = \delta_a(q, \downarrow_\bullet) = (q, \downarrow_1^\bullet) \quad \underbrace{\delta_a^{\text{root}}(q, \uparrow_1^\bullet) = \delta_a(q, \uparrow_1^\bullet) = (q, \downarrow_2^\bullet)}_{\text{“after returning from the 1st child, the traversal starts visiting the 2nd child”}} \quad \delta_a(q, \uparrow_2^\bullet) = (q, \uparrow_\bullet)$$

“after returning from the 1st child, the traversal starts visiting the 2nd child”

In the definitions of tree-walking automata or transducers in the literature, e.g. [9, 16, 19], transitions often do not have access to this provenance, but instead they can depend on the “child number” i of the current node (such that the node is an i -th child of its parent). One can easily simulate one variant with the other; but if neither of these features were available, the machine model would be strictly weaker [25, Theorem 5.5].

Our main motivation for using provenances instead of child numbers is that, according to [39, Sections 5 and 6], being “direction-determinate” – i.e. knowing which previous node the current configuration came from – is important in the reversible case. This can indeed be observed in the proof of Claim 3.9. (Directed states are used in [15] for a similar reason.)

Let us say that a map $\delta: X \rightarrow \text{Tree}(\Sigma, Y)$ is Y -leaf-injective whenever in the family of trees $(\delta(x) \mid x \in X)$, each $y \in Y$ occurs at most once: y appears in at most one tree of the family, and if it does appear, this tree has a single leaf with label y .

► **Definition 3.8.** A tree-walking transducer $\text{Tree}(\Gamma) \rightarrow \text{Tree}(\Sigma)$ is reversible when all maps δ_a and δ_a^{root} for $a \in \Gamma$ are $(Q \times \{\uparrow_\bullet, \circ, \downarrow_1, \dots, \downarrow_{\text{rk}(a)}^\bullet\})$ -leaf-injective.

Example 3.4 is reversible, but not Example 3.6.

▷ **Claim 3.9 (The meaning of reversibility).** Fix a reversible tree-walking transducer and an input tree. Any configuration C has at most one predecessor configuration, i.e. one configuration C' whose image by the computation-step function contains C as a leaf.

Proof. Let $(q, p, v) = C$, and assume that $(q', p', v') = C'$ exists. The provenance information p tells us where v' is situated in the input tree relatively to v , so we determine v' along with its label a . Among the leaves of $\delta_a(q', p')$ (or $\delta_a^{\text{root}}(q', p')$ if v' is the root), there must be one of the form (q, p) where q and p are related (in a one-to-one fashion) as defined just above Example 3.6. By leaf-injectivity this uniquely determines q' and p' . ◁

4 From the Interaction Abstract Machine to (reversible) TWTs

This section finally discusses the proof of Theorem 1.4. (For lack of space, we focus on stating some key definitions and illustrating the ideas on examples; rigorous details are left to the technical report.) By definition, computing the image of an input tree by a λ -transducer involves normalizing a λ -term. Unfortunately, iterating β -reductions until a normal form is reached requires too much working memory to be implemented by a finite-state device, such as a tree-walking transducer. That is why we rely on other ways to normalize terms of base type o , namely variants of the Interaction Abstract Machine (IAM).

The purely affine IAM. Let us start with a machine that can normalize a purely affine term $v : o$. Intuitively, it moves a token around the *edges* of the syntax tree of v ; we represent the situation where the token is on the edge connecting the subterm t to its context C (such that $C\langle t \rangle = v$) and is moving **down** (resp. **up**) as $C\langle \underline{t} \rangle$ (resp. $C\langle \bar{t} \rangle$). The token carries a small amount of additional information: a “tape” which is a stack using the symbols \bullet and \circ .

We reuse the formalism of tree-generating machines from the previous section, and we denote by X^* (with the Kleene star) the set of lists with elements in X .

- **Definition 4.1.** Let $v : o$ be purely affine. The tree-generating machine $\text{PAIAM}(v)$ has:
- configurations of the form (d, t, C, T) where $d \in \{\downarrow, \uparrow\}$, $C\langle t \rangle = v$ and $T \in \{\bullet, \circ\}^*$ – which we abbreviate as $(C\langle \underline{t} \rangle, T)$ when $d = \downarrow$ or $(C\langle \bar{t} \rangle, T)$ when $d = \uparrow$;
 - the initial configuration $(\underline{t}, \varepsilon) = (\downarrow, t, \langle \cdot \rangle, \varepsilon)$;

■ the following computation-step function:

$$\begin{array}{ll}
 (C\langle \underline{t}u \rangle, T) \mapsto (C\langle \underline{t}u \rangle, \bullet \cdot T) & (C\langle \overline{t}u \rangle, \bullet \cdot T) \mapsto (C\langle \overline{t}u \rangle, T) \\
 (C\langle \underline{t}\overline{u} \rangle, T) \mapsto (C\langle \underline{t}u \rangle, \circ \cdot T) & (C\langle \overline{t}u \rangle, \circ \cdot T) \mapsto (C\langle \underline{t}u \rangle, T) \\
 (C\langle \lambda x. \underline{t} \rangle, T) \mapsto (C\langle \lambda x. \underline{t} \rangle, \bullet \cdot T) & (C\langle \lambda x. \underline{t} \rangle, \bullet \cdot T) \mapsto (C\langle \lambda x. \underline{t} \rangle, T) \\
 (C\langle \lambda x. D\langle \underline{x} \rangle \rangle, T) \mapsto (C\langle \lambda x. D\langle x \rangle \rangle, \circ \cdot T) & (C\langle \lambda x. D\langle x \rangle \rangle, \circ \cdot T) \mapsto (C\langle \lambda x. D\langle \overline{x} \rangle \rangle, T) \\
 (C\langle \underline{c} \rangle, \bullet^{\text{rk}(c)} \cdot T) \mapsto c((C\langle \overline{c} \rangle, \circ \cdot T), (C\langle \overline{c} \rangle, \bullet \cdot \circ \cdot T), \dots, (C\langle \overline{c} \rangle, \bullet^{\text{rk}(c)-1} \cdot \circ \cdot T))
 \end{array}$$

The last rule handles constants $c : o^{\text{rk}(c)} \multimap o$ coming from a fixed ranked alphabet Σ . When $\text{rk}(c) = 0$, the right-hand side is simply the tree with a single node labeled by c ; otherwise, it is a tree of height 1, whose leaves are all configurations.

► **Example 4.2.** Let u and $(t_x)_{x \in \{a,b,c\}}$ be the terms defining the purely affine λ -transducer from Example 1.1. On the term $v = u (t_a (t_b t_c) t_c)$, we have the following IAM execution (where $C \rightsquigarrow^n C'$ means that C rewrites into C' in n steps):

$$\begin{aligned}
 (\underline{v}, \varepsilon) &\rightsquigarrow (\underline{u} (t_a (t_b t_c) t_c), \bullet) \rightsquigarrow ((\lambda f. \underline{f} \ 0) (t_a (t_b t_c) t_c), \varepsilon) \rightsquigarrow ((\lambda f. \underline{f} \ 0) (\dots), \bullet) \\
 &\rightsquigarrow (\overline{(\lambda f. \underline{f} \ 0)} (t_a (t_b t_c) t_c), \circ \bullet) \rightsquigarrow (u (\underline{t_a} (t_b t_c) t_c), \bullet) \rightsquigarrow^2 (u (\underline{t_a} (t_b t_c) t_c), \bullet \bullet \bullet) \\
 &\rightsquigarrow^4 (u ((\lambda \ell. \lambda r. \lambda x. \underline{\ell} (r x)) (t_b t_c) t_c), \bullet) \rightsquigarrow (u (\overline{t_a} (t_b t_c) t_c), \circ \bullet) \rightsquigarrow^2 (\dots \underline{t_b} \dots, \bullet \bullet) \\
 &\rightsquigarrow^3 (u (t_a ((\lambda f. \lambda x. \underline{S} (f x)) t_c) t_c), \bullet) \rightsquigarrow \underline{S}((u (t_a ((\lambda f. \lambda x. \overline{S} (f x)) t_c) t_c), \circ)) \\
 &\rightsquigarrow^* \dots [\text{many steps}] \dots \quad \text{output node (will be the root of the output tree)} \\
 &\rightsquigarrow^* S(S((u (\dots \underline{t_c}), \bullet))) \rightsquigarrow S^3((u (t_a (t_b t_c) \overline{S}), \circ)) \rightsquigarrow S^3((u (\underline{t_a} (t_b t_c) \underline{S}), \circ \circ)) \\
 &\rightsquigarrow^2 S^3((u ((\lambda \ell. \lambda r. \lambda x. \underline{\ell} (r x)) (t_b t_c) t_c), \circ \circ)) \rightsquigarrow S^3((\dots (\overline{r} x) \dots, \circ)) \\
 &\rightsquigarrow S^3((\dots (r \underline{x}) \dots, \varepsilon)) \rightsquigarrow S^3((u ((\lambda \ell. \lambda r. \lambda x. \underline{\ell} (r x)) (t_b t_c) t_c), \circ)) \\
 &\rightsquigarrow^2 S^3((u (\overline{t_a} (t_b t_c) t_c), \bullet \bullet \circ)) \rightsquigarrow^2 S^3((u (\underline{t_a} (t_b t_c) t_c), \circ)) \rightsquigarrow S^3((\underline{u} \dots, \circ \circ)) \\
 &\rightsquigarrow S^3(((\lambda f. \overline{f} \ 0) \dots, \circ)) \rightsquigarrow S^3(((\lambda f. \underline{f} \ 0) \dots, \varepsilon)) \rightsquigarrow S(S(S(0)))
 \end{aligned}$$

Aside from our bespoke extension dedicated to constants from Σ , all the other rules in Definition 4.1 come from the “Linear IAM” described by Accattoli et al. [1, Section 3] (see also [46, §3.1]) – we refer to those papers for high-level explanations of these rules. Despite its name, the Linear IAM also works for affine terms (cf. [32, §3.3.4]). In particular, when run on closed normal forms of type o , namely encoding of trees by Prop. 2.5, the IAM outputs the encoded tree.

► **Proposition 4.3.** For each tree $\tau \in \text{Tree}(\Sigma)$, $(\underline{\tau}, \varepsilon) \rightsquigarrow^* \tau$.

Proof. We strengthen the statement, considering the reduction $(C\langle \underline{\tau} \rangle, \varepsilon) \rightsquigarrow^* \tau$. Then, we proceed by induction on the structure of τ . If τ is a leaf, then we have $\tilde{\tau} = c$ and $\text{rk}(c) = 0$. Thus, we have $(C\langle \underline{c} \rangle, \varepsilon) \rightsquigarrow c$, which concludes this case. Otherwise τ is not a leaf, and in particular it is of the form $c (\tau_1 \dots \tau_k)$, where $k = \text{rk}(c) \geq 1$. Then we have:

$$\begin{aligned}
 (C\langle \underline{c} \underline{\tau}_1 \dots \underline{\tau}_k \rangle, \varepsilon) &\rightsquigarrow^k (C\langle \underline{c} \underline{\tau}_1 \dots \underline{\tau}_k \rangle, \bullet^k) \\
 &\rightsquigarrow c \left((C\langle \overline{c} \underline{\tau}_1 \dots \underline{\tau}_k \rangle, \circ), \dots, (C\langle \overline{c} \underline{\tau}_1 \dots \underline{\tau}_k \rangle, \bullet^{k-1} \cdot \circ) \right) \\
 &\rightsquigarrow^* c \left((C\langle \underline{c} \underline{\tau}_1 \underline{\tau}_2 \dots \underline{\tau}_k \rangle, \varepsilon), \dots, (C\langle \underline{c} \underline{\tau}_1 \dots \underline{\tau}_{k-1} \underline{\tau}_k \rangle, \varepsilon) \right) \\
 &\rightsquigarrow^* c (\tau_1 \dots \tau_k)
 \end{aligned}$$

Where the last reduction comes from applying the induction hypothesis to each configuration. ◀

The soundness of the machine comes from the standard fact (see e.g. [1, 31]) that what is computed by the IAM, in our case the tree, is invariant by β -reduction, the details are in the technical report [38].

► **Theorem 4.4** (Soundness of the purely affine IAM). *For any purely affine term $v : o$, the output of $\text{PAIAM}(v)$ is the unique $\tau \in \text{Tree}(\Sigma)$ such that $\tilde{\tau}$ is the normal form of v .*

Simulating λ -transducers by tree-walking transducers. Fix a purely affine λ -transducer given by $t_a : A^{\text{rk}(a)} \multimap A$ for $a \in \Gamma$ and $u : A \multimap o$. Thanks to Proposition 2.2, we may assume that u and every t_a are in normal form. Proposition 2.4 then tells us that since A is purely affine, so are these terms – and therefore, so is $u \hat{\tau}((t_a)_{a \in \Gamma}) : o$ for any input $\tau \in \text{Tree}(\Gamma)$. Thus, we can use the purely affine IAM to compute the tree encoded by its normal form – which, by definition, gives us the image of τ by our λ -transducer.

To prove the first part of Theorem 1.4, we just need to simulate (in an appropriate sense, made precise in the technical report [38]) $\text{PAIAM}(u \hat{\tau}((t_a)_{a \in \Gamma}))$ by a TWT running on τ . Example 4.5 below illustrates how this works: a configuration of $\text{PAIAM}(u \hat{\tau}((t_a)_{a \in \Gamma}))$ is represented by a configuration on input τ of a TWT in which each state consists of

- a formal symbol in $\{\mathbf{U}, \mathbf{T}, \nabla, \Delta\}$;
 - if the symbol is \mathbf{U} (resp. \mathbf{T}), a position in u (resp. $t_a \diamond_1 \dots \diamond_{\text{rk}(a)}$ for some $a \in \Gamma$);
 - a tape in $\{\bullet, \circ\}^*$ that appears in the run of $\text{PAIAM}(u \hat{\sigma}((t_a)_{a \in \Gamma}))$ for some $\sigma \in \text{Tree}(\Gamma)$.
- The information in the first two components is clearly bounded, but this is not so obvious for the third one. We shall address this issue using Corollary 4.7 below, after the example.

► **Example 4.5.** We translate Example 1.1 to a TWT that has the following run on the input $a_1(b_2(c_3), c_4)$ – note how the steps correspond to those of the IAM run in Example 4.2. This run visits the same input nodes as Example 3.4, in the same order. The only difference is that it stays for longer on each node (\circ appears very frequently).

$$\begin{aligned}
(l, \circ, a_1) &\rightsquigarrow (\mathbf{U}(\underline{\lambda f. f \ 0}, \bullet), \circ, a_1) \rightsquigarrow^3 (\mathbf{U}(\overline{\lambda f. f \ 0}, \circ\bullet), \circ, a_1) \rightsquigarrow (\nabla(\bullet), \circ, a_1) \\
&\rightsquigarrow (\mathbf{T}(\underline{t_a \ \diamond_1 \ \diamond_2}, \bullet\bullet), \circ, a_1) \rightsquigarrow^5 (\mathbf{T}((\lambda\ell. \lambda r. \lambda x. \underline{\ell} (r \ x)) \ \diamond_1 \ \diamond_2, \bullet), \circ, a_1) \\
&\rightsquigarrow (\mathbf{T}(\overline{t_a \ \diamond_1 \ \diamond_2}, \circ\bullet), \circ, a_1) \rightsquigarrow (\nabla(\bullet), \downarrow\bullet, b_2) \rightsquigarrow (\mathbf{T}(\underline{t_b \ \diamond_1}, \bullet\bullet), \circ, b_2) \\
&\rightsquigarrow^3 (\mathbf{T}((\lambda f. \lambda x. \underline{S} (f \ x)) \ \diamond_1, \bullet), \circ, b_2) \rightsquigarrow S((\mathbf{T}((\lambda f. \lambda x. \overline{S} (f \ x)) \ \diamond_1, \circ), \circ, b_2)) \\
&\rightsquigarrow^* \dots [\text{many steps}] \dots \\
&\rightsquigarrow^* S(S((\nabla(\bullet), \downarrow\bullet, c_4))) \rightsquigarrow S^3((\Delta(\circ), \uparrow\bullet, a_1)) \rightsquigarrow (\mathbf{T}(\underline{t_a \ \diamond_1 \ \diamond_2}, \circ\circ), \circ, a_1) \\
&\rightsquigarrow^7 S^3(\mathbf{T}(\overline{t_a \ \diamond_1 \ \diamond_2}, \bullet\bullet\circ), \circ, a_1) \rightsquigarrow^2 S^3((\Delta(\circ), \circ, a_1)) \rightsquigarrow S^3(\mathbf{U}(\underline{u}, \circ\circ), \circ, a_1) \\
&\rightsquigarrow^2 S^3((\mathbf{U}(\lambda f. f \ \underline{0}, \varepsilon), \circ, a_1)) \rightsquigarrow S(S(S(0)))
\end{aligned}$$

Finite states via a typing invariant. To bound the size of tapes $T \in \{\bullet, \circ\}^*$, we leverage the type system. The idea is that a tape that appears in an IAM run “points to” an occurrence of the base type o in the type of the current subterm. Formally, we define inductively:

$$A \not\downarrow \varepsilon = A \quad (A \multimap B) \not\downarrow (\circ \cdot T) = A \not\downarrow T \quad (A \multimap B) \not\downarrow (\bullet \cdot T) = B \not\downarrow T$$

(thus, $o \not\downarrow T$ is undefined for $T \neq \varepsilon$). We then have the following invariant on configurations:

► **Proposition 4.6** (compare with [32, Lemma 32 in §3.3.5]). *Suppose that either $(C(\underline{t}), T)$ or $(C(\overline{t}), T)$ appears in a run of $\text{PAIAM}(v)$ for some $v : o$. If the (not necessarily closed) term t is given the type A as part of a typing derivation for $v : o$, then $A \not\downarrow T = o$.*

Since $|T| \leq \text{height}(A)$ is a necessary condition for $A \not\downarrow T$ to be defined, we immediately get:

► **Corollary 4.7.** *The sizes of the tapes that appear in a run of $\text{PAIAM}(v)$ are bounded by the maximum, over all subterms t of v , of the height of the syntax tree of the type of v .*

68:12 Slightly Non-Linear Higher-Order Tree Transducers

Therefore, the tapes that can appear in a run of $\text{PAIAM}(u \widehat{\tau}((t_a)_{a \in \Gamma}))$ have their size bounded depending only on the subterms of u and of each t_a , independently of $\tau \in \text{Tree}(\Gamma)$. This ensures that the TWT we build from a purely affine λ -transducer has a finite set of states – which concludes our exposition of the key ingredients for the first half of Theorem 1.4.

The almost purely affine case. To prove the second half of Theorem 1.4, we add non-standard rules for let-bindings and !-boxes to the Interaction Abstract Machine.

► **Definition 4.8.** *Let $v : o$ be an almost purely affine term. $\text{APAIAM}(v)$ is the extension of $\text{PAIAM}(v)$ (cf. Definition 4.1) with the following new cases in the computation-step function:*

$$\begin{aligned} (C\langle \text{let } !x = u \text{ in } t \rangle, T) &\mapsto (C\langle \text{let } !x = u \text{ in } \underline{t} \rangle, T) & (C\langle !\underline{t} \rangle, T) &\mapsto (C\langle \underline{!t} \rangle, T) \\ (C\langle \text{let } !x = u \text{ in } \bar{t} \rangle, T) &\mapsto (C\langle \text{let } !x = u \text{ in } \underline{\bar{t}} \rangle, T) \\ (C\langle \text{let } !x = u \text{ in } D\langle \underline{x} \rangle \rangle, T) &\mapsto (C\langle \text{let } !x = \underline{u} \text{ in } D\langle x \rangle \rangle, T) \end{aligned}$$

▷ **Claim 4.9.** Proposition 4.6 extends to $\text{APAIAM}(v)$ with $!A \not\leq T = A \not\leq T$.

The last rule (and the rule for !-boxes) in Definition 4.8 break(s) a key duality principle at work in the purely affine IAM (and suggested by the layout of Definition 4.1): if any rule – except the one for constants from Σ – sends a configuration κ_1 to another configuration κ_2 , then there is a dual rule sending κ_2^\perp to κ_1^\perp , where $(C\langle \underline{t} \rangle, T)^\perp = (C\langle \bar{t} \rangle, T)$ and conversely $(C\langle \bar{t} \rangle, T)^\perp = (C\langle \underline{t} \rangle, T)$.

In fact, our new rule for let-bound variables \underline{x} *cannot* have a dual, because it is not injective. Indeed, consider a term of the form $C\langle \text{let } !x = u \text{ in } t \rangle$ where t contains multiple occurrences of x , i.e. $t = D_1\langle x \rangle = D_2\langle x \rangle$ for some contexts $D_1 \neq D_2$ – this may happen, since let-bound variables are not affine. Then for any $T \in \{\bullet, \circ\}^*$, the computation-step function sends both $(C\langle \text{let } !x = u \text{ in } D_1\langle \underline{x} \rangle \rangle, T)$ and $(C\langle \text{let } !x = u \text{ in } D_2\langle \underline{x} \rangle \rangle, T)$ to the same configuration $(C\langle \text{let } !x = \underline{u} \text{ in } t \rangle, T)$ due to the \underline{x} rule. This is why *reversible* TWTs can simulate the purely affine IAM, but not the *almost* purely affine IAM.

To be sure that the missing dual rule is unnecessary, we show that configurations of the form $(C\langle \text{let } !x = \bar{u} \text{ in } t \rangle, T)$ cannot occur in an actual run. Assume the opposite for the sake of contradiction. The typing rule for let-bindings forces the type of u to have the form $!A$, and by almost pure affineness, it must be $!o$. By Claim 4.9, $!o \not\leq T = o \not\leq T$; therefore, $T = \varepsilon$, which contradicts another invariant:

► **Proposition 4.10.** *If $(C\langle \bar{t} \rangle, T)$ (resp. $(C\langle \underline{t} \rangle, T)$) appears in a run of $\text{APAIAM}(v)$ for some almost purely affine $v : o$, then T contains an odd (resp. even) number of \circ s.*

(The same reasoning also shows that $(C\langle !\bar{t} \rangle, T)$ cannot occur in a run).

Having ruled out these problematic configurations, we can establish soundness for the almost purely affine IAM exactly as before, extending Theorem 4.4.

► **Proposition 4.11** (Soundness of the almost purely affine IAM). *For any almost purely affine term $v : o$, the output of $\text{APAIAM}(v)$ is the unique $\tau \in \text{Tree}(\Sigma)$ such that $\tilde{\tau}$ is the normal form of v .*

The simulation by tree-walking transducers then follows the same pattern than in the purely affine case (except that we do not get reversibility).

5 From the almost !-depth 1 IAM to invisible pebbles

Now that we have seen how to prove Theorem 1.4, let us apply the same methodology to the almost !-depth 1 case, with another variation on the Interaction Abstract Machine.

The key challenge is that we can no longer rule out positions of the form $C\langle \text{let } !x = \bar{u} \text{ in } t \rangle$ for the IAM token. If x has multiple occurrences in t , we need some information to know which of these occurrences we should move to. The standard solution to this problem is to enrich the IAM configurations with another data structure – cf. the “boxes stack” of [14] or the “log” of [1]. In our simple low-depth case, a stack of variable occurrences will be enough.

The almost !-depth 1 IAM. Let $v : o$ be an almost !-depth 1 term. To compute its normal form, we introduce a tree-generating machine whose configurations are of the form $(C\langle \underline{t} \rangle, T, L)$ or $(C\langle \bar{t} \rangle, T, L)$, where $C\langle t \rangle = v$, $T \in \{\bullet, \circ, l\}^*$, and *logs* L and logged positions l are defined by mutual induction as follows (please notice that $n \in \{0, 1\}$ for !-depth 1 terms):

$$l ::= (D_n, L_n) \quad L_0 ::= \varepsilon \quad L_n ::= l \cdot L_{n-1}$$

The initial configuration is $(\underline{v}, \varepsilon, \varepsilon)$. To define the computation-step function, we start by reusing all the rules of the almost purely affine IAM (Definition 4.8) except the \underline{x} -rule for let-bound x , and the one for !-boxes, adapting them so that they do not change the log L . We then add the rules below, where C_i ranges over contexts of depth $i \in \{0, 1\}$ (as defined in Section 2) and $A \neq o$. Please notice that transition rules now depend also on the *type* of the current subterm, indeed we have to distinguish between the “almost” and the “depth-1” exponentials:

$$\begin{aligned} (C_0\langle \text{let } !x = u \text{ in } D_n\langle \underline{x}^{!A} \rangle \rangle, T, L_n) &\mapsto (C_0\langle \text{let } !x = \underline{u}^{!A} \text{ in } D_n\langle x \rangle \rangle, (D_n, L_n) \cdot T, \varepsilon) \\ (C_0\langle \text{let } !x = \bar{u}^{!A} \text{ in } D_n\langle x \rangle \rangle, (D_n, L_n) \cdot T, \varepsilon) &\mapsto (C_0\langle \text{let } !x = u \text{ in } D_n\langle \bar{x}^{!A} \rangle \rangle, T, L_n) \\ (C\langle \text{let } !x = u \text{ in } D_n\langle \underline{x}^{!o} \rangle \rangle, T, L_n \cdot L) &\mapsto (C\langle \text{let } !x = \underline{u}^{!o} \text{ in } D_n\langle x \rangle \rangle, T, L) \quad (\text{non-reversible}) \\ (C_0\langle \underline{t}^{!A} \rangle, l \cdot T, \varepsilon) &\mapsto (C_0\langle \underline{t}^A \rangle, T, l) \quad (C_0\langle \bar{t}^A \rangle, T, l) \mapsto (C_0\langle \bar{t}^{!A} \rangle, l \cdot T, \varepsilon) \\ (C\langle \underline{t}^{!o} \rangle, T, L) &\mapsto (C\langle \underline{t}^o \rangle, T, L) \quad (\text{non-reversible}) \end{aligned}$$

Again, this device, which is just a specialization of the standard IAM (but again non-reversible in order to handle linearly the almost affine terms), successfully normalizes almost !-depth 1 terms of base type. Next, we would like to simulate it by some automaton model, to get a counterpart of Theorem 1.4 in this setting. The problem now is that both the log L and the tape T do not fit into the finite state of a tree-walking transducer, since their size cannot be statically bounded. Therefore, we need to target a more powerful machine model.

Invisible pebbles. Luckily, a suitable device has already been introduced by Engelfriet, Hoogboom and Samwel [18]: the *invisible pebble tree transducer* (IPTT). Informally, it is a TWT extended with the ability to put down pebbles on input nodes. The pebbles have colors that are taken in a finite set. They can be later examined and removed: an IPTT can check whether the *last pebble to have been put down* is on the current position, and if so, it can observe its color, and perhaps decide to remove it. The “invisible” part means that only the last pebble can be seen. Thus, the lifetimes of the pebbles follow a *stack discipline* (last put down, first removed). The number of pebbles used in a computation may be unbounded.

► **Definition 5.1** ([18]). An invisible pebble tree transducer $\text{Tree}(\Gamma) \rightarrow \text{Tree}(\Sigma)$ is made of:

- a finite set of states Q with an initial state $q_0 \in Q$
- a finite set of colors \mathcal{C}
- a (partial) transition function that sends tuples consisting of
 - an input letter $a \in \Gamma$, a state $q \in Q$, a provenance $p \in \{\downarrow_\bullet, \circlearrowleft, \uparrow_1^\bullet, \dots, \uparrow_{\text{rk}(a)}^\bullet\}$
 - a boolean `isRoot` which must be false if $p = \downarrow_\bullet$.
 - a value z which is either a color in \mathcal{C} or the symbol `None`
 to $\text{Tree}(\Sigma, Q \times (\{\uparrow_\bullet, \circlearrowleft, \downarrow_1^\bullet, \dots, \downarrow_k^\bullet, \text{remove}\} \cup \{\text{put}_c \mid c \in \mathcal{C}\}))$
prohibited if `isRoot` is true prohibited if $z = \text{None}$

Note that removing z in the arguments and `remove/putc` in the codomain would just yield an alternative presentation of tree-walking transducers (Definition 3.5).

The set of configurations of an invisible pebble tree transducer on an input tree τ is

$$\underbrace{Q \times \{\downarrow_\bullet, \circlearrowleft, \uparrow_1^\bullet, \dots\}}_{\text{TWT configuration}} \times \underbrace{\{\text{nodes of } \tau\}}_{\text{pebble stack}} \times (\mathcal{C} \times \{\text{nodes of } \tau\})^*$$

The transition function of the IPTT determines a computation-step function by extending Definition 3.5 in the expected way (the transducer stays at the same node after a `remove` or `putc` instruction). Here is an example of a configuration over $\tau = a_1(b_2(c_3), c_4)$ for some invisible pebble tree transducer: $(q, \downarrow_\bullet, c_3, [(a, b_2), (b, c_3)])$. The top of the stack is the leftmost element the list: it is an \mathbf{a} -colored pebble on position b_2 . Since this differs from the current node c_3 , the transducer does not see any pebble ($z = \text{None}$) even though there is a \mathbf{b} -colored pebble on c_3 further down the stack. If we execute the instruction `puta` while transitioning to state q' , we get the configuration $(q', \circlearrowleft, c_3, [(a, c_3), (a, b_2), (b, c_3)])$. In that new configuration, the IPTT now sees the topmost pebble ($z = \mathbf{a}$) and is thus allowed to `remove` it.

Outcome of the simulation. We use a single stack implementation (detailed in the technical report [38]) of the IAM presented above to compile λ -transducers to IPTTs as in the previous section, thus establishing the following comparison in expressive power:

► **Lemma 5.2.** *Almost !-depth 1 λ -transducer \subseteq invisible pebble tree transducer \equiv MSOT-S².*

Here, the equivalence between IPTT and MSOT-S² is a rephrasing of a result of Engelfriet et al. [18, Theorem 53], as explained in [35, §3.3].

6 Expressiveness of λ -transducers with preprocessing

Now, let us prove Theorems 1.5 and 1.7. We first note that the left-to-right inclusions are immediate consequences of Theorem 1.4 and Lemma 5.2 combined with the following facts:

- MSOT-S (and, therefore, MSOT-S²) are closed under precomposition by MSO relabeling (cf. [7, Section 3] where MSOT-S are called “MSO term graph transductions”);
- TWT \subset MSOT-S (a slight variant of [7, Theorem 9], cf. [35, §3.2]);
- TWT of linear growth \subset MSOT (see e.g. [19, §6.2]), and purely affine λ -transducers have linear growth because the size of purely affine terms is non-increasing during β -reduction. (“ f has linear growth” means that $|f(t)| = O(|t|)$.) Next, we turn to the converse inclusions.

MSOT \subseteq purely affine λ -transducer \circ MSO relabeling. We derive this from the results of Gallot, Lemay and Salvati [22, 23]. First, we introduce a slight generalization of their machine model for MSOTs [22, §2.3]. Their model involves bottom-up regular lookahead, but as usual in automata theory, this feature can be simulated by preprocessing by an MSO relabeling; this is why we do not include it in our version.

► **Definition 6.1.** A GLS-transducer $\text{Tree}(\Gamma) \rightarrow \text{Tree}(\Sigma)$ consists of:

- a finite set Q of states, with a family $(A_q)_{q \in Q}$ of purely affine types;
- an initial state $q_0 \in Q$ and an output term $u : A_{q_0} \multimap o$ – which, like all the terms t below, may use constants $c : o^{\text{rk}(c)} \multimap o$ for $c \in \Sigma$;
- for each $q \in Q$ and $a \in \Gamma$, a rule $q \langle a(x_1, \dots, x_{\text{rk}(a)}) \rangle \rightarrow t \ q_1 \langle x_1 \rangle \dots q_{\text{rk}(a)} \langle x_{\text{rk}(a)} \rangle$ where the q_i and $t : A_{q_1} \multimap \dots \multimap A_{q_{\text{rk}(a)}} \multimap A_q$ are chosen depending on (q, a) .

The semantics is that a GLS-transducer performs a top-down traversal $q_0 \langle \tau \rangle \rightarrow^* \tau^\downarrow$ of its input tree τ which builds a λ -term $\tau^\downarrow : A_{q_0}$. The normal form of $u \tau^\downarrow$ then encodes the output tree. Our model is a bit more syntactically permissive than that of Gallot et al. (theirs would correspond to using linear rather than affine terms, and forcing u to be $\lambda x. x$ – so $A_{q_0} = o$); therefore, it can compute at least everything that their model can:

► **Theorem 6.2** (from [22, Theorem 3]). *MSOT \subseteq GLS-transducer \circ MSO relabeling.*

We derive our desired result on λ -transducers in two steps.

- Every GLS-transducer can be made “type-constant”: $\exists A : \forall q, A_q = A$. This uses an encoding trick, detailed in the technical report [38], that preserves pure affineness, but not linearity.
- A type-constant GLS-transducer can be turned into an MSO relabeling (that adds to each node its top-down propagated state) followed by a purely affine λ -transducer (which is just a GLS-transducer with $|Q| = 1$).

From Theorem 6.2, we thus get $\text{MSOT} \subseteq \text{purely affine } \lambda\text{-transducer} \circ (\text{MSO relabeling})^2$, and since MSO relabelings are closed under composition [7, §3], we are done.

MSOT-S \subseteq almost purely affine λ -transducer \circ MSO relabeling. Following the same recipe as above, we reduce this to Gallot et al.’s characterization of MSOT-S [22, Theorem 3]. Since they use Kanazawa’s almost linear λ -terms [28] (which we discussed in the introduction), we need to translate such terms into our almost purely affine terms. Let us introduce the abbreviation $\lambda!x. t = (\lambda y. \text{let } !x = y \text{ in } t)$ where y is a fresh variable. We define inductively: $?c = \lambda!x_1. \dots \lambda!x_{\text{rk}(c)}. !(c x_1 \dots x_{\text{rk}(c)})$ for constants c in a ranked alphabet Σ , and

$$?x = \begin{cases} !x & \text{if } x : o \\ x & \text{otherwise} \end{cases} \quad ?(\lambda x. t) = \begin{cases} \lambda!x. ?t & \text{if } x : o \\ \lambda x. ?x & \text{otherwise} \end{cases} \quad ?(t u) = (?t) (?u)$$

▷ **Claim 6.3.** Let $t : A$ be almost affine as defined in [27]. Then $?t$ is almost purely affine, with type $A\{o := !o\}$. When $t : o$, if $t \rightarrow_\beta^* \tilde{\tau}$ (for some $\tau \in \text{Tree}(\Sigma)$) then $?t \rightarrow_\beta^* !\tilde{\tau}$.

The inductive rule for application implies that $?(u \widehat{\tau}((t_a)_{a \in \Gamma})) = (?u) \widehat{\tau}((?t_a)_{a \in \Gamma})$, so the above claim allows us to translate λ -transducers using almost affine terms *à la* Kanazawa.

MSOT-S² \subseteq almost !-depth 1 λ -transducer \circ relabeling. Having just finished proving Theorem 1.5 above, we may use it right away:

$$\begin{aligned} \text{MSOT-S}^2 &\equiv \text{almost purely affine } \lambda\text{-transducer} \circ \text{MSO relabeling} \circ \text{MSOT-S} \quad (\text{Thm. 1.5}) \\ &\equiv \text{almost purely affine } \lambda\text{-transducer} \circ \text{MSOT-S} \quad (*) \\ &\equiv (\text{almost purely affine } \lambda\text{-transducer})^2 \circ \text{MSO relabeling} \quad (\text{Thm. 1.5}) \end{aligned}$$

The line (*) above relies on the fact that MSOT-S are closed under *post*composition by MSO relabelings, cf. [35, §3.3]. To conclude, we apply the composition property of λ -transducers (Proposition 1.8), noting that if A and B are almost purely affine types, then $A\{o := B\}$ is almost !-depth 1 (indeed, every ‘!’ in it is applied to either o or B).

7 Conclusion

In this paper, we established several expressivity results relating a typed λ -calculus to tree transducers. This can be seen as furthering Nguyễn and Pradic’s “implicit automata” research programme [37], even though the formal setting is slightly different; indeed, we settle one of their conjectures in Corollary 1.2. From a purely automata-theoretic perspective, our characterization of MSOT-S² is the first that involves a “one-way” device, performing a single bottom-up pass on its input (modulo preprocessing).

The equivalences between “one-way” λ -transducers and tree-walking / invisible pebble tree transducers can be seen as a trade-off between a sophisticated memory (higher-order data) and freedom of movement on the input (tree-walking reading head). This is arguably a sort of qualitative space/time trade-off (more movement means more computation steps). This is similar to the reasons that led the Geometry of Interaction to be used in implicit computational complexity when dealing with space complexity classes – an application area pioneered by Schöpp [44, 45] and leading to several further works [13, 31]. These successes even led to the belief that the GoI should give a reasonable space cost model, that is to say comparable with the one of Turing machines; but this belief is now known to be wrong in the general case of the untyped λ -calculus [2].

7.1 More related work

Katsumata [29] has connected a categorical version of the GoI (the “Int-construction”) to attribute grammars [30], which are “essentially [a] notational variation” on tree-walking transducers (quoting Courcelle & Engelfriet [12, §8.7]). Recently, Pradic and Price [40] have used a “planar” version of this categorical GoI in order to prove an “implicit automata” theorem. Further GoI-automata connections of this kind are discussed in [36, §1.1].

Our methodology of connecting λ -calculus and automata via abstract machines may be compared to Salvati and Walukiewicz’s [43] use of Krivine machines in the theory of higher-order recursion schemes. Clairambault and Murawski [11] also compile affine recursion schemes to automata using a game semantics that can be seen as a denotational counterpart of the operational Interaction Abstract Machine. Ghica exploited ideas from the GoI and game semantics, to design a compiler from a higher-order functional language directly to digital circuits [24], in particular targeting Mealy machines.

► **Remark 7.1.** The aforementioned work [11] is a rare example of application of some GoI variant to a setting that features the *additive connectives* of linear logic. It yields a translation from λ -terms to infinite-state systems, making it unsuited to our purposes. In most versions of the GoI, the support for additives is not as satisfactory as their handling of additive-free linear logic – a well-known issue in the linear logic community.

This obstruction also motivated our choice, discussed in the introduction, to follow the approach of Gallot et al. [22, 23] rather than Nguyễn and Pradic’s “implicit automata” [37, 34]. Indeed, the latter’s solution to overcome the limitations evidenced by Corollary 1.2 is to work with a linear λ -calculus with additive connectives, enabling more flexible linear usage patterns. This is analogous (see [34, Remark 6.0.1] for an actual technical connection) to moving from “strongly single use” to “single use” macro tree transducers [20, Section 5].

Salvati has shown [41] that the string languages defined by abstract categorical grammars, which are very close to our purely linear λ -transducers, coincide with the output languages of tree-to-string tree-walking transducers. He explains in his habilitation thesis [42, §3.2] that the proof ideas are similar to a game semantics of multiplicative linear logic – and the latter is closely related to GoI, as mentioned above. It would be interesting to understand to which extent his approach implicitly resembles ours, despite a very different presentation.

We also note that in the same paper that introduces almost linear λ -terms [28], Kanazawa studies a notion of “links in typed λ -terms” that looks like a form of GoI. However, these links are only well-behaved for λ -terms in normal form, while the Interaction Abstract Machine does not have this drawback. Finally, let us stress that our use of the IAM has the advantage, compared to the aforementioned works [29, 11, 41, 26], of adapting to the presence of the exponential modality “!”. This is crucial in our proof of Theorem 1.7.

7.2 Perspectives

The obvious direction for further work is to study the MSOT-S^{k+1} and almost !-depth k hierarchies for $k \geq 2$. While the argument at the end of Section 6 easily generalizes to show that the former is included in the latter, we have no reason to believe that they coincide. As a more modest conjecture (“!-depth 1” means “no nested ‘!’s”):

► **Conjecture 7.2.** *!-depth 1 λ -transducer \circ MSO relabeling \equiv MSOT \circ MSOT-S.*

We believe that the *reversible* tree-walking transducers that we have introduced also deserve to be studied further. Indeed, we expect that they should be closed under composition (cf. [15] over strings) and verify the “single-use restriction” of [12, §8.2]; the latter would imply that they can be translated into MSO transductions.

References

- 1 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The machinery of interaction. In *PPDP '20: 22nd International Symposium on Principles and Practice of Declarative Programming, Bologna, Italy, 9-10 September, 2020*, pages 4:1–4:15. ACM, 2020. doi:10.1145/3414080.3414108.
- 2 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The space of interaction. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2021. doi:10.1109/LICS52264.2021.9470726.
- 3 Beniamino Accattoli and Delia Kesner. The structural λ -calculus. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.
- 4 Rajeev Alur and Loris D’Antoni. Streaming Tree Transducers. *Journal of the ACM*, 64(5):1–55, August 2017. doi:10.1145/3092842.
- 5 Toshiyasu Arai. 10th Asian Logic Conference. *The Bulletin of Symbolic Logic*, 15(2):246–265, 2009. doi:10.2178/bs1/1243948490.

- 6 Andrew Barber. Dual Intuitionistic Linear Logic. Technical report ECS-LFCS-96-347, LFCS, University of Edinburgh, 1996. URL: <http://www.lfcs.inf.ed.ac.uk/reports/96/ECS-LFCS-96-347/>.
- 7 Roderick Bloem and Joost Engelfriet. A Comparison of Tree Transductions Defined by Monadic Second Order Logic and by Attribute Grammars. *Journal of Computer and System Sciences*, 61(1):1–50, August 2000. doi:10.1006/jcss.1999.1684.
- 8 Mikołaj Bojańczyk. Who to cite: MSO transductions, December 2019. URL: <https://web.archive.org/web/20230810161232/https://www.mimuw.edu.pl/~bojan/posts/who-to-cite-mso-transductions>.
- 9 Mikołaj Bojańczyk and Thomas Colcombet. Tree-walking automata do not recognize all regular languages. *SIAM Journal on Computing*, 38(2):658–701, 2008. doi:10.1137/050645427.
- 10 Mikołaj Bojańczyk and Amina Doumane. First-order tree-to-tree functions, 2020. Corrected version with erratum of a LICS 2020 paper. arXiv:2002.09307v2.
- 11 Pierre Clairambault and Andrzej S. Murawski. On the Expressivity of Linear Recursion Schemes. In Peter Rossmanith, Pinar Heggenes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.50.
- 12 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic. A language-theoretic approach*. Encyclopedia of Mathematics and its applications, Vol. 138. Cambridge University Press, June 2012. Collection Encyclopedia of Mathematics and Applications, Vol. 138. URL: <https://hal.archives-ouvertes.fr/hal-00646514>.
- 13 Ugo Dal Lago and Ulrich Schöpp. Computation by interaction for space-bounded functional programming. *Information and Computation*, 248:150–194, 2016. doi:10.1016/j.ic.2015.04.006.
- 14 Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal λ -machines. *Theoretical Computer Science*, 227(1):79–97, September 1999. doi:10.1016/S0304-3975(99)00049-3.
- 15 Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. On reversible transducers. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 113:1–113:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.113.
- 16 Joost Engelfriet. The time complexity of typechecking tree-walking tree transducers. *Acta Informatica*, 46(2):139–154, 2009. doi:10.1007/s00236-008-0087-y.
- 17 Joost Engelfriet. Context-free grammars with storage, 2014. Revised version of a 1986 technical report. arXiv:1408.0683.
- 18 Joost Engelfriet, Hendrik Jan Hoogeboom, and Bart Samwel. XML navigation and transformation by tree-walking automata and transducers with visible and invisible pebbles. *Theoretical Computer Science*, 850:40–97, January 2021. doi:10.1016/j.tcs.2020.10.030.
- 19 Joost Engelfriet, Kazuhiro Inaba, and Sebastian Maneth. Linear-bounded composition of tree-walking tree transducers: linear size increase and complexity. *Acta Informatica*, 58(1-2):95–152, 2021. doi:10.1007/s00236-019-00360-8.
- 20 Joost Engelfriet and Sebastian Maneth. Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations. *Information and Computation*, 154(1):34–91, October 1999. doi:10.1006/inco.1999.2807.
- 21 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26(1/2):131–192, 1988. doi:10.1007/BF02915449.
- 22 Paul Gallot, Aurélien Lemay, and Sylvain Salvati. Linear high-order deterministic tree transducers with regular look-ahead. In Javier Esparza and Daniel Král’, editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 38:1–38:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.38.

- 23 Paul D. Gallot. *Safety of transformations of data trees: tree transducer theory applied to a verification problem on shell scripts*. PhD thesis, Université de Lille, December 2021. URL: <https://theses.hal.science/tel-03773108>.
- 24 Dan R. Ghica. Geometry of Synthesis: A Structured Approach to VLSI Design. In Martin Hofmann and Matthias Felleisen, editors, *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*, pages 363–375. ACM, 2007. doi:10.1145/1190216.1190269.
- 25 Tsutomu Kamimura and Giora Slutzki. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control*, 49(1):10–51, 1981. doi:10.1016/S0019-9958(81)90438-1.
- 26 Makoto Kanazawa. A lambda calculus characterization of MSO definable tree transductions, September 2008. Talk given at the 10th Asian Logic Conference, Kobe University, Japan. Slides available at https://makotokanazawa.ws.hosei.ac.jp/talks/asian_logic.pdf. Abstract available at [5, p. 250–251].
- 27 Makoto Kanazawa. Almost affine lambda terms. In Andrzej Indrzejczak, Janusz Kaczmarek, and Michał Zawidzki, editors, *Trends in Logic XIII. Gentzen’s and Jaśkowski’s Heritage. 80 Years of Natural Deduction and Sequent Calculi*, pages 131–148. Wydawnictwo Uniwersytetu Łódzkiego, 2014.
- 28 Makoto Kanazawa. Parsing and generation as datalog query evaluation. *IfCoLog Journal of Logics and their Applications (FLAP)*, 4(4), 2017. Long version of an ACL 2007 paper. URL: <http://www.collegepublications.co.uk/downloads/ifcolog00013.pdf>.
- 29 Shin-ya Katsumata. Attribute grammars and categorical semantics. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II – Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 271–282. Springer, 2008. doi:10.1007/978-3-540-70583-3_23.
- 30 Donald E. Knuth. The genesis of attribute grammars. In Pierre Deransart and Martin Jourdan, editors, *Attribute Grammars and their Applications, International Conference WAGA, Paris, France, September 19-21, 1990, Proceedings*, volume 461 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1990. doi:10.1007/3-540-53101-7_1.
- 31 Damiano Mazza. Simple parsimonious types and logarithmic space. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPICs*, pages 24–40. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.CSL.2015.24.
- 32 Damiano Mazza. *Polyadic Approximations in Logic and Computation*. Habilitation à diriger des recherches, Université Paris XIII (Sorbonne Paris Nord), November 2017. URL: <https://theses.hal.science/tel-04238579>.
- 33 Anca Muscholl and Gabriele Puppis. The Many Facets of String Transducers. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.STACS.2019.2.
- 34 Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory*. PhD thesis, Université Paris XIII (Sorbonne Paris Nord), December 2021. URL: <https://theses.hal.science/tel-04132636>.
- 35 Lê Thành Dũng Nguyễn. Two or three things i know about tree transducers, 2024. arXiv:2409.03169.
- 36 Lê Thành Dũng Nguyễn, Camille Noûs, and Cécilia Pradic. Two-way automata and transducers with planar behaviours are aperiodic, 2023. arXiv:2307.11057.

- 37 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ -calculi I: aperiodicity in a non-commutative logic. 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 135:1–135:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.135.
- 38 Lê Thành Dũng Nguyễn and Gabriele Vanoni. Slightly non-linear higher-order tree transducers. *CoRR*, abs/2402.05854, 2024. doi:10.48550/arXiv.2402.05854.
- 39 Alexander Okhotin. Graph-walking automata: From whence they come, and whither they are bound. In Michal Hospodár and Galina Jirásková, editors, *Implementation and Application of Automata – 24th International Conference, CIAA 2019, Košice, Slovakia, July 22-25, 2019, Proceedings*, volume 11601 of *Lecture Notes in Computer Science*, pages 10–29. Springer, 2019. doi:10.1007/978-3-030-23679-3_2.
- 40 Cécilia Pradic and Ian Price. Implicit automata in λ -calculi iii: affine planar string-to-string functions. *Electronic Notes in Theoretical Informatics and Computer Science*, Volume 4 – Proceedings of MFPS XL, December 2024. doi:10.46298/entics.14804.
- 41 Sylvain Salvati. Encoding second order string ACG with deterministic tree walking transducers. In Shuly Wintner, editor, *The 11th conference on Formal Grammar*, FG Online Proceedings, pages 143–156, Malaga, Spain, 2006. Paola Monachesi; Gerald Penn; Giorgio Satta; Shuly Wintner, CSLI Publications. URL: <https://web.stanford.edu/group/cslipublications/cslipublications/FG/2006/salvati.pdf>.
- 42 Sylvain Salvati. *Lambda-calculus and formal language theory*. Habilitation à diriger des recherches, Université de Bordeaux, December 2015. URL: <https://theses.hal.science/tel-01253426>.
- 43 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Mathematical Structures in Computer Science*, 26(7):1304–1350, October 2016. doi:10.1017/S0960129514000590.
- 44 Ulrich Schöpp. Space-efficient computation by interaction. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 606–621. Springer, 2006. doi:10.1007/11874683_40.
- 45 Ulrich Schöpp. Stratified bounded affine logic for logarithmic space. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wrocław, Poland, Proceedings*, pages 411–420. IEEE Computer Society, 2007. doi:10.1109/LICS.2007.45.
- 46 Gabriele Vanoni. *On Reasonable Space and Time Cost Models for the λ -Calculus*. PhD thesis, Alma Mater Studiorum – Università di Bologna, June 2022. doi:10.48676/unibo/amsdottorato/10276.




A Dichotomy Theorem for Ordinal Ranks in MSO

Damian Niwiński   

Institute of Informatics, University of Warsaw, Poland

Paweł Parys   

Institute of Informatics, University of Warsaw, Poland

Michał Skrzypczak   

Institute of Informatics, University of Warsaw, Poland

Abstract

We focus on formulae $\exists X. \varphi(\vec{Y}, X)$ of monadic second-order logic over the full binary tree, such that the witness X is a well-founded set. The ordinal rank $\text{rank}(X) < \omega_1$ of such a set X measures its depth and branching structure. We search for the least upper bound for these ranks, and discover the following dichotomy depending on the formula φ . Let η_φ be the minimal ordinal such that, whenever an instance \vec{Y} satisfies the formula, there is a witness X with $\text{rank}(X) \leq \eta_\varphi$. Then η_φ is either strictly smaller than ω^2 or it reaches the maximal possible value ω_1 . Moreover, it is decidable which of the cases holds. The result has potential for applications in a variety of ordinal-related problems, in particular it entails a result about the closure ordinal of a fixed-point formula.

2012 ACM Subject Classification Theory of computation \rightarrow Automata over infinite objects; Theory of computation \rightarrow Tree languages

Keywords and phrases dichotomy result, limit ordinal, countable ordinals, nondeterministic tree automata

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.69

Related Version *Full Version*: <https://arxiv.org/abs/2501.05385>

Funding All authors supported by the National Science Centre, Poland (grant no. 2021/41/B/ST6/03914).

Acknowledgements The authors would like to thank Marek Czarnecki for preliminary discussions on related subjects.

1 Introduction

The concept of well-founded relation plays a central role in foundations of mathematics. It gives rise to ordinal numbers, which underlie the basic results in set theory, for example that any two sets can be compared in cardinality. Well-foundedness is no less important in the realm of computer science, where it often underlies the proofs of *termination* of non-deterministic processes, especially when no efficient bound on the length of a computation is known. In such cases, the complexity of possible executions is usually measured using an ordinal called *rank*. Such a rank can be seen as a measure of the *depth* of the considered partial order, taking into account suprema of lengths of possible descending chains. Estimates on a rank can provide upper-bounds on the computational complexity of the considered problem [26].

In this work, we adopt the perspective of mathematical foundations of program verification and model-checking. We focus on the monadic second-order logic (MSO) interpreted in the infinite binary tree (with the left and right successors as the only non-logical predicates), which is one of the reference formalisms in the area [29]. The famous Rabin Tree Theorem [25] established its decidability, but – half a century after its introduction – the theory is still an object of study. On one hand, it has led to numerous extensions, often shifting the decidability



© Damian Niwiński, Paweł Parys, and Michał Skrzypczak;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 69; pp. 69:1–69:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



result far beyond the original theory (see e.g. [4, 24]). On the other hand, a number of natural questions regarding Rabin's theory remain still open, including a large spectrum of *simplification* problems. For example, we still do not know whether we can decide if a given formula admits an equivalent form with all quantifiers restricted to finite sets. Similar questions have been studied in related formalisms like μ -calculus or automata; for example if we can effectively minimise the Mostowski index of a parity tree automaton [11, 15], or the $\mu\nu$ -alternation depth of a μ -calculus formula [5].

On positive side, some decidability questions have been solved by reduction to the original theory. For example, it has been observed [22] that for a given formula $\varphi(\vec{X})$, the *cardinality* of the family of tuples of sets \vec{X} satisfying $\varphi(\vec{X})$ can be computed; this cardinality can be either finite, \aleph_0 , or \mathfrak{c} . Later on, Bárány, Kaiser, and Rabinovich [3] proved a more general result: they studied *cardinality quantifiers* $\exists^{\geq \kappa} X. \varphi(\vec{Y}, X)$, stating that there are at least κ distinct sets X satisfying $\varphi(\vec{Y}, X)$, and showed that these quantifiers can be expressed in the standard syntax of MSO; thus the extended theory remains decidable.

In the present work, instead of asking *how many* sets X witness to the formula $\exists X. \varphi(\vec{Y}, X)$, we ask how *complex* these witnesses must be in terms of their depth-and-branching structure. A set X of nodes of a tree is *well-founded* if it contains no infinite chain with respect to the descendant order. In this case, a countable ordinal number $\text{rank}(X)$ is well defined (see Section 2 below); intuitively, the smaller $\text{rank}(X)$, the simpler the set X is, in terms of its *branching structure*.

We consider formulae of the form $\exists X. \varphi(\vec{Y}, X)$, where φ is an arbitrary formula of MSO (it may contain quantifiers). We assume that, whenever the formula is satisfied for some valuation of variables \vec{Y} , the value of X witnessing the formula is a well-founded set. Note that well-foundedness of a set is expressible in MSO (it suffices to say that each branch contains only finitely many nodes in X), hence the requirement can be expressed within φ . For a fixed formula as above, we ask what is the minimal ordinal η_φ , such that the rank of a witness can be bounded by η_φ ,

$$\eta_\varphi \stackrel{\text{def}}{=} \sup_{\vec{Y}} \min_X \text{rank}(X), \quad (1.1)$$

where \vec{Y} and X range, as expected, over the values satisfying $\varphi(\vec{Y}, X)$.

Since η_φ is a supremum of countable ordinals, its value is at most ω_1 (the least uncountable ordinal). It can be achieved, for example, by the formula “ $X = Y$ and Y is a well-founded set”, as there are well-founded sets of arbitrarily large countable ranks. On the other hand, for each pair of natural numbers (k, l) , one can construct a formula φ with $\text{rank } \eta_\varphi = \omega \cdot k + l$ in an analogous way to Czarnecki [12], see Lemma 9.4 below. The main result of this work shows that no other ordinals can be obtained:

► **Theorem 1.1.** *For any formula $\exists X. \varphi(\vec{Y}, X)$ as above, the ordinal η_φ is either strictly smaller than ω^2 or equal to ω_1 . Moreover, it can be effectively decided which of the cases holds. In the former case, it is possible to compute a number $N \in \mathbb{N}$ such that $\eta_\varphi < \omega \cdot N$.*

We also show that, in contrast to the aforementioned cardinality quantifiers, the property that $\text{rank}(X)$ is smaller than ω^2 cannot be expressed directly in MSO (see Corollary 9.3).

The proof of Theorem 1.1 develops the game-based technique used previously to characterise certain properties of MSO-definable tree languages (see e.g. [9, 28]). Each application of this technique requires a specific game, designed in a way which reflects the studied property. The game should have a finite arena and be played between two perfectly-informed players \exists and \forall . The winning condition of the game is given by a certain ω -regular set.

Then, the seminal result of Büchi and Landweber [8] yields that the game is determined and the winner of this game can be effectively decided. The construction of the game is such that a winning strategy of each of the players provides a witness of either of the considered possibilities: if \exists wins then $\eta_\varphi = \omega_1$ and if \forall wins then $\eta_\varphi < \omega \cdot N$ for some computable N . The crucial difficulty of this approach lies in a proper definition of the game, so that both these implications actually hold.

Related work

In the context of μ -calculus, one asks how many iterations are needed to reach a fixed point; this aspect concerns complexity of model checking (cf. [7, 14]), as well as expressive power of the logic (cf. [7, 6]). Recall that, in an infinite structure, a least fixed point $\mu X. F(X)$ is in general reached in a transfinite number of iterations: $\emptyset, F(\emptyset), \dots, F^\alpha(\emptyset), \dots$, where, for a limit ordinal α , $F^\alpha(\emptyset) = \bigcup_{\xi < \alpha} F^\xi(\emptyset)$. It is therefore natural to ask if, for a given formula, one can effectively find a *closure ordinal* η_F , such that, in any model, the fixed point can be reached in η_F , but in general not less, iterations. Fontaine [18] effectively characterised the formulae such that in each model the fixed point is reached in a finite number of steps. Czarnecki [12] observed that some formulae have no closure ordinals but, for each ordinal $\eta < \omega^2$, there is a formula whose closure ordinal is η . He also raised the following question.

► **Question 1.2** (Czarnecki [12]). *Is there a μ -calculus formula of the form $\mu X. F(X)$ that has a closure ordinal $\eta_F \geq \omega^2$?*

Gouveia and Santocanale [19] exhibited an example of a formula with an essential alternation of the least and greatest fixed-point operators whose closure ordinal is ω_1 ; clearly this limit can be achieved only in uncountable models. In general, it remains open whether a formula $\mu X. F(X)$ of the μ -calculus may have a *countable* closure ordinal $\eta_F \geq \omega^2$. Afshari and Leigh [2] claimed a negative answer for formulas of the alternation-free fragment of μ -calculus; however, as the authors have later admitted [1], the proof contained some gaps. In a recent paper [1], Afshari, Barlucchi, and Leigh update the proof and extend the result to formulae of the so-called Σ -fragment of the μ -calculus. More specifically, the authors consider systems of equations $X_1 = F_1(X_1, \dots, X_n), \dots, X_n = F_n(X_1, \dots, X_n)$, where the formulae F_i may contain closed sub-formulae of the full μ -calculus, but the variables X_i do not fall in the scope of any fixed-point operators. The authors show that if a countable number of iterations η suffice to reach the least fixed point of such a system in any Kripke frame then $\eta < \omega^2$. (Note that, by Bekić principle, each component of such a solution can be expressed by a single formula with the nested μ -operators.)

As a direct consequence of our result, we obtain an alternative proof for formulae of the form $\mu X. F(X)$, where X does not fall in the scope of any fixed-point operator, but arbitrary closed subformulae may appear in F . (This corresponds to the case of $n = 1$ in the equation system of [1].) We show that no such formula has a closure ordinal η_F satisfying $\omega^2 \leq \eta_F < \omega_1$, and moreover, it can be decided whether $\eta_F < \omega^2$; see Section 10. This is achieved via a well-known reduction of the μ -calculus to the MSO theory of the binary tree, based on the tree-model property [7], and a natural encoding of a tree model.

In the studies of topological complexity of MSO-definable tree languages, it has been also observed that dichotomies may help to solve decision problems. An open problem related to the aforementioned question of definability with finite-set quantifiers, is whether we can decide if a tree language belongs to the Borel hierarchy (in general, it need not). A positive answer is known if a tree language is given by a deterministic parity automaton [23], based on the following dichotomy: such a language is either Π_1^1 -complete (very hard), or on the

level Π_3^0 (relatively low) of Borel hierarchy. Skrzypczak and Walukiewicz [28] gave a proof in the case when a tree language is given by a non-deterministic Büchi tree automaton, inspired by a rank-related dichotomy conjectured in the previous work of the first author [27, Conjecture 4.4]. There, an ordinal has been associated with each Büchi tree automaton, and it turns out (in view of [28]) that this ordinal either equals ω_1 or is smaller than ω^2 , in which case the tree language is Borel. It should be mentioned that a procedure to decide if a Büchi definable tree language is weakly definable was given earlier by Colcombet et al. [10].

Relation between ordinals and automata has been also considered in the studies of automatic structures. In particular, Delhommé [13] showed that automatic ordinals are smaller than ω^ω while tree-automatic ordinals (defined in terms of automata on finite trees) can go higher, but not above ω^{ω^ω} . Later, Finkel and Todorćević [17] showed that ω -tree automatic ordinals are smaller than ω^{ω^ω} as well. These results may appear in contrast with our more restrictive bound of ω^2 ; however, representation by automatic structures is in general more powerful than expressibility in MSO, so the two approaches are not directly related.

2 Basic notions

$\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of natural numbers. We use standard notation for ordinal numbers, with 0 being the least ordinal number, ω being the least infinite ordinal, and ω_1 the least uncountable ordinal. Although ω and \mathbb{N} coincide as sets, we distinguish the two notations to emphasise the perspective.

Words. An alphabet A is any finite non-empty set, whose elements are called *letters*. A *word* over A is a finite sequence of letters $w = w_0 \cdots w_{n-1}$, with $w_i \in A$ for $i < n$, and n being the *length* of w . The empty word, denoted ε , is the unique word of length 0. By uv we denote the *concatenation* of words $u = u_0 \cdots u_{n-1}$ and $v = v_0 \cdots v_{m-1}$, that is, $uv = u_0 \cdots u_{n-1}v_0 \cdots v_{m-1}$.

A^* denotes the set of all finite words over the alphabet A , while A^ω denotes the set of all ω -words over A , that is, functions $\alpha: \mathbb{N} \rightarrow A$. If $\alpha \in A^\omega$ is an infinite word and $n \in \mathbb{N}$ then by $\alpha \upharpoonright_n$ we denote the finite word consisting of the first n letters of α , that is, $\alpha_0 \cdots \alpha_{n-1} \in A^*$.

Trees. Let L and R denote two distinct letters called *directions*. For a direction $d \in \{\mathsf{L}, \mathsf{R}\}$, the *opposite direction* is denoted $\bar{d} \neq d$. Words over the alphabet $\{\mathsf{L}, \mathsf{R}\}$ are called *nodes*, with the empty word ε often called the *root*. An ω -word $\alpha \in \{\mathsf{L}, \mathsf{R}\}^\omega$ is called an *infinite branch*. A node u is called the *parent* of its *children* $u\mathsf{L}$ and $u\mathsf{R}$.

A (full infinite binary) *tree* over an alphabet A is a function $t: \{\mathsf{L}, \mathsf{R}\}^* \rightarrow A$, assigning letters $t(u) \in A$ to nodes $u \in \{\mathsf{L}, \mathsf{R}\}^*$. The set of all trees over A is denoted Tr_A . A *subtree* of a tree $t \in \text{Tr}_A$ in a node $u \in \{\mathsf{L}, \mathsf{R}\}^*$ is the tree $t \upharpoonright_u$ over A defined by taking $t \upharpoonright_u(v) = t(uv)$.

Automata. Instead of working with formulae of MSO, we use another equivalent formalism, namely non-deterministic parity tree automata [29]. An *automaton* is a tuple $\mathcal{A} = (A, Q, q_{\mathsf{I}}, \Delta, \Omega)$, where A is an alphabet, Q is a finite set of *states*, $q_{\mathsf{I}} \in Q$ is an *initial state*, $\Delta \subseteq Q \times A \times Q \times Q$ is a *transition relation*, and $\Omega: Q \rightarrow \{i, \dots, j\} \subseteq \mathbb{N}$ is a *priority mapping*. A *run* of an automaton \mathcal{A} from a state $q \in Q$ over a tree $t \in \text{Tr}_A$ is a tree $\rho \in \text{Tr}_Q$ such that $\rho(\varepsilon) = q$ and for every node $u \in \{\mathsf{L}, \mathsf{R}\}^*$ the quadruple

$$(\rho(u), t(u), \rho(u\mathsf{L}), \rho(u\mathsf{R})),$$

is a transition of \mathcal{A} (i.e., belongs to Δ).

A sequence of states $(q_0, q_1, \dots) \in Q^\omega$ is *accepting* if $\limsup_{n \rightarrow \infty} \Omega(q_n)$ is an even natural number. A run ρ is *accepting* if for every infinite branch $\alpha \in \{\mathsf{L}, \mathsf{R}\}^\omega$ the sequence of states $(\rho(\alpha \upharpoonright_n))_{n \in \mathbb{N}}$ that appear in ρ on α is accepting.

A tree $t \in \text{Tr}_A$ is *accepted* by an automaton \mathcal{A} from a state $q \in Q$ if there exists an accepting run ρ of \mathcal{A} from the state q over t . The *language* of \mathcal{A} , denoted $L(\mathcal{A}) \subseteq \text{Tr}_A$, is the set of trees which are accepted by the automaton from the initial state q_1 . A language $L \subseteq \text{Tr}_A$ is *regular* if it is $L(\mathcal{A})$ for some automaton \mathcal{A} .

We now recall the famous theorem of Rabin, which allows us to transform MSO formulae into equivalent tree automata.

► **Theorem 2.1** ([25], see also [29]). *For every MSO formula $\varphi(X_0, \dots, X_{n-1})$ the set of valuations satisfying φ is a regular language over the alphabet $\{0, 1\}^n$.*

We say that an automaton is *pruned* if every state $q \in Q$ appears in some accepting run, that is, there exists an accepting run ρ of \mathcal{A} over a tree t such that $\rho(u) = q$ for some node $u \in \{\mathsf{L}, \mathsf{R}\}^*$. Note that every automaton can effectively be pruned, without affecting its language, by detecting and removing states that do not appear in any accepting run.

Games. We use the standard framework of two-player games of infinite duration (see e.g., [8, 16]). The arena of such a game is given as a graph with both players having perfect information about the current position. The winning condition of the game is given by a language of infinite plays won by one of the players.

Ranks. Recall that an ordinal η can be seen as the linearly ordered set of all ordinals smaller than η . Given a set $X \subseteq \{\mathsf{L}, \mathsf{R}\}^*$, a *counting function* is a function $\mathfrak{C}: X \rightarrow \eta$ such that $\mathfrak{C}(u) < \mathfrak{C}(v)$ whenever $u \in X$ is a proper descendant of $v \in X$ (such a function exists for some η whenever X is well-founded). The *rank* of a well-founded set X is the least ordinal η for which a counting function from X to η exists.

Taking the automata-based perspective, instead of working with monadic variables \vec{Y} and X , we consider labellings of the tree by certain finite alphabets. In particular, a set of nodes $X \subseteq \{\mathsf{L}, \mathsf{R}\}^*$ can be identified with its characteristic function, that is, a tree $x \in \text{Tr}_{\{0,1\}}$ over the alphabet $\{0, 1\}$. Such a tree is *well-founded* if no infinite branch contains infinitely many nodes labelled by 1. The set of well-founded trees is denoted $\text{WF} \subseteq \text{Tr}_{\{0,1\}}$. Likewise, we define the *rank* of $x \in \text{WF}$, denoted $\text{rank}(x)$, as the least η for which there is a counting function from the set of 1-labelled nodes of x to η . The considered rank is analogous with the standard rank of well-founded trees (e.g., [21, Section 2.E]).

► **Example 2.2.** The tree $x_0 \in \text{Tr}_{\{0,1\}}$ with all nodes labelled by 0 has rank 0.

Consider a tree x_ω where a node v has label 1 if $v = \mathsf{R}^i \mathsf{L}^j$ with $1 \leq j \leq i$. It is a *comb*: the rightmost branch is labelled by zeros, and below its i -th node we have a tooth of i nodes labelled by ones, going left. The rank of x_ω is ω .

Such combs can be nested: suppose that $x_{\omega \cdot 2}$ starts analogously to x_ω , but below every tooth we again insert x_ω (i.e., $x_{\omega \cdot 2} \upharpoonright_{\mathsf{R}^i \mathsf{L}^{i+1}} = x_\omega$ for every i); then $\text{rank}(x_{\omega \cdot 2}) = \omega \cdot 2$. Repeating this, we can insert $x_{\omega \cdot n}$ below every tooth of x_ω , and obtain $x_{\omega \cdot (n+1)}$ of rank $\omega \cdot (n+1)$, for every $n \in \mathbb{N}$.

Then, we can place every $x_{\omega \cdot n}$ at node $\mathsf{R}^n \mathsf{L}$, below a 0-labelled rightmost branch; the resulting tree has rank ω^2 . In a similar manner we can create a tree having rank equal to any countable ordinal η .

3 Problem formulation

We begin by formulating the problem under consideration. Instead of working with a formula $\varphi(\vec{Y}, X)$, we assume that A is some alphabet and Γ is a regular language over the alphabet $A \times \{0, 1\}$. We identify a tree τ over $A \times \{0, 1\}$ with a pair (t, x) , where $t \in \text{Tr}_A$ and $x \in \text{Tr}_{\{0,1\}}$ such that $\tau(u) = (t(u), x(u))$ for every node $u \in \{\mathbb{L}, \mathbb{R}\}^*$. Thus, $\Gamma \subseteq \text{Tr}_{A \times \{0,1\}}$ can be seen as a relation whose elements are pairs (t, x) . We additionally require that whenever $(t, x) \in \Gamma$ then x is well-founded, which means that Γ (treated as a relation) is contained in $\text{Tr}_A \times \text{WF}$. We say that such a relation is *regular* if it is regular as a language over $A \times \{0, 1\}$.

Let $\pi_A(\Gamma)$ be the projection of Γ onto the A coordinate, that is, the set of those trees $t \in \text{Tr}_A$ for which there exists a (necessarily well-founded) tree $x \in \text{Tr}_{\{0,1\}}$ such that $(t, x) \in \Gamma$. Similarly, for a tree $t \in \text{Tr}_A$ by Γ_t we denote the *section* of Γ over the tree t , that is, the set $\{x \in \text{Tr}_{\{0,1\}} \mid (t, x) \in \Gamma\}$.

The following definition is just a reformulation of Formula (1.1) in terms of a relation Γ .

► **Definition 3.1.** *The closure ordinal of a relation $\Gamma \subseteq \text{Tr}_A \times \text{WF} \subseteq \text{Tr}_{A \times \{0,1\}}$ (or of an automaton recognising it) is defined as*

$$\eta_\Gamma \stackrel{\text{def}}{=} \sup_{t \in \pi_A(\Gamma)} \min_{x \in \Gamma_t} \text{rank}(x).$$

► **Example 3.2.** Consider the following automaton \mathcal{A} over the alphabet $\{\mathbf{b}, \mathbf{c}\} \times \{0, 1\}$. Its states are $p_0, q_1, q_2, q_3, r_0, r_1$ with q_1 being initial, where the subscript provides the priority of a state (i.e., $\Omega(p_i) = \Omega(q_i) = \Omega(r_i) = i$). The transitions are, for all $i \in \{0, 1\}$, $j \in \{1, 2, 3\}$, and $a \in \{\mathbf{b}, \mathbf{c}\}$,

$$\begin{array}{lll} (p_0, (a, 0), p_0, p_0), & (q_j, (\mathbf{c}, 0), q_3, p_0), & (q_j, (\mathbf{c}, 0), p_0, r_0), \\ (q_j, (\mathbf{b}, 0), q_2, p_0), & (q_j, (\mathbf{c}, 0), p_0, q_3), & (r_i, (\mathbf{b}, i), r_1, r_0), \\ (q_j, (\mathbf{b}, 0), p_0, q_1), & (q_j, (\mathbf{c}, 0), r_0, p_0), & (r_i, (\mathbf{c}, 0), p_0, p_0). \end{array}$$

In this example, we should see \mathbf{c} -labelled nodes as separators, splitting the whole tree into \mathbf{b} -labelled fragments. Let us see when a pair $(t, x) \in \text{Tr}_{\{\mathbf{b}, \mathbf{c}\} \times \{0,1\}}$ can be accepted. Note first that \mathcal{A} accepts (t, x_0) from p_0 for every tree t , and for x_0 having all nodes labelled by 0. Next, observe that states q_j become aligned along a single branch, either finite or infinite. If in t there is a branch that infinitely often goes left, but visits only finitely many \mathbf{c} -labelled nodes, then we can align the q_j states along this branch, and (t, x_0) will be accepted. Indeed, just below every \mathbf{c} we have q_3 (state of large odd priority; has to occur finitely often); below every \mathbf{b} , if we go left we have q_2 , and if we go right we have q_1 (so the parity condition requires infinitely many left-turns). Another possibility is that the branch with states q_j is finite, and below some \mathbf{c} -labelled node the state changes to r_0 . Now the run sends r_1 to every left child, and r_0 to every right child; hence every left child in x should have label 1, and every right child – label 0. We can continue the zone of states r_i until reaching a node labelled by \mathbf{c} ; such a node allows us to change the state into p_0 and accept anything below. The acceptance condition requires that there are only finitely many states r_1 (hence also nodes with label 1 in x) on every branch; the tree x is necessarily well-founded.

This determines the optimal rank of a witness x for a tree t . Namely, if in t there is a branch that infinitely often goes left, but visits only finitely many \mathbf{c} -labelled nodes, we have a witness of rank 0. Otherwise, we should consider every zone of \mathbf{b} -labelled nodes in t , surrounded by \mathbf{c} -labelled nodes; consider x having 1 in every left child in that zone; and take the minimum of ranks of such trees x , over all choices of zones (not including the topmost

zone, above the first c on a branch). Thus, every witness of a tree t has rank at least η if and only if every such zone results in rank at least η , and the former case does not hold. Such a tree exists for every $\eta < \omega_1$, so the closure ordinal of \mathcal{A} is ω_1 .

4 The dichotomy game

We now move to the definition of the game $\mathcal{G}_{\mathcal{A}}$ designed to decide the dichotomy from Theorem 1.1.

Note that for every regular relation $\Gamma \subseteq \text{Tr}_{A \times \{0,1\}}$ there exists another regular relation $\Gamma' \subseteq \text{Tr}_{A \times \{0,1\}}$ with the same closure ordinal, but such that $\pi_A(\Gamma')$ is the set of all trees over A . To see this, it is enough to take $\Gamma' = \Gamma \cup \{(t, x') \mid x' \in \text{WF} \wedge \neg \exists x.(t, x) \in \Gamma\}$. Then $\min_{x \in \Gamma_t} \text{rank}(x) = \min_{x \in \Gamma'_t} \text{rank}(x)$ for $t \in \pi_A(\Gamma)$ and $\min_{x \in \Gamma'_t} \text{rank}(x) = 0$ for $t \notin \pi_A(\Gamma)$.

Towards the proof of Theorem 1.1, we consider some relation $\Gamma \subseteq \text{Tr}_{A \times \{0,1\}}$ such that $\Gamma \subseteq \text{Tr}_A \times \text{WF}$ and $\pi_A(\Gamma) = \text{Tr}_A$. Reformulating Theorem 1.1, we need to show that either $\eta_{\Gamma} < \omega \cdot N$ for some $N \in \mathbb{N}$, or $\eta_{\Gamma} = \omega_1$, and we can effectively decide which case holds. We assume that Γ is given by a pruned (i.e., all states appear in some accepting run) automaton $\mathcal{A} = (A, Q, q_{\mathbb{I}}, \Delta, \Omega)$.

First, a *side* is a symbol $s \in \{\mathbb{T}, \mathbb{R}\}$ (which stands for “trunk” and “reach”), and a *mode* is a symbol $m \in \{\textcircled{1}, \textcircled{2}\}$ (which stands for non-branching and binary-branching).

A quadruple $(\delta, s, m, d) \in \Delta \times \{\mathbb{T}, \mathbb{R}\} \times \{\textcircled{1}, \textcircled{2}\} \times \{\mathbb{L}, \mathbb{R}\}$ such that $s = \mathbb{R}$ implies $m = \textcircled{1}$ (i.e., $m = \textcircled{2}$ is allowed only for $s = \mathbb{T}$) is called a *selector for* (δ, s) , where δ is of the form $(q, (a, i), q_{\mathbb{L}}, q_{\mathbb{R}})$ with $a \in A$, $i \in \{0, 1\}$. Such a selector *agrees with* a direction $d' \in \{\mathbb{L}, \mathbb{R}\}$ if either $m = \textcircled{2}$ or $d' = d$ (i.e., both directions d' are fine if $m = \textcircled{2}$, but if $m = \textcircled{1}$ then we require $d' = d$). The *output side* of a selector (δ, s, m, d) in direction d' , where $\delta = (q, (a, i), q_{\mathbb{L}}, q_{\mathbb{R}})$, is

- \mathbb{T} if $s = \mathbb{T}$ and $d' = d$, or $s = \mathbb{R}$ and $i = 1$, and
- \mathbb{R} otherwise: if $s = \mathbb{T}$ and $d' \neq d$, or $s = \mathbb{R}$ and $i = 0$.

A *state-flow* is a triple $((q, s), m, (q', s'))$, where $q, q' \in Q$, $s, s' \in \{\mathbb{T}, \mathbb{R}\}$, and $m \in \{\textcircled{1}, \textcircled{2}\}$. A *flow* μ is a set of state-flows. The set $\{(q', s') \mid ((q, s), m, (q', s')) \in \mu\}$ is called the *image* of μ . Note that the number of all possible state-flows, hence also of all possible flows, is finite.

Given a flow μ , we say that a flow $\bar{\mu} \subseteq \mu$ is a *back-marking* of μ if for every pair (q', s') in the image of μ , precisely one state-flow leading to (q', s') belongs to $\bar{\mu}$. This state flow is called *back-marked* for (q', s') .

Given a sequence of flows, μ_1, μ_2, \dots we can define their *composition* as the graph with vertices $(q, s, n) \in Q \times \{\mathbb{T}, \mathbb{R}\} \times \mathbb{N}$ and with a directed edge from (q, s, n) to $(q', s', n+1)$ labelled by m for every state-flow $((q, s), m, (q', s')) \in \mu_{n+1}$, $n \in \mathbb{N}$. Note that in a flow there may be two state-flows with the same pairs (q, s) and (q', s') , with modes $m = \textcircled{1}$ and $m = \textcircled{2}$, leading to two parallel edges in this graph.

Assume that we are given a set $Q' \subseteq Q$ and a letter $a \in A$. By $\Delta_a(Q')$ we denote the set of transitions of the form $(q, (a, i), q_{\mathbb{L}}, q_{\mathbb{R}})$ with $q \in Q'$.

Finally, for two sets of states $T, R \subseteq Q$ we denote $\langle T, R \rangle \stackrel{\text{def}}{=} (T \times \{\mathbb{T}\}) \cup (R \times \{\mathbb{R}\})$. Note that every subset of $Q \times \{\mathbb{T}, \mathbb{R}\}$ can be uniquely represented as $\langle T, R \rangle$ for some $T, R \subseteq Q$.

We can now move to the definition of the crucial game $\mathcal{G}_{\mathcal{A}}$, used to prove the desired dichotomy. The positions of $\mathcal{G}_{\mathcal{A}}$ are of the form $\langle T, R \rangle \subseteq Q \times \{\mathbb{T}, \mathbb{R}\}$ (formally, one also needs additional auxiliary positions to represent the situation between particular steps of a round; we do not refer to these positions explicitly). The initial position is $\langle \{q_{\mathbb{I}}\}, \emptyset \rangle$, where $q_{\mathbb{I}}$ is the initial state of the automaton \mathcal{A} . The consecutive steps in a round $n \in \mathbb{N}$ from a position $\langle T_n, R_n \rangle$ are as follows:

1. \forall declares a subset $T'_n \subseteq T_n$.
2. \exists declares a letter $a_n \in A$.
3. \exists declares a set F_n of selectors, containing one selector for each $(\delta, s) \in (\Delta_{a_n}(T'_n) \times \{\mathsf{T}\}) \cup (\Delta_{a_n}(R_n) \times \{\mathsf{R}\})$.
4. \forall declares a direction $d_{n+1} \in \{\mathsf{L}, \mathsf{R}\}$.
5. We define a flow μ_{n+1} as the set containing the state-flows $((q, s), m, (q_{d_{n+1}}, s'))$ for each selector $(\delta, s, m, d) \in F_n$ that agrees with direction d_{n+1} , where s' is the output side of the selector in the direction d_{n+1} , and $\delta = (q, (a_n, i), q_{\mathsf{L}}, q_{\mathsf{R}})$ (so q is the source state of δ and $q_{d_{n+1}}$ is the state sent by δ in direction d_{n+1}).
6. \forall declares some back-marking $\bar{\mu}_{n+1}$ of the flow μ_{n+1} .

The new position of the game, (T_{n+1}, R_{n+1}) , is the image of the flow μ_{n+1} .

Given a play Π of the game, the winning condition for \exists is the disjunction **A**) \vee **B**) of the following parts.

- A**) In the graph obtained as a composition of the back-markings $\bar{\mu}_1, \bar{\mu}_2, \dots$ there exists a path which infinitely many times changes sides between T and R .
- B**) In the graph obtained as a composition of the flows μ_1, μ_2, \dots every infinite path either is rejecting or contains infinitely many state-flows of mode $\textcircled{2}$.

Above, while saying that a path going through $(q_0, s_0, 0), (q_1, s_1, 1), \dots$ is rejecting, we mean that the sequence of states $(q_n)_{n \in \mathbb{N}}$ is rejecting, that is, $\limsup_{n \rightarrow \infty} \Omega(q_n)$ is odd.

► **Remark 4.1.** The arena of the game $\mathcal{G}_{\mathcal{A}}$ is finite and the winning condition defined above is ω -regular. Thus, the theorem of Büchi and Landweber [8] applies: one can effectively decide the winner of $\mathcal{G}_{\mathcal{A}}$. Moreover, there exists a computable bound M such that whoever wins $\mathcal{G}_{\mathcal{A}}$, can win using a finite-memory strategy that uses at most M memory states.

The following proposition formalises the relation between $\mathcal{G}_{\mathcal{A}}$ and Theorem 1.1.

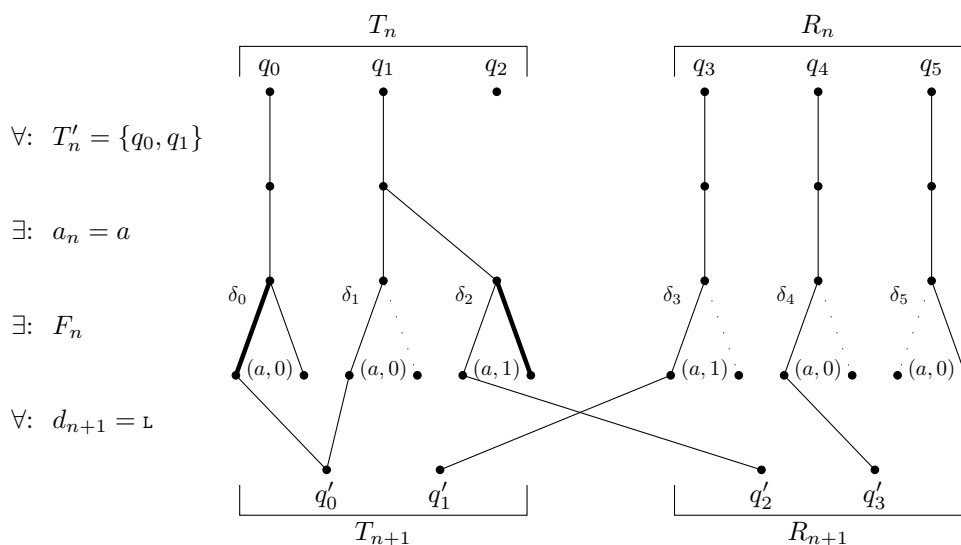
► **Proposition 4.2.** *Assume that \mathcal{A} is a pruned automaton which recognises a relation $\Gamma \subseteq \text{Tr}_{\mathcal{A}} \times \text{WF}$ such that $\pi_{\mathcal{A}}(\Gamma) = \text{Tr}_{\mathcal{A}}$ (i.e., the projection of Γ is full). Then, we have the following two implications:*

1. *if player \exists wins $\mathcal{G}_{\mathcal{A}}$ then $\eta_{\Gamma} = \omega_1$;*
2. *if player \forall wins $\mathcal{G}_{\mathcal{A}}$ then $\eta_{\Gamma} < \omega \cdot N$, for a number $N \in \mathbb{N}$ computable based on the automaton \mathcal{A} .*

Depiction of a round of $\mathcal{G}_{\mathcal{A}}$. Figure 4.1 depicts an example of a round of the game $\mathcal{G}_{\mathcal{A}}$. First \forall declares a subset $T'_n = \{q_0, q_1\}$ effectively removing the state q_2 from the T side. Note that some states might repeat on both sides. Once the letter $a_n = a$ is chosen by \exists , there are six possible transitions from the considered states: one from each of them, except the state q_1 which has two possible transitions, δ_1 over $(a, 0)$ and δ_2 over $(a, 1)$. \exists declares a set of selectors

$$F_n = \left\{ (\delta_0, \mathsf{T}, \textcircled{2}, \mathsf{L}), (\delta_1, \mathsf{T}, \textcircled{1}, \mathsf{L}), (\delta_2, \mathsf{T}, \textcircled{2}, \mathsf{R}), (\delta_3, \mathsf{R}, \textcircled{1}, \mathsf{L}), (\delta_4, \mathsf{R}, \textcircled{1}, \mathsf{L}), (\delta_5, \mathsf{R}, \textcircled{1}, \mathsf{R}) \right\}.$$

Thus, the selectors for the transitions δ_0 and δ_2 are in the mode $\textcircled{2}$, while the remaining selectors are in the mode $\textcircled{1}$. Once \forall chooses the direction $d_{n+1} = \mathsf{L}$, we gather into new sets T_{n+1} and R_{n+1} the states sent by the transitions in the direction L . The transitions δ_0 and δ_1 provide their left states into T_{n+1} . However, in the case of transition δ_2 the direction of the selector is R while the direction chosen by \forall is L and therefore the left state of this transition goes to R_{n+1} . Regarding the transitions on the R side, δ_3 provides its state to T_{n+1} because this transition is over $(a, 1)$. The transition δ_4 provides its state to R_{n+1} because



■ **Figure 4.1** A depiction of a round of the game $\mathcal{G}_{\mathcal{A}}$.

this transition is over $(a, 0)$. Finally, the transition δ_5 does not provide its state anywhere, because its mode is $\textcircled{1}$ and its direction is R , different than the direction L chosen by \forall . The states provided by the transitions δ_0 and δ_1 are both q'_0 and are therefore merged. This means that the final flow μ_{n+1} is

$$\left\{ ((q_0, \text{T}), \textcircled{2}, (q'_0, \text{T})), ((q_0, \text{T}), \textcircled{1}, (q'_0, \text{T})), ((q_1, \text{T}), \textcircled{2}, (q'_2, \text{R})), \right. \\ \left. ((q_3, \text{R}), \textcircled{1}, (q'_1, \text{T})), ((q_4, \text{R}), \textcircled{1}, (q'_3, \text{R})) \right\}.$$

When \forall chooses a back-marking $\bar{\mu}_{n+1}$ of the flow μ_{n+1} , he needs to make a choice which state-flow to select for the pair (q'_0, T) as there are two possible choices.

5 Intuitions

Let us explain the intuitions behind the game. First very generally: the game is designed so that \exists wins when there are trees t requiring witnesses x of arbitrarily large ranks. Conversely, \forall wins if there is a bound η such that every tree t has a witness x of rank at most η . But because this is a finite game with an ω -regular winning condition, if \forall wins then he wins with a finite-memory winning strategy; from such a strategy we can deduce that the bound η is actually of the form $\omega \cdot N$ for some natural number N depending on the size of the memory of \forall .

Having the above in mind, let us discuss details of the game. The role of \exists should be to show a tree t (whose all witnesses x have large rank), so we allow her to propose a label of a node in step 2. However, as usually in games, we do not continue in both children of the current node, but we rather ask \forall to choose one direction (d_{n+1} in step 4), where \exists has to continue creating the tree, and where \forall thinks that it is impossible to continue.

Recall now that arbitrarily large countable ordinals can be obtained by alternately applying two operations in a well-founded way: add one, and take the supremum of infinitely many ordinals. Similarly, in a tree x of a large rank, we can always find a *comb*: an infinite branch – a trunk – such that from infinitely many nodes on the side of this trunk we can reach a node labelled by 1, below which the rank is again large (but slightly smaller). In these

places we can repeat this process, that is, again find an analogous comb, and so on, obtaining a tree of nested combs that is well-formed but itself has a large rank. Obviously the converse holds as well: such a tree of nested combs having large rank can exist inside x only if x has a large rank. Let us also remark that in the case of a finite-memory strategy of \forall we obtain that such combs in x can be nested at most $N \in \mathbb{N}$ times, which itself implies that the rank of x is at most $\omega \cdot N$.

Thus, in order to show that the constructed tree t allows only witnesses x of large rank, \exists shows a nested comb structure. But this has to be done for every x such that $(t, x) \in \Gamma$, so, in a sense, for every run of \mathcal{A} over (t, x) for some x . As usual, we cannot require from \forall to choose a run on the fly, during the game; an interesting (even an accepting) run of \mathcal{A} can only be fixed after the whole tree (i.e., the whole future of the play) is fixed. This means that during the game we have to trace all possible runs of \mathcal{A} . However, there are infinitely many runs, so we cannot do that. To deal with that, we keep track of all “interesting” states in the sets T_n, R_n , and we consider all possible transitions from them in step 3. This makes the situation of \exists a bit worse: she has to make decisions based only on the current state, not knowing the past of the run (the same state may emerge after two different run prefixes). But it turns out that \exists can handle that; in our proof this corresponds to positionality of strategies in an auxiliary game considered in Section 7.

Now, how exactly does \exists show the nested comb structure? This is done via selectors proposed in the sets F_n . For states in T_n (i.e., on the “trunk” side) \exists has to show in which direction the trunk continues. Moreover, \exists has to show places where on side of the trunk we can reach label 1 followed by a nested comb; in these places \exists plays mode ② (“branch”). If \forall chooses a direction in which the trunk (as declared by \exists) continues, we trace the resulting state again on the T side. If he chooses the opposite direction, and the mode is ① (no branching here), we just stop tracing this run. But if \forall chooses the non-trunk direction while the mode is ②, the resulting state ends up on the R (“reach”) side. The role of \exists is now to show a direction in which we can reach a node with label 1. If \forall follows this direction, and the label of x is 0, we continue searching for label 1 on the R side. When label 1 is found, we put the resulting state on the T side; \exists has to show a next comb. The **B**) part of the winning condition ensures for every accepting run that the trunk of each comb has infinitely many branching points (i.e., points with mode ②) and that on the R side we stay only for finitely many steps (as mode ② does not occur there). Note that by arbitrarily composing transitions we may obtain also rejecting runs; the **B**) condition does not require anything for them.

There is one more issue taken into account in the design of the game: \exists should be obliged to produce the combs nested arbitrarily many times, but not infinitely many times. The number of *nestings* (i.e., of switches between sides T and R) is controlled by \forall . When he is satisfied with the nesting depth provided by \exists for runs ending in some state, he can remove this step from the position in step 1, and let \exists provide appropriate comb structures only from remaining states (we allow this removal only on the T side, but we could equally well allow it on the R side, or on both sides). The **A**) part of the winning condition obliges \forall to indeed remove a state after seeing finitely many nestings. To see the usefulness of back-marking used to formulate this condition, consider a situation with two runs leading to some state: one with already many nestings of combs provided by \exists , and other where we are still on the trunk of the first comb. Because \exists should provide many nested combs for all runs, in such a situation \forall should still be able to analyse the latter run. To this end, he can select the latter run as the back-marked history of the considered state, and continue waiting for further nested combs, without worrying that he will lose by the **A**) part of the winning condition due to the former run.

► **Example 5.1.** Let us see how the game $\mathcal{G}_{\mathcal{A}}$ behaves for the automaton \mathcal{A} from Example 3.2. Recall that the closure ordinal of \mathcal{A} is ω_1 , so \exists should be able to win.

The strategy of \exists from a position (T'_n, R_n) is as follows. If $T'_n \cup R_n$ contains a state r_i , then \exists plays letter **b**, otherwise letter **c**. Then \exists proposes selectors: transitions δ originating from states q_j are handled by selectors (δ, s, m, d) with mode $m = \textcircled{1}$ and with direction d being such that δ sends state p_0 to the opposite direction \bar{d} (i.e., $d = \text{L}$ for $\delta = (q_j, (a, 0), q_2, p_0)$, etc.); transitions originating from r_i are handled by direction $d = \text{R}$, and by mode $m = \textcircled{2}$ on the **T** side and mode $m = \textcircled{1}$ on the **R** side. As we argue below, p_0 never becomes an element of $T_n \cup R_n$, so transitions from p_0 need not to be handled.

The initial position is $(\{q_1\}, \emptyset)$. Here \exists plays letter **c** and some selectors with mode $\textcircled{1}$, and \forall chooses a direction. There are two selectors (δ, s, m, d) that agree with this direction, and they send there states q_3 and r_0 . Thus the next position is $(\{q_3, r_0\}, \emptyset)$, reached by the flow $\{((q_1, \text{T}), \textcircled{1}, (q_3, \text{T})), ((q_1, \text{T}), \textcircled{1}, (r_0, \text{T}))\}$. This time \exists plays letter **b**. If \forall goes right, the new position is $(\{q_1, r_0\}, \emptyset)$, reached by the flow $\{((q_3, \text{T}), \textcircled{1}, (q_1, \text{T})), ((r_0, \text{T}), \textcircled{2}, (r_0, \text{T}))\}$; this position behaves identically to the previous one, as from q_1 and q_3 we have the same transitions. If \forall goes left, the new position is $(\{q_2\}, \{r_1\})$, reached by the flow $\{((q_3, \text{T}), \textcircled{1}, (q_2, \text{T})), ((r_0, \text{T}), \textcircled{2}, (r_1, \text{R}))\}$. From $(\{q_2\}, \{r_1\})$ once again letter **b** is played. Note that the only transition from r_1 reading letter **b** on the first coordinate, reads 1 on the second coordinate. So, if \forall goes right, the new position is back to $(\{q_1, r_0\}, \emptyset)$, reached by the flow $\{((q_2, \text{T}), \textcircled{1}, (q_1, \text{T})), ((r_1, \text{R}), \textcircled{1}, (r_0, \text{T}))\}$; if he goes left, he reaches $(\{q_2\}, \emptyset)$, which behaves like the initial position.

It is also possible that \forall erases a state from the **T** side (i.e., plays T'_n being a proper subset of T_n). If state r_0 is erased, we end up in a position $(\{q_j\}, \emptyset)$, being like the initial position. We may also have positions $(\{r_0\}, \emptyset)$ and $(\emptyset, \{r_1\})$, and flows $\{((r_0, \text{T}), \textcircled{2}, (r_0, \text{T}))\}$, $\{((r_0, \text{T}), \textcircled{2}, (r_1, \text{R}))\}$, $\{((r_1, \text{R}), \textcircled{1}, (r_0, \text{T}))\}$ between them. Finally, we may also reach (\emptyset, \emptyset) .

Note that in our example there is always only one state-flow leading to every pair (q, s) ; in consequence, every state-flow is back-marked.

Let us now check the winning condition. One possibility is that infinitely many letters **c** were played. In these moments no state r_i was present in the position, so the only infinite path in the composition of flows is the path going through appropriate q_j states. But after seeing every **c** this state was q_3 , so the path is rejecting; part **B)** of the winning condition is satisfied (we may also have no infinite path, if the q_j state was removed by \forall , but then condition **B)** holds even more). The opposite case is that from some moment on, only letter **b** was played. We then also have an infinite path going, from some moment, through the r_i states (this path really exists: if \forall removes the state r_0 , then letter **c** is played). Note that whenever \forall goes left, this path changes sides from **T** to **R**, and in the next round returns back to the **T** side. If this happens infinitely often, \exists wins by condition **A)**. Otherwise, from some moment on \forall constantly goes right. After that, the path going through the r_i states has all state-flows of mode $\textcircled{2}$, and the other path (if exists) remains in state q_1 , so it is rejecting; condition **B)** is satisfied.

6 Soundness

We begin by proving Item 1 of Proposition 4.2: if \exists wins $\mathcal{G}_{\mathcal{A}}$ then $\eta_{\Gamma} = \omega_1$. To this end, assume that \exists wins the game and fix any winning strategy for her. For every countable ordinal $\eta > 0$, our goal is to construct a tree t such that $\text{rank}(x) \geq \eta$ whenever $(t, x) \in \Gamma$. We do this by inductively unravelling the fixed strategy of \exists . During this process, we keep track of

69:12 A Dichotomy Theorem for Ordinal Ranks in MSO

- the current node $v \in \{\mathsf{L}, \mathsf{R}\}^*$,
- the current position $(\langle T_v, R_v \rangle)$ of the game,
- a mapping κ_v , which assigns some ordinals $\leq \eta$ to all elements of the set $(\langle T_v, R_v \rangle)$.

Initially $v = \varepsilon$, the position $(\langle T_\varepsilon, R_\varepsilon \rangle)$ is the initial position of the game (i.e., $(\{q_{\mathsf{I}}\}, \emptyset)$), and $\kappa_\varepsilon = \{(q_{\mathsf{I}}, \mathsf{T}) \mapsto \eta\}$. Then, in every node v , the letter declared by \exists provides a label of this node in t , and while moving to a child vd of v we trace a play in which \forall declares the respective direction d . What remains is to declare the remaining choices of \forall and say how the mapping κ_v is updated. The role of the ordinal $\kappa_v(q, s)$ is to provide a lower bound for ranks of witnesses x such that $(t|_v, x)$ can be accepted by \mathcal{A} from the state q . While going down the tree, this ordinal decreases whenever we change the side from T to R ; when it becomes 0, \forall removes the respective state from the set T_v in his move. The back-markings are declared by \forall in a way that maximises the ordinals $\kappa_v(q, s)$ of the respective state-flows.

It remains to show that if $(t, x) \in \Gamma$ then $\text{rank}(x) \geq \eta$. This is achieved by considering any accepting run ρ of \mathcal{A} over (t, x) . Since all possible transitions of \mathcal{A} are taken into account in each round of $\mathcal{G}_{\mathcal{A}}$, one can trace the run ρ in the simulated plays of $\mathcal{G}_{\mathcal{A}}$. The policy of updating the mapping κ_v inductively assures that for every node v and currently traced side s_v we have $(\rho(v), s_v) \in (\langle T_v, R_v \rangle)$ and $\text{rank}(x|_v) \geq \kappa_v(\rho(v), s_v)$. Moreover, this policy guarantees that condition **A**) is never satisfied in these plays. Thus, condition **B**) must hold, inductively guaranteeing the inequality on $\text{rank}(x|_v)$.

7 Auxiliary game

Before moving towards a proof of the other implication, we need to be able to construct reasonable strategies of \exists in $\mathcal{G}_{\mathcal{A}}$. This is achieved by considering an auxiliary game, based directly on $\mathcal{G}_{\mathcal{A}}$, when considering a single transition of \mathcal{A} at each round. The players of the game are called Automaton (responsible for choosing transitions and choices of \forall in $\mathcal{G}_{\mathcal{A}}$) and Pathfinder (responsible for choices of \exists in $\mathcal{G}_{\mathcal{A}}$). The game is denoted $\mathcal{H}_{\mathcal{A}, t, N}$ and depends on the fixed automaton \mathcal{A} , a tree $t \in \text{Tr}_{\mathcal{A}}$, and a number $N \in \mathbb{N}$. Positions of $\mathcal{H}_{\mathcal{A}, t, N}$ are triples $(v, q, s) \in \{\mathsf{L}, \mathsf{R}\}^* \times Q \times \{\mathsf{T}, \mathsf{R}\}$, plus some additional auxiliary positions, to which we do not refer explicitly.

For a node $v \in \text{dom}(t)$ and a state $q \in Q$ define

$$\text{val}_t(v, q) = \inf \{ \text{rank}(x) \mid (t|_v, x) \text{ can be accepted from } q \}.$$

We assume $\inf \emptyset = \infty$ (which is greater than all ordinals).

In positions (v, q, s) such that $\text{val}_t(v, q) = 0$ the game reaches an *immediate victory* in which Pathfinder wins. A round from a position (v, q, s) such that $\text{val}_t(v, q) > 0$ consists of the following steps:

1. Automaton declares a transition $\delta = (q, (t(v), i), q_{\mathsf{L}}, q_{\mathsf{R}})$ from the current state q over $(t(v), i)$ for some $i \in \{0, 1\}$.
2. Pathfinder declares a selector (δ, s, m, d) for (δ, s) .
3. Automaton declares a direction d' that agrees with the selector.

Let $v' = vd'$, $q' = q_{d'}$, and let s' be the output side of the selector in the direction d' . Now, for every possible number $k \leq N$, the following four conditions of *immediate victory* may end the game, making Automaton win:

$$\begin{aligned}
s &= \text{T} \wedge \text{val}_t(v, q) \geq \omega \cdot k \wedge s' = \text{T} \wedge \text{val}_t(v', q') < \omega \cdot k, \\
s &= \text{T} \wedge \text{val}_t(v, q) \geq \omega \cdot k \wedge s' = \text{R} \wedge \text{val}_t(v', q') \leq \omega \cdot (k-1), \\
s &= \text{R} \wedge \text{val}_t(v, q) > \omega \cdot k \wedge s' = \text{R} \wedge \text{val}_t(v', q') \leq \omega \cdot k, \\
s &= \text{R} \wedge \text{val}_t(v, q) > \omega \cdot k \wedge s' = \text{T} \wedge \text{val}_t(v', q') < \omega \cdot k.
\end{aligned}$$

If no immediate victory happened, the game proceeds to the new position which is (v', q', s') . Note that the conditions of immediate victory depend only on (v, q, s) and (v', q', s') , so can be directly hardwired in the structure of the game.

An infinite play (i.e., without any immediate victory) of $\mathcal{H}_{\mathcal{A}, t, N}$ is won by Pathfinder if the sequence of visited states is rejecting or a selector with mode ② is played infinitely often.

The main result about $\mathcal{H}_{\mathcal{A}, t, N}$ is the following lemma. First, the winning condition of $\mathcal{H}_{\mathcal{A}, t, N}$ guarantees that if Pathfinder wins the game then he wins it positionally. To prove that he wins, one assumes that Automaton wins from some position, which leads to a contradiction.

► **Lemma 7.1.** *Pathfinder has a positional strategy in $\mathcal{H}_{\mathcal{A}, t, N}$ that is winning from every position of the game (we call such a strategy uniform).*

8 Completeness

We now move to the proof of Item 2 of Proposition 4.2: if \forall wins $\mathcal{G}_{\mathcal{A}}$ then $\eta_{\Gamma} < \omega \cdot N$ for some computable $N \in \mathbb{N}$. First observe the following important consequence of the winning condition of the game.

► **Proposition 8.1.** *There exists a computable bound $N \in \mathbb{N}$ such that if \forall wins $\mathcal{G}_{\mathcal{A}}$ then he has a winning strategy that guarantees the following. In every play Π (either finite or infinite) consistent with this strategy, every path in the graph obtained as the composition of back-markings $\bar{\mu}_1, \bar{\mu}_2, \dots$ contains less than N changes of sides from T to R.*

Compute the bound N as above. Assume that \forall wins the game $\mathcal{G}_{\mathcal{A}}$ and fix his winning strategy satisfying the thesis of the above proposition. We claim that then the closure ordinal of the automaton is bounded by $\omega \cdot N$. Assume for the sake of contradiction that there exists a tree t which does not have a witnessing tree x of rank smaller than $\omega \cdot N$. The above assumption about t implies that $\text{val}_t(\varepsilon, q_{\Gamma}) \geq \omega \cdot N$.

Based on the tree t , we now construct a play that is consistent with the fixed \forall 's strategy, but is won by \exists , leading to a contradiction. Together with the current position of the game, $(\langle T_n, R_n \rangle)$, we store some node v_n of the tree, starting with $v_0 = \varepsilon$. In order to construct the play, we need to provide choices of \exists : she declares a letter based on the label of v_n in the fixed tree t , and a set of selectors based on a fixed uniform positional strategy π_0 of Pathfinder in $\mathcal{H}_{\mathcal{A}, t, N}$, given by Lemma 7.1. The current node v_n is then updated according to the direction declared by \forall in $\mathcal{G}_{\mathcal{A}}$.

For a pair $(q, s) \in \langle T_n, R_n \rangle$ denote by $\text{hist}_n(q, s)$ the number of switches from the side T to the side R on the back-marked history of (q, s) . We keep a *val-preserving invariant* saying that for every pair $(q, s) \in \langle T_n, R_n \rangle$ with $k = N - \text{hist}_n(q, s)$ either

1. $s = \text{T}$ and $\text{val}_t(v_n, q) \geq \omega \cdot k$, or
2. $s = \text{R}$ and $\text{val}_t(v_n, q) > \omega \cdot k$.

Note that the val-preserving invariant is initially met, because $\text{val}_t(\varepsilon, q_{\Gamma}) \geq \omega \cdot N$. Note also that always $k > 0$ because Proposition 8.1 implies that $\text{hist}_n(q, s) < N$. The fact that the Pathfinder's strategy π_0 is winning in $\mathcal{H}_{\mathcal{A}, t, N}$ can be used to deduce that the constructed play is won by \exists in $\mathcal{G}_{\mathcal{A}}$. This concludes the proof of Proposition 4.2, hence also of Theorem 1.1.

9 Definability of ranks in MSO

We now move to a study of definability of particular rank bounds in MSO.

With some analogy to the cardinality quantifier by Bárány et al. [3], one can propose a quantifier $\exists^{\leq \eta} X. \varphi(\vec{Y}, X)$, expressing that there exists a well-founded set X of rank at most η such that $\varphi(\vec{Y}, X)$ holds. Note that this can be equivalently rewritten using the predicate $\text{rank}(X) \leq \eta$, defined by $\exists^{\leq \eta} Z. X = Z$. Below we show that the predicate $\text{rank}(X) \leq \eta$, and consequently the respective quantifier, cannot be expressed in MSO except for the basic case of natural numbers (or ω_1). Definability of this predicate in MSO boils down to checking if the following language is regular:

$$L_{\leq \eta} \stackrel{\text{def}}{=} \{x \in \text{Tr}_{\{0,1\}} \mid \text{rank}(x) \leq \eta\}.$$

Clearly, $\text{rank}(x) < \omega_1$ holds for every well-founded tree, thus it remains to consider $\eta < \omega_1$.

We first show that these languages are regular for $\eta < \omega$.

► **Fact 9.1.** *For every $l < \omega$ the language $L_{\leq l}$ is regular.*

Going beyond ω , the languages stop being regular. First, a simple pumping argument shows the following.

► **Lemma 9.2.** *For all ordinals $\eta < \omega_1$ and all $l < \omega$ the language $L_{\leq \eta + \omega + l}$ is not regular.*

The argument from Lemma 9.2 applies in particular to all ordinals η such that $\omega \leq \eta < \omega^2$, whereas the ordinals $\eta \geq \omega^2$ are covered by Theorem 1.1. Thus we may conclude.

► **Corollary 9.3.** *For an ordinal $\eta < \omega_1$ the language $L_{\leq \eta}$ is regular if and only if $\eta < \omega$.*

The above corollary does not exclude the possibility that a higher ordinal η may be a supremum of a regular subset of $L_{\leq \eta}$.

► **Lemma 9.4.** *For each pair of natural numbers k, l there exists a regular language $L \subseteq \text{WF}$ such that $\sup_{x \in L} \text{rank}(x) = \omega \cdot k + l$.*

► **Remark 9.5.** Clearly no countable formalism can define all the languages $L_{\leq \eta}$ for $\eta < \omega_1$. However, certain formalisms can go beyond MSO. For instance, the logic WMSO+U (for which the satisfiability problem is known to be decidable [4]) is capable of defining the language $L_{\leq \omega}$: a tree x has rank at most ω if below every node u labelled by 1 there is a bound K on the number of nodes labelled by 1 that can appear on branches of x passing through u . This construction can be iterated to define the languages $L_{\leq \omega \cdot k + l}$ and possibly even beyond that.

10 Closure ordinals

In this section we show how a negative answer to Czarnecki's question on closure ordinals can be derived from the present result. We use the standard syntax and semantics of modal μ -calculus [7], with its formulae constructed using the following grammar:

$$F ::= a \mid X \mid \mu X. F \mid \nu X. F \mid F_1 \vee F_2 \mid F_1 \wedge F_2 \mid \diamond F \mid \square F,$$

where $a \in A$ is a letter from a fixed alphabet, X is a variable from some fixed set of variables, μ and ν are the least and the greatest *fixed-point operators*, and \diamond and \square are the standard modalities (“exists a successor” and “for all successors”). For technical convenience we make

an assumption that, in each point of our model, exactly one proposition (letter in A) is satisfied. For the sake of readability we often identify a formula with its semantics, that is, we read a closed formula as a subset of the domain, and a formula with k free variables as a k -ary function over its powerset.

Given an ordinal number η , we use the standard notation $\mu^\eta X. F(X)$ for the η -approximation of the fixed point in a given model (which amounts to the η 's iteration $F^\eta(\emptyset)$). Given a model τ , we define the *closure ordinal of $\mu X. F(X)$ in τ* as the least ordinal η such that $\mu X. F(X) = \mu^\eta X. F(X)$. Then, the *closure ordinal of $\mu X. F(X)$* (as considered in Question 1.2) is the supremum of these ordinals over all models τ (or ∞ if the supremum does not exist). We aim at providing an (alternative to Afshari, Barlucchi, and Leigh [1]) proof of the following theorem.

► **Theorem 10.1.** *Let $F(X)$ be a μ -calculus formula in which the variable X does not occur in scope of any fixed-point operator. Then, the closure ordinal of $\mu X. F(X)$ is either strictly smaller than ω^2 , or at least ω_1 and it can be effectively decided which of the cases holds.*

Note that we allow arbitrary closed formulae of μ -calculus to be nested in F ; however, we do not cover the whole μ -calculus, because of the restriction on occurrences of X . This stays in line with the fragment considered by Afshari, Barlucchi, and Leigh [1] (as explained in Section 1), but we additionally provide a decision procedure that makes the dichotomy effective.

Towards a proof of Theorem 10.1, as a first step, we eliminate from F all occurrences of X that are not in scope of any modal operator; this can be done without changing the closure ordinal. Next, using standard techniques we obtain the following lemma.

► **Lemma 10.2.** *Let $F(X)$ be a formula as in Theorem 10.1, with all occurrences of X being in scope of a modal operator. The following three conditions are equivalent for every countable limit ordinal η :*

1. *the closure ordinal of $\mu X. F(X)$ is bounded by η ;*
2. *for every model τ that is a countable tree with its root in $\mu X. F(X)$, this root belongs to $\mu^\eta X. F(X)$;*
3. *for every model τ that is a countable tree with its root in $\mu X. F(X)$, there exists a well-founded set $Z \subseteq \text{dom}(\tau)$ containing the root, such that $F(Z) \supseteq Z$ and $\text{rank}(Z) \leq \eta$.*

A countable tree τ , occurring in Items 2 and 3 above, can be seen as a function $\tau: X \rightarrow A$ from a prefix-closed subset $X \subseteq \mathbb{N}^*$ to a finite alphabet A . Now, recall a natural encoding $(n_1, n_2, \dots, n_k) \mapsto \mathbb{R}^{n_1} \mathbb{L} \mathbb{R}^{n_2} \mathbb{L} \cdots \mathbb{R}^{n_k} \mathbb{L}$ of \mathbb{N}^* into $\{\mathbb{L}, \mathbb{R}\}^*$. This encoding preserves the prefix order on \mathbb{N}^* and moreover preserves ranks of well-founded sets. Take the relation Γ_F that contains $(t, x) \in \text{Tr}_A \times \text{Tr}_{\{0,1\}}$ if:

- t encodes a model τ ,
- x encodes a set $Z \subseteq \text{dom}(\tau)$,
- the root of τ belongs to $\mu X. F(X)$,
- the root of τ belongs to Z , and $F(Z) \supseteq Z$.

The μ -calculus formulae $F(Z)$ and $\mu X. F(X)$ can be rewritten into MSO [7] and then modified to read the above encoding of τ in the binary tree, instead of τ itself. It follows that the relation Γ_F is MSO-definable.

Observe that Item 3 of Lemma 10.2 can be rephrased by saying that the closure ordinal of Γ_F is bounded by η . Applying Theorem 1.1 to Γ_F , and then Lemma 10.2, we have one of two possibilities: If the closure ordinal of Γ_F is smaller than ω^2 , then it is bounded by ω^2 , then also the closure ordinal of $\mu X. F(X)$ is bounded by $\omega \cdot N$ for some $N \in \mathbb{N}$; then also the closure ordinal of $\mu X. F(X)$ is bounded by $\omega \cdot N < \omega^2$.

Otherwise, the closure ordinal of Γ_F is ω_1 , so it is not bounded by any countable limit ordinal; then also the closure ordinal of $\mu X. F(X)$ is not bounded by any countable limit ordinal, hence it is at least ω_1 . This concludes the proof of Theorem 10.1.

► **Remark 10.3.** One may ask if Theorem 10.1 is merely a consequence or it is in some sense equivalent to our main result Theorem 1.1. To the best of our understanding, Theorem 10.1 does not transfer back to the general realm of MSO-definable relations, as in Theorem 1.1. One of the reasons is that the iterations of fixed points are required to proceed in a monotone fashion, driven by the internal formula F ; while in full MSO one can express arbitrary correspondence between the parameters \vec{Y} and a well-founded witness X .

11 Conclusions

This work contributes to the study of expressive power of the MSO theory of binary tree. We investigate to what extent this theory can express properties of well-founded trees, and in particular distinguish between their ordinal ranks. We observe that the ability of explicit expression of properties of ranks is practically limited to statements of the form: all trees X satisfying $\varphi(X)$ have $\text{rank}(X) < N$, for a fixed $N \in \mathbb{N}$ (cf. Corollary 9.3 above). However the implicit expressive power of MSO logic goes much higher. In particular, our main result (Theorem 1.1) allows us to decide whether the property

$$\exists X. \varphi(\vec{Y}, X) \wedge X \text{ is well-founded with } \text{rank}(X) < \omega^2,$$

is generally true (for all \vec{Y}), although the property itself is not expressible in MSO.

There is, however, a number of questions that remain to be answered. As ordinals smaller than ω^2 can be effectively represented, we would like to have an effective procedure that, given a formula φ , computes the exact bound, that is, (a representation of) the least ordinal η_φ that can be substituted for ω^2 in the construction above. Even more elementarily, given an MSO-definable set L of well-founded trees, we would like to compute the supremum of ranks of trees in L . These questions are subjects of ongoing research.

A more far-reaching direction is to relate the techniques of the present paper to the open problem of computing the Mostowski index, mentioned in Introduction. The parity condition itself imposes well-foundedness restriction on the occurrences of each odd label m in the fragments of tree where this label is the highest. Colcombet and Löding [11] have approached the index problem (still unsolved) by reducing it to the boundedness problem for distance automata (see also Idir and Lehtinen [20] for a simplified version of this reduction). One may consider an alternative approach towards the index problem by studying the ordinal ranks which arise from the well-foundedness restriction of the parity condition.



References

- 1 Bahareh Afshari, Giacomo Barlucchi, and Graham E. Leigh. The limit of recursion in state-based systems. In *FICS*, 2024. URL: https://www.irif.fr/_media/users/saurin/fics2024/pre-proceedings/fics-2024-afshari-et-al.pdf.
- 2 Bahareh Afshari and Graham E. Leigh. On closure ordinals for the modal mu-calculus. In *CSL*, volume 23 of *LIPICs*, pages 30–44. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPICs.CSL.2013.30.
- 3 Vince Bárány, Łukasz Kaiser, and Alexander M. Rabinovich. Expressing cardinality quantifiers in monadic second-order logic over trees. *Fundam. Inform.*, 100(1-4):1–17, 2010. doi:10.3233/FI-2010-260.

- 4 Mikołaj Bojańczyk. Weak MSO+U with path quantifiers over infinite trees. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2014. doi:10.1007/978-3-662-43951-7_4.
- 5 Julian C. Bradfield. Simplifying the modal mu-calculus alternation hierarchy. In *STACS*, volume 1373 of *Lecture Notes in Computer Science*, pages 39–49. Springer, 1998. doi:10.1007/BFB0028547.
- 6 Julian C. Bradfield, Jacques Duparc, and Sandra Quickert. Transfinite extension of the mu-calculus. In *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 384–396. Springer, 2005. doi:10.1007/11538363_27.
- 7 Julian C. Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In *Handbook of Model Checking*, pages 871–919. Springer, 2018. doi:10.1007/978-3-319-10575-8_26.
- 8 Julius R. Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
- 9 Lorenzo Clemente and Michał Skrzypczak. Deterministic and game separability for regular languages of infinite trees. In *ICALP*, volume 198 of *LIPICs*, pages 126:1–126:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.126.
- 10 Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. Deciding the weak definability of Büchi definable tree languages. In *CSL*, volume 23 of *LIPICs*, pages 215–230. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013. doi:10.4230/LIPICs.CSL.2013.215.
- 11 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 2008. doi:10.1007/978-3-540-70583-3_33.
- 12 Marek Czarnecki. How fast can the fixpoints in modal mu-calculus be reached? In *FICS*, pages 35–39. Laboratoire d’Informatique Fondamentale de Marseille, 2010. URL: <https://hal.archives-ouvertes.fr/hal-00512377/document#page=36>.
- 13 Christian Delhommé. Automaticité des ordinaux et des graphes homogènes. *Comptes Rendus de L’Académie des Sciences, Mathématiques*, 339(1):5–10, 2004.
- 14 Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Derivation tree analysis for accelerated fixed-point computation. *Theor. Comput. Sci.*, 412(28):3226–3241, 2011. doi:10.1016/J.TCS.2011.03.020.
- 15 Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Rabin-Mostowski index problem: A step beyond deterministic automata. In *LICS*, pages 499–508. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.56.
- 16 Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on graphs. *CoRR*, abs/2305.10546, 2023. doi:10.48550/arXiv.2305.10546.
- 17 Olivier Finkel and Stevo Todorćević. Automatic ordinals. *Int. J. Unconv. Comput.*, 9(1-2):61–70, 2013. URL: <http://www.oldcitypublishing.com/journals/ijuc-home/ijuc-issue-contents/ijuc-volume-9-number-1-2-2013/ijuc-9-1-2-p-61-70/>.
- 18 Gaëlle Fontaine. Continuous fragment of the mu-calculus. In *CSL*, volume 5213 of *Lecture Notes in Computer Science*, pages 139–153. Springer, 2008. doi:10.1007/978-3-540-87531-4_12.
- 19 Maria J. Gouveia and Luigi Santocanale. \aleph_1 and the modal μ -calculus. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:1)2019.
- 20 Olivier Idir and Karoliina Lehtinen. Mostowski index via extended register games. *CoRR*, abs/2412.16793, 2024. doi:10.48550/arXiv.2412.16793.
- 21 Alexander Kechris. *Classical descriptive set theory*. Springer-Verlag, New York, 1995.
- 22 Damian Niwiński. On the cardinality of sets of infinite trees recognizable by finite automata. In *MFCs*, volume 520 of *Lecture Notes in Computer Science*, pages 367–376. Springer, 1991. doi:10.1007/3-540-54345-7_80.

- 23 Damian Nawiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003. doi:10.1016/S0304-3975(02)00452-8.
- 24 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.
- 25 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
- 26 Sylvain Schmitz. *Algorithmic Complexity of Well-Quasi-Orders. (Complexité algorithmique des beaux pré-ordres)*. École normale supérieure Paris-Saclay, 2017. URL: <https://tel.archives-ouvertes.fr/tel-01663266>.
- 27 Michał Skrzypczak. *Descriptive Set Theoretic Methods in Automata Theory – Decidability and Topological Complexity*, volume 9802 of *Lecture Notes in Computer Science*. Springer, 2016. doi:10.1007/978-3-662-52947-8.
- 28 Michał Skrzypczak and Igor Walukiewicz. Deciding the topological complexity of Büchi languages. In *ICALP*, volume 55 of *LIPICs*, pages 99:1–99:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ICALP.2016.99.
- 29 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages (3)*, pages 389–455. Springer, 1997. doi:10.1007/978-3-642-59126-6_7.



Colorful Vertex Recoloring of Bipartite Graphs

Boaz Patt-Shamir  

School of Electrical Engineering, Tel Aviv University, Israel

Adi Rosén 

CNRS and Université Paris Cité, France

Seeun William Umboh  

School of Computing and Information Systems, The University of Melbourne, Australia

ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Melbourne, Australia

Abstract

We consider the problem of vertex recoloring: we are given n vertices with their initial coloring, and edges arrive in an online fashion. The algorithm is required to maintain a valid coloring by means of vertex recoloring, where recoloring a vertex incurs a cost. The problem abstracts a scenario of job placement in machines (possibly in the cloud), where vertices represent jobs, colors represent machines, and edges represent “anti affinity” (disengagement) constraints. Online coloring in this setting is a hard problem, and only a few cases were analyzed. One family of instances which is fairly well-understood is bipartite graphs, i.e., instances in which two colors are sufficient to satisfy all constraints. In this case it is known that the competitive ratio of vertex recoloring is $\Theta(\log n)$.

In this paper we propose a generalization of the problem, which allows using additional colors (possibly at a higher cost), to improve overall performance. Concretely, we analyze the simple case of bipartite graphs of bounded largest *bond* (a bond of a connected graph is an edge-cut that partitions the graph into two connected components). From the upper bound perspective, we propose two algorithms. One algorithm exhibits a trade-off for the uniform-cost case: given $\Omega(\log \beta) \leq c \leq O(\log n)$ colors, the algorithm guarantees that its cost is at most $O(\frac{\log n}{c})$ times the optimal offline cost for two colors, where n is the number of vertices and β is the size of the largest bond of the graph. The other algorithm is designed for the case where the additional colors come at a higher cost, $D > 1$: given Δ additional colors, where Δ is the maximum degree in the graph, the algorithm guarantees a competitive ratio of $O(\log D)$. From the lower bounds viewpoint, we show that if the cost of the extra colors is $D > 1$, no algorithm (even randomized) can achieve a competitive ratio of $o(\log D)$. We also show that in the case of general bipartite graphs (i.e., of unbounded bond size), any deterministic online algorithm has competitive ratio $\Omega(\min(D, \log n))$.

2012 ACM Subject Classification Theory of computation \rightarrow Online algorithms; Theory of computation \rightarrow Graph algorithms analysis; Theory of computation \rightarrow Adversary models

Keywords and phrases online algorithms, competitive analysis, resource augmentation, graph coloring

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.70

Related Version *Full Version*: <https://arxiv.org/abs/2501.05796>

Funding *Boaz Patt-Shamir*: This research was supported by the Israel Science Foundation, grant No. 1948/21.

Adi Rosén: Research partially supported by CNRS grant IEA ALFRED and ANR grant AlgoriDAM.

Seeun William Umboh: Funded by CNRS grant IEA ALFRED and by the Australian Government through the Australian Research Council DP240101353.

1 Introduction

Motivation. In the cloud, jobs are placed on machines according to multiple criteria. Sometimes, during the execution of a job, it turns out that the job is in conflict with another job, in the sense that the two jobs must not run on the same machine. Such conflicts may



© Boaz Patt-Shamir, Adi Rosén, and Seeun William Umboh;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 70; pp. 70:1–70:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



arise due to limited resources on a single machine, or due to security considerations, or other reasons. Whenever such a conflict between two jobs located in the same machine is detected, the system needs to migrate one of the conflicting jobs (or both) so as to separate them. This situation was abstracted in [1] as the “vertex recoloring” (or “disengagement”) problem. Generally speaking, the problem is stated as an online problem, where initially we are given n vertices (representing jobs) colored in k colors (representing machines). Edges (representing conflicts) arrive online, and the algorithm is asked to output a valid vertex coloring after the arrival of each edge. Recoloring a vertex incurs a cost, and the cost incurred by an algorithm is the total cost of vertex recoloring events by the algorithm. As usual with online problems, the performance measure is the competitive ratio, namely the worst-case ratio between the cost paid by the online algorithm in question and the best cost possible (offline).

In this paper we seek to generalize the problem so as to make it both richer theoretically and more realistic practically. Our starting point is the following. Recalling the motivating scenario of job migration as outlined above, we note that in many cases, it is possible to acquire more machines on which jobs may be executed. Therefore, we are interested in understanding what are the consequences of allowing the online algorithm to use more colors. Our reference cost is still the optimal cost when using the initial k colors, but now we allow the online algorithm to use more colors.

The idea is not new: in fact, it was used in the very first paper on competitive analysis [12] (for cache size). Later work called this type of comparison “resource augmentation” (see, e.g., [5]). However, in this paper, still motivated by the job migration scenario, we propose an additional generalization: we also consider the case in which the additional colors come at a greater cost. We are not aware of any previous study of such a model.¹ We believe that this approach, which we call *weighted* resource augmentation, may be appropriate in other scenarios as well.

Our results. The problem of vertex recoloring is natural and applicable in many situations, but one has to bear in mind that it entails the problem of graph coloring, which is notoriously hard, computationally speaking. Following the approach of [1], we focus on polynomially-solvable instances of coloring. Specifically, in this paper we consider bipartite graphs, i.e., we assume that after all edges have arrived, the graph is 2-colorable. In [1] it was shown that the competitive ratio of online vertex recoloring is $\Theta(\log n)$ in the case of bipartite graphs, where n is the number of vertices. We study the effect of using more colors in this case. In the following, the competitive ratio is w.r.t. the 2-color optimal solution. For general bipartite graphs and deterministic algorithms it turns out that extra colors do not really help:

► **Theorem 1.** *Let \mathcal{I} be the set of instances of recoloring with two basic colors and n special colors, where recoloring by a basic color costs 1 and recoloring by a special color costs D . Then for every deterministic online recoloring algorithm there is an instance in \mathcal{I} with competitive ratio $\Omega(\min\{\log n, D\})$.*

(We note that competitive ratio D is trivial to achieve when given n special colors. See Lemma 5.) We therefore restrict our attention to a subclass of bipartite graphs, namely bipartite graphs of bounded *bond* size. A bond of a connected graph is a set of edges whose removal partitions the graphs into two connected components (alternatively, a minimal edge cut, cf. [4], and see Definition 6). Our main result in this paper is the following:

¹ In paging, this could be interpreted as follows: The offline solution uses a cache of size k , while the online solution has the cache of size k and an additional cache whose *replacement cost may be greater*.

► **Theorem 2.** *Suppose that the final graph is bipartite, with all vertex degrees at most Δ , and with bond size at most β for each of its connected components. Then there is a deterministic algorithm that uses Δ special colors of cost at most D , whose competitive ratio is $O(\log D + \beta^2)$.*

We note that in acyclic graphs, the largest bond size is 1. The following result shows that the competitive ratio of Theorem 2 cannot be improved even if we restrict the instances to have largest bond size 1.

► **Theorem 3.** *Consider instances in which the final graph is a collection of paths, and there is an infinite set of special colors, each of cost $D > 1$. The competitive ratio of any (possibly randomized) recoloring algorithm in this class of instances is $\Omega(\log D)$.*

On the other hand, if $D = 1$ (i.e., the cost of all colors is 1), another algorithm provides a way to reduce the competitive ratio at the price of using more colors.

► **Theorem 4.** *Suppose that the final graph is bipartite with bond at most β . Then for any given $\frac{1}{2 \log \beta + 3} < \epsilon \leq 1$ there is a deterministic online algorithm that uses $O(\epsilon \log n)$ special colors of unit cost, and guarantees competitive ratio $O(\epsilon^{-1})$.*

Our techniques. Our algorithms use Algorithm 1 of [1], denoted \mathcal{A} in this paper, as a black box. To facilitate our algorithms, however, we need to develop a refined analysis of Algorithm \mathcal{A} . In Lemma 17 we prove a tighter upper bound on the cost of \mathcal{A} when applied to an input sequence that satisfies a certain condition (“moderate sequences,” cf. Definition 7). Using this bound, the main idea in our algorithms is to filter the input sequence so as to make it moderate, and to feed that sequence to \mathcal{A} for simulation (which we know to have good performance). The difference between our algorithms is what to do with the other edges, and how to determine the color of vertices which may be affected by the filtering.

The approach taken by our Algorithm \mathcal{B} (whose performance is stated Theorem 2), is to use special colors, essentially in greedy fashion, to resolve conflicts that involve edges that were rejected from simulation due to the moderate sequence condition. We show that this approach can guarantee $O(\log D)$ competitiveness while using Δ additional colors. The other approach, taken by our Algorithm \mathcal{C} , is to have a hierarchical set of instantiations (simulations) of \mathcal{A} . If an edge is rejected by one instantiation, one of its endpoints is sent to the next instantiation in the hierarchy, which uses a new set of colors; and if this fails, we send that vertex to the next instantiation etc. We show that the number of edges in this hierarchy of simulations decreases exponentially as a function of a parameter of the “moderation” of the input sequence, which allows us to prove Theorem 4.

We note that the largest bond size proves to be a very useful graph parameter, as it measures, in some precise sense, how far a graph is from a tree. Our Lemma 10 gives us a tool which may be handy in other contexts as well.

Related work. As mentioned above, the problem of vertex recoloring was introduced in [1], where vertices have weights, and the cost of recoloring a vertex is its weight. It is shown that for bipartite graphs, competitive ratio of $O(\log n)$ can be achieved by a deterministic online algorithm, and that no randomized algorithm has competitive ratio $o(\log n)$. Tight bounds are also presented for $(\Delta + 1)$ coloring for randomized and deterministic algorithms where Δ denotes the maximal degree in the graph. Recently, Rajaraman and Wasim [11] considered the capacitated setting where there is a bound B on the number or weight of vertices in each color. They give “traditional” resource-augmented algorithms: the algorithms are allowed to

violate the capacity bound by a $(1 + \epsilon)$ where ϵ is an arbitrarily small constant. They also study the $(1 + \epsilon)$ -overprovisioned setting where the algorithm is allowed Δ colors and the maximum degree of the graph is bounded by $(1 - \epsilon)\Delta$.

Recoloring (or coloring with recourse) has been considered previously in the context of dynamic data structures [6, 3, 13, 10]. In these papers, no initial coloring is given and the competitive ratio is not analyzed; their measure of performance is the absolute number of recolorings. Competitive analysis is implicit in [3, 13] which is bicriteria: the arrival model is similar to ours but the final graph may be arbitrary, and the goal is to minimize both the update time and the number of colors used. In this line of work, it is assumed that the algorithm has access to an oracle that can be queried about the chromatic number of a graph. The best general result is due to Solomon and Wein [13], who give a deterministic algorithm with $O(d)$ amortized running time using $O\left(\frac{\log^3 n}{d} \chi(G)\right)$ colors. Henzinger et al. [9] give better results for bounded arboricity graphs.

Although the largest bond and maximum cut of a graph may seem superficially similar, they are quite different, both in value and complexity. For example, finding the largest graph bond in bipartite graphs is NP-hard [7], but max-cut is trivial in bipartite graphs, and it is polynomially computable in planar graphs [8]; any tree has a max-cut of size $\Omega(n)$ but largest bond size 1.

Paper organization. The remainder of this paper is organized as follows. In Section 2, we formalize the problem and introduce some notation. In Section 3, we prove our main result Theorem 2. In Section 4, we consider the uniform case and prove Theorem 4. Additional material is included in the full version: we prove our lower bounds Theorem 3 and Theorem 1, and present a slightly improved version of Algorithm \mathcal{B} . A short conclusion is presented in Section 6.

2 Problem Statement and Notation

Problem statement. We consider the following model. Initially we are given a palette P of k basic colors, and a superset $P^* \supseteq P$ of colors. The extra colors in $P^* \setminus P$ are called *special* colors. Each color j is assigned a cost $\text{cost} : P^* \rightarrow \mathbb{R}^+$, such that $\text{cost}(j) = 1$ for all basic colors, and $1 \leq \text{cost}(j) \leq D$ for all special colors, where $D \geq 1$ is some given parameter. We are also given a set of n vertices V with an initial coloring $c_0 : V \rightarrow P$ using only the basic colors. The online input is a sequence of edges e_1, e_2, \dots , where each edge is an unordered pair of distinct vertices. We define the graph G_i by $G_i = (V, \{e_1, \dots, e_i\})$. In response to the arrival of each edge e_i , the algorithm outputs a coloring $c_i : V \rightarrow P^*$ such that none of the edges in G_i is monochromatic.

Given an algorithm A , an initial coloring c_0 and an edge sequence $\sigma = (e_1, \dots, e_\ell)$, $A(c_0, \sigma)$ is a sequence of colorings c_1, \dots, c_ℓ . Define the cost of A on an instance (c_0, σ) as

$$\text{cost}_A(c_0, \sigma) = \sum_{i=1}^{\ell} \sum_{v \in V} (1 - \delta_{c_{i-1}(v)c_i(v)}) \text{cost}(c_i(v)) ,$$

where δ is the Kronecker delta. That is, whenever a vertex is recolored, the algorithm pays the cost of its new color. In this paper we assume that the input is such that the final graph can be colored using P , i.e., G_ℓ is k -colorable. Given an initial coloring c_0 , the *offline cost* of an input sequence σ with respect to the basic colors P is

$$\text{OPT}_k(c_0, \sigma) = \min \left\{ \sum_{v \in V} (1 - \delta_{c_0(v)c^*(v)}) \mid c^* : c^* \text{ is a valid } k\text{-coloring of } G_i \text{ by } P \right\}.$$

In other words, the offline cost of σ w.r.t. k is the least cost of recoloring V using basic colors so that G_i has no monochromatic edges.

Let $I = (c_0, \sigma)$ denote an instance. The competitive ratio of an algorithm A with respect to k is $\sup_I \{\text{cost}_A(I)/\text{OPT}_k(I)\}$. Let us make a quick observation about the problem.

► **Lemma 5.** *Any instance I of recoloring with k basic colors can be solved by an online algorithm using $n - k$ special colors at cost at most $2D \cdot \text{OPT}_k(I)$, where n is the number of vertices and D is the cost of a special color.*

Proof. Consider the final graph, with vertices colored by the initial coloring. Any feasible solution must recolor a vertex cover of the monochromatic edges in this graph. Therefore, denoting the size of such a minimum vertex cover by q , we have that $\text{OPT}_k(I) \geq q$. On the other hand, we can maintain in an online manner a 2-approximate vertex cover (e.g., using an online version of the algorithm of Bar-Yehuda and Even [2]), and every time a vertex enters the online cover, we recolor it with a distinct new color. The special colors never collide because each is used at most once, and hence the total cost is at most $2qD$. ◀

Additional notation.

- Given a sequence σ , $\sigma[i]$ denotes the prefix of the first i elements of σ . Given sequences σ and σ' , $\sigma \circ \sigma'$ denotes the sequence obtained by concatenating σ and σ' .
- Given a graph $G = (V, E)$ and $V' \subseteq V$, $G[V']$ denotes the graph induced by V' , i.e., the graph with vertices V' whose edges are all edges of E with both endpoints in V' .

Graph bond. The largest bond size is a graph parameter related to max cut, but it is quite different. Intuitively, the graph bond is a measure of how close the graph is to a tree. We give below a definition that generalizes the standard one (e.g., [7]) to graphs with multiple connected components.

► **Definition 6.** *Let $G = (V, E)$ be a graph with $k \geq 1$ connected components. An edge set $B \subseteq E$ is a **bond** of G if $(V, E \setminus B)$ has exactly $k + 1$ connected components. The size of the largest bond of G is denoted $\beta(G)$.*

Note that $\beta(G) = 1$ if and only if G is a forest. Also note that the size of the largest bond of a graph is monotone in its edge set. More formally, if $G = (V, E)$ and $G' = (V, E')$ are two graphs with the same vertex set, then $E \subseteq E'$ implies $\beta(G) \leq \beta(G')$.

We remark that an alternative definition for bond is a *minimal edge cut* [4]: an edge cut of $G = (V, E)$ is an edge set $B \subseteq E$ such that $(V, E \setminus B)$ has more connected components than G , and an edge cut B is *minimal* if no proper subset of B is an edge cut of G .

3 Non-Uniform Cost

In this section we prove our main result. Intuitively, we show that while the competitive ratio for general bipartite graphs is known to be $\Theta(\log n)$, by using $\Delta + 1$ additional colors of cost (at most) D , one can reduce the competitive ratio to $\Theta(\log D)$ in graphs whose largest bond is small. In the full version, we show that we can achieve the same competitive ratio with only Δ additional colors, using a somewhat more complicated algorithm.

Our algorithm uses the algorithm of Azar *et al.* [1] for bipartite graphs as a black box; henceforth, we refer to Algorithm 1 of [1] as \mathcal{A} . The basic strategy of our algorithm is as follows. The input sequence of edges σ is split in two: a subsequence denoted σ^{sim} is sent to a simulation by algorithm \mathcal{A} (which uses only the two basic colors), and the remaining edges are sent to a procedure called **recx**, which uses the additional special colors. Determining which edge goes to σ^{sim} depends on the size of the connected components of its two endpoints, and by the number of vertices already recolored by \mathcal{A} in them. The idea is to control the simulation of \mathcal{A} so that its cost remains within the range of $O(\log D)$ competitiveness, and use the special colors only once we know that we can pay for them. The bond size affects the total cost due to the edges that are not sent to the simulation.

3.1 Algorithm \mathcal{B} : Specification

To describe the algorithm, we need some notation. Recall that $\sigma[i]$ is the sequence of the first i edges of σ . We use E_i to denote the set of edges in $\sigma[i]$: $E_i = \{e_1, e_2, \dots, e_i\}$. We use σ_i^{sim} to denote the subsequence of edges sent to the simulation of \mathcal{A} in steps $1, \dots, i$, and $E[i]_{\text{sim}}$ to denote the corresponding set of edges. R_i is used to denote the set of vertices ever recolored by algorithm \mathcal{A} when executed on input σ_i^{sim} . We also define R to be $R_{|\sigma|}$.

We now specify the algorithm, using the parameters D (the maximal cost of a color), and $\alpha \in (0, 1)$, a parameter of our choice which indirectly controls how many of the edges will be sent to the simulation: the smaller α is, less edges will be sent to the simulation.

We also define the following concepts.

► **Definition 7.** Consider a sequence of edges σ and a connected component C of the graph which is edge-induced by the set of edges in σ .

- C is **small** if $|C| \leq D$ and **large** otherwise, where $|C|$ is the number of vertices in C .
- C is σ -**light** if $|R \cap C|/|C| \leq \alpha$, and σ -**heavy** otherwise, where R is the set of vertices ever recolored by algorithm \mathcal{A} on input σ .

► **Definition 8.** Consider an input sequence σ , and two values D and α .

- An input sequence σ is called (D, α) -**moderate** if for every edge $e_i = (u, v)$ in σ , it holds that if u and v are in two distinct connected components C_u and C_v with respect to e_1, \dots, e_{i-1} , then either C_u or C_v is either small or $\sigma[i-1]$ -light (i.e., no edge in σ connects two distinct components which are large and $\sigma[i-1]$ -heavy).
- The two series of subsequences σ_i^{sim} and σ_i^{exc} of σ are defined inductively as follows.

$$\sigma_i^{\text{sim}} = \begin{cases} \perp & \text{if } i = 0 \\ \sigma_{i-1}^{\text{sim}} \circ e_i & \text{if } i > 0 \text{ and } \sigma_{i-1}^{\text{sim}} \circ e_i \text{ is } (D, \alpha)\text{-moderate} \\ \sigma_{i-1}^{\text{sim}} & \text{if } i > 0 \text{ and } \sigma_{i-1}^{\text{sim}} \circ e_i \text{ is not } (D, \alpha)\text{-moderate} \end{cases}$$

$$\sigma_i^{\text{exc}} = \begin{cases} \perp & \text{if } i = 0 \\ \sigma_{i-1}^{\text{exc}} & \text{if } i > 0 \text{ and } \sigma_{i-1}^{\text{sim}} \circ e_i \text{ is } (D, \alpha)\text{-moderate} \\ \sigma_{i-1}^{\text{exc}} \circ e_i & \text{if } i > 0 \text{ and } \sigma_{i-1}^{\text{sim}} \circ e_i \text{ is not } (D, \alpha)\text{-moderate} \end{cases}$$

(Informally, e_i is appended to $\sigma_{i-1}^{\text{sim}}$ if it keeps the moderation property, and to $\sigma_{i-1}^{\text{exc}}$ otherwise.)

The pseudocode for the algorithm appears in the following page.

3.2 Analysis

We now turn to analyze the algorithm.

The interesting part about the algorithm is that the simulation of \mathcal{A} is unaware of some of the edges. We start by showing that Algorithm \mathcal{B} produces a valid coloring.

► **Lemma 9.** *After every step i , c_i is a valid coloring of the graph $G_i = (V, E_i)$.*

Proof. First observe that once a vertex becomes special, then it is always colored by a special color. Likewise, only special vertices are colored by special colors.

Algorithm \mathcal{B}

State:

- Each vertex u has an *actual* color $c(u)$ and a *simulated* color $\bar{c}(u)$. The simulated coloring \bar{c} is the coloring maintained by the simulation of \mathcal{A} . The initial actual colors are given as input, and the initial simulated colors are the initial actual colors.
- Each vertex records whether its simulated color was ever changed by the simulation of \mathcal{A} . This allows the algorithm to maintain the set R_i .
- Each vertex has an indication whether it is “special” or not. Initially all vertices are not special.

Action:

Upon the arrival of edge $e_i = (u_i, v_i)$:

- a. If $\sigma_{i-1}^{\text{sim}} \circ e_i$ is (D, α) -moderate then
 - send e_i to \mathcal{A} (which updates $\bar{c}(\cdot)$); $// \sigma_i^{\text{sim}} = \sigma_{i-1}^{\text{sim}} \circ e_i$
 - set $c(w) := \bar{c}(w)$ for every non-special vertex w .
- b. Else
 - If both u_i and v_i are not special then mark u_i as *special*.
- c. Invoke **recx** (u_i, v_i)

Procedure **recx (u, v) :**

1. If u and v are special then
 - if $e = (u, v)$ is monochromatic then
 - recolor u with a free special color. $//$ there are $\Delta + 1$ special colors
2. Else
 - If u is special then
 - if u is colored by a basic color then
 - recolor u with a free special color. $//$ first time u is colored special
3. Else
 - If v is special then $//$ this case cannot happen because of the way **recx** is used.
 - if v is colored by a basic color then
 - recolor v with a free special color.

We prove the lemma by induction on i . The base case is $i = 0$, in which $E_0 = \emptyset$ and hence any coloring is valid. For the inductive step, we first consider all edges but the new edge e_i and for those we proceed in two sub-steps. Then we consider the new edge e_i .

For all edges but e_i , if e_i is not a simulated edge and is not sent to \mathcal{A} , then the first sub-step is empty. If e_i is a simulated edge and is sent to \mathcal{A} , then the colors of some non-special vertices may change. After this sub-step, for each (old) edge: (1) if its two endpoints are non-special then the edge is not monochromatic by the correctness of \mathcal{A} ; (2) if its two endpoints are special, then their color does not change in this sub-step and the edge is not monochromatic by the induction hypothesis; (3) if one endpoint is special and the other is not, then one and only one endpoint is colored by a special color, and hence the edge is not monochromatic.

The second sub-step is the invocation of **recx** on the input edge. Following this invocation one vertex might be recolored to a special color. Clearly any (old) edge with at least one node not special remains non-monochromatic by the induction hypothesis. For an (old) edge with two special endpoints, if their color is not changed, they remain non-monochromatic by the induction hypothesis. For such an edge for which one of its endpoints changed color by **recx**, the code ensures that it is non-monochromatic after the execution of **recx**.

70:8 Colorful Vertex Recoloring of Bipartite Graphs

Now as to the edge e_i itself, we have a number of cases depending whether it is a simulated edge and the status (special or not) of its two endpoints when step i begins.

1. If its two endpoints are non-special when step i starts:
 - if e_i is a simulated edge: at the end of the first sub-step e_i is not monochromatic by the correctness of \mathcal{A} ; none of its endpoints is marked special and hence recx does not recolor any node and e_i remains non-monochromatic.
 - if e_i is not a simulated edge: One of its endpoints is marked special; recx recolors that vertex by a special color, while the other endpoint remains colored by a basic color; hence e_i is not monochromatic.
2. If its two endpoints are special when step i starts, then regardless of whether e_i is a simulated edge or not the code of recx ensures that e_i is not monochromatic.
3. If one endpoint is special and the other is not when step i starts, then regardless of whether e_i is a simulated edge or not, the status of neither vertex changes, and one of them remains colored by a special color and the other by a basic color, hence e_i is not monochromatic. ◀

We now turn to analyze the competitiveness of our algorithm. To this end, we first prove the following property about graphs with bounded bond size.

► **Lemma 10.** *Let $G = (V, E)$ be a connected graph with largest bond size at most β , and let $\{V_1, \dots, V_k\}$ be a partition of V such that the induced subgraph $G[V_i]$ is connected, for every $1 \leq i \leq k$. Then the number of edges which are not contained in any of these induced subgraphs (i.e., the number of edges with endpoints in two different parts of the partition) is at most $(k - 1)\beta$.*

Proof. Consider the multi-graph $G' = (V', E')$ obtained from G by contracting each V_i to a single node and eliminating self-loops. We shall prove that $|E'| \leq (k - 1)\beta$.

First, we note that $|V'| = k$, and that the size of the largest bond of G' is at most β . We now prove, by induction on k , that if a loop-free multigraph $G' = (V', E')$ has k nodes and bond size at most β , then $|E'| \leq (k - 1)\beta$. The base case is $k = 1$; in this case, E' is empty since G' does not contain self loops.

For the inductive step, fix any $k \geq 2$. Let v be an arbitrary node in V' , and let $\bar{V} = V' \setminus \{v\}$. We proceed according to the connectivity of \bar{V} . If $G'[\bar{V}]$ is connected, then the degree of v is at most β . Since $|\bar{V}| = k - 1$, by the induction hypothesis and the fact that the size of the largest bond of $G'[\bar{V}]$ is at most the size of the largest bond of G' (which is at most β), we have that $|E'| \leq \beta + (k - 2)\beta = (k - 1)\beta$.

Otherwise, $G'[\bar{V}]$ is disconnected. Let C_1 be the set of vertices of an arbitrary connected component of \bar{V} , and denote $C_2 = \bar{V} \setminus C_1$. Let $C_1^v = C_1 \cup \{v\}$, and $C_2^v = C_2 \cup \{v\}$. Trivially $|C_1^v| + |C_2^v| = k + 1$, and $m_1 + m_2 = |E'|$, where m_1 and m_2 are the number of edges in $G'[C_1^v]$ and in $G'[C_2^v]$, respectively. Since the largest bond of a subgraph is no larger than the largest bond of the original graph, and since the number of nodes in each of $G'[C_1^v]$ and $G'[C_2^v]$ is strictly less than k , by the inductive hypothesis we have that $m_1 \leq (|C_1^v| - 1)\beta$ and $m_2 \leq (|C_2^v| - 1)\beta$. It follows that $|E'| = m_1 + m_2 \leq (k - 1)\beta$. ◀

► **Corollary 11.** *Let $G = (V, E)$ be a graph with $\beta(G) \leq \beta$, and let C_1, \dots, C_k be a collection of disjoint subsets of vertices in V such that $G[C_i]$ is connected for all $1 \leq i \leq k$. Then $|\{(v, u) \in E : v \in C_i, u \in C_j, i \neq j\}| \leq \beta(k - 1)$, i.e., the number of edges with endpoints in two different subsets is at most $\beta(k - 1)$.*

Proof. Let V_1, \dots, V_k, V_{k+1} be any partition of V such that:

- for each $1 \leq i \leq k$, $C_i \subseteq V_i$, and $G[C_i]$ is connected; and
- V_{k+1} is exactly the set of vertices which are not connected in G to any vertex of $\bigcup_{i=1}^k C_i$.

This can be achieved by associating each vertex of $V \setminus V_{k+1}$ with the set C_i closest to it (like in a Voronoi partition). Note that we may assume without loss of generality that V_{k+1} is empty: otherwise, consider the graph $G' \stackrel{\text{def}}{=} G[V \setminus V_{k+1}]$: Clearly, by the definition of V_{k+1} , the number of edges connecting vertices in different C_i in G and G' is the same.

Since every edge between two subsets C_i, C_j connects different parts of the partition V_i and V_j , the result follows from Lemma 10, when applied to each connected component of G separately. ◀

We proceed to bound from above the cost of Algorithm \mathcal{B} . The cost consists of two parts: the cost incurred by Procedure `recx` and the cost due to the simulation of \mathcal{A} . We start by stating a number of facts due to the definition of the algorithm.

► **Fact 12.** *A vertex which is colored by a special color after step i must belong to a connected component of $E[i - 1]_{\text{sim}}$ that has more than αD vertices of R .*

Proof. Observe that a node v is colored by a special color only in procedure `recx`, when it is a special vertex. Moreover, it is marked special only when an edge $e_j = (v, u)$ arrives, and that edge is in $E \setminus E_{\text{sim}}$. It follows that there is an edge e_j , for $j \leq i$, which connects two large and heavy connected components of $E[j - 1]_{\text{sim}}$. Hence, v is part of a large connected component of $E[j - 1]_{\text{sim}}$ which has more than αD vertices of R . ◀

Recalling the sequences σ^{exc} and σ^{sim} defined in Definition 8 as a function of any sequence σ , we prove the following lemma that allows us to (indirectly) give an upper bound on the number of vertices that require “special treatment”.

► **Lemma 13.** *For any given σ , D and α , it holds that $|\sigma^{\text{exc}}| \leq \frac{\beta|R|}{\alpha D}$, where R is the set of vertices whose colors are modified when σ^{sim} is given as input to \mathcal{A} .*

Proof. In order to prove the lemma we maintain, as the input sequence σ^{sim} proceeds, sets of vertices, B_1, B_2, \dots, B_s , which are pairwise disjoint; furthermore, all vertices in each B_i belong to the same connected component of the input graph (with respect to all edges, not only those in σ^{sim}). We call the B_i sets *witness sets*. We construct the witness sets online as follows. Initially, there are no witness sets. When an edge $e = (u, v) \in \sigma$ arrives, we modify the witness sets as follows.

- I If none of $\{u, v\}$ is already in one of the existing witness sets, and if the connected component that contains e (after e is added to the graph) has at most αD vertices from R : do nothing.
- II If none of $\{u, v\}$ is already in one of the existing witness sets, and if the connected component that contains e (after e is added to the graph) has more than αD vertices from R : create a witness set which contains all the vertices in that connected component.
- III If one of $\{u, v\}$, w.l.o.g. u , is already in an existing witness set, say B , and v is not: add all the vertices of the connected component of v to B . (As we prove below in Point 4, these vertices do not yet belong to any witness set.)
- IV If both $\{u, v\}$ are already in some witness sets (possibly the same one): do nothing.

Intuitively, witness sets are created when a connected component passes the “critical mass” threshold of αD vertices from R , and vertices that do not belong to any witness set are added to the witness set they encounter, i.e., the set of the first vertex that belongs to a witness set, to which they are connected.

The following properties follow from a straightforward induction on the input sequence.

70:10 Colorful Vertex Recoloring of Bipartite Graphs

1. All the vertices of a given witness set are in the same connected component of σ . This is because a new witness set is created from connected vertices. Moreover, additions to a witness set are always in the form of vertices connected to vertices already in that witness set.
2. All the vertices in a connected component of σ^{sim} with more than αD vertices of R are in some witness set (not necessarily all in the same witness set). This follows by induction on the arrival of new edges of σ^{sim} and the actions relative to witness sets taken when this happens.
3. Every two vertices u, v which are in the same connected component of σ are either both in some witness set (not necessarily the same), or both are not in any witness set. This follows from the fact that the actions taken regarding witness sets exclude the possibility of an edge with exactly one endpoint in a witness set.
4. If a vertex w belongs to some witness set B at some point of the execution, then w belongs to B in the remainder of the execution. This follows from the fact that a vertex w becomes a member of witness set B' only in Cases II or III above.
 In Case II, w did not belong before to any witness set because both u and v did not belong to any witness set and w is in the same connected component of either u or v . Hence, by Point 3, w is not in any witness set.
 In Case III, similarly, w is connected to v and since v does not belong to any witness set, then by Point 3 w is not in any witness set.

It follows from the above properties that the total number of witness sets is at most $\frac{|R|}{\alpha D}$, because each witness set that is created requires at least αD distinct vertices of R , vertices do not change their witness set, and the sets are pairwise disjoint.

Next, we claim that when an edge e is added to σ^{exc} (i.e., it connects two distinct large-and-heavy connected components C, C' of σ^{sim}), then the vertices of C and C' already belong to two distinct witness sets. By definition, each of C and C' contains more than αD vertices of R , and hence, by Point (2) above, the endpoints of e are already associated with witness sets. To show that the endpoints of e belong to distinct witness sets, assume, by way of contradiction, that the endpoints of e belong to the same witness set. Then from Point 1 we have that C and C' are already connected in σ (but, by assumption, they are not connected in σ^{sim}). Let e_0 be the first edge (according to the order of the arrival of the edges) in σ^{exc} that completes a path from some node in C to some node in C' . Since $e_0 \notin \sigma^{\text{sim}}$, it must be the case that when e_0 arrived, both of its endpoints belonged to two large and heavy connected components of σ^{sim} . Point 2 ensures then that the vertices of C and C' are already in witness sets. Point 1 implies that these witness sets must be distinct, because before the arrival of e_0 , C and C' are not connected to each other in σ . The sets to which the vertices of C and C' belong are the same, and hence distinct, when e arrives, by Point 4. This is in contradiction to the assumption that the witness sets are the same.

Thus, an edge is added to σ^{exc} only when its endpoints are in two *distinct* witness sets. Since there can be at most $\frac{|R|}{\alpha D}$ such sets, we get from Corollary 11 and the assumption that the graph has bond size at most β , that there are at most $\frac{\beta|R|}{\alpha D}$ such edges. ◀

► **Lemma 14.** *The total cost incurred in procedure `recx` is $O(\beta^2|R|/\alpha)$, where R is the set of vertices ever recolored by \mathcal{A} .*

Proof. Procedure `recx` only recolors vertices that are marked special. For each special vertex v , the procedure `recx` recolors v from a basic color to a special color exactly once in the step when v becomes special, and might recolor v again when a new edge (v, u) arrives, and also u is marked special. Let V' be the set of vertices ever marked special, and E' the set of edges such that when they arrive both their endpoints are marked special.

A node v is marked special only when an edge $e_i = (v, u)$ arrives and $e_i \in E \setminus E_{\text{sim}}$. By Lemma 13 $|E \setminus E_{\text{sim}}| \leq O(\frac{\beta|R|}{\alpha D})$, and hence $|V'| \leq O(\frac{\beta|R|}{\alpha D})$.

Let $|V'| = \ell$, and $V' = \{u_j : 1 \leq j \leq \ell\}$. By applying Corollary 11 with sets $C_1 = \{u_1\}, \dots, C_\ell = \{u_\ell\}$, we have that $|E'| \leq O(\frac{\beta^2|R|}{\alpha D})$.

The cost of **recx** is therefore $O(|V'| + |E'|) \cdot D = O(\frac{\beta|R|}{\alpha D} + \frac{\beta^2|R|}{\alpha D}) \cdot D = O(\frac{\beta^2|R|}{\alpha})$. ◀

Next, we bound from above the cost incurred by the simulation of \mathcal{A} . We need a refined analysis of Algorithm \mathcal{A} . In the following two lemmas, we recall properties of \mathcal{A} from [1]. The first lemma relates the number of vertices recolored by \mathcal{A} to the optimal cost.

► **Lemma 15** ([1], Lemma 3.4). *Let $\text{OPT}_2[i]$ denote the offline cost for input $\sigma[i]$ when using two colors of unit cost. Then $|R_i| \leq 3 \cdot \text{OPT}_2[i]$.*

The next lemma is used to upper-bound the number of times a vertex is recolored by \mathcal{A} .

► **Lemma 16** ([1], Proposition 3.5). *Let $\mathbf{r}(i)$ denote the number of vertices recolored by \mathcal{A} in step i . There exists a subset of steps $I^+ \subseteq \{1, \dots, |\sigma|\}$ such that $\sum_{i \in I^+} \mathbf{r}(i) \geq \frac{1}{7} \sum_{i=1}^{|\sigma|} \mathbf{r}(i)$. Moreover, for every $i \in I^+$, if the arriving edge connects two distinct connected components C and C' , and \mathcal{A} recolors C , then $|C' \cup C| \geq \frac{5}{4}|C|$.*

In [1], Lemma 15 and Lemma 16 are used to show an $O(\log n)$ bound on the competitive ratio of \mathcal{A} . Here, we prove a tighter bound on the competitive ratio when the input sequence is moderate (cf. Definition 7). The proof uses amortized analysis by designing an appropriate charging scheme.

► **Lemma 17**. *If σ is a (D, α) -moderate input sequence, then $\text{cost}_{\mathcal{A}}(\sigma) \leq O(|R| \log D / (1 - \alpha))$.*

Proof. Let R_i be the set of vertices recolored by \mathcal{A} so far after it processed the i th input edge. Let I^+ be the subset of steps given by Lemma 16; it suffices to bound from above the cost incurred by \mathcal{A} on steps in I^+ . We will do this via a charging scheme.

The high-level intuition behind the charging scheme is as follows. Let $i \in I^+$, and suppose e_i connects components C_i and C'_i , and that in response, \mathcal{A} recolors C_i (recall that we consider bipartite graphs and two colors, hence a monochromatic edge must connect two distinct connected components). Our goal is to charge a the load of 1 on certain vertices and show that (1) the total charge in each step i is at least a constant fraction of $|C_i \cap R|$ or $|C'_i \cap R|$ and (2) every vertex is charged $O(\log D)$ times in total over the course of the input sequence. There are *four* cases to consider and each case will use a different type of charging. (1) The first is when C_i is small, i.e. $|C_i| \leq D$. In this case, we charge every vertex in C_i . Lemma 16 implies that a vertex can only be charged by this charging type a total of $O(\log D)$ times. (2) The second case is when C_i is large and light. In this case, we charge to the vertices of C_i that are newly recolored by the algorithm. Since C_i is light, the number of vertices that are recolored for the first time by \mathcal{A} in this step is sufficiently large. Since this type of charging charges newly recolored vertices, a vertex can be charged by this charging type at most once.

The remaining possibilities concern case (3) when C_i is large and heavy. We consider two subcases. (3.1) The first subcase is when C_i is large and heavy and C'_i is small and heavy. In this case, we charge to $C'_i \cap R$. Lemma 16 implies that $|R \cap C'_i|$ is at least an α fraction of $|C'_i|$. Moreover, since the vertices in C'_i are part of a large component after this step, a vertex can be charged by this charging type at most once. The final and most interesting case is (3.2) when C'_i is light. To handle this case, we maintain, for each (maximal) connected component C that exists at a given time, a subset $X(C)$ of vertices. We will charge to the vertices of

70:12 Colorful Vertex Recoloring of Bipartite Graphs

$X(C_i)$ in this case. The sets X are defined inductively as described in the pseudocode of the charging scheme below. As we prove below, each vertex is charged by this type of charging only once, we the size of the sets X is large enough to compensate for the cost of recoloring.

Note that these cases are exhaustive as C_i and C'_i cannot be both large and heavy, otherwise the edge would not have been input to the simulation \mathcal{A} . The following summarizes in a formal manner the charging scheme described above. We then formally prove the desired properties of the charging scheme as stated informally above.

Charging scheme

- Initially, $X(\{v\}) = \{v\}$ for every vertex v .
- When a new edge $e_i = (u_i, v_i)$ arrives connecting components C_i and C'_i :
 1. Suppose \mathcal{A} recolors C_i
 - a. Charge vertices as follows:
 - i. If C_i is small, then charge 1 to each vertex in C_i
 - ii. If C_i is large and light, then charge 1 to each vertex in $C_i \setminus R_{i-1}$, i.e., the vertices of C_i that were never recolored before step i .
 - iii. If C_i is large and heavy, C'_i small and heavy, then charge 1 to each vertex in $R_{i-1}(C'_i)$
 - iv. If C_i is large and heavy, C'_i light, then charge 1 to each vertex in $X(C_i)$.
 - b. If C'_i is light, then $X(C_i \cup C'_i) = (C_i \cup C'_i) \setminus R_i$; else $X(C_i \cup C'_i) = X(C_i) \cup X(C'_i)$
 2. If \mathcal{A} does not recolor, then $X(C_i \cup C'_i) = X(C_i) \cup X(C'_i)$

Observe that every vertex charged is either in C_i , the component being recolored, or was previously recolored (case 1(a)iii). Thus every vertex that is charged is in R . It now remains to show that in each step, the number of vertices being charged is at least a constant fraction of the number of vertices being recolored – i.e. the total charge received by the vertices can pay for the recoloring cost – and that each vertex is not charged more than $O(\log D)$ times.

▷ **Claim 18.** Consider a connected component C that exists after an arbitrary step i , and the set $X(C)$ defined by the charging scheme. For every $C \in \mathcal{P}$, we have $|X(C)| \geq |C|(1 - \alpha)/5$.

Proof. We prove this claim by induction on $|C|$, the base case being $|C| = 1$. By definition for C , such that $|C| = 1$, $X(C) = C$ and hence the claim hold, since $0 < \alpha < 1$. For the inductive step, suppose that $C = A \cup A'$, and that $|X(A)| \geq |A|(1 - \alpha)/5$ and $|X(A')| \geq |A'|(1 - \alpha)/5$. If $X(C) = X(A) \cup X(A')$ (which occurs in two distinct cases in the specifications of the charging scheme) then, since $A \cap A' = \emptyset$,

$$|X(C)| = |X(A)| + |X(A')| \geq (|A| + |A'|)(1 - \alpha)/5 = |C|(1 - \alpha)/5.$$

Otherwise, $X(C) = (A \cup A') \setminus R_i$. Then, when e_i arrived, \mathcal{A} recolored one of the two connected components A, A' and the other connected component was light. Suppose w.l.o.g. that \mathcal{A} recolored A , and A' was light. Then, we have

$$C \setminus R_i \subseteq A' \setminus R_i = A' \setminus R_{i-1}.$$

By the lightness of A' , we have that $|A' \setminus R_{i-1}| \geq |A'|(1 - \alpha) \geq |C|(1 - \alpha)/5$ by Lemma 16. This concludes the induction argument. ◁

▷ **Claim 19.** Suppose \mathcal{A} recolors component C_i at step i . Then the number of vertices being charged is at least $\Omega(|C_i|(1 - \alpha))$.

Proof. We show that the number of vertices being charged is at least $|C|(1 - \alpha)/20$ in every case. In case 1(a)i, we charge C so this clearly holds. In case 1(a)ii, lightness of C implies that $|C \setminus R_{i-1}| \geq |C|(1 - \alpha)/5 \geq |C|(1 - \alpha)/20$. In case 1(a)iii, we have

$|R_{i-1}(C')| \geq |C'|(1 - \alpha)/5 \geq |C'|(1 - \alpha)/20$ where the first inequality is due to C' being heavy and the second due to Lemma 16. For case 1(a)iv, we get that $|X(C)| \geq |C|(1 - \alpha)/5$ from Claim 18. \triangleleft

\triangleright **Claim 20.** Every vertex is charged at most $O(\log D)$ times.

Proof. By Lemma 16, the number of times a vertex v is charged due to case 1(a)i is $O(\log D)$. The number of times a vertex v is charged due to line 1(a)ii is at most once since it is recolored and now belongs to R_i . The number of times it can be charged due to 1(a)iii is at most once since it now belongs to a large component. The number of times it can be charged due to 1(a)iv is at most once since afterwards it belongs to R_i and thus cannot be in $X(A)$ for any future component, due to the way the set X is modified in 1b. \triangleleft

Combining the fact that only vertices in R can be charged and Claims 19 and 20, we are done proving Lemma 17. \blacktriangleleft

We are now ready to conclude the analysis of Algorithm \mathcal{B} .

Proof of Theorem 2. By Lemma 14, the total cost incurred by recoloring using a special color is $O(|R| \cdot \frac{\beta^2}{\alpha})$. Next, note that by the code the sequence given as input to \mathcal{A} , σ^{sim} , is (D, α) -moderate. It therefore follows from Lemma 17 that the total cost due to the simulation of \mathcal{A} is $O(|R| \log D / (1 - \alpha))$. Since by Lemma 15 the optimal cost is $\Omega(|R|)$, we are done by picking, say, $\alpha = 1/2$. \blacktriangleleft

4 Uniform cost

In this section, we still consider vertex recoloring of bipartite graphs, but now in the setting where that all colors, basic and special, have the same cost. We thus present an algorithm which has better competitive ratio if more than two colors are available, covering the spectrum between an $O(\log n)$ competitive ratio with no special colors, and a $O(\log \beta)$ competitive ratio with $O(\log_\beta n)$ special colors, where β is an upper bound on the size of graph bonds. Specifically, we prove Theorem 4, reproduced below.

\blacktriangleright **Theorem 4.** *Suppose that the final graph is bipartite with bond at most β . Then for any given $\frac{1}{2 \log \beta + 3} < \epsilon \leq 1$ there is a deterministic online algorithm that uses $O(\epsilon \log n)$ special colors of unit cost, and guarantees competitive ratio $O(\epsilon^{-1})$.*

The idea is to use $O(\epsilon \log n)$ instances of Algorithm \mathcal{A} (Alg. 1 of [1]) for recoloring bipartite graphs.

High-Level Description. We push the ideas of simulation and promoting vertices to be “special”, used in algorithm \mathcal{B} , even further. Roughly speaking, the main idea of the algorithm is to create multiple (possibly overlapping) subinstances of the problem, such that the input sequence of each subinstance is moderate, as in Definition 7. Each subsequence is fed into a different instance of \mathcal{A} , which uses a distinct pair of colors. At every point in time, each vertex is associated with some instance j , which determines its actual color.

More specifically, denote the j th instance of \mathcal{A} by \mathcal{A}_j , its input sequence by σ^j , and the coloring that it maintains at any time by \mathcal{c}^j . Each vertex v is associated at any given time with one of the instances, denoted $\text{level}(v)$. Initially, $\text{level}(v) = 1$ for all $v \in V$. We define $V^j \stackrel{\text{def}}{=} \{v \in V \mid \text{level}(v) = j\}$. \mathcal{A}_j determines the coloring of V^j : $\mathcal{c}^j : V^j \rightarrow \{2j - 1, 2j\}$. Our

70:14 Colorful Vertex Recoloring of Bipartite Graphs

algorithm maintains the following invariants: (1) each subsequence σ^j is moderate; (2) for each level j , the coloring c^j is a valid coloring of V^j with respect to $G[V^j]$ where G is the graph consisting of every edge in the entire input sequence, not just those of σ^j .

Suppose that when an edge $e_i = (u_i, v_i)$ arrives, u_i and v_i have the same color. This can only happen if they have the same level j . We first check if appending e_i to σ^j is still moderate. If it is, then we append and send e_i to \mathcal{A}_j , and recolor vertices of level j accordingly. Otherwise, we choose one of them, arbitrarily, to promote to level $j + 1$.

When we promote a vertex u to level k , in order to maintain invariant (2), we attempt to append, one-by-one, each edge (u, v) in which $\text{level}(v) = k$ to σ^k , in arbitrary order. In this process, if for some edge (u, v) , we are unable to append it to σ^k without maintaining the moderate property, we promote u to the next level. Note a vertex cannot be promoted indefinitely as it eventually ends up at a level with no other vertices. Pseudocode of our algorithm is given below.

Algorithm C: Coloring a bipartite graph of largest bond size β using $O(\epsilon \log n)$ colors.

- a. Set $\text{level}(v) := 1$ for all $v \in V$.
- b. Set c^1 to be the initial vertex coloring.
- c. Fix an arbitrary initial coloring c^j for $j > 1$, say $c^j(v) = 2j$ for all $v \in V$.
- d. Let $\tau = \max\{2^{1/\epsilon}, 4\beta^2/\alpha\}$. *// α is a constant parameter in $(0, 1)$*
- e. When edge $e_i = (u_i, v_i)$ arrives:
 - (1) If $\text{level}(v_i) \neq \text{level}(u_i)$ then return; *// different colors*
 - (2) Let $j = \text{level}(v_i)$ *// $j = \text{level}(u_i)$ as well*
 - (3) If $\sigma^j \circ (e_i)$ is (τ, α) -moderate, then send e_i to \mathcal{A}_j *// e_i is appended to σ^j*
 - (4) Else invoke **promote** $(u_i, j + 1)$ *// e_i is j -excess; u_i is chosen arbitrarily*
- f. Output coloring $c(v) = c^{\text{level}(v)}(v)$ for all $v \in V$.

Procedure promote (u, k) :

1. Let $V' := V^k \cup \{u\}$ *// u gets the initial color of c^k*
2. Let σ' be an arbitrary ordering of the edges $e = (u, v)$ arrived so far, such that $\text{level}(v) = k$.
3. For each edge e' in σ' :
 - a. If $\sigma^k \circ (e')$ is (τ, α) -moderate, send e' to \mathcal{A}_k *// e' is appended to σ^k*
 - b. Else invoke **promote** $(u, k + 1)$ and return *// e' is k -excess*
4. $\text{level}(u) := k$ *// promotion to level j was successful*

4.1 Analysis

▷ **Claim 21.** Let $e_i = (u, v)$. At any time $t \geq i$, if $\text{level}(u) = \text{level}(v) = k$, then $e_i \in \sigma^k$.

Proof. At any time $t \geq i$, if at the beginning of time step t , $\text{level}(u) \neq \text{level}(v)$, we are done as the levels do not change (step 5(a)). If at the beginning of time step t $\text{level}(u) = \text{level}(v) = k$ and $\sigma^k \circ e_i$ is moderate then e_i is added σ^k (step 5(c)), and the claim holds. In the remaining case (Step 5(d)), u is promoted and v is not, hence their level will not be the same at the end of time step t . ◀

The correctness of the algorithm is now straightforward.

► **Lemma 22.** *The coloring produced by Algorithm C is correct.*

Proof. Follows from the fact that at any time (1) two vertices of different levels are not colored by the same color, (2) if an edge $e = (u, v)$ exists and $\text{level}(u) = \text{level}(v) = k$, then $e \in \sigma^k$ (3) the correctness of \mathcal{A}_k for all k . ◀

We now proceed to give an upper bound on the cost paid by Algorithm \mathcal{C} and on the number of colors it uses. Recoloring occurs in \mathcal{C} either by some \mathcal{A}_j or by procedure `promote` (which recolors to the initial color of the corresponding level). We analyze each separately.

Let us denote by $\text{OPT}_2(\sigma)$ the optimal cost of recoloring an input sequence σ , when only the two initial colors of unit cost are available; the initial coloring of the vertices is understood from the context. Regarding recoloring by some \mathcal{A}_j , we have the following. Let R_j denote the set of vertices whose color was altered by \mathcal{A}_j .

► **Lemma 23.** *For all j , $\text{cost}_{\mathcal{A}}^j(\sigma^j) = O(|R_j| \log \tau / (1 - \alpha))$.*

Proof. Follows from Lemma 17 and the fact that σ^j is (τ, α) -moderate by construction. ◀

We note that the recoloring done by the algorithm are either (1) changes of $c^k(v)$, for v such that $\text{level}(v) = k$, done by \mathcal{A}_k , or (2) change of the color of v due to the change of $\text{level}(v)$, in procedure `promote`. We now analyze the cost due to the latter; the former was analyzed in the above lemma.

We use the following definition.

► **Definition 24.** *An input edge $e = (u, v)$ is called j -excess if it caused `promote`($u, j + 1$) to be invoked, i.e., either of the two lines (1) Step e4 or (2) Step 3b of Procedure `promote`, was reached when e was processed. We denote the set of all j -excess edges by F_j .*

Clearly, the total cost due to recoloring by `promote` is at most $\sum_j |F_j|$, because each excess edge causes at most one vertex to be recolored (indirectly) when `promote` changes the level of a vertex. Then by Lemma 13 we have:

► **Lemma 25.** $|F_j| \leq \beta \frac{|R_j|}{\alpha\tau}$ for every $j \geq 1$.

We now have that the total cost of the algorithm is

$$\text{cost}_{\mathcal{C}}(\sigma) \leq \sum_{j \geq 2} |F_j| + \sum_{j \geq 1} \text{cost}_{\mathcal{A}}(\sigma^j) \leq O\left(\frac{\beta}{\alpha\tau} + \frac{\log \tau}{1 - \alpha}\right) \sum_{j \geq 1} |R_j|. \quad (1)$$

Next, we bound $\sum_{j \geq 1} |R_j|$. We will need the following lemma.

► **Lemma 26.** *Let E^j denote the set of edges in σ^j . Then for every $j > 1$, $|E^j| \leq \beta^2 \cdot \frac{|R_{j-1}|}{\alpha\tau}$.*

Proof. Let S_j be the set of vertices u to which `promote`(u, j) was applied at some point in the algorithm. Clearly, $|S_j| = |F_{j-1}|$, because each $(j - 1)$ -excess edge promotes a single vertex to level j . It follows that $|S_j| = |F_{j-1}| \leq \beta \cdot \frac{|R_{j-1}|}{\alpha\tau}$ by Lemma 25. Now, $E^j \subseteq G[S_j]$, because an edge is appended to σ^j only if both its endpoints are in V^j at that time. Since E^j is a subset of the edges of a graph with largest bond size β , we have from Corollary 11 that $|E^j| \leq \beta(|S_j| - 1)$. We therefore conclude that $|E^j| \leq \beta^2 \cdot \frac{|R_{j-1}|}{\alpha\tau}$. ◀

We can now prove that $|R_j|$ decreases exponentially with j , for large enough τ .

► **Proposition 27.** $|R_j| \leq 2\beta^2 \cdot \frac{|R_{j-1}|}{\alpha\tau}$ for every $j > 1$.

Proof. Since \mathcal{A}_j recolors a vertex only if it is incident on an input edge, we have that $|R_j| \leq 2|E^j|$. The result follows from Lemma 26. ◀

Proposition 27 immediately gives an upper bound on the number of colors used by Algorithm \mathcal{C} .

70:16 Colorful Vertex Recoloring of Bipartite Graphs

► **Corollary 28.** *With $\alpha = \frac{1}{2}$ and $1/\epsilon \geq 2 \log \beta + 3$, Algorithm \mathcal{C} uses at most $O(\epsilon^{-1} \log n)$ colors.*

Proof. Let $\gamma = \frac{2\beta^2}{\alpha\tau}$. By Step d of Algorithm \mathcal{C} , $\gamma < 1$. Clearly, $|R_1| \leq n$, and hence, by Proposition 27 we have that $|R_j| \leq n \cdot \gamma^j$. Hence, for $j > \log_{(1/\gamma)} n$ we have that $|R_j| < 1$. The result follows, since $\log(1/\gamma) = \log \tau - O(\log \beta) = \Theta(1/\epsilon)$. ◀

We can now conclude the analysis of Algorithm \mathcal{C} .

Proof of Theorem 4. Fix $\alpha = 1/2$. The number of colors is given by Corollary 28. By Inequality (1), the total cost of the algorithm is at most

$$O\left(\frac{\beta}{\alpha\tau} + \frac{\log \tau}{1-\alpha}\right) \sum_{j \geq 1} |R_j|.$$

Since $\epsilon > \frac{1}{2 \log \beta + 3}$ and $\alpha = 1/2$, we get that $\tau = 2^{1/\epsilon}$ and so $\frac{\beta}{\alpha\tau} + \frac{\log \tau}{1-\alpha} \leq O(1/\epsilon)$. Moreover, Proposition 27 implies that $\sum_{j \geq 1} |R_j| \leq O(1)|R_1|$. By Lemma 15, we have that $|R_1| \leq O(1)\text{OPT}_2(\sigma^1) \leq O(1)\text{OPT}_2(\sigma)$. Thus, $\sum_{j \geq 1} |R_j| \leq O(1)\text{OPT}_2(\sigma)$. Therefore, the total cost of the algorithm is at most $O(1/\epsilon)\text{OPT}_2(\sigma)$, as desired. ◀

5 Lower Bounds

In this section we prove two lower bounds on online recoloring. The first, Theorem 3, says that no matter how many extra colors we use, if they cost D , then the best competitive ratio one can hope for is $O(\log D)$ even if the underlying graph is acyclic. The second lower bound, Theorem 1, shows that if there is no upper bound on the bond size of the graph, the competitive ratio of any online recoloring algorithm is at least $\Omega(\min(\log n, D))$.

5.1 Acyclic Graphs

In this subsection we show that the competitive ratio of Algorithm \mathcal{B} cannot be improved, even by randomized algorithms, as stated by Theorem 3 reproduced below.

► **Theorem 3.** *Consider instances in which the final graph is a collection of paths, and there is an infinite set of special colors, each of cost $D > 1$. The competitive ratio of any (possibly randomized) recoloring algorithm in this class of instances is $\Omega(\log D)$.*

Proof. By Yao's Principle, it suffices to show the existence of an input distribution on which the expected cost of any deterministic algorithm is $\Omega(\log D)$ times the optimal cost. We describe such a distribution. Initially, there are n vertices, each colored by one of the two basic (unit cost) colors. Without loss of generality, let us assume that n is a multiple of $\geq D/\log D$ vertices. For ease of explanation, we shall assume that the vertices are arranged left-to-right on a line. The input sequence is constructed in H phases, where $H \stackrel{\text{def}}{=} \lceil \log D - \log \log D \rceil$. In phase $1 \leq h \leq H$, edges arrive so that at the end of the phase, each connected component is a path with 2^h vertices. Specifically, for each $1 \leq j \leq n/2^h$, an edge connecting random endpoints of the $(2j-1)$ -th leftmost path and the $2j$ -th leftmost path is inserted. Note that after the last phase, we have a set of paths of length 2^H vertices each. Note that for every phase h , the set of connected components at the end of the phase is predetermined; it is only the order in which the vertices in a connected component appear in the corresponding path that is randomized.

Fix an arbitrary deterministic recoloring algorithm A . We now bound from below the expected cost of A on the input distribution defined above. We first prove the following lemma.

► **Lemma 29.** *Let A^* be an arbitrary deterministic algorithm that uses only the two basic colors. Let k be any integer power of 2 not greater than 2^H . Consider a randomized input sequence as described above and a connected component consisting of k vertices after $\log k$ phases. Let σ be the restriction of the input sequence to the connected component. Then we have $\Pr[\text{cost}_{A^*}(\sigma) \geq k(\log k - 2)/8] \geq 1/2$.*

Proof. In phase $h > 1$, there are $k/2^h$ merges of paths each containing 2^{h-1} vertices. Observe that with probability exactly $1/2$ a given merge is monochromatic (i.e., the merging new edge connects two vertices of the same color) independently of all other merges (in the same and other phases). This is because for a given merge, each of the two merged paths have one end colored with each of the two basic colors. Further observe that a monochromatic merge at phase $h > 0$ results in recoloring of exactly 2^{h-1} vertices. Let Z_h be the indicator random variable indicating whether at least half of the merges in phase h are monochromatic. Clearly, if $Z_h = 1$ then the algorithm recolors at least $\frac{1}{2} \frac{k}{2^h} \cdot 2^{h-1} = k/4$ vertices in phase h . Moreover, for any $1 < h < \log k$, even when conditioned on all Z_j (except Z_h), it holds that $\Pr[Z_h = 1] = 1/2$.

It therefore follows that $\text{cost}_{A'} \geq \sum_{h=2}^{\log k} Z_h \cdot k/4$, and

$$\Pr \left[\text{cost}_{A^*} \geq \frac{k(\log k - 2)}{8} \right] \geq \Pr \left[\sum_{h=2}^{\log k} Z_h \cdot \frac{k}{4} \geq \frac{k(\log k - 2)}{8} \right] \geq \frac{1}{2},$$

where the last inequality follows from the fact that $\sum_{h=2}^{\log k-1} Z_h$ is a sum of $(\log k - 2)$ independent Bernoulli random variables with parameter $1/2$, and so their sum is at least $(\log k - 2)/2$ with probability $1/2$. ◀

We now continue with the proof of Theorem 3. Consider one of the paths of length 2^H that exist after the last phase. We give a lower bound on the expected cost of A for re-coloring the vertices of that path. Let A_D be the indicator random variable indicating if A uses at least once an extra color (of cost D) for recoloring the vertices of the given path. There are two cases to consider.

- (1) If $\Pr[A_D] \geq 1/4$, then $\Pr[\text{cost}_A \geq D] \geq 1/4$, and hence $E[\text{cost}_A] \geq D/4$.
- (2) Otherwise, $\Pr[A_D] < 1/4$. In this case, define a deterministic online recoloring algorithm A' that uses only the basic colors as follows. A' mimics A so long as A uses basic colors. Once A uses a special color (if it does), A' stops mimicking A and continues to recolor (properly), using only the basic colors.

We make the following observations.

- $\Pr[\text{cost}_{A'} \geq 2^H(H - 2)/8] \geq 1/2$ by Lemma 29.
- The probability that A' and A diverge in their colorings on the path under consideration (and hence *do not have* the same cost for that path) is less than $1/4$.

It follows that with probability more than $1/4$, $\text{cost}_A = \text{cost}_{A'}$, and that cost is at least

$$\begin{aligned} \text{cost}_A &\geq 2^H \cdot \frac{H - 2}{8} \\ &= (2^{\lceil \log D - \log \log D \rceil}) \cdot \frac{\lceil \log D - \log \log D \rceil - 2}{8} \\ &\geq \frac{D}{\log D} \cdot \frac{\log D - \log \log D - 2}{8} \\ &= \Omega(D). \end{aligned}$$

Therefore, in both cases the expected cost of A for recolorings of the vertices in the path that we consider is $\Omega(D)$.

The bound above is the cost for a single component of length 2^H . By linearity of expectation, the expected cost of A for all components is $\frac{n}{2^H} \cdot \Omega(D)$. On the other hand the cost of the optimal algorithms is $O(n)$ for any input sequence. Hence we have a lower bound on the competitive ratio of $\Omega\left(\frac{(n/2^H) \cdot D}{n}\right) = \Omega\left(\frac{D}{2^H}\right) = \Omega\left(\frac{D}{2^{\log D - \log \log D}}\right) = \Omega(\log D)$. ◀

5.2 Bipartite Graphs with Unbounded Bond Size

In this subsection we show that no deterministic online recoloring algorithm can have good competitive ratio for all bipartite graphs with large bond, even when special colors are available. In fact, using special colors cannot improve the competitive ratio beyond the easy $O(D)$ bound of Lemma 5 if the bond may be large.

► **Theorem 1.** *Let \mathcal{I} be the set of instances of recoloring with two basic colors and n special colors, where recoloring by a basic color costs 1 and recoloring by a special color costs D . Then for every deterministic online recoloring algorithm there is an instance in \mathcal{I} with competitive ratio $\Omega(\min\{\log n, D\})$.*

Proof. We construct an adversarial input sequence that consists of phases. At any point in time, a vertex is said to be *special* if the algorithm has recolored it at some earlier point to a special color; the vertex is called *basic* otherwise. Furthermore, at each phase we characterize the connected components as *active* or *inactive*. A component is said to be active if at least half of its vertices are basic, and it is said to be inactive otherwise. Let c_1, c_2 denote the two basic colors. Initially, each vertex is given either the color c_1 or c_2 . An active component is said to be *c-dominated*, for $c \in \{c_1, c_2\}$ if at least a quarter of its vertices are colored c (a component may be both c_1 -dominated and c_2 -dominated).

The construction maintains the following invariant:

■ At the start of phase $h > 0$, every active component has 2^{h-1} vertices.

In phase h , we match active components of the same dominating color in pairs. For each such pair (C, C') of connected components of the same dominating color, we add a perfect matching between their vertex sets such that at least a quarter of the matching edges (i.e., at least 2^{h-3} edges) are monochromatic. The process ends once there is at most one active component of dominating color c_1 and at most one active component of dominating color c_2 .

We claim that the cost paid by the algorithm for the above input sequence is at least $\Omega(\min\{n \log n, nD\})$. To prove the claim, let k be the number of special vertices at the end of the input sequence, and let M be the number of edges that were monochromatic when they arrived. Clearly, the cost for the algorithm is at least $\max\{kD, M\} \geq (kD + M)/2$. Now, if $k \geq n/4$, then the algorithm's cost is $\Omega(nD)$ and the claim holds. Otherwise, $k < n/4$, and hence, by the definition of active components, at the end of the execution, there are at most $2k < n/2$ vertices in inactive components. It follows that in each phase there are at least $n/2$ vertices in active components, and therefore at least $n/4$ vertices in active components of the same dominating color, which implies that in each phase, the number of introduced monochromatic edges is $\Omega(n)$. Further, note that if $k < n/4$ then the number of phases is at least $\log n - 3$: before every phase $h \leq \log n - 2$, each connected component contains less than $2^{\log n - 3} = n/8$ vertices, and hence there are at least 4 active connected components, so there are at least two connected components dominated by the same color. It follows that the total cost of the algorithm is $\Omega(n \log n)$.

In summary, we have shown that the cost of the online algorithm for the input sequence specified above is $\Omega(\min\{n \log n, nD\})$. As the optimal cost is $O(n)$ the theorem follows. ◀

We note that the largest bond size of any active connected component C in the construction above is $\Omega(|C|)$.

6 Conclusions

In this paper we studied competitive vertex recoloring with weighted augmentation. We have shown that the competitive ratio of recoloring bipartite graphs may be reduced if the online algorithm can use additional colors, even if they are more costly than the basic two colors. Beyond the direct technical contribution, we believe that we introduced some conceptual contributions:

- The approach of weighted resource augmentation is new, as far as we know.
- The concept of the largest graph bond as a way to generalize algorithmic results quantitatively.

It seems that these ideas can be useful in dealing with other problems of competitive analysis.

References

- 1 Yossi Azar, Chay Machluf, Boaz Patt-Shamir, and Noam Touitou. Competitive vertex recoloring. *Algorithmica*, 85(7):2001–2027, 2023. doi:10.1007/s00453-022-01076-x.
- 2 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.
- 3 Luis Barba, Jean Cardinal, Matias Korman, Stefan Langerman, André van Renssen, Marcel Roeloffzen, and Sander Verdonschot. Dynamic graph coloring. *Algorithmica*, 81(4):1319–1341, 2019. doi:10.1007/s00453-018-0473-y.
- 4 J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 1st edition, 2008.
- 5 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- 6 Bartłomiej Bosek, Yann Disser, Andreas Emil Feldmann, Jakub Pawlewicz, and Anna Zych-Pawlewicz. Recoloring Interval Graphs with Limited Recourse Budget. In *Proc. 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, volume 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:23, Dagstuhl, Germany, 2020. doi:10.4230/LIPIcs.SWAT.2020.17.
- 7 Gabriel L. Duarte, Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Daniel Lokshantov, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, and Uéverton S. Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83:1421–1458, January 2021. doi:10.1007/S00453-020-00789-1.
- 8 F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3):221–225, 1975. doi:10.1137/0204019.
- 9 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Explicit and implicit dynamic coloring of graphs with bounded arboricity. *CoRR*, abs/2002.10142, 2020. arXiv:2002.10142.
- 10 Manas Jyoti Kashyop, N. S. Narayanaswamy, Meghana Nasre, and Sai Mohith Potluri. Trade-offs in dynamic coloring for bipartite and general graphs. *Algorithmica*, 85(4):854–878, 2023. doi:10.1007/s00453-022-01050-7.
- 11 Rajmohan Rajaraman and Omer Wasim. Competitive capacitated online recoloring. In Timothy 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 95:1–95:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.95.
- 12 Daniel M. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.
- 13 Shay Solomon and Nicole Wein. Improved dynamic graph coloring. *ACM Trans. Algorithms*, 16(3):1–24, June 2020. doi:10.1145/3392724.

Unfairly Splitting Separable Necklaces

Patrick Schnider ✉ 

Department of Computer Science, ETH Zürich, Switzerland

Linus Stalder ✉

Department of Computer Science, ETH Zürich, Switzerland

Simon Weber ✉ 

Department of Computer Science, ETH Zürich, Switzerland

Abstract

The Necklace Splitting problem is a classical problem in combinatorics that has been intensively studied both from a combinatorial and a computational point of view. It is well-known that the Necklace Splitting problem reduces to the discrete Ham Sandwich problem. This reduction was crucial in the proof of PPA-completeness of the Ham Sandwich problem. Recently, Borzeczowski, Schnider and Weber [ISAAC'23] introduced a variant of Necklace Splitting that similarly reduces to the α -Ham Sandwich problem, which lies in the complexity class UEOPL but is not known to be complete. To make this reduction work, the input necklace is guaranteed to be n -separable. They showed that these necklaces can be fairly split in polynomial time and thus this subproblem cannot be used to prove UEOPL-hardness for α -Ham Sandwich. We consider the more general *unfair* necklace splitting problem on n -separable necklaces, i.e., the problem of splitting these necklaces such that each thief gets a desired fraction of each type of jewels. This more general problem is the natural necklace-splitting-type version of α -Ham Sandwich, and its complexity status is one of the main open questions posed by Borzeczowski, Schnider and Weber. We show that the unfair splitting problem is also polynomial-time solvable, and can thus also not be used to show UEOPL-hardness for α -Ham Sandwich.

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

Keywords and phrases Necklace splitting, n -separability, well-separation, Ham Sandwich, alpha-Ham Sandwich, unfair splitting, fair division

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.71

Related Version *Full Version*: <https://arXiv.org/abs/2408.17126>

Funding *Simon Weber*: Swiss National Science Foundation under project no. 204320.

1 Introduction

One of the most famous theorems in fair division is the *Ham Sandwich theorem* [23]. It states that any d point sets in \mathbb{R}^d can be simultaneously *bisected* by a single hyperplane. The Ham Sandwich theorem is closely related to another fair division theorem that lives in \mathbb{R} ; the *Necklace Splitting theorem*. It states that given n point sets in \mathbb{R} (a *necklace* with n types of *jewels*), we can split the real number line at n points such that when we partition the resulting pieces alternately, each of the two parts contains exactly half of the jewels of each type. In fact, the Necklace Splitting theorem can be proven by lifting the necklace to the *moment curve* in \mathbb{R}^n , which is the curve parameterised by $(t, t^2, t^3, \dots, t^n)$, and then applying the Ham Sandwich theorem.

Under some additional assumptions on the input points, the Ham Sandwich theorem can be significantly strengthened: If the input point sets are *well-separated* and in general position, the α -*Ham Sandwich theorem* [22] says that we can not only simultaneously *bisect* each point set, but we can for each i choose any number $1 \leq \alpha_i \leq |P_i|$, and find a single



© Patrick Schnider, Linus Stalder, and Simon Weber;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 71; pp. 71:1–71:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



hyperplane that cuts off exactly α_i points from each point set P_i . Informally, a family of point sets is well-separated if the union of any subfamily can be separated from the union of the complement subfamily by a single hyperplane. Borzechowski, Schnider and Weber [6] introduced an analogue of this well-separation condition for necklaces: A necklace is n -separable, if any subfamily can be separated from the complement subfamily by splitting the necklace at at most n points. It is shown in [6] that a necklace is n -separable if and only if its lifting to the moment curve is well-separated.

Existence theorems such as the Ham Sandwich and the Necklace Splitting theorem can also be viewed from the lens of computational complexity. The corresponding problems are *total search problems*, i.e., problems in which a solution is always guaranteed to exist, but the task is to actually find a solution. In this setting, the strategy used above to prove the Necklace Splitting theorem using the Ham Sandwich theorem also yields a *reduction* from the Necklace Splitting problem to the Ham Sandwich problem. This reduction was crucial for establishing the PPA-hardness of the Ham Sandwich problem, which is since known to be PPA-complete [12]. The α -Ham Sandwich problem is known to lie in the subclass UEOPL \subseteq PPA [8], but no matching hardness result is known. It is thus natural to ask whether UEOPL-hardness of α -Ham Sandwich could be proven by reduction from a Necklace Splitting problem on n -separable necklaces. Borzechowski, Schnider and Weber [6] showed that the classical (*fair*) Necklace Splitting problem on n -separable necklaces is polynomial-time solvable and thus very unlikely to be UEOPL-hard. However, the natural necklace-splitting-type analogue of α -Ham Sandwich would actually allow for *unfair* splittings on n -separable necklaces, where the first of the two thieves should get exactly α_i jewels of type i , for some input vector α . We settle the complexity status of this problem variant by providing a polynomial-time algorithm. This completely disqualifies Necklace Splitting variants on n -separable necklaces as possible problems to prove UEOPL-hardness of α -Ham Sandwich.

We would like to note that necklace splitting and its extensions to higher dimensions and larger numbers of thieves, as well as other related problems have enjoyed much research interest [1, 2, 4, 5, 7, 10, 11, 13, 14, 17, 20, 21].

1.1 Results

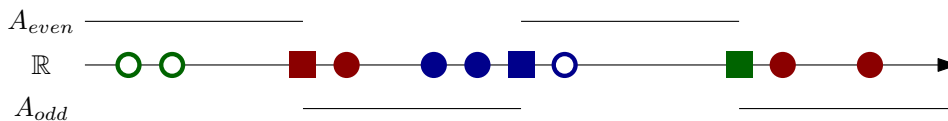
Before we can state our results we have to begin with the most crucial definitions.

► **Definition 1** (Necklace). *A necklace is a family $C = \{C_1, \dots, C_n\}$ of disjoint finite point sets in \mathbb{R} . The sets C_i are called colours, and each point $p \in C_i$ is called a bead of colour i .*

We align the following definition of α -cuts in necklaces as closely as possible with the definition of α -cuts in the α -Ham Sandwich theorem (which we will define later). We require that an α -cut of a necklace splits the necklace at exactly one bead per colour, called the *cut point*. Furthermore, the parity of the permutation of how these cut points appear along \mathbb{R} determines whether the first part of the necklace is given to the first or the second thief. Let us make this definition more formal.

► **Definition 2** (α -Cut). *Let $C = \{C_1, \dots, C_n\}$ be a necklace. A set of n cut points s_1, \dots, s_n such that for all i we have $s_i \in C_i$ defines the subset C^+ of $C_1 \cup \dots \cup C_n$ that is assigned to the first thief as follows. We first add the points $s_0 := -\infty$ and $s_{n+1} := \infty$, and let $\pi : \{0, \dots, n+1\} \rightarrow \{0, \dots, n+1\}$ be the permutation such that $s_{\pi(0)} < \dots < s_{\pi(n+1)}$. We then get the two sets of closed intervals $A_{\text{even}} := \{[s_{\pi(i)}, s_{\pi(i+1)}] \mid 0 \leq i \leq n, i \text{ even}\}$ and $A_{\text{odd}} := \{[s_{\pi(i)}, s_{\pi(i+1)}] \mid 0 \leq i \leq n, i \text{ odd}\}$. Let A^+ be $A_{\text{sgn}(\pi)}$, i.e., the set of intervals corresponding to the parity of π . Then, $\{s_1, \dots, s_n\}$ is called an α -cut of C for the vector $\alpha = (\alpha_1, \dots, \alpha_n)$, where $\alpha_i = |A^+ \cap C_i|$.*

Definition 2 is illustrated in Figure 1.



■ **Figure 1** A 3-separable necklace with three colours C_1 (red), C_2 (green), and C_3 (blue). In each colour, the point illustrated as a square is chosen as the cut point s_i . This defines the intervals A_{even} illustrated above the necklace, and the intervals A_{odd} illustrated below. The cut points are coloured 1 3 2 when ordered increasingly. Since the parity of the permutation 0 1 3 2 4 is odd, $A^+ = A_{odd}$. The filled points are thus exactly those in A^+ , and A^- consists of the square cut points and the empty circles. Thus this cut is (the unique) (4, 1, 3)-cut.

With this definition, n -separability of the necklace and the α -Ham Sandwich theorem guarantee a unique α -cut for every vector α fulfilling $1 \leq \alpha_i \leq |C_i|$, as we will see later. We are now ready to define the α -NECKLACE-SPLITTING problem.

► **Definition 3** (α -NECKLACE-SPLITTING). *Given an n -separable necklace C with n colours, and a vector $\alpha = (\alpha_1, \dots, \alpha_n)$ with $1 \leq \alpha_i \leq |C_i|$, the α -NECKLACE-SPLITTING problem is to find the unique α -cut of C .*

We are now ready to introduce our results.

► **Theorem 4.** *α -NECKLACE-SPLITTING is polynomial-time solvable.*

We contrast this result by showing that without the promise of n -separability, the associated decision problem is NP-complete.

► **Theorem 5.** *Given a necklace C with n colours, and a vector $\alpha = (\alpha_1, \dots, \alpha_n)$ with $1 \leq \alpha_i \leq |C_i|$, the problem of deciding whether C has an α -cut is NP-complete.*

Note that Borzechowski, Schnider and Weber [6] proved that it is possible to check whether a given necklace is n -separable in polynomial time. This is in contrast to the Ham Sandwich setting, where it has been shown that checking well-separation is co-NP-complete [3].

1.2 Proof Techniques

The algorithm of Borzechowski, Schnider and Weber [6] relies on the following core observation: If some colour only appears in the necklace in two *components* (i.e., consecutive intervals of \mathbb{R} that contain only points of this colour), the smaller of the two components can be discarded since the cut point of that colour must lie in the larger component. While for fair splitting this follows quite immediately – the larger component must be split, otherwise the cut cannot be fair – this observation does not generalise to unfair splitting. We thus have to use a more complicated approach.

We first show that under the promise of n -separability the number of colours that consist of more than two consecutive components is bounded by a constant. For each of these colours we can guess in which component the cut point of this colour lies. For each guess we reduce each of these colours to the single component containing the cut point, and try to compute the α -cut for the resulting necklace. For one of these guesses, the resulting α -cut must be adaptable to an α -cut of the original necklace. To compute the α -cut for these necklaces (in which now every colour consists of at most two components), we reduce the necklace further according to some reduction rules similar to those in [6]. This process will eventually come

to an end; we will reach an *irreducible* necklace. Here comes the crucial part of our proof: We will show that for each n , the irreducible necklaces with n colours all have the same *walk graph* (as introduced in [6] and below in Section 2). We translate α -NECKLACE-SPLITTING on irreducible necklaces into an integer linear program (ILP), and use the rigid structure of irreducible necklaces to show that the *primal graph* of this ILP has constant treewidth. Using the FPT algorithm of Jansen and Kratsch [15], we can thus solve these ILPs efficiently.

For the NP-hardness proof of the decision version of α -NECKLACE-SPLITTING we use a reduction from E3-SAT.

2 Preliminaries

Let us first formally introduce separability, as defined by Borzechowski, Schnider and Weber [6].

► **Definition 6 (Separability).** *A necklace C is k -separable if for all $A \subseteq C$ there exist k separator points $s_1 < \dots < s_k \in \mathbb{R}$ that separate A from $C \setminus A$. More formally, if we alternately label the intervals $(-\infty, s_1], [s_1, s_2], \dots, [s_{k-1}, s_k], [s_k, \infty)$ with A and \bar{A} (starting with either A or \bar{A}), for every interval I labelled A we have $I \cap \bigcup_{c \in (C \setminus A)} c = \emptyset$ and for every interval I' labelled \bar{A} we have $I' \cap \bigcup_{c \in A} c = \emptyset$.*

The separability $\text{sep}(C)$ of a necklace C is the minimum integer $k \geq 0$ such that C is k -separable.

We call each maximal set of consecutive points that have the same colour c a *component* of c . We say a colour c is an *interval*, if it consists of exactly one component. In other words, a colour c is an interval if its convex hull does not intersect any other colour c' .

Borzechowski, Schnider and Weber further showed that n -separability is strongly related to the notion of *well-separation*.

► **Definition 7.** *Let $P_1, \dots, P_k \subset \mathbb{R}^d$ be point sets. They are well-separated if and only if for every non-empty index set $I \subset [k]$, the convex hulls of the two disjoint subfamilies $\bigcup_{i \in I} P_i$ and $\bigcup_{i \in [k] \setminus I} P_i$ can be separated by a hyperplane.*

► **Lemma 8 ([6]).** *Let C be a set of n colours in \mathbb{R} . Let C' be the set of subsets of \mathbb{R}^n obtained by lifting each point in each colour of C to the n -dimensional moment curve using the function $f(t) = (t, t^2, \dots, t^n)$. Then the set C is n -separable if and only if C' is well-separated.*

The following theorem due to Steiger and Zhao [22] shows that we can always unfairly bisect well-separated point sets.

► **Lemma 9 (α -Ham-Sandwich Theorem, [22]).** *Let $P_1, \dots, P_n \subset \mathbb{R}^n$ be finite well-separated point sets in general position, and let $\alpha_1, \dots, \alpha_n$ be positive integers with $\alpha_i \leq |P_i|$, then there exists a unique $(\alpha_1, \dots, \alpha_n)$ -cut, i.e., a hyperplane H that contains a point from each colour and such that for the closed positive halfspace H^+ bounded by H we have $|H^+ \cap P_i| = \alpha_i$. Here, the positive side of a hyperplane H containing one point p_i per point set P_i is determined by the orientation of these p_i , i.e., for any point $h \in H^+$ the simplex (p_1, \dots, p_n, h) is oriented positively.*

Note that the $(\alpha_1, \dots, \alpha_n)$ -cut in Lemma 9 is defined by one point from each colour, and the positive side of the cut is determined by the orientation of these points on the hyperplane. This is the motivation of the similar restrictions in our definition of an α -cut in α -NECKLACE-SPLITTING (Definition 2).

Through the classical reduction of Necklace Splitting to the Ham-Sandwich problem obtained by lifting the points to the moment curve, as it appeared in many works before [9, 12, 16, 19], we can easily obtain the following theorem.

► **Theorem 10.** *α -NECKLACE-SPLITTING always has a unique solution.*

Proof (sketch). By Lemma 8, the point sets lifted to the moment curve are well-separated, and thus Lemma 9 applies. α -NECKLACE-SPLITTING thus always has a solution, and uniqueness follows from the fact that two different solutions of α -NECKLACE-SPLITTING would lift to different solutions of α -Ham Sandwich. ◀

To argue about the separability of necklaces, we use the view of *walk graphs* that were also introduced in [6].

► **Definition 11 (Walk graph).** *Given a necklace C , the walk graph G_C is the multigraph with $V = C$ and with every potential edge $\{a, b\} \in \binom{V}{2}$ being present with the multiplicity equal to the number of pairs of points $p \in a, p' \in b$ that are neighbouring.*

Note that given a necklace C as a set of point sets, the walk graph can be built in linear time in the size of the necklace $N := \sum_{c \in C} |c|$.

Recall that a graph is *Eulerian* if it contains a Eulerian tour, a closed walk that uses all edges exactly once. A graph is *semi-Eulerian* if it contains a Eulerian path, a (not necessarily closed) walk that uses all edges exactly once.

► **Observation 12.** *The walk graph of a necklace is connected and semi-Eulerian, and thus at most two vertices have odd degree.*

The separability of a necklace turns out to be equivalent to the max-cut in its walk graph.

► **Definition 13 (Cut).** *In a (multi-)graph G on the vertices V , a cut is a subset $A \subseteq V$. The size $\mu(A)$ of a cut A is the number of edges $\{u, v\}$ in G such that $u \in A$ and $v \notin A$. The max-cut, denoted by $\mu(G)$, is the largest size of any cut $A \subseteq V$.*

► **Lemma 14 ([6]).** *For every necklace C , we have $\text{sep}(C) = \mu(G_C)$.*

3 Tractability of the Search Problem

In this section we prove Theorem 4 by providing a polynomial-time algorithm for α -NECKLACE-SPLITTING. As mentioned above, the algorithm works in two main phases.

In the first phase, the necklace is first reduced to a necklace where each colour consists of at most two components by guessing the correct component to cut for colours with three or more components. The necklace with only colours with at most two components is then further reduced to an *irreducible* necklace.

In the second phase, we reduce necklace splitting in an irreducible necklace to a labelling problem of its walk graph. This labelling problem is then modelled as an integer linear program, which turns out to be tractable in polynomial time. To prove that this ILP is tractable, we prove some strong structural properties about the walk graphs of irreducible necklaces.

3.1 Reducing Necklaces

Instead of solving α -NECKLACE-SPLITTING, we will actually solve the following slightly more general problem. Given a vector $\alpha = (\alpha_1, \dots, \alpha_n)$, we define the complement vector $\bar{\alpha}$ as the vector $(|C_1| - \alpha_1 + 1, \dots, |C_n| - \alpha_n + 1)$. If α denotes the number of points per point set on the positive side of a cut, $\bar{\alpha}$ denotes the number of points on the other side, both sides including the cut points. Since the cut parity can change in our reduction steps and thus the positive side may become the negative side and vice versa, the following problem is nicer to solve recursively than α -NECKLACE-SPLITTING.

► **Definition 15.** α - $\bar{\alpha}$ -NECKLACE-SPLITTING

Input: An n -separable necklace $C = \{C_1, \dots, C_n\}$,
a vector $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha_i \in \{1, \dots, |C_i|\}$.

Output: A pair (S, \bar{S}) , where S is the unique α -cut and \bar{S} the unique $\bar{\alpha}$ -cut.

3.1.1 Reducing to at most two components per colour

In this section we will algorithmically reduce α - $\bar{\alpha}$ -NECKLACE-SPLITTING to the following subproblem, where each colour in the necklace consists of at most two components.

► **Definition 16.** α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂

Input: An n -separable necklace $C = \{C_1, \dots, C_n\}$, where each colour has at most 2 components, and a vector $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha_i \in \{1, \dots, |C_i|\}$.

Output: A pair (S, \bar{S}) , where S is the unique α -cut and \bar{S} the unique $\bar{\alpha}$ -cut.

Our reduction is based on the following observation, proven in the full version of this paper.

► **Lemma 17.** Let C be an n -separable necklace with n colours for $n \geq 8$. Then in the necklace at least one of the following must be true:

- (i) there are two neighbouring intervals, or
- (ii) there is no colour with more than four components, and at most two colours have more than two components.

The proof of Lemma 17 as well as the proofs of many other structural results on n -separable necklaces in this paper and in [6] rely on Lemma 14 and the following bound that is a corollary of a theorem of Poljak and Turzík.

► **Corollary 18** ([18]). A connected (multi-)graph G with n vertices and m edges has a maximum cut $\mu(G)$ of at least $\omega(G) := \frac{m}{2} + \frac{n-1}{4}$.

This corollary directly gives a bound on the number of edges in the walk graph of an n -separable necklace. By this we can also bound the degree distribution of the vertices in the walk graph and therefore also the distribution of the numbers of components of colours in the necklace.

Algorithmically, the conditions (i) and (ii) can be used as follows. If there are two neighbouring intervals, since an α -cut must go through exactly one bead of each colour, there must be a cut in both intervals. Moreover, in between these cuts there is no other cut point. Therefore, we remove the two intervals and solve on the newly obtained necklace recursively. Finally, we add the cuts at the right positions in the removed intervals.

If there are no neighbouring intervals, condition (ii) states that at most two colours have more than two components. The strategy will be to look at these colours and test out every possible component per colour. Again by condition (ii), this requires at most

$4 \cdot 4 = 16 \in O(1)$ tests. That is, for each colour with more than two components we fix a component and remove all other components of that colour. In this smaller necklace (that still consists of n colours) we recursively solve for an α -cut. The necklace for the recursive call will then be a necklace where each colour has at most two components, so we need to solve an α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ instance in the recursive step.

Since there must be an α -cut, one combination of components must lead to a smaller necklace whose α -cut can be augmented by inserting the cut of each colour in the fixed component.

Note that for our recursive calls to be valid, we also need that removing neighbouring intervals yields a $(n - 2)$ -separable necklace on $n - 2$ colours, and removing all but one component from a colour does not destroy n -separability either. Both of these facts are shown in [6, Lemmas 19 and 20].

This lets us conclude the following proposition.

► **Proposition 19.** *Let $T_2(n, N)$ be the time to solve α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ on an n -separable necklace with n colours and a total of N beads. Then α - $\bar{\alpha}$ -NECKLACE-SPLITTING on an n -separable necklace with a total of N beads and n colours can be solved in at most $O(T_2(n, N) + n \cdot N)$ time.*

Proof. We describe an algorithm for α - $\bar{\alpha}$ -NECKLACE-SPLITTING in this proof, pseudocode of this algorithm can be found in the full version of this paper.

As a base case, for n small enough ($n < 8$) we simply apply a brute force algorithm that iterates through every possible cut and checks if it has found the α -cut or $\bar{\alpha}$ -cut. Otherwise, the algorithm proceeds as follows.

In a first step, neighbouring intervals are removed from the necklace and the α - and $\bar{\alpha}$ -cut is computed in the obtained necklace recursively. Then in these cuts the right cuts are added in the removed intervals to get the α - and $\bar{\alpha}$ -cut. We need to make sure to maintain the cut parity when inserting these cuts, so the algorithm checks first if the cut parity switches when inserting these cuts and if yes, it swaps the α - and $\bar{\alpha}$ -cuts obtained by the recursive call. That is, if the cut parity switches, the algorithm uses the obtained $\bar{\alpha}$ -cut and turns it into the α -cut by inserting the correct cuts in the removed intervals (and similarly turns the α -cut into an $\bar{\alpha}$ -cut).

If there are no neighbouring intervals in the necklace, the algorithm determines the set of all colours with at least three components, call this set C^3 . If C^3 is empty, the necklace consists only of colours each having at most two components, so we can use an algorithm for α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ to compute the α - and $\bar{\alpha}$ -cut.

Otherwise, if C^3 is not empty, the algorithm iterates over all combinations of components of colours in C^3 . That is, each iteration considers a choice $(c_{i_1}, \dots, c_{i_{|C^3|}})$ of exactly one component c_{i_k} of each colour C_{i_k} in C^3 . In this iteration, all components from colours in C^3 except the components from the current choice are removed. We obtain a necklace that still consists of n colours but each colour has at most two components. Therefore, we can use an algorithm for α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ to find an α - and $\bar{\alpha}$ -cut in the new necklace. However, since we removed beads from the colours in C^3 the α vector might be invalid for this necklace. We therefore use a dummy α value of 1 for these colours. Then the algorithm checks if the obtained cuts can be turned to the corresponding α - or $\bar{\alpha}$ -cut by shifting the cuts along the beads within the components of the current iteration.

To show that the algorithm is correct, we need to show that the algorithm returns the correct result in the case when removing neighbouring intervals, and in the case when removing the components from colours with at least three components.

We first consider the case when removing neighbouring intervals. Note that inserting cuts in neighbouring intervals either changes the parity of the cut permutation for all cuts or for no cut. This can be seen as moving neighbouring intervals as a pair cannot add additional inversions to the cut permutation. Thus, moving them to the beginning of the necklace, there are always either an even or an odd number of inversions, regardless of the cut to augment. We can therefore indeed check whether inserting the cuts in the intervals changes parity and if it does we swap the α - and $\bar{\alpha}$ -cut of the necklace from the recursive call. This way, we make sure that when inserting the correct cut points in the removed intervals, the obtained cuts are indeed valid α - and $\bar{\alpha}$ -cuts. This shows that in this case the algorithm returns the unique α - and $\bar{\alpha}$ -cuts.

Next we show that the cut returned for the case when there are colours with at least three components is correct. We only argue for the α -cut, the case for the $\bar{\alpha}$ -cut is analogous. Notice that in the unique α -cut there is exactly one component per colour in C^3 where the cut lies in. Therefore, there must be one iteration for exactly that choice of components. For that iteration, the cut obtained by the call of the algorithm for α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ is a valid α -cut in the necklace C' for all colours except the ones in C^3 , where C' is the necklace obtained by removing all components from colours in C^3 except the components from the current choice. Then the α -cut in C' can be shifted to an α -cut in C . Observe that we do not have to worry about changing cut parities, since the necklace C' works on the same set of colours and the cuts are only shifted within components, so the cut permutation does not change. Hence, the cut obtained in this case will be the correct α -cut.

For the runtime analysis note that the brute force step takes $\mathcal{O}(N)$ time. Moreover, as long as there are neighbouring intervals the algorithm takes $\mathcal{O}(N)$ per recursive call. In each recursive call the number of colours decreases by 2, so there are at most $\mathcal{O}(n)$ iterations. Hence, the runtime for removing neighbouring intervals is at most $\mathcal{O}(n \cdot N)$.

By Lemma 17 we have $|C^3| \leq 2$ and each colour in C^3 has at most four components. Therefore, the number of iterations over components of C^3 is constant, where each iteration takes time $\mathcal{O}(T_2(n, N) + N)$.

Hence, the total runtime is $\mathcal{O}(n \cdot N + T_2(n, N) + N) = \mathcal{O}(T_2(n, N) + n \cdot N)$. \blacktriangleleft

3.1.2 Further reductions until irreducibility

To solve the α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ problem we perform some further reductions to reach a necklace that we cannot further reduce using any techniques known to us, which we will call irreducible.

► **Definition 20.** *An n -separable necklace C is called irreducible if and only if all of the following conditions hold.*

- (i) *All colours have at most two components,*
- (ii) *there are no neighbouring intervals,*
- (iii) *neither the first nor the last component is an interval,*
- (iv) *the first and the last component are of different colours.*

We thus reduce α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ to the following subproblem.

► **Definition 21.** α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING

Input: *An n -separable irreducible necklace $C = \{C_1, \dots, C_n\}$,
a vector $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha_i \in \{1, \dots, |C_i|\}$.*

Output: *A pair (S, \bar{S}) , where S is the unique α -cut and \bar{S} the unique $\bar{\alpha}$ -cut.*

To perform this reduction, we search for violations of any of the conditions (ii) to (iv) in Definition 20. Note that although we already removed all neighbouring intervals in Section 3.1.1, neighbouring intervals may have been reintroduced by removing some components from the necklace when removing components from colours, or may be reintroduced now when dealing with any of the other three violations in the further reduction.

In the following we show how the α -cut can be computed for each of the violations to conditions (ii) to (iv). We already know how to deal with violations to condition (ii), i.e., neighbouring intervals, from the previous subsection.

If condition (iii) is violated because the first component is an interval, the idea is to remove that colour and solve recursively on the smaller instance C' . To obtain an α -cut we take either the α -cut or the $\bar{\alpha}$ -cut in C' and add the cut the first component at the correct bead. Note that if the first component is of colour C_i and the number of colours $C_j \in C'$ such that $j < i$ is odd, then augmenting the cut will switch the parity of the cut permutation, since adding the first cut in $C_i \notin C'$ adds an odd number of inversions to the permutation. Moreover, adding a cut in the first component flips the positive and the negative side of the rest of the necklace once more. Thus, if the first component is such that the cut permutation parity does *not* flip, we extend the $\bar{\alpha}$ -cut of C' , and otherwise the α -cut.

The same idea is applied if the last component is an interval C_i . We again remove this colour, solve recursively and add a cut in the last component. In this case, adding the cut to the last component does not additionally flip the positive and negative side of the rest of the necklace, but the permutation flips parity if the number of colours $C_j \in C'$ such that $j > i$ is odd.

It is easy to see that removing an interval at the beginning or the end of the necklace decreases the separability by 1, since this interval can always be put on the correct side of a cut so one cut is needed to separate it from the rest. Thus, C' is indeed $(n - 1)$ -separable, and the recursive call is legal.

If condition (iv) is violated because the first and last component are of the same colour, we use a similar idea. We obtain the necklace C' by removing this colour. Note again that C' is $(n - 1)$ -separable. We again use recursion to get an α - and $\bar{\alpha}$ -cut for the remaining colours in C' . The α -cut in C can now be found by trying to augment both the α -cut and the $\bar{\alpha}$ -cut of C' by inserting a cut in the first or last component. At least one of the two augmentations must succeed as removing the cut in C_i from the unique α -cut in C must yield either the α - or the $\bar{\alpha}$ -cut of C' . Similarly, we can also find the $\bar{\alpha}$ -cut of C .

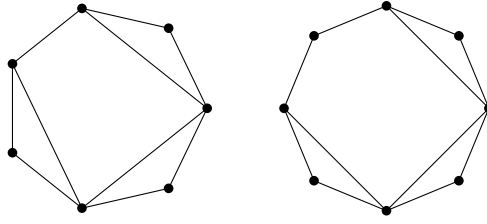
For the sake of completeness we give pseudocode for the way these reductions can be implemented in the full version of this paper. From the arguments above it follows that the algorithm is correct and runs in $\mathcal{O}(T_{irr}(n, N) + n \cdot N)$ time, where $T_{irr}(n, N)$ is the time needed to solve α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING. This proves the following proposition.

► **Proposition 22.** *Let $T_{irr}(n, N)$ be the time to solve α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING on an n -separable irreducible necklace with n colours and a total of N beads. Then α - $\bar{\alpha}$ -NECKLACE-SPLITTING₂ on an n -separable necklace with a total of N beads and n colours can be solved in at most $\mathcal{O}(T_{irr}(n, N) + n \cdot N)$ time.*

3.2 Structure of Irreducible Necklaces

To be able to solve α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING, we will first analyse the structure of the walk graphs of irreducible necklaces.

71:10 Unfairly Splitting Separable Necklaces



■ **Figure 2** The graphs N_7 to the left and N_8 to the right.

First, we claim that the walk graphs $G = (V, E)$ of irreducible necklaces with n colours can be characterised as follows. A proof of this can be found in the full version of this paper, but the lemma follows quite straightforwardly from Definition 20.

► **Lemma 23.** *A graph $G = (V, E)$ on n vertices is the walk graph of an irreducible necklace with n colours if and only if it satisfies all of the following conditions.*

- a) G is semi-Eulerian but not Eulerian,
- b) for all vertices $v \in V$ we have $\deg(v) \in \{2, 3, 4\}$,
- c) there are no adjacent vertices of degree 2,
- d) the maximum cut of G is at most $\mu(G) \leq n$.

If a graph G is the walk graph for some irreducible necklace, we call G an irreducible walk graph. We will see that all irreducible walk graphs on n vertices are isomorphic. In particular, we claim that the walk graph is isomorphic to the graph N_n defined as follows.

► **Definition 24.** *The cycle graph C_n is the cycle on the vertex set $[n]$. The graph P_{n-1}^{odd} is the graph with vertex set $[n]$ and edge set*

$$E(P_{n-1}^{\text{odd}}) = \left\{ \{2i-1, 2i+1\} \mid i \in \left[\lfloor \frac{n-1}{2} \rfloor \right] \right\},$$

that is P_{n-1}^{odd} is the graph on vertex set $[n]$ with a path of length $\lfloor (n-1)/2 \rfloor$ starting at vertex 1 and skipping every other vertex. Now the graph N_n obtained by taking the union of the edges in C_n and in P_{n-1}^{odd} is called the irreducible necklace graph of size n .

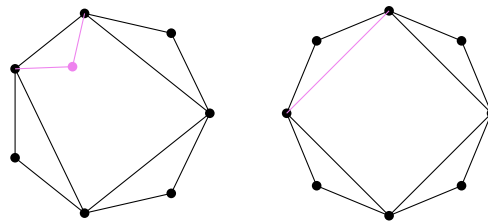
See Figure 2 for two examples of these graphs. Solving α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING heavily relies on the following proposition.

► **Proposition 25.** *Let C be an irreducible necklace with n colours for some $n \geq 3$. The walk graph G of C is isomorphic to N_n .*

The proof of this proposition is quite technical and lengthy. Moreover, the proof itself is not instructive for the further design of the algorithm. We thus only present the proof in the full version of this paper.

3.3 Splitting Irreducible Necklaces

As mentioned above, we will solve α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING by formulating it as an edge labelling problem in the walk graph, which will then be formulated as an integer linear program (ILP). While at first glance a reduction to an ILP might be counterintuitive due to the NP-completeness of ILP, it turns out that in our special case the ILP is tractable in polynomial time.



■ **Figure 3** Turning the walk graph to the label graph for $n = 7$ (left) and $n = 8$ (right). The coloured edges and vertices are added to the walk graph to obtain the label graph.

3.3.1 The cut labelling problem

Recall that the edges of the walk graph correspond to intervals between beads of the necklace – where one of the beads is the last bead of one component and the other is the first bead of a component of some different colour. Any choice of one component per colour to put a cut point puts some of these intervals on the positive side of the cut, and others on the negative side. In this way, such a choice of components induces a labelling of the edges of the walk graph by “positive” and “negative”. Of course, not every labelling corresponds to a cut. We will now collect some properties of labellings that do correspond to cuts.

For every component of a colour c there are two edges (except if the component appears at the beginning or at the end of the necklace): one edge corresponding to the change of colours when entering the component and one edge when leaving the component. We call these pairs of edges *traversals*. For a vertex v define $\text{trav}(v) := \{(e, e') \in E \times E \mid e, e' \text{ is a traversal of } v\}$ to be the set of traversals of v . The two edges of a traversal are labelled the same if and only if the corresponding component is not chosen to contain a cut point. Thus, if we now consider a vertex corresponding to an interval, it must have exactly one positively labelled incident edge, and one negatively labelled incident edge. Since a colour has exactly one component with a cut point, a bicomponent that is neither the first nor the last colour of the necklace must have either one positive and three negative, or one negative and three positive incident edges.

The edges of the walk graph only contain information about the intervals between components of the necklace. However, we additionally know that if n is even, the interval from $-\infty$ to the first cut point and the interval from the last cut point to ∞ must be on the same side of the cut, as there is an even number of cut points. Similarly for n odd, the two intervals are on opposite sides of the cut. To capture this information in the edge labelling, we slightly modify our walk graph. We add an edge between the vertices corresponding to the first and last component if n is even, or a subdivided edge (with a new degree two vertex) between these two vertices if n is odd. This newly obtained graph is called the *label graph* (see Figure 3). We see that a choice of one component per colour also induces a labelling of these newly added edges. When n is odd, we see that the two edges incident to the newly introduced vertex v must have different labels, just like if v was a regular vertex corresponding to an interval.

So far, all of our properties are invariant under flipping the labelling of all edges. However, by the cut permutation, any choice of one component per colour fixes the positive side of the cut. We can see that for every labelling fulfilling the previous properties, exactly one of the labelling and its inverse are actually the labelling induced by some cut.

71:12 Unfairly Splitting Separable Necklaces

We thus have now found a characterisation of the labellings that are induced by choices of one component per colour to cut. However, we are interested only in α -cuts. We thus would now like to characterise the labellings that are induced by α -cuts.

Clearly, if both edges of a traversal (e, e') are labelled negative, all beads of the corresponding component c of colour C_v must lie on the negative side of the cut. This cut can only be an α -cut if α_c is low enough, i.e., $\alpha_v \leq |C_v| - |c|$. Similarly, if both edges were labelled positive, we would have to have $\alpha_v \geq |c| + 1$. It turns out that if an edge labelling of the label graph fulfils this condition for some fixed α , we can actually place the cut points in the corresponding components to get an α -cut. We thus define the following problem.

For notational convenience we let $e(v, i, 1)$ and $e(v, i, 2)$ be the two edges corresponding to the i -th traversal of the vertex v . That is, $(e(v, i, 1), e(v, i, 2)) \in \text{trav}(v)$, and this traversal corresponds to the i -th component of that colour. In our case $e(v, i, j)$ is only defined for $i \in \{1, 2\}$ for every vertex; for intervals only for $i = 1$. We furthermore define $w_v(i)$ as the size of the i -th component of colour v .

► **Definition 26.** We define α -CUT-LABELLING as the following problem.

Input: The label graph $G = (V, E)$ of some n -separable irreducible necklace $C = \{C_1, \dots, C_n\}$, and a vector $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha_i \in \{1, \dots, |C_i|\}$.

Output: A subset of the edges $\mathcal{P} \subseteq E$ to be labelled positively (the negatively labelled edges are $\mathcal{N} := E \setminus \mathcal{P}$) such that:

- (1) $\forall v \in V : |\{e \in \mathcal{P} \mid v \in e\}| \in \{1, 3\}$,
- (2) $\forall v \in V, i \in \{1, 2\} : \{e(v, i, 1), e(v, i, 2)\} \subseteq \mathcal{P} \implies \alpha_v \geq w_v(i) + 1$,
- (3) $\forall v \in V, i \in \{1, 2\} : \{e(v, i, 1), e(v, i, 2)\} \subseteq \mathcal{N} \implies \alpha_v \leq |C_v| - w_v(i)$,
- (4) all edges in \mathcal{P} lie on the positive side of the cut induced by the labelling $(\mathcal{P}, \mathcal{N})$.

A solution of an α -CUT-LABELLING instance for a given α -vector is called an α -labelling. The concluding result of this section is the following.

► **Proposition 27.** Let $T_{lab}(n)$ be the time required to solve α -CUT-LABELLING on the label graph of any irreducible necklace with n colours. We can solve α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING on any necklace with n colours and N total beads in time $\mathcal{O}(T_{lab}(n, N) + N)$.

Proof. To solve α - $\bar{\alpha}$ -NECKLACE-SPLITTING on the necklace C we simply solve α -CUT-LABELLING and $\bar{\alpha}$ -CUT-LABELLING on its label graph and translate the labellings back to α - and $\bar{\alpha}$ -cuts. We only consider the α -labelling, since the case for $\bar{\alpha}$ works exactly the same way.

We begin by placing a cut point in the first or last bead of each component that is cut according to the α -labelling. Thanks to conditions (1) and (4) of an α -labelling, this already yields an α' -cut for some other α' . The cut points can now be moved within their components. Moving the cut point of colour v within its component only changes the number of points on the positive side of colour v and leaves other colours unaffected. Moving the cut point from one side of the component to the other allows us to include any number of points of the component on the positive side, from 1 point up to all points. Thanks to conditions (2) and (3), each cut point can be moved such that exactly α_v points of the colour are on the positive side. We have thus reached an α -cut we can return. Since this movement can be computed for each colour individually, it can be performed in $O(N)$. The label graph can also be computed in $O(N)$, and thus the statement follows. ◀

Note that for a given α , there must be a unique α -labelling, since applying the strategy in the proof above to distinct α -labellings would yield distinct α -cuts, a contradiction to the α -Ham Sandwich theorem.

3.3.2 An ILP formulation

In this section we model α -CUT-LABELLING as an Integer Linear Program (ILP). To do this, we formulate an ILP such that any solution of the ILP corresponds to a labelling fulfilling conditions (1)–(3) in Definition 26 and vice versa. Condition (4) can then be addressed by observing that for a given instance of α -CUT-LABELLING there are only a constant number of labellings fulfilling conditions (1)–(3): namely the labelling induced by the α -cut and the labelling induced by the $\bar{\alpha}$ -cut, but with the label of all edges swapped. Instead of merely solving for one solution of the ILP, we will later just enumerate all feasible solutions and check the result against condition (4).

Our ILP formulation will only use binary variables. For the sake of clarity we use **bold** face for variables in our ILP. To model that each edge is either labelled positively or negatively, we introduce two binary variables per edge. For each edge e in the label graph add two binary variables \mathbf{p}_e and \mathbf{n}_e with the interpretation that $\mathbf{p}_e = 1 \iff e$ is labelled positive and $\mathbf{n}_e = 1 \iff e$ is labelled negative. Since any edge must have exactly one label we add the constraint $\mathbf{p}_e + \mathbf{n}_e = 1$ for every edge e .

To model constraints on traversals, we introduce new binary variables $\mathbf{p}_{v,i}$ and $\mathbf{n}_{v,i}$ for every vertex v and possible indices of traversals i . The interpretation of these variables are that in the i -th traversal of v both the edges are labelled both positive or both negative, respectively. To keep these new variables consistent with the variables for edges, we wish to add the constraints $\mathbf{p}_{v,i} = \mathbf{p}_{e(v,i,1)} \cdot \mathbf{p}_{e(v,i,2)}$ and $\mathbf{n}_{v,i} = \mathbf{n}_{e(v,i,1)} \cdot \mathbf{n}_{e(v,i,2)}$. Note that these constraints are not linear, but they can be linearised as follows.

$$\begin{array}{ll} \mathbf{p}_{v,i} \leq \mathbf{p}_{e(v,i,1)} & \mathbf{n}_{v,i} \leq \mathbf{n}_{e(v,i,1)} \\ \mathbf{p}_{v,i} \leq \mathbf{p}_{e(v,i,2)} & \mathbf{n}_{v,i} \leq \mathbf{n}_{e(v,i,2)} \\ \mathbf{p}_{v,i} \geq \mathbf{p}_{e(v,i,1)} + \mathbf{p}_{e(v,i,2)} - 1 & \mathbf{n}_{v,i} \geq \mathbf{n}_{e(v,i,1)} + \mathbf{n}_{e(v,i,2)} - 1 \end{array}$$

We also wish to encode whether the edges corresponding to the i -th traversal have the same label, no matter what this label is. We thus introduce the variables $\mathbf{s}_{v,i}$ for each vertex v and its traversals i . They are related to $\mathbf{p}_{v,i}$ and $\mathbf{n}_{v,i}$ as follows.

$$\mathbf{s}_{v,i} = \mathbf{p}_{v,i} + \mathbf{n}_{v,i}$$

Now we use these variables to encode condition (1) of a cut labelling. This can be done by the following constraints.

$$\begin{array}{ll} \forall v \in V, \deg(v) = 4 : & \mathbf{s}_{v,1} + \mathbf{s}_{v,2} = 1 \\ \forall v \in V, \deg(v) = 2 : & \mathbf{s}_{v,1} = 0 \end{array}$$

Lastly, we need to encode the conditions the α -vector poses on the labelling, that is conditions (2) and (3) of a cut labelling. Note that these conditions are only necessary for vertices with multiple traversals, since for intervals the conditions are always satisfied.

Condition (2) can be implemented using the constraints $\mathbf{p}_{v,i} \cdot w_v(i) + 1 \leq \alpha_v$. For condition (3) we need to be more careful since only using $\mathbf{n}_{v,i} \cdot w_v(i) \geq \alpha_v$ results in a violated condition if $\mathbf{n}_{v,i} = 0$. We thus need to include the case when $\mathbf{n}_{v,i} = 0$ and add a trivial upper bound on α_v . This can be done by using $|C_v|$, the total number of beads for that vertex. We obtain the following constraints for all vertices v with $\deg(v) = 4$.

$$\begin{aligned}
\mathbf{n}_{v,1} \cdot w_v(2) + (1 - \mathbf{n}_{v,1}) \cdot |C_v| &\geq \alpha_v \\
\mathbf{n}_{v,2} \cdot w_v(1) + (1 - \mathbf{n}_{v,2}) \cdot |C_v| &\geq \alpha_v \\
\mathbf{p}_{v,1} \cdot w_v(1) + 1 &\leq \alpha_v \\
\mathbf{p}_{v,2} \cdot w_v(2) + 1 &\leq \alpha_v
\end{aligned}$$

Note that in general ILPs also optimise an objective function. However, we are only interested in the satisfying assignments. Let us now summarise the complete α -CUT-LABELLING-ILP.

► **Definition 28.** α -CUT-LABELLING-ILP

Let $G = (V, E)$ be a label graph of some n -separable necklace $C = \{C_1, \dots, C_n\}$ with colours consisting of at most two components. The α -CUT-LABELLING-ILP is the ILP given by the feasibility of the following constraints.

$$\begin{aligned}
\forall e \in E : & \quad \mathbf{p}_e, \mathbf{n}_e \in \{0, 1\} \\
\forall v \in V, i \in [\deg(v)/2] : & \quad \mathbf{p}_{v,i}, \mathbf{n}_{v,i}, \mathbf{s}_{v,i} \in \{0, 1\} \\
\forall e \in E : & \quad \mathbf{p}_e + \mathbf{n}_e = 1 \\
\forall v \in V, i \in [\deg(v)/2] : & \quad \mathbf{p}_{v,i} \leq \mathbf{p}_{e(v,i,1)} \\
& \quad \mathbf{p}_{v,i} \leq \mathbf{p}_{e(v,i,2)} \\
& \quad \mathbf{p}_{e(v,i,1)} + \mathbf{p}_{e(v,i,2)} - 1 \leq \mathbf{p}_{v,i} \\
& \quad \mathbf{n}_{v,i} \leq \mathbf{n}_{e(v,i,1)} \\
& \quad \mathbf{n}_{v,i} \leq \mathbf{n}_{e(v,i,2)} \\
& \quad \mathbf{n}_{e(v,i,1)} + \mathbf{n}_{e(v,i,2)} - 1 \leq \mathbf{n}_{v,i} \\
& \quad \mathbf{s}_{v,i} = \mathbf{p}_{v,i} + \mathbf{n}_{v,i} \\
\forall v \in V, \deg(v) = 2 : & \quad \mathbf{s}_{v,1} = 0 \\
\forall v \in V, \deg(v) = 4 : & \quad \mathbf{s}_{v,1} + \mathbf{s}_{v,2} = 1 \\
& \quad \mathbf{n}_{v,1} \cdot w_v(2) + (1 - \mathbf{n}_{v,1}) \cdot |C_v| \geq \alpha_v \\
& \quad \mathbf{n}_{v,2} \cdot w_v(1) + (1 - \mathbf{n}_{v,2}) \cdot |C_v| \geq \alpha_v \\
& \quad \mathbf{p}_{v,1} \cdot w_v(1) + 1 \leq \alpha_v \\
& \quad \mathbf{p}_{v,2} \cdot w_v(2) + 1 \leq \alpha_v
\end{aligned}$$

By construction, α -CUT-LABELLING-ILP is equivalent to conditions (1)–(3) of α -CUT-LABELLING in the sense that any solution of the former can be turned to a labelling fulfilling the conditions and vice versa, by simply considering the \mathbf{p}_e and \mathbf{n}_e as the encoding of a labelling.

3.3.3 Solving cut labelling for irreducible necklaces

In this section we show that α -CUT-LABELLING-ILP is solvable in polynomial time. To do that we leverage the structure of the ILP given by the fact that the underlying necklace is irreducible. We wish to use the following FPT algorithm by Jansen and Kratsch.

► **Theorem 29** ([15]). *For an ILP instance \mathcal{I} where each variable is in the domain \mathcal{D} , feasibility can be decided in time $\mathcal{O}(|\mathcal{D}|^{\mathcal{O}(\text{tw}(P(\mathcal{I})))} \cdot |\mathcal{I}|)$, where $\text{tw}(P(\mathcal{I}))$ denotes the treewidth of the primal graph of the instance. Moreover, if the ILP only has a constant number of feasible solutions, they can be enumerated in the same time bound.*

The primal graph of an ILP encodes which pairs of variables occur in constraints together. Treewidth is a very well-studied graph parameter that describes how “tree-like” a graph is, with lower treewidth denoting that the graph is more “tree-like”. We omit the exact definitions here and refer to the full version of this paper. The following proposition shown in the full version of this paper using a hierarchical approach states that the primal graph of the α -CUT-LABELLING-ILP has a constant treewidth. Since the ILP is boolean, the domain is $\mathcal{D} = \{0, 1\}$, and thus from this Theorem 29 gives us polynomial runtime for enumerating all solutions of α -CUT-LABELLING-ILP, and thus for solving α -CUT-LABELLING.

► **Proposition 30.** *Let $C = \{C_1, \dots, C_n\}$ be an irreducible necklace with n colours. Then the primal graph of the α -CUT-LABELLING-ILP of that necklace has treewidth at most 55.*

We are now ready to prove the following.

► **Proposition 31.** *α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING can be solved in $\mathcal{O}(n \cdot N)$ time.*

Proof. Theorem 29 implies an $\mathcal{O}(n^2)$ algorithm for α -CUT-LABELLING-ILP, using that $|\mathcal{D}| = |\{0, 1\}| \in \mathcal{O}(1)$ and $\text{tw}(P(\mathcal{I})) \in \mathcal{O}(1)$ by Proposition 30. The size of the ILP instance $|\mathcal{I}|$ is given by the number of entries of the matrix defining the instance \mathcal{I} . Thus we have $|\mathcal{I}| \in \mathcal{O}(n \cdot n)$, as we have a constant number of variables and constraints per colour. As argued above, the ILP only has a constant number of feasible solutions. Hence, we can enumerate these efficiently, and determine which of these satisfy condition (4) of α -CUT-LABELLING. This check can be implemented in $\mathcal{O}(N)$. This shows that α -CUT-LABELLING can be solved in time $\mathcal{O}(n^2 + N)$. Therefore, by Proposition 27 α - $\bar{\alpha}$ -IRR-NECKLACE-SPLITTING can be solved in $\mathcal{O}(n^2 + N) \in \mathcal{O}(n \cdot N)$ time, using that $n \in \mathcal{O}(N)$. ◀

Together with Propositions 19 and 22, we can conclude that α - $\bar{\alpha}$ -NECKLACE-SPLITTING, and thus α -NECKLACE-SPLITTING, can be solved in $\mathcal{O}(n \cdot N)$ time. We have thus proven Theorem 4.

4 NP-Completeness of the Decision Problem

In this section we prove Theorem 5, i.e., we show NP-completeness of deciding whether an arbitrary necklace has an α -cut. Since an α -cut can be used as a yes-certificate and verified in polynomial time, this problem is clearly in NP.

Instead of showing NP-hardness of this problem directly, we instead show NP-hardness of the following problem.

► **Definition 32.** $\alpha/\bar{\alpha}$ -NECKLACE-DECIDING

Input: A necklace $C = \{C_1, \dots, C_n\}$ and $\alpha = (\alpha_1, \dots, \alpha_n)$, $\alpha_i \in \{1, \dots, |C_i|\}$

Decide: Does C have a cut that is either an α -cut or an $\bar{\alpha}$ -cut?

$\alpha/\bar{\alpha}$ -NECKLACE-DECIDING can be seen as the problem of deciding whether there exist cut points that split the necklace into two sides, one of which has size α , but not necessarily the positive side of the cut defined via the cut parity. Clearly, $\alpha/\bar{\alpha}$ -NECKLACE-DECIDING can be solved by testing individually whether C has an α -cut, and whether it has an $\bar{\alpha}$ -cut. Thus, the following proposition immediately implies Theorem 5.

► **Proposition 33.** *$\alpha/\bar{\alpha}$ -NECKLACE-DECIDING is NP-hard.*

71:16 Unfairly Splitting Separable Necklaces

Proof. We reduce from E3-SAT. Let $\Phi = C_1 \wedge \cdots \wedge C_m$ be an instance of E3-SAT. Each clause C_i consists of exactly three variables. We now construct a necklace ζ and a vector α such that ζ has an α - or $\bar{\alpha}$ -cut if and only if Φ is satisfiable. As discussed above, we can instead show that Φ is satisfiable if and only if there exist cut points (one per colour) such that *some* chosen side contains the desired number of points α .

To construct ζ , we use the following types of beads. The two types of beads P and N are used to enforce that certain parts of the necklace are on the positive or negative side (we will set $\alpha(P) = |P|$ and $\alpha(N) = 1$). For every variable x we use the types x_0^A and x_0^B to construct a gadget which encodes the truth value of x . For every clause C_i such that $x \in C_i$ we additionally use the types x_i^A and x_i^B to read out the value of x at the clause C_i . To transfer the value of x , we use yet another type of bead x_T . For clauses C_i we only need one more type of bead, which we simply name C_i .

We also need multiple types of beads as *separator beads*. A separator bead only occurs once, and is used to enforce that there is a cut at its location. We denote a separator bead type by S_* where the subscript indicates the part of the necklace the separator bead belongs to.

Finally, we need two types of beads a, b that are only used to enforce the positive side of the cut. The necklace starts with the string $abab$ and we set α such that both beads of a must be on the positive side. Since there must be exactly one cut through one bead of a and another cut through the single occurrence of b , the only way of having both a and the b on the positive side is to cut the second occurrence of a , and to put the unbounded interval from $-\infty$ to the first occurrence of a on the positive side of the cut, or to cut through the first occurrence of a , and still putting the unbounded interval from $-\infty$ to the first bead of a on the positive side, which will then also put the second bead of a on the positive side.

The rest of the necklace consists of two main parts. The first part is the encoding part where we encode the assignment of variables and the satisfiability of each clause. In the second part we enforce the position of cuts for some types of beads. In the following we describe both parts, starting with the encoding part.

The encoding part first contains the following string for each variable x

$$P x_0^A \underbrace{x_T \dots x_T}_{k \text{ times}} x_0^B P x_0^A x_0^B P$$

where k is the number of occurrences of the variable x , that is, the number of clauses C_i such that $x \in C_i$ (either positively or negatively). Then, the encoding part contains the following string for each clause C_i and each variable $x \in C_i$.

$$\begin{array}{ll} P x_i^A & C_i \quad x_i^B P x_i^A x_T \quad x_i^B P, \quad \text{if } x \text{ appears as a positive literal in } C_i, \\ P x_i^A & x_i^B P x_i^A x_T \quad C_i \quad x_i^B P, \quad \text{if } x \text{ appears as a negative literal in } C_i. \end{array}$$

Note that the only difference between these two strings is the placement of the bead of type C_i .

We prove that under the following assumptions the encoding part properly encodes true assignments to Φ . These assumptions will be enforced by the α -vector and the second part of the necklace. We first assume that there is no cut in any bead of types P, x_T, C_i within the encoding part. Furthermore, we assume that a cut is correct if and only if the positive side of the cut *within the encoding part* contains all beads of type P , half of each x_T (note that x_T occurs an even number of times), at most two beads of type C_i , and both beads of types x_i^A and x_i^B (for all i including 0). Recall that a bead at which we cut the necklace is counted towards the positive side.

The only way to make the x_i^A and x_i^B types have both beads on the positive side is by either cutting through the first x_i^A and first x_i^B , or by cutting through the second x_i^A and second x_i^B . The first case will encode the assignment $x = \text{“true”}$ and the latter encodes $x = \text{“false”}$.

The cuts for $i = 0$ corresponding to a true assignment will move the k beads of type x_T between x_0^A and x_0^B to the negative side. Since we need half of the beads of type x_T on the positive side, this implies that for the cuts in x_i^A and x_i^B for $i \neq 0$, the bead of type x_T must be on the positive side. Hence, the cuts in x_i^A and x_i^B must encode the same assignment as the cuts in x_0^A and x_0^B .

We see that the bead of type C_i between x_i^A and x_i^B is on the negative side of the cut if and only if C_i is satisfied by the variable x . We thus see that at most two of the beads of type C_i are on the positive side if and only if C_i is fulfilled by one of its literals. Hence, we can conclude that under the assumptions listed above, the encoding part correctly encodes correct assignments to Φ .

In the second part of the necklace we will now first enforce that the cut points of type P, x_T , and C_i will lie in this second part.

We add the following three strings. For each clause C_i we add the string

$$P C_i C_i C_i S_i P,$$

where S_i is a new separator bead. Still assuming that P is not getting cut, this enforces that one of the three beads of type C_i is cut.

Then, for each variable x we add the string

$$\underbrace{P x_T \dots P x_T}_{k \text{ times}} \underbrace{N x_T \dots N x_T}_{k \text{ times}} N S_x,$$

where again k is the number of clauses containing x , and S_x is a new separator bead. Assuming P and N to not be cut, this enforces the k -th occurrence of x_T to be cut, putting k of the beads of type x_T in this substring on the positive side, and k on the negative side.

Finally, we add the string

$$P N.$$

Since the number of types of beads is even (P comes paired with N , x_i^A comes paired with x_i^B , C_i is paired with S_i and x_T is paired with S_x), we can see that this last bead of type N must be cut: If it was not cut, it would be on the positive side of the cut, since both unbounded intervals must belong to the positive side. However, the cut point of type N is another bead that would be counted to the positive side, violating $\alpha(N) = 1$. If however this last bead of type N is cut, also the last bead of type P must be cut by a symmetric argument.

We thus see that the second part of the string enforces the cuts we assumed in the encoding part. Furthermore, since the second part has exactly half of the beads of type x_T and one up to three beads of type C_i on the positive side, the following α -vector exactly gives us all of the assumptions on the number of beads on the positive side we made in the encoding part.

$$\begin{aligned}
\alpha(a) &= 2, \alpha(b) = 1, \\
\alpha(P) &= |P|, \alpha(N) = 1, \\
\alpha(x_i^A) &= 2, \alpha(x_i^B) = 2 && \text{for all variables } x \text{ and all } i \text{ including } 0, \\
\alpha(x_T) &= \frac{|x_T|}{2} && \text{for all variables } x_T, \text{ note that } |x_T| \text{ is even,} \\
\alpha(C_i) &= 3 && \text{for all clauses } C_i, \\
\alpha(S_*) &= 1 && \text{for any separator } S_*.
\end{aligned}$$

We have argued before that a cut with the correct number of points α on the positive side corresponds to a satisfying assignment of Φ . On the flip side, it is clear that a satisfying assignment can be turned into a cut by cutting the x_i^A, x_i^B at the corresponding places and the C_i at the correct of the three consecutive occurrences, depending on how many literals in C_i are true.

The necklace ζ can be built in polynomial time in m , and we thus get that $\alpha/\bar{\alpha}$ -NECKLACE-DECIDING is NP-hard. \blacktriangleleft

For an example of this reduction see the full version of this paper.

References

- 1 Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. doi:10.1016/0001-8708(87)90055-7.
- 2 Noga Alon and Douglas B. West. The Borsuk-Ulam theorem and bisection of necklaces. In *Proceedings of the American Mathematical Society*, volume 98, pages 623–628. American Mathematical Society, 1986. doi:10.1090/S0002-9939-1986-0861764-9.
- 3 Helena Bergold, Daniel Bertschinger, Nicolas Grelier, Wolfgang Mulzer, and Patrick Schnider. Well-Separation and Hyperplane Transversals in High Dimensions. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*, volume 227 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2022.16.
- 4 Paul Bonsma, Thomas Epping, and Winfried Hochstättler. Complexity results on restricted instances of a paint shop problem for words. *Discrete Applied Mathematics*, 154(9):1335–1343, 2006. 2nd Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2003). doi:10.1016/j.dam.2005.05.033.
- 5 Michaela Borzechowski, John Fearnley, Spencer Gordon, Rahul Savani, Patrick Schnider, and Simon Weber. Two Choices Are Enough for P-LCPs, USOs, and Colorful Tangents. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.32.
- 6 Michaela Borzechowski, Patrick Schnider, and Simon Weber. An FPT Algorithm for Splitting a Necklace Among Two Thieves. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation (ISAAC 2023)*, volume 283 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2023.15.
- 7 Steven J. Brams and Alan D. Taylor. An Envy-Free Cake Division Protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995. doi:10.1080/00029890.1995.11990526.


- 8 Man-Kwun Chiu, Aruni Choudhary, and Wolfgang Mulzer. Computational Complexity of the α -Ham-Sandwich Problem. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.ICALP.2020.31.
- 9 Jesús De Loera, Xavier Goaoc, Frédéric Meunier, and Nabil Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *Bulletin of the American Mathematical Society*, 56(3):415–511, 2019. doi:10.1090/bull/1653.
- 10 Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012. doi:10.1287/opre.1120.1116.
- 11 Thomas Epping, Winfried Hochstättler, and Peter Oertel. Complexity results on a paint shop problem. *Discrete Applied Mathematics*, 136(2):217–226, 2004. The 1st Cologne-Twente Workshop on Graphs and Combinatorial Optimization. doi:10.1016/S0166-218X(03)00442-6.
- 12 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 638–649, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316334.
- 13 Charles H. Goldberg and Douglas B. West. Bisection of circle colorings. *SIAM Journal on Algebraic Discrete Methods*, 6(1):93–106, 1985. doi:10.1137/0606010.
- 14 Charles R. Hobby and John R. Rice. A moment problem in L1 approximation. *Proceedings of the American Mathematical Society*, 16(4):665–670, 1965. doi:10.2307/2033900.
- 15 Bart M. P. Jansen and Stefan Kratsch. A structural approach to kernels for ILPs: Treewidth and total unimodularity. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 779–791. Springer Berlin Heidelberg, 2015. doi:10.1007/978-3-662-48350-3_65.
- 16 Jiří Matoušek. *Lectures on Discrete Geometry*. Graduate Texts in Mathematics. Springer New York, 2002. doi:10.1007/978-1-4613-0039-7.
- 17 Frédéric Meunier and András Sebő. Paintshop, odd cycles and necklace splitting. *Discrete Applied Mathematics*, 157(4):780–793, 2009. doi:10.1016/j.dam.2008.06.017.
- 18 Svatopluk Poljak and Daniel Turzik. A polynomial time heuristic for certain subgraph optimization problems with guaranteed worst case bound. *Discrete Mathematics*, 58(1):99–104, 1986. doi:10.1016/0012-365X(86)90192-5.
- 19 Sambuddha Roy and William Steiger. Some combinatorial and algorithmic applications of the Borsuk–Ulam theorem. *Graphs and Combinatorics*, 23(1):331–341, June 2007. doi:10.1007/s00373-007-0716-1.
- 20 Erel Segal-Halevi, Shmuel Nitzan, Avinatan Hassidim, and Yonatan Aumann. Envy-free division of land. *Mathematics of Operations Research*, 45(3):896–922, 2020. doi:10.1287/moor.2019.1016.
- 21 Forest W. Simmons and Francis Edward Su. Consensus-halving via theorems of Borsuk–Ulam and Tucker. *Mathematical Social Sciences*, 45(1):15–25, 2003. doi:10.1016/S0165-4896(02)00087-2.
- 22 William Steiger and Jihui Zhao. Generalized ham-sandwich cuts. *Discrete & Computational Geometry*, 44(3):535–545, 2010. doi:10.1007/s00454-009-9225-8.
- 23 Arthur H. Stone and John W. Tukey. Generalized “sandwich” theorems. *Duke Math. J.*, 9(2):356–359, 1942. doi:10.1215/S0012-7094-42-00925-6.

Card-Based Protocols Imply PSM Protocols

Kazumasa Shinagawa  

Ibaraki University, Ibaraki, Japan

National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Koji Nuida  

Institute of Mathematics for Industry (IMI), Kyushu University, Fukuoka, Japan

National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Abstract

Card-based cryptography is the art of cryptography using a deck of physical cards. While this area is known as a research area of recreational cryptography and is recently paid attention in educational purposes, there is no systematic study of the relationship between card-based cryptography and the other “conventional” cryptography. This paper establishes the first generic conversion from card-based protocols to private simultaneous messages (PSM) protocols, a special kind of secure multiparty computation. Our compiler supports “simple” card-based protocols, which is a natural subclass of finite-runtime protocols. The communication complexity of the resulting PSM protocol depends on how many cards are opened in total in all possible branches of the original card-based protocol. This result shows theoretical importance of such “opening complexity” of card-based protocols, which had not been focused in this area. As a consequence, lower bounds for PSM protocols imply those for simple card-based protocols. In particular, if there exists no PSM protocol with subexponential communication complexity for a function f , then there exists no simple card-based protocol with subexponential opening complexity for the same f .

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Card-based cryptography, private simultaneous messages

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.72

Funding This work was supported by Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University: FY2022 Short-term Visiting Researcher “On Minimal Construction of Private Simultaneous Messages Protocols” (2022a006), FY2023 Short-term Visiting Researcher “On the Relationship between Physical and Non-physical Secure Computation Protocols” (2023a009), and FY2024 Short-term Visiting Researcher “Private Simultaneous Messages and Card-Based Cryptography” (2024a015).

Kazumasa Shinagawa: This work was supported in part by JSPS KAKENHI Grant Numbers 21K17702 and 23H00479, and JST CREST Grant Number MJCR22M1.

Koji Nuida: This work was supported in part by JSPS KAKENHI Grant Number JP22K11906.

1 Introduction

1.1 Background

Secure computation allows a set of parties, each with a secret input, to compute an output value of a function of their inputs without revealing any information on the inputs beyond the output value. Although secure computation is typically assumed to be implemented on electronic devices, there is a line of research to implement secure computation using a deck of physical cards, which is called *card-based cryptography* [8, 11, 21].

In card-based cryptography, an important research topic is to determine the minimum number of cards required for secure computation of a function. For the logical AND function, for example, upper and lower bounds on the number of cards have been studied by constructing a protocol and by proving impossibility for a certain number of cards [16–19], respectively.



© Kazumasa Shinagawa and Koji Nuida;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 72; pp. 72:1–72:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Other than AND protocols, lower bounds are so far known only for COPY protocols [16, 21] and protocols for generating a random permutation without fixed points [14], and as far as we know, no lower bounds for general functions have been explored.

On the other hand, in “conventional” (or “electronic”) cryptography, there have been derived some lower bounds for general functions. For example, lower bounds on the communication complexity have been studied for secure multiparty computation (MPC) [10], private simultaneous messages (PSM) [2, 4, 5, 12], conditional disclosure of secrets (CDS) [1, 13], and secret sharing (SS) [9].

If there is some technical relationship between card-based and conventional cryptography, lower bounds in card-based cryptography might be obtained from lower bounds in conventional cryptography. Unfortunately, however, there has been no systematic study of the relationship between card-based and conventional cryptography so far.

In this line of research, den Boer [11], in the historically first paper on card-based protocols, proposed a conventional cryptographic protocol based on an idea from a card-based protocol. However, there is no other research on constructing conventional cryptographic protocols from card-based protocols; this line of research has not progressed at all for more than 30 years, long enough for many cryptographers to believe that there is no more technical relationship between card-based and conventional cryptography.

1.2 Our Contribution

In this paper, we study a technical relationship between card-based and conventional cryptography. In particular, we develop a generic conversion from card-based to PSM protocols.

The model of PSM was initially introduced by Feige, Kilian, and Naor [12] and later generalized by Ishai and Kushilevitz [15]. A PSM protocol is a kind of secure computation protocols with one-round and unidirectional communication from n input players, having input-independent pre-shared common randomness, to an output player called a referee.

Our compiler supports a class of finite-runtime card-based protocols, which we name *simple* protocols (see Section 4.1): Roughly speaking, a finite-runtime protocol is said to be simple if, throughout an execution, all cards in each step are face-down except when opening cards. When the total number of possibly opened cards in a simple protocol is t , the resulting PSM protocol has communication complexity nt . This parameter t , which we name the *opening complexity* of the protocol, is a new complexity measure for card-based protocols not well investigated in the previous studies. It is an interesting research direction to design card-based protocols with small opening complexity.

For a special case, we consider *single-shuffle full-open (SSFO)* protocols (see Section 3). Since SSFO protocols are simple and the opening complexity of an SSFO protocol is exactly the same as the number of cards d used in the protocol, the communication complexity of the resulting PSM protocol is nd .

The main contribution of this study is deepening our understanding of card-based cryptography by connecting to the world of conventional cryptography. We believe that our work enhances a new direction for future research on card-based cryptography: An interesting research topic is to determine a class of functions for which our conversion yields efficient PSM protocols; Another interesting topic is to develop deeper connections between card-based protocols and PSM protocols or other kinds of conventional cryptographic protocols.

As a direct consequence of our conversion, lower bounds on the opening complexity t in card-based protocols will be derived from lower bounds on the communication complexity of PSM protocols (Corollary 5). If there exists no PSM protocol for any function with subexponential communication complexity, which is an unproven but plausible statement

and is stated as Conjecture 6, then there exists no simple protocol computing any function with subexponential opening complexity (Corollary 7). To prove an unconditional non-trivial lower bound for card-based protocols, we need a *super-quadratic* lower bound for PSM protocols, i.e., $\omega(kn^2)$ for a function $f: (\{0, 1\}^k)^n \rightarrow \{0, 1\}$. Unfortunately, as far as we know, the state-of-the-art lower bound is $\Omega(kn^2/\log(nk))$ by Ball and Randolph [5] and no super-quadratic lower bound has been proved yet. We believe that our result provides a new motivation for obtaining such a lower bound.

Another application of our conversion is to provide a new method to construct PSM protocols. As a concrete example, we construct a card-based protocol for an *indirect storage access* (ISA, for short) function [23] and then convert it to a PSM protocol with communication complexity $O(KL^2 + K^2L \log L)$, where K and L are parameters for the input length. Our protocol is more efficient than the existing constructions based on branching programs (BP, for short) and gate evaluation secret sharing (GESS, for short), both of which, to the best of our knowledge, are the only existing constructions for efficient PSM protocols effective for the ISA function.

2 Preliminaries

For an integer $n \geq 2$, we denote $[n] := \{1, 2, \dots, n\}$. For an integer $k \geq 1$, we denote the k -th symmetric group by S_k . For a k -bit string $s = (s_1, s_2, \dots, s_k) \in \{0, 1\}^k$ and a permutation $\pi \in S_k$, a permuted string of s by π , denoted by $\pi(s)$, is defined by $\pi(s) := (s_{\pi^{-1}(1)}, s_{\pi^{-1}(2)}, \dots, s_{\pi^{-1}(k)})$. For example, for a string $s = 111000 \in \{0, 1\}^6$ and a cyclic permutation $\pi = (1\ 2\ 3\ 4\ 5\ 6) \in S_6$, we have $\pi(s) = 011100$. We denote by X^* the set of finite sequences (of various lengths) from a set X .

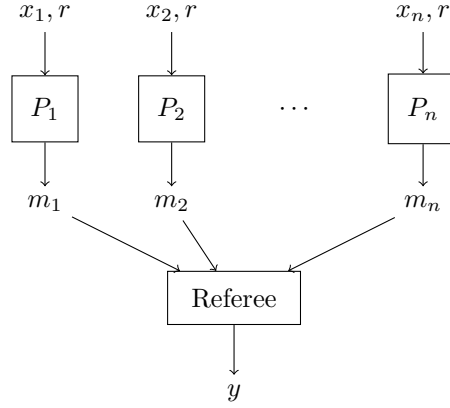
2.1 PSM Protocols

Feige, Kilian, and Naor [12] introduced a minimal model of secure two-party computation protocols. Ishai and Kushilevitz [15] generalized the model to the multiparty case, which is called *private simultaneous messages (PSM)*.

Suppose that n players P_1, P_2, \dots, P_n hold $x_1, x_2, \dots, x_n \in X$, respectively. They want to tell the output value $f(x_1, x_2, \dots, x_n)$ of a given function $f: X^n \rightarrow Y$ only to the *referee*, who is not one of the n players, without revealing any information on the inputs beyond the output value. The n players are assumed to share a *common randomness* and each of them is allowed to send a single message to the referee. The model of PSM is summarized in Figure 1.

Now we define the syntax and requirements for PSM protocols formally. Let X, Y be finite sets and $f: X^n \rightarrow Y$ a function. Let R, M_1, M_2, \dots, M_n be finite sets and $M := M_1 \times M_2 \times \dots \times M_n$. A *private simultaneous messages (PSM) protocol* for f is an $(n+2)$ -tuple $(\mathcal{R}, \text{Enc}_1, \text{Enc}_2, \dots, \text{Enc}_n, \text{Dec})$, where \mathcal{R} is a probability distribution over R , $\text{Enc}_i: X \times R \rightarrow M_i$ is the i -th *encoding function* ($1 \leq i \leq n$), and $\text{Dec}: M \rightarrow Y$ is the *decoding function*. The protocol proceeds as follows: First, a common randomness $r \leftarrow \mathcal{R}$ is chosen and distributed to the n players P_1, P_2, \dots, P_n . Then each player P_i holding (x_i, r) , where $x_i \in X$ is an input of P_i , computes a message $m_i = \text{Enc}_i(x_i, r)$ and sends it to the referee. Finally, on receiving m_1, m_2, \dots, m_n , the referee determines an output value $y = \text{Dec}(m_1, m_2, \dots, m_n)$.

The protocol is said to be *correct* if we have $\text{Dec}(\text{Enc}_1(x_1, r), \dots, \text{Enc}_n(x_n, r)) = f(\vec{x})$ for any $\vec{x} = (x_1, \dots, x_n) \in X^n$ and any $r \in R$ such that $\Pr[r \leftarrow \mathcal{R}] > 0$. The protocol is said to be *secure* if there exists an algorithm Sim called a simulator such that for any $\vec{x} \in X^n$,



■ **Figure 1** The model of PSM protocols.

the distribution of $\text{Sim}(f(\vec{x}))$ equals the distribution of $(\text{Enc}_1(x_1, \mathcal{R}), \dots, \text{Enc}_n(x_n, \mathcal{R}))$. The *communication complexity* of the protocol is defined by $\sum_{i=1}^n \log_2 |M_i|$. The *randomness complexity* of the protocol is defined by the Shannon entropy $H(\mathcal{R})$ of \mathcal{R} .

2.2 Card-Based Protocols

Mizuki and Shizuya [21] proposed a computational model for card-based protocols, referred to as the *Mizuki–Shizuya model*. We review the Mizuki–Shizuya model below. (We also follow the well-summarized description of [19].)

A *deck* \mathcal{D} is a finite and non-empty multiset of symbols. For a symbol $c \in \mathcal{D}$, $\frac{c}{\uparrow}$ and $\frac{c}{\downarrow}$ denote a *face-up card* and *face-down card* with symbol c , respectively, where ‘?’ is a special backside symbol that is not contained in \mathcal{D} . For a card (i.e., face-up card or face-down card) α , $\text{top}(\alpha)$ is defined by $\text{top}(\frac{c}{\uparrow}) := c$ and $\text{top}(\frac{?}{\downarrow}) := ?$, and the *flipped card* of α is defined to be $\frac{?}{\downarrow}$ if $\alpha = \frac{c}{\uparrow}$ and to be $\frac{c}{\uparrow}$ if $\alpha = \frac{?}{\downarrow}$.

In this paper, we mainly deal with a *two-colored deck* \mathcal{D} , which consists of two types of symbols \clubsuit and \heartsuit . The face-up cards $\frac{\clubsuit}{\uparrow}$ and $\frac{\heartsuit}{\uparrow}$ are depicted by $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, respectively, and a face-down card is depicted by $\boxed{?}$.

A *sequence of cards* from \mathcal{D} is obtained by permuting \mathcal{D} and, for each symbol $c \in \mathcal{D}$, choosing a face-up card $\frac{c}{\uparrow}$ or a face-down card $\frac{?}{\downarrow}$. For example, when $\mathcal{D} = [\clubsuit, \heartsuit, \heartsuit]$ (where the square brackets represent a multiset), $\Gamma = (\frac{\heartsuit}{\downarrow}, \frac{\clubsuit}{\uparrow}, \frac{?}{\downarrow})$ is a sequence of cards from \mathcal{D} . For a sequence of cards $\Gamma = (\alpha_1, \dots, \alpha_d)$, $\text{vis}(\Gamma) := (\text{top}(\alpha_1), \dots, \text{top}(\alpha_d))$ is called the *visible sequence* of Γ . We denote the set of all visible sequences of card sequences from \mathcal{D} by $\text{Vis}^{\mathcal{D}}$.

For a sequence of cards $\Gamma = (\alpha_1, \dots, \alpha_d)$, we can apply the following three types of *actions*:

- (perm, π) for $\pi \in S_d$ is an action that converts Γ to $\pi(\Gamma) := (\alpha_{\pi^{-1}(1)}, \alpha_{\pi^{-1}(2)}, \dots, \alpha_{\pi^{-1}(d)})$.
- (shuf, Π, \mathcal{F}) for a subset $\Pi \subseteq S_d$ and a probability distribution \mathcal{F} over Π is an action that converts Γ to $\pi(\Gamma)$, where $\pi \in \Pi$ is drawn from \mathcal{F} . Here, no player should know which permutation is chosen by the shuffle when it is applied to a sequence of face-down cards.
- (turn, T) for a subset $T \subseteq [d]$ is an operation that converts Γ to $(\beta_1, \dots, \beta_d)$ where β_i is the flipped card of α_i if $i \in T$ and $\beta_i = \alpha_i$ otherwise.

The set of actions for card sequences from \mathcal{D} is denoted by $\text{Action}^{\mathcal{D}}$.

A *Mizuki–Shizuya (MS) protocol* is defined by a tuple (\mathcal{D}, U, Q, A) , where \mathcal{D} is a deck, U is a set of input sequences (of cards), Q is a set of states including the *initial state* q_0 and the set of *final states* $Q_{\text{fin}} \subseteq Q$, and $A: (Q \setminus Q_{\text{fin}}) \times \text{Vis}^{\mathcal{D}} \rightarrow Q \times \text{Action}^{\mathcal{D}}$ is a

partial function called an *action function*. An MS protocol starts with an input sequence $\Gamma_0 \in U$ and the initial state q_0 . For the current sequence Γ_i and the current state q , if $A(\text{vis}(\Gamma_i), q) = (q', \text{action})$, the MS protocol applies **action** to Γ_i and updates the state to q' . The MS protocol terminates when entering a final state $q \in Q_{\text{fin}}$. The list of all sequences $(\Gamma_0, \Gamma_1, \dots, \Gamma_k)$ for a protocol execution is called the *sequence trace* of the protocol execution, and $(\text{vis}(\Gamma_0), \text{vis}(\Gamma_1), \dots, \text{vis}(\Gamma_k))$ is called the *visible sequence trace* of the protocol execution.

We note that an MS protocol is defined as a mechanism for converting an input sequence into an output sequence, and does not explicitly address the function to be computed. To associate a function $f: X^n \rightarrow Y$ with an MS protocol (\mathcal{D}, U, Q, A) , we need to give a map $\text{in}: X^n \rightarrow U$ called an *input function* and a map $\text{out}: (\text{Vis}^{\mathcal{D}})^* \rightarrow Y$ called an *output function*. For an input $\vec{x} \in X^n$, the input sequence of the protocol is set to $\text{in}(\vec{x}) \in U$, and the output of the protocol is defined by $\text{out}(v_0, v_1, \dots, v_k) \in Y$ when the visible sequence trace is $(v_0, v_1, \dots, v_k) \in (\text{Vis}^{\mathcal{D}})^{k+1}$.

Let in_i be the function representing the i -th card of the input sequence, i.e., $\text{in}(\vec{x}) := (\text{in}_1(\vec{x}), \dots, \text{in}_d(\vec{x}))$. We naturally assume that the output of each function in_i depends only on at most one input, say $x_j \in X$ (including the case that the output of in_i is constant), referred to as the *locality* of in .

A *protocol* is defined by a tuple $(\mathcal{D}, U, Q, A, \text{in}, \text{out})$, where (\mathcal{D}, U, Q, A) is an MS protocol, and in and out are input and output functions, respectively. A protocol for a function f is said to be *correct* if for all inputs $\vec{x} \in X^n$, the protocol outputs $f(\vec{x})$ whenever it terminates. A protocol for f is said to be *secure* if there exists an algorithm Sim called a simulator such that for any $\vec{x} \in X^n$, the distribution of $\text{Sim}(f(\vec{x}))$ equals the distribution of the visible sequence trace of the protocol execution with input \vec{x} . A protocol is said to be *finite-runtime* if it terminates within a fixed number of steps. A protocol is said to be *Las Vegas* if its runtime is finite in expectation.

We give two remarks on the above definition. A pair of face-down cards with different symbols is called a *commitment* to a bit, via the encoding rule $\spadesuit \heartsuit = 0$ and $\heartsuit \spadesuit = 1$. Our definition of in can deal with input commitments: For a sequence of n commitments to $x_1, x_2, \dots, x_n \in \{0, 1\}$, in our definition, the input function in can be represented by $\text{in}(\vec{x}) = \left(\frac{?}{x_1}, \frac{?}{\bar{x}_1}, \frac{?}{x_2}, \frac{?}{\bar{x}_2}, \dots, \frac{?}{x_n}, \frac{?}{\bar{x}_n} \right)$ using the encoding rule $\clubsuit = 0$ and $\heartsuit = 1$, where \bar{x}_i denotes the negation of a bit x_i . Our definition also captures other encoding rules as long as they satisfy the locality.

There are two types of protocols in the area of card-based cryptography: a *committed-format protocol* and a *non-committed-format protocol*. A protocol of the former type produces a sequence of commitments as output, while a protocol of the latter type publicly outputs the value $f(x_1, x_2, \dots, x_n)$. Our definition of the correctness and security supposed non-committed-format protocols, but committed-format protocols can be obviously converted to non-committed-format ones by just opening the output commitments and hence our proposed conversion method is also applicable to them.

For a protocol $\mathcal{P} = (\mathcal{D}, U, Q, A, \text{in}, \text{out})$ (or an MS protocol $\mathcal{P} = (\mathcal{D}, U, Q, A)$), the *protocol diagram* of \mathcal{P} is defined as the directed edge-labeled graph as follows:

- The set of vertices is Q .
- A directed edge labeled by $\text{action} \in \text{Action}^{\mathcal{D}}$ from q to q' exists if and only if there exists a visible sequence $v \in \text{Vis}^{\mathcal{D}}$ such that $A(q, v) = (q', \text{action})$.

See Figure 2 in Section 4.1 for an example of the protocol diagram of the four-card trick [20]. Note that a protocol diagram can be viewed as a reduced version of the Koch–Walzer–Härtel diagram [19]. We remark that any finite-runtime protocol can be easily converted to a (functionally equivalent) protocol whose protocol diagram forms a finite tree.

3 Our Conversion for Special Case

In order to explain the essential idea of our proposed conversion before a general description given in Section 4 below, in this section, we first describe our conversion for a special type of card-based protocols which we call single-shuffle full-open (SSFO) if it proceeds as follows:

1. The input sequence $\text{in}(\vec{x})$ is given:

$$\underbrace{\boxed{?} \boxed{?} \dots \boxed{?} \boxed{?}}_{\text{in}(\vec{x})}.$$

2. Apply a shuffle ($\text{shuf}, \Pi, \mathcal{F}$) to the sequence as follows:

$$\underbrace{\boxed{?} \boxed{?} \dots \boxed{?} \boxed{?}}_{\text{in}(\vec{x})} \rightarrow \underbrace{\boxed{?} \boxed{?} \dots \boxed{?} \boxed{?}}_{\pi(\text{in}(\vec{x}))},$$

where $\pi \in \Pi$ is chosen according to \mathcal{F} .

3. Turn over all cards and determine the output value:

$$\underbrace{\boxed{?} \boxed{?} \dots \boxed{?} \boxed{?}}_{\pi(\text{in}(\vec{x}))} \rightarrow \boxed{\heartsuit} \boxed{\clubsuit} \dots \boxed{\clubsuit} \boxed{\heartsuit} \Rightarrow y \in Y.$$

In Section 3.1, we demonstrate our conversion for an SSFO protocol called the five-card trick. In Section 3.2, we give our conversion for any SSFO protocol.

3.1 PSM Protocol from Five-Card Trick

The five-card trick is a five-card AND protocol proposed by den Boer [11]. First, we recall the protocol description of the five-card trick.

1. The input sequence $\text{in}(\vec{x})$ is given as follows, where $\vec{x} = (x_1, x_2) \in \{0, 1\}^2$:

$$\underbrace{\boxed{?} \boxed{?}}_{\bar{x}_1} \underbrace{\boxed{?}}_{\heartsuit} \underbrace{\boxed{?} \boxed{?} \boxed{?}}_{x_2}.$$

2. Apply a shuffle ($\text{shuf}, \langle (1\ 2\ 3\ 4\ 5) \rangle$) to the sequence as follows:

$$\underbrace{\boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?}}_{\text{in}(\vec{x})} \rightarrow \underbrace{\boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?}}_{\pi(\text{in}(\vec{x}))},$$

where $\pi = (1\ 2\ 3\ 4\ 5)^r$ for a uniformly random $r \in \{0, 1, 2, 3, 4\}$.

3. Turn over all cards. Output 0 if they are $\heartsuit\clubsuit\heartsuit\clubsuit\heartsuit$ up to cyclic shifts, and 1 if they are $\heartsuit\heartsuit\clubsuit\clubsuit$ up to cyclic shifts.

Now we convert the above protocol to a PSM protocol. Let Alice and Bob be the players holding $x_1, x_2 \in \{0, 1\}$, respectively, and Charlie the referee.

First, following the encoding rule $\clubsuit = 0$ and $\heartsuit = 1$, the input sequence $\text{in}(x_1, x_2)$, which is the sequence of symbols for the cards, can be represented by

$$\text{in}(x_1, x_2) := (\bar{x}_1, x_1, 1, x_2, \bar{x}_2) \in \{0, 1\}^5.$$

Although neither Alice nor Bob alone can compute the whole of $\text{in}(x_1, x_2)$, Alice can compute

$$s_1(x_1) := (\bar{x}_1, x_1, 1, 0, 0) \in \{0, 1\}^5,$$

and Bob can compute

$$s_2(x_2) := (0, 0, 0, x_2, \overline{x_2}) \in \{0, 1\}^5.$$

We note that, by taking the component-wise XOR, we have

$$s_1(x_1) \oplus s_2(x_2) = (\overline{x_1}, x_1, 1, x_2, \overline{x_2}) = \text{in}(x_1, x_2).$$

In other words, thanks to the locality of in , Alice and Bob can jointly compute the input sequence $\text{in}(x_1, x_2)$ in a distributed manner.

Next, we notice that Alice and Bob can apply a shuffle to the sequence *in their heads* by sharing a randomness of the shuffle as common randomness. In particular, by sharing a random cyclic permutation $\pi \in \langle (1\ 2\ 3\ 4\ 5) \rangle$, Alice and Bob can compute $\pi(s_1(x_1))$ and $\pi(s_2(x_2))$, respectively. We emphasize that

$$\pi(s_1(x_1)) \oplus \pi(s_2(x_2)) = \pi(s_1(x_1) \oplus s_2(x_2)) = \pi(\text{in}(x_1, x_2))$$

by the property of permutations.

Finally, to hide their inputs, Alice and Bob compute $m_1 := \pi(s_1(x_1)) \oplus r$ and $m_2 := \pi(s_2(x_2)) \oplus r$, respectively, where $r \in \{0, 1\}^5$ is also shared as common randomness, and send m_1 and m_2 to Charlie. Thanks to the masking by the random r , each of the messages m_1 and m_2 alone does not leak any information on x_1 and x_2 , respectively. Unmasking by Charlie requires computation of XOR for the two messages to cancel the term r :

$$\begin{aligned} m &:= m_1 \oplus m_2 = (\pi(s_1(x_1)) \oplus r) \oplus (\pi(s_2(x_2)) \oplus r) \\ &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) = \pi(\text{in}(x_1, x_2)), \end{aligned}$$

which only tells Charlie the result $\pi(\text{in}(x_1, x_2))$ of the XOR and does not tell individual sequences $\pi(s_1(x_1))$ nor $\pi(s_2(x_2))$. Then Charlie outputs 0 if m is equal to 10101 up to cyclic shifts, and 1 if m is equal to 11100 up to cyclic shifts.

The resulting PSM protocol is summarized as follows.

1. The common randomness consists of a cyclic permutation $\pi \in \langle (1\ 2\ 3\ 4\ 5) \rangle$ and a 5-bit string $r \in \{0, 1\}^5$, both are chosen uniformly at random.
2. Alice (resp. Bob) computes $m_1 = \pi(s_1(x_1)) \oplus r$ (resp. $m_2 = \pi(s_2(x_2)) \oplus r$) and sends it to Charlie.
3. On receiving m_1 and m_2 , Charlie determines an output value y as follows:

$$y := \begin{cases} 1 & \text{if } m_1 \oplus m_2 \in \{00111, 10011, 11001, 11100, 01110\}; \\ 0 & \text{if } m_1 \oplus m_2 \in \{01011, 10101, 11010, 01101, 10110\}. \end{cases}$$

The communication complexity of the protocol is 10 bits. The randomness complexity of the protocol is $5 + \log_2 5$ bits.

3.2 PSM Protocol from Any Single-Shuffle Full-Open Protocol

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \text{in}, \text{out})$ be any SSFO protocol whose shuffle operation is $(\text{shuf}, \Pi, \mathcal{F})$. By a similar observation to Section 3.1 thanks to the locality of the function in , we can define a function $s_i : X \rightarrow \{0, 1\}^d$ ($i \in [n]$) such that $s_1(x_1) \oplus \cdots \oplus s_n(x_n) = \text{in}(\vec{x})$ for any $\vec{x} \in X^n$.

The resulting PSM protocol $\widehat{\mathcal{P}}$ is given as follows, where $H(\mathcal{F})$ denotes the Shannon entropy of the distribution \mathcal{F} .

The PSM protocol $\widehat{\mathcal{P}}$ from any SSFO protocol \mathcal{P}

Input: $\vec{x} = (x_i)_{i \in [n]} \in X^n$

Common Randomness: $(\pi, (r_i)_{i \in [n]})$ is generated as follows:

- Choose a permutation $\pi \in \Pi$ according to \mathcal{F} .
- Choose $r_1, r_2, \dots, r_{n-1} \in \{0, 1\}^d$ uniformly at random, and set $r_n := r_1 \oplus r_2 \oplus \dots \oplus r_{n-1}$.

Encoding Function: Output $m_i := \pi(s_i(x_i)) \oplus r_i$.

Decoding Function: Compute $m := m_1 \oplus m_2 \oplus \dots \oplus m_n$. The output value y is equal to the output of the protocol \mathcal{P} when the opened symbol is m .

Communication Complexity: nd

Randomness Complexity: $(n-1)d + H(\mathcal{F})$

► **Theorem 1.** *Let \mathcal{P} be an SSFO protocol with correctness and security for $f: X^n \rightarrow Y$ using d cards. Then the above PSM protocol $\widehat{\mathcal{P}}$ is correct and secure for f with communication complexity nd .*

Proof. From the following computation, we have:

$$\begin{aligned} m &= m_1 \oplus m_2 \oplus \dots \oplus m_n \\ &= (\pi(s_1(x_1)) \oplus r_1) \oplus (\pi(s_2(x_2)) \oplus r_2) \oplus \dots \oplus (\pi(s_n(x_n)) \oplus r_n) \\ &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) \oplus \dots \oplus \pi(s_n(x_n)) \oplus r_1 \oplus r_2 \oplus \dots \oplus r_n. \end{aligned}$$

By the choice of r_n in $\widehat{\mathcal{P}}$, we have $r_1 \oplus r_2 \oplus \dots \oplus r_n = 0$. Hence we have

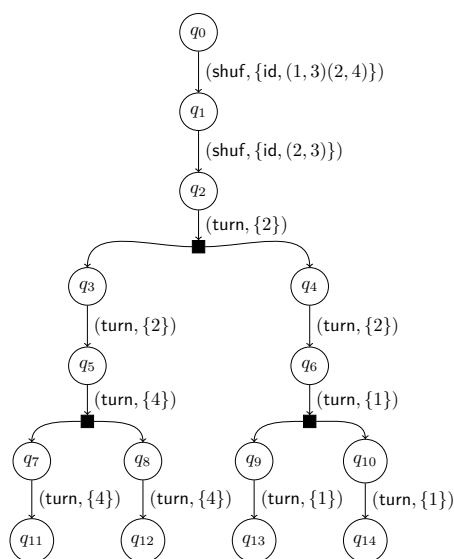
$$\begin{aligned} m &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) \oplus \dots \oplus \pi(s_n(x_n)) \\ &= \pi(s_1(x_1) \oplus s_2(x_2) \oplus \dots \oplus s_n(x_n)) = \pi(\text{in}(\vec{x})), \end{aligned}$$

which is exactly equal to the opened symbols of \mathcal{P} . Thus, the correctness of $\widehat{\mathcal{P}}$ follows from the correctness of \mathcal{P} .

We prove the security of $\widehat{\mathcal{P}}$ by constructing a simulator Sim for $\widehat{\mathcal{P}}$. Given an output value $y \in Y$, Sim first invokes a simulator for \mathcal{P} on input y , which outputs a visible sequence trace $(?^d, v) \in (\text{Vis}^{\mathcal{D}})^*$ of a protocol execution. Here, $v \in \{0, 1\}^d$ represents the opened values at the end of the protocol \mathcal{P} , therefore v has the same conditional distribution as m in the protocol $\widehat{\mathcal{P}}$ conditioned on the output value y . Then Sim chooses $r'_1, r'_2, \dots, r'_{n-1} \in \{0, 1\}^d$ uniformly at random, and outputs $(r'_1, r'_2, \dots, r'_{n-1}, r'_1 \oplus \dots \oplus r'_{n-1} \oplus v)$. Now the XOR of the n components is v , which (as mentioned above) has the same conditional distribution as m . Moreover, for each of the first $n-1$ component, say the i -th, the message $m_i = \pi(s_i(x_i)) \oplus r_i$ is uniformly random thanks to the random choice of r_i . Hence the distribution of $\text{Sim}(y)$ equals the distribution of the messages, therefore $\widehat{\mathcal{P}}$ is secure. ◀

4 Our Conversion for General Case

In this section, we give our conversion for *simple* protocols. In Section 4.1, we define the class of simple protocols. In Section 4.2, we give an overview of our conversion. In Section 4.3, we describe our conversion for simple protocols.



■ **Figure 2** The protocol diagram of the four-card trick.

4.1 Simple Protocols

► **Definition 2.** A card-based protocol $\mathcal{P} = (\mathcal{D}, U, Q, A, \text{in}, \text{out})$ is said to be simple if \mathcal{P} satisfies the following conditions:

Finite-Tree: The protocol diagram of \mathcal{P} forms a finite tree.

Face-Down: Any input sequence in U consists of face-down cards.

Mono-Opening: Every turn operation (turn, T) in \mathcal{P} satisfies $|T| = 1$. (Intuitively speaking, every turn operation flips a single card.)

Instant-Turn: Whenever a turn operation (turn, T) was applied to a sequence of all face-down cards at the last step, the next operation is the same turn operation (turn, T) . (Intuitively speaking, if a face-down card is opened, then it is immediately faced down.)

Here we note that, from the viewpoint of feasibility, focusing only on the simple card-based protocols does not decrease the generality of our conversion, because any finite-runtime protocol \mathcal{P} can be converted to a simple protocol (see Remark 3 below). We remark that almost all existing finite-runtime card-based protocols are already simple or can be trivially converted to simple (e.g., when some multiple cards are opened simultaneously, decomposing this step into a series of opening and closing of each single card in order to satisfy the mono-opening property). See Figure 2 for an example of the protocol tree of the four-card trick [20].

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \text{in}, \text{out})$ be a simple protocol. Then any turn operation in \mathcal{P} is classified into the following two types:

- The operation opens a single card from the whole sequence of face-down cards; referred to as an *open operation*. In this case, since we deal with two-colored decks, there are at most two subsequent states after this operation. We call this operation a *branch operation* if there are two subsequent states, and a *non-branch operation* otherwise.
- The operation faces down the card that was opened at the previous step; referred to as a *close operation*.

Now we observe that for each non-final state $q \in Q \setminus Q_{\text{fin}}$, the operation performed at the current step is uniquely determined from the state q . Indeed, when an open operation was performed at the previous step, the operation at the current step must be a close operation

72:10 Card-Based Protocols Imply PSM Protocols

for the unique opened card due to the instant-turn property. On the other hand, for the other case, the current visible sequence is the trivial visible sequence $v = (?, ?, \dots, ?)$, therefore the operation $A(q, v)$ in fact depends solely on q . This yields the following partitions for the set of non-final states:

$$\begin{aligned} Q \setminus Q_{\text{fin}} &= Q_{\text{turn}} \sqcup Q_{\text{perm}} \sqcup Q_{\text{shuf}} \\ &= (Q_{\text{turn}}^+ \sqcup Q_{\text{turn}}^-) \sqcup Q_{\text{perm}} \sqcup Q_{\text{shuf}} \\ &= ((Q_{\text{branch}} \sqcup Q_{\text{nonbranch}}) \sqcup Q_{\text{turn}}^-) \sqcup Q_{\text{perm}} \sqcup Q_{\text{shuf}}, \end{aligned}$$

where Q_{turn} , Q_{turn}^+ , Q_{turn}^- , Q_{branch} , $Q_{\text{nonbranch}}$, Q_{perm} , and Q_{shuf} are the sets of all non-final states for which the next operation is a turn operation **turn**, an open operation, a close operation (hence $Q_{\text{turn}} = Q_{\text{turn}}^+ \sqcup Q_{\text{turn}}^-$), a branch operation, a non-branch operation (hence $Q_{\text{turn}}^+ = Q_{\text{branch}} \sqcup Q_{\text{nonbranch}}$), a permutation operation **perm**, and a shuffle operation **shuf**, respectively.

The *opening complexity* of \mathcal{P} is defined by $|Q_{\text{turn}}^+|$. Note that this notion can be defined similarly for any finite-runtime protocol that is not necessarily simple.

► **Remark 3.** We note that any finite-runtime protocol can be converted into a simple protocol without affecting its correctness and security. As already stated in Section 2.2, finite-tree can be easily obtained. Mono-opening can be obtained by replacing each $(\text{turn}, \{i_1, \dots, i_k\})$ with $(\text{turn}, \{i_1\}), \dots, (\text{turn}, \{i_k\})$. Instant-turn can be obtained by appending close operations for every open operations. The most non-trivial part is to obtain face-down. Note that if a protocol is not face-down, then an input sequence may contain some face-up cards depending on the inputs, and some shuffle operations are applied to a sequence containing face-up cards called a *branching shuffle*. To obtain face-down, we modify the protocol as follows:

- We associate a pair of auxiliary cards (faced down in default) to each card, representing one-bit information by the order of two cards (where swapping these two cards corresponds to bit flipping). Then:
 - We change each face-up card in $\text{in}(\vec{x})$ to be faced-down (satisfying instant-opening) while recording this fact to auxiliary cards. At the beginning of the protocol, we read information in auxiliary cards by opening them, and turn suitable cards to recover the original $\text{in}(\vec{x})$.
 - Each step of the protocol is changed to: (1) turn down each face-up main (i.e., non-auxiliary) card while recording to auxiliary cards which main cards are to be face-up; (2a) in permutation/shuffle, main and auxiliary cards are synchronized; (2b) turn for main cards are emulated by updating information in (i.e., permuting) auxiliary cards; and (3) read information in auxiliary cards, turn up main cards suitably, and reset auxiliary cards. This avoids permutation/shuffle operations for face-up cards.

4.2 Overview of Our Conversion

Let \mathcal{P} be a simple protocol for $f: X^n \rightarrow Y$. From the finite-runtime property, we can assume that the protocol diagram of \mathcal{P} forms a tree.

If all permutations in the shuffle operations are fixed, the order of the sequence of cards at any step in an execution is determined. This is because the probability in an execution is taken only for shuffle operations. So, the basic idea of our conversion is to share the permutations chosen in shuffle operations as common randomness among the players, similar to the protocol in Section 3.

Let $c_1, c_2, \dots, c_t \in \{0, 1\}$ be the opened symbols when a sequence of permutations $\vec{\pi}$ in the shuffle operations and the input $\vec{x} \in X^n$ are given. Similar to the protocol in Section 3, we can observe that for each c_i , at least one player can compute it if $\vec{\pi}$ is shared

as common randomness. Therefore, each player P_i can compute a message $m_i \in \{0, 1\}^t$ such that $m_1 \oplus m_2 \oplus \dots \oplus m_n = (c_1, c_2, \dots, c_t)$. Since the output value of \mathcal{P} is determined by the opened values, the referee can compute the output value from the messages and the correctness of the obtained PSM protocol $\widehat{\mathcal{P}}$ can be satisfied.

The remaining thing we need to consider is the security of $\widehat{\mathcal{P}}$. See Figure 2. This protocol has three possible opened symbols c_1, c_2, c_3 corresponding to q_2, q_4, q_8 , respectively, but only two symbols are revealed in an execution as follows: If $c_1 = 0$, the left path (c_1, c_2) should be revealed; If $c_1 = 1$, the right path (c_1, c_3) should be revealed. Since any leakage of the values outside the path would compromise the security, the values outside the path must be hidden from the referee. To hide these values, some player can replace the i -th bit of the player's message with a random bit if he notices that c_i is not on the path.

The above idea is summarized as follows.

1. The common randomness consists of a sequence of permutations in all shuffle operations, an n -tuple of t -bit random numbers $(r_1, r_2, \dots, r_n) \in (\{0, 1\}^t)^n$ such that $r_1 \oplus r_2 \oplus \dots \oplus r_n = 0^t$, and other random bits to hide the values outside the path.
2. Each player P_i computes $m_i \in \{0, 1\}^t$ such that $m_1 \oplus m_2 \oplus \dots \oplus m_n$ equals the opened values (c_1, c_2, \dots, c_t) .
3. Each player P_i updates m_i to hide the values c_i outside the path.
4. Each player P_i sets $m_i \leftarrow m_i \oplus r_i$ and sends it to the referee.
5. On receiving m_1, m_2, \dots, m_n , the referee obtains the opened symbols on the path from $m_1 \oplus m_2 \oplus \dots \oplus m_n$, and determines an output value y based on simple protocol \mathcal{P} .

4.3 Our Conversion for Simple Protocols

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \text{in}, \text{out})$ be a simple protocol for $f: X^n \rightarrow Y$. We can assume that the protocol diagram of \mathcal{P} forms a tree.

For each $q \in Q_{\text{shuf}}$, let $(\text{shuf}, \Pi_q, \mathcal{F}_q)$ be the shuffle operation of q . As stated in Section 4.2, given a sequence of permutations $\vec{\pi} = (\pi_q)_{q \in Q_{\text{shuf}}}$ for $\pi_q \in \Pi_q$, the order of the sequence of cards in a protocol execution is determined. In particular, given a sequence of permutations $\vec{\pi}$ and an input $\vec{x} \in X^n$, the opened symbol for $q \in Q_{\text{turn}}^+$ is determined.

Fix a sequence of permutations $\vec{\pi}$ and an input $\vec{x} \in X^n$. We define the *responsibility* for $q \in Q_{\text{turn}}^+$ as follows. We say that the first player P_1 is responsible for q if the opened value at q is a constant bit or depends on $x_1 \in X$. We say that the i -th ($i \neq 1$) player P_i is responsible for q if the opened value at q is not a constant bit and depends on $x_i \in X$. We can observe that, for any $q \in Q_{\text{turn}}^+$, exactly one player P_i is responsible for q .

Let $c_q \in \{0, 1\}$ be the opened symbol at $q \in Q_{\text{turn}}^+$ when $\vec{\pi}$ and \vec{x} are given. We define a map $\text{val}_{i,q}$ for $i \in [n]$ and $q \in Q_{\text{turn}}^+$ as follows:

$$\text{val}_{i,q}(\vec{\pi}, x_i) := \begin{cases} c_q & \text{if } P_i \text{ is responsible for } q; \\ 0 & \text{otherwise.} \end{cases}$$

Since exactly one player is responsible for q , we have

$$\sum_{i=1}^n \text{val}_{i,q}(\vec{\pi}, x_i) = c_q,$$

i.e., the opened value c_q is computed by the players in a distributed manner.

For a state $q \in Q$, the *descendants* of q is recursively defined as follows: a child of q is a descendant of q , and a child of a descendant of q is a descendant of q . For a state $q \in Q$, we define $\text{descen}(q)$ as follows:

$$\text{descen}(q) := \{q' \in Q_{\text{turn}}^+ \mid q' \text{ is a descendant of } q\}.$$

72:12 Card-Based Protocols Imply PSM Protocols

For a branch state $q \in Q_{\text{branch}}$, let $q_b \in Q$ ($b \in \{0, 1\}$) be the child of q corresponding to the opened value b , and we define $\text{descen}(q, b)$ as follows:

$$\text{descen}(q, b) := \{q'_b\} \cup \text{descen}(q'_b),$$

where q'_b is either $q'_b = q_b$ if $q_b \in Q_{\text{turn}}^+$ or the nearest descendant of q_b in Q_{turn}^+ .

The resulting PSM protocol is given as follows, where $H(\mathcal{F}_q)$ denotes the Shannon entropy of the distribution \mathcal{F}_q for $q \in Q_{\text{shuf}}$ and $t := |Q_{\text{turn}}^+|$ denotes the opening complexity of \mathcal{P} .

The PSM protocol $\widehat{\mathcal{P}}$ from any simple protocol \mathcal{P}

Input: $\vec{x} = (x_i)_{i \in [n]} \in X^n$

Common Randomness: $(\vec{\pi}, \vec{r}, \vec{s})$ is generated as follows:

- For each $q \in Q_{\text{shuf}}$ whose operation is $(\text{shuf}, \Pi_q, \mathcal{F}_q)$, choose a permutation $\pi_q \in \Pi_q$ according to \mathcal{F}_q . Set $\vec{\pi} := (\pi_q)_{q \in Q_{\text{shuf}}}$.
- Choose $r_i = (r_i[q])_{q \in Q_{\text{turn}}^+} \in \{0, 1\}^t$ for $1 \leq i \leq n-1$ uniformly at random, and set $r_n := r_1 \oplus r_2 \oplus \dots \oplus r_{n-1}$. Set $\vec{r} := (r_i)_{i \in [n]}$.
- For each $q \in Q_{\text{branch}}$ and $q' \in \text{descen}(q)$, choose $s_{q'}^{(q)} \in \{0, 1\}$ uniformly at random. Set $\vec{s} := (s_{q'}^{(q)})_{q \in Q_{\text{branch}}, q' \in \text{descen}(q)}$.

Encoding Function: Output $m_i := (m_i[q])_{q \in Q_{\text{turn}}^+} \in \{0, 1\}^t$ as follows:

1. For each $q \in Q_{\text{turn}}^+$, set $m_i[q] \leftarrow \text{val}_{i,q}(\vec{\pi}, x_i)$.
2. For each $q \in Q_{\text{branch}}$, if the player i is responsible for q , do the following:
 - Let $c_q \in \{0, 1\}$ be the opened value at q .
 - For each $q' \in \text{descen}(q, \bar{c}_q)$, compute $m_i[q'] \leftarrow m_i[q'] \oplus s_{q'}^{(q)}$.
3. For each $q \in Q_{\text{turn}}^+$, set $m_i[q] \leftarrow m_i[q] \oplus r_i[q]$.

Decoding Function: Compute $m := m_1 \oplus m_2 \oplus \dots \oplus m_n$. Since $m = (m[q])_{q \in Q_{\text{turn}}^+}$ represents a path on the tree, the referee outputs the value corresponding to the path. **Communication Complexity:** nt **Randomness Complexity:** $(n-1)t + \sum_{q \in Q_{\text{branch}}} |\text{descen}(q)| + \sum_{q \in Q_{\text{shuf}}} H(\mathcal{F}_q)$

► **Theorem 4.** Let \mathcal{P} be a simple protocol with correctness and security for $f: X^n \rightarrow Y$ with opening complexity t . Then the above PSM protocol $\widehat{\mathcal{P}}$ is correct and secure for f with communication complexity nt .

Proof. Let \vec{x} be an input and $\vec{\pi}$ be all permutations in shuffle operations. Given \vec{x} and $\vec{\pi}$, let $Q_{\vec{x}, \vec{\pi}}$ be the set of all states $q \in Q_{\text{turn}}^+$ that actually appear during the execution. By the definition of encoding function in $\widehat{\mathcal{P}}$, we can observe that $m_i[q] = \text{val}_{i,q}(\vec{\pi}, x_i) \oplus r_i[q]$ if and only if $q \in Q_{\vec{x}, \vec{\pi}}$. For any $q \in Q_{\vec{x}, \vec{\pi}}$, we have

$$\begin{aligned} m[q] &= m_1[q] \oplus \dots \oplus m_n[q] \\ &= (\text{val}_{1,q}(\vec{\pi}, x_1) \oplus r_1[q]) \oplus \dots \oplus (\text{val}_{n,q}(\vec{\pi}, x_n) \oplus r_n[q]) \\ &= \text{val}_{1,q}(\vec{\pi}, x_1) \oplus \dots \oplus \text{val}_{n,q}(\vec{\pi}, x_n) \oplus r_1[q] \oplus r_2[q] \oplus \dots \oplus r_n[q]. \end{aligned}$$

By the choice of r_n in $\widehat{\mathcal{P}}$, we have $r_1[q] \oplus \dots \oplus r_n[q] = 0$ for any $q \in Q_{\text{turn}}^+$. Hence we have

$$m[q] = \text{val}_{1,q}(\vec{\pi}, x_1) \oplus \dots \oplus \text{val}_{n,q}(\vec{\pi}, x_n) = c_q,$$

where $c_q \in \{0, 1\}$ is the opened symbol at $q \in Q_{\text{turn}}^+$ of \mathcal{P} . Thus, the correctness of $\widehat{\mathcal{P}}$ follows from the correctness of \mathcal{P} .

We prove the security of $\widehat{\mathcal{P}}$ by constructing a simulator Sim for $\widehat{\mathcal{P}}$. Given an output value $y \in Y$, Sim first invokes a simulator for \mathcal{P} on input y , which outputs a visible sequence trace $\vec{v} \in (\text{Vis}^{\mathcal{D}})^*$ of \mathcal{P} . From \vec{v} , the path of the execution in the protocol tree of \mathcal{P} is determined. Let $c_1, c_2, \dots, c_k \in \{0, 1\}$ be the opened values in \vec{v} and $q_1, q_2, \dots, q_k \in Q_{\text{turn}}^+$ be the states corresponding to them. Define $m' = (m'[q])_{q \in Q_{\text{turn}}^+} \in \{0, 1\}^t$ as follows:

$$m'[q] := \begin{cases} c_q & \text{if } q \in \{q_1, q_2, \dots, q_k\}; \\ 0 & \text{otherwise.} \end{cases}$$

Then the simulator Sim chooses $r'_1, r'_2, \dots, r'_{n-1} \in \{0, 1\}^t$ uniformly at random and outputs $(r'_1, \dots, r'_{n-1}, r'_1 \oplus \dots \oplus r'_{n-1} \oplus m')$. For $q \in Q_{\text{turn}}^+$, the message $(m_1[q], \dots, m_n[q])$ is uniformly random conditioned on $\sum_{i=1}^n m_i[q] = c_q$ thanks to the random choice of $(r_1[q], \dots, r_n[q])$. For $q \notin Q_{\text{turn}}^+$, $(m_1[q], \dots, m_n[q])$ is uniformly random thanks to the random choice of \vec{s} . Hence the distribution of $\text{Sim}(y)$ equals the distribution of the messages, therefore $\widehat{\mathcal{P}}$ is secure. \blacktriangleleft

4.4 Extension of Our Conversion to Up-Down Cards

A deck of *up-down cards* [22] consists of two types of cards $\boxed{\uparrow}$ and $\boxed{\downarrow}$, which are transformed to each other by 180° rotation. The extension of our conversion to up-down cards is somewhat straightforward. The only difference is that random permutations $\pi_q \in \Pi_q$ in the common randomness are replaced with *extended permutations* defined below.

Let ρ be the 180° rotation operation, and define $\text{CMap} := \{\text{id}, \rho\}$ as the set of rotation operations. An *extended permutation* (over d cards) is defined by a pair of d rotation operations $(\rho_1, \dots, \rho_d) \in \text{CMap}^d$ and a permutation $\pi \in S_d$. For a sequence of d cards $\Gamma := (\alpha_1, \alpha_2, \dots, \alpha_d)$, we define an action of an extended permutation $\omega := ((\rho_1, \dots, \rho_d), \pi) \in \text{CMap}^d \times S_d$ by

$$\omega(\Gamma) := (\rho_1(\alpha_{\pi^{-1}(1)}), \dots, \rho_d(\alpha_{\pi^{-1}(d)})),$$

i.e., it first permutes the sequence of cards according to π and then applies the rotation operations. Here, by defining the operation “ \circ ” as

$$\omega \circ \omega' := ((\rho_1 \circ \rho'_{\pi^{-1}(1)}), \dots, \rho_d \circ \rho'_{\pi^{-1}(d)}), \pi \pi')$$

for $\omega = ((\rho_j)_{j \in [d]}, \pi)$, $\omega' = ((\rho'_j)_{j \in [d]}, \pi') \in \text{CMap}^d \times S_d$, the set $\text{CMap}^d \times S_d$ forms a monoid with $\text{id} := ((\text{id}, \dots, \text{id}), \text{id})$ being the identity element, which is so-called the wreath product $\text{CMap} \wr S_d$. In this extended model, perm and shuffle operations are extended to the set of extended permutations $\text{CMap} \wr S_d$ instead of S_d . Other definitions are the same as the Mizuki–Shizuya model.

Theorems 1 and 4 also hold for up-down cards. In particular, an SSFO protocol using d up-down cards implies a PSM protocol with communication complexity nd , and a simple protocol using up-down cards with opening complexity t implies a PSM protocol with communication complexity nt . In Section 5.2, we demonstrate our conversion for a card-based protocol using up-down cards.

5 Application

5.1 Lower Bounds of Card-Based Protocols

From our conversion method in Section 4, lower bounds on the opening complexity for card-based protocols are immediately implied to lower bounds on the communication complexity of PSM protocols. This is summarized by Corollary 5.

► **Corollary 5.** *Let ℓ be a lower bound on the communication complexity of PSM protocols for a function $f : X^n \rightarrow Y$. Then for any simple protocol computing f , the opening complexity t must satisfy $t \geq \ell/n$. In particular, for any SSFO protocol computing f , the number of cards d must satisfy $d \geq \ell/n$.*

In 1994, Feige, Kilian, and Naor [12] constructed a two-player PSM protocol for any function $f : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$ with communication complexity $O(2^k)$. After 20 years, Beimel, Ishai, Kumaresan, and Kushilevitz [6] improved it to $O(2^{k/2})$. For multi-player setting, in 2018, Beimel, Kushilevitz, and Nissim [7] constructed an n -player PSM protocol for any function $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$ with communication complexity $O(2^{kn/2})$. In 2021, Assouline and Liu [3] improved it to $O(2^{k(n-1)/2})$ for infinitely many n . Although the communication complexity has gradually improved, the communication complexity of general-purpose protocols is still exponential with respect to the total input length nk , and improving it to subexponential seems difficult at least based on the existing techniques. We summarize this observation in the following conjecture.

► **Conjecture 6.** *There exists no PSM protocol for $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$ with subexponential communication complexity $2^{o(kn)}$.*

From Conjecture 6, we obtain a lower bound on the opening complexity.

► **Corollary 7.** *Assume Conjecture 6 holds. Then there exists no simple protocol for $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$ with opening complexity $2^{o(kn)}$. In particular, there exists no SSFO protocol for $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$ with $2^{o(kn)}$ cards.*

Proof. If there exists a card-based protocol with $2^{o(kn)}$, there exists a PSM protocol with $n \cdot 2^{o(kn)} = 2^{o(kn)}$, contradicting to Conjecture 6. ◀

Of course, whether Corollary 7 holds is an open problem, and thus whether such a subexponential lower bound holds is not yet proven. However, we believe that the previous work of PSM protocols provides a piece of evidence that such a subexponential lower bound might hold in card-based cryptography.

One might imagine that a non-trivial lower bound for card-based protocols could be obtained from existing lower bounds for PSM protocols. In 2022, Ball and Randlph [5] showed a quadratic lower bound of PSM protocols based on the *modified Nečiporuk measure* $G^*(f)$ for a function $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$, which is a measure of function complexity. They proved that the communication complexity of PSM protocols computing $f : (\{0, 1\}^k)^n \rightarrow \{0, 1\}$ is greater than or equal to $G^*(f)/2$, and a random function f has $G^*(f) = \Omega(\frac{kn^2}{\log_2(kn)})$, implying a quadratic lower bound of PSM protocols.

Now we try to obtain a lower bound of card-based protocols from the Ball–Randlph’s lower bound. From Corollary 5, we obtain a lower bound $d = \Omega(\frac{kn}{\log_2(kn)})$ on the number of cards d of SSFO protocols, but this is not a strong bound because a card-based protocol naturally requires $\Omega(kn)$ cards for representing a kn -bit input. Unfortunately, since Ball and Randlph also showed that $G^*(f) \leq \frac{kn^2}{\log_2(kn)}$ for any f , we cannot obtain any better lower bound using the Nečiporuk measure. In order to obtain a non-trivial lower bound of card-based protocols, we need to obtain a super-quadratic lower bound $\omega(kn^2)$ of PSM protocols. As far as we know, no super-quadratic lower bound of PSM protocols has been proven so far, but our result provides a new motivation for obtaining such a lower bound.

5.2 PSM Protocol for Indirect Storage Access Function

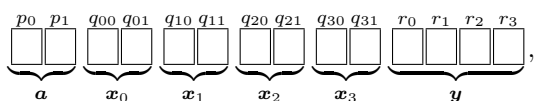
An *indirect storage access* (ISA, in short) function [23] is a function $f_{\text{ISA}}^{k,\ell} : \{0, 1\}^{k+\ell K+L} \rightarrow \{0, 1\}$, where $K = 2^k$ and $L = 2^\ell$, defined as follows:

$$f_{\text{ISA}}^{k,\ell}(\mathbf{a}, \mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{K-1}, \mathbf{y}) := y_{|\mathbf{x}_{|\mathbf{a}|}|},$$

where $\mathbf{a} = (a_0, \dots, a_{k-1}) \in \{0, 1\}^k$, $\mathbf{x}_j = (x_{j,0}, \dots, x_{j,\ell-1}) \in \{0, 1\}^\ell$ ($0 \leq j \leq K-1$), $\mathbf{y} = (y_{0\dots 0}, y_{0\dots 01}, \dots, y_{11\dots 1}) \in \{0, 1\}^L$, and for a bit string $b = (b_0, b_1, \dots, b_{t-1}) \in \{0, 1\}^t$, $|b|$ represents an integer $\sum_{i=0}^{t-1} b_i 2^i$.

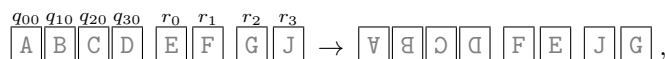
We describe our protocol for $f_{\text{ISA}}^{k,\ell}$ using up-down cards when $k = \ell = 2$. The general case follows similarly. The protocol proceeds as follows:

1. Arrange the input sequence as follows:



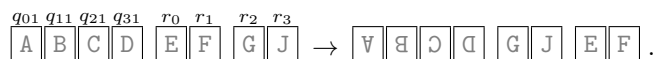
where p_i , $q_{i,j}$, and r_i are labels of positions.

2. Apply a shuffle ($\text{shuf}, \{\text{id}, \omega_0\}$) for ω_0 defined as follows:

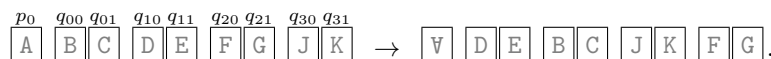


where A, B, ..., J are alphabets with no 180°-rotational symmetry. (Note that these characters are only used to represent the rotation and permutation, and are not actually written on the cards. In the following, we will use the same notation in the protocol.)

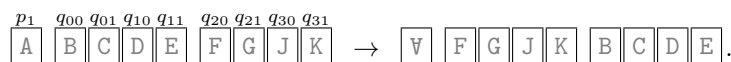
3. Apply a shuffle ($\text{shuf}, \{\text{id}, \omega_1\}$) for ω_1 defined as follows:



4. Apply a shuffle ($\text{shuf}, \{\text{id}, \omega'_0\}$) for ω'_0 defined as follows:



5. Apply a shuffle ($\text{shuf}, \{\text{id}, \omega'_1\}$) for ω'_1 defined as follows:



6. Open the first k cards. Let $\tilde{\mathbf{a}} \in \{0, 1\}^k$ be the k -tuple of the opened values.
7. Open the ℓ consecutive cards from the $(k+1+|\tilde{\mathbf{a}}|\ell)$ -th card. Let $\tilde{\mathbf{x}} \in \{0, 1\}^\ell$ be the ℓ -tuple of the opened values.
8. Open the $(k+\ell K+1+|\tilde{\mathbf{x}}|)$ -th card and output the opened value.

For the general case, since the number of opened cards is $k+\ell+1$, the opening complexity is $2^{k+\ell+1} = 2KL$. Thus the communication complexity of the resulting PSM protocol is $2KL(k+\ell K+L) = O(KL^2 + K^2L\ell)$. The communication complexity of Ishai–Kushilevitz's protocol for $f_{\text{ISA}}^{k,\ell}$ is $O(K^2L^3 + K^3L^2\ell)$ since $f_{\text{ISA}}^{k,\ell}$ can be computed by a BP of size $O(KL)$, and that of Kolesnikov's protocol for $f_{\text{ISA}}^{k,\ell}$ is $O(K^2L^2(k+\ell)^2)$ since $f_{\text{ISA}}^{k,\ell}$ can be computed by a formula of depth $2(k+\ell)$. Therefore, our protocol is more efficient than those protocols.

6 Conclusion

In this paper, we showed a generic conversion from card-based to PSM protocols. The significance of our conversion is to show a direct relationship from card-based to conventional cryptography for the first time, which was previously thought to be of little relevance.

Finally, we list open problems and future research directions as follows:

Deriving unconditional lower bounds for card-based protocols: An open problem is to derive unconditional lower bounds for card-based protocols from those of PSM protocols. To obtain such a lower bound for card-based protocols, it is sufficient to obtain a super-quadratic lower bound $\omega(kn^2)$ on the communication complexity of the PSM protocols for $f: (\{0, 1\}^k)^n \rightarrow \{0, 1\}$. However, as already mentioned in Section 5.1, a lower bound from the Nečiporuk measure will never be super-quadratic. Thus, it seems essential to develop new techniques to obtain super-quadratic lower bounds.

Constructing efficient PSM protocols: An open problem is to determine a class of functions for which our conversion yields efficient PSM protocols. Since the resulting PSM protocol has communication complexity linear to the opening complexity of the underlying protocol (or the number of cards for SSFO protocols), the following questions are worthy towards obtaining efficient PSM protocols with polynomial communication:

- What is the class of functions for which a simple card-based protocol exists with opening complexity polynomial in the input length?
- What is the class of functions for which an SSFO card-based protocol exists with polynomial number of cards in the input length?

Further relations among card-based and PSM protocols: An open problem is to develop deeper connections between card-based protocols and PSM or other kinds of conventional cryptographic protocols as follows:

- Is it possible to improve the efficiency (i.e., the communication complexity of the resulting PSM protocol) of our conversion?
- Can we establish a conversion in the opposite direction, i.e., from a certain subclass of PSM protocols back to card-based protocols?

References

- 1 Benny Applebaum, Barak Arkis, Pavel Raykov, and Prashant Nalini Vasudevan. Conditional disclosure of secrets: Amplification, closure, amortization, lower-bounds, and separations. *SIAM J. Comput.*, 50(1):32–67, 2021. doi:10.1137/18M1217097.
- 2 Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayevitz. The communication complexity of private simultaneous messages, revisited. *J. Cryptol.*, 33(3):917–953, 2020. doi:10.1007/S00145-019-09334-Y.
- 3 Léonard Assouline and Tianren Liu. Multi-party psm, revisited: Improved communication and unbalanced communication. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography – 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 194–223. Springer, 2021. doi:10.1007/978-3-030-90453-1_7.
- 4 Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: How friendly can a garbling-friendly PRF be? In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPICs*, pages 86:1–86:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.86.

- 5 Marshall Ball and Tim Randolph. A note on the complexity of private simultaneous messages with many parties. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPICs*, pages 7:1–7:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITC.2022.7.
- 6 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342. Springer, 2014. doi:10.1007/978-3-642-54242-8_14.
- 7 Amos Beimel, Eyal Kushilevitz, and Pnina Nissim. The complexity of multiparty PSM protocols and related models. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 287–318. Springer, 2018. doi:10.1007/978-3-319-78375-8_10.
- 8 Claude Crépeau and Joe Kilian. Discreet solitary games. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 1993. doi:10.1007/3-540-48329-2_27.
- 9 László Csirmaz. The size of a share must be large. *J. Cryptol.*, 10(4):223–231, 1997. doi:10.1007/S001459900029.
- 10 Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 2014. doi:10.1007/978-3-662-44381-1_12.
- 11 Bert den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 208–217. Springer, 1989. doi:10.1007/3-540-46885-4_23.
- 12 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994. doi:10.1145/195058.195408.
- 13 Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 485–502. Springer, 2015. doi:10.1007/978-3-662-48000-7_24.
- 14 Yuji Hashimoto, Koji Nuida, Kazumasa Shinagawa, Masaki Inamura, and Goichiro Hanaoka. Toward finite-runtime card-based protocol for generating a hidden random permutation without fixed points. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 101-A(9):1503–1511, 2018. doi:10.1587/TRANSFUN.E101.A.1503.
- 15 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings*, pages 174–184. IEEE Computer Society, 1997. doi:10.1109/ISTCS.1997.595170.

- 16 Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. The minimum number of cards in practical card-based protocols. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017 – 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 126–155. Springer, 2017. doi:10.1007/978-3-319-70700-6_5.
- 17 Alexander Koch. The landscape of optimal card-based protocols. *IACR Cryptol. ePrint Arch.*, page 951, 2018. URL: <https://eprint.iacr.org/2018/951>.
- 18 Alexander Koch, Michael Schrempf, and Michael Kirsten. Card-based cryptography meets formal verification. *New Gener. Comput.*, 39(1):115–158, 2021. doi:10.1007/S00354-020-00120-0.
- 19 Alexander Koch, Stefan Walzer, and Kevin Härtel. Card-based cryptographic protocols using a minimal number of cards. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015 – 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 – December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 783–807. Springer, 2015. doi:10.1007/978-3-662-48797-6_32.
- 20 Takaaki Mizuki, Michihito Kumamoto, and Hideaki Sone. The five-card trick can be done with four cards. In Xiaoyun Wang and Kazuo Sako, editors, *Advances in Cryptology – ASIACRYPT 2012 – 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 598–606. Springer, 2012. doi:10.1007/978-3-642-34961-4_36.
- 21 Takaaki Mizuki and Hiroki Shizuya. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Sec.*, 13(1):15–23, 2014. doi:10.1007/S10207-013-0219-4.
- 22 Takaaki Mizuki and Hiroki Shizuya. Practical card-based cryptography. In Alfredo Ferro, Fabrizio Luccio, and Peter Widmayer, editors, *Fun with Algorithms – 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, volume 8496 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2014. doi:10.1007/978-3-319-07890-8_27.
- 23 John E. Savage. *Models of computation – exploring the power of computing*. Addison-Wesley, 1998.

Dominating Set, Independent Set, Discrete k -Center, Dispersion, and Related Problems for Planar Points in Convex Position

Anastasiia Tkachenko  

Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Haitao Wang  

Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Abstract

Given a set P of n points in the plane, its unit-disk graph $G(P)$ is a graph with P as its vertex set such that two points of P are connected by an edge if their (Euclidean) distance is at most 1. We consider several classical problems on $G(P)$ in a special setting when points of P are in convex position. These problems are all NP-hard in the general case. We present efficient algorithms for these problems under the convex position assumption.

- For the problem of finding the smallest dominating set of $G(P)$, we present an $O(kn \log n)$ time algorithm, where k is the smallest dominating set size. We also consider the weighted case in which each point of P has a weight and the goal is to find a dominating set in $G(P)$ with minimum total weight; our algorithm runs in $O(n^3 \log^2 n)$ time. In particular, for a given k , our algorithm can compute in $O(kn^2 \log^2 n)$ time a minimum weight dominating set of size at most k (if it exists).
- For the discrete k -center problem, which is to find a subset of k points in P (called *centers*) for a given k , such that the maximum distance between any point in P and its nearest center is minimized. We present an algorithm that solves the problem in $O(\min\{n^{4/3} \log n + kn \log^2 n, k^2 n \log^2 n\})$ time, which is $O(n^2 \log^2 n)$ in the worst case when $k = \Theta(n)$. For comparison, the runtime of the current best algorithm for the continuous version of the problem where centers can be anywhere in the plane is $O(n^3 \log n)$.
- For the problem of finding a maximum independent set in $G(P)$, we give an algorithm of $O(n^{7/2})$ time and another randomized algorithm of $O(n^{37/11})$ expected time, which improve the previous best result of $O(n^6 \log n)$ time. Our algorithms can be extended to compute a maximum-weight independent set in $G(P)$ with the same time complexities when points of P have weights.
 - If we are looking for an (unweighted) independent set of size 3, we derive an algorithm of $O(n \log n)$ time; the previous best algorithm runs in $O(n^{4/3} \log^2 n)$ time (which works for the general case where points of P are not necessarily in convex position).
 - If points of P have weights and are not necessarily in convex position, we present an algorithm that can find a maximum-weight independent set of size 3 in $O(n^{5/3+\delta})$ time for an arbitrarily small constant $\delta > 0$. By slightly modifying the algorithm, a maximum-weight clique of size 3 can also be found within the same time complexity.
- For the dispersion problem, which is to find a subset of k points from P for a given k , such that the minimum pairwise distance of the points in the subset is maximized. We present an algorithm of $O(n^{7/2} \log n)$ time and another randomized algorithm of $O(n^{37/11} \log n)$ expected time, which improve the previous best result of $O(n^6)$ time.
 - If $k = 3$, we present an algorithm of $O(n \log^2 n)$ time and another randomized algorithm of $O(n \log n)$ expected time; the previous best algorithm runs in $O(n^{4/3} \log^2 n)$ time (which works for the general case where points of P are not necessarily in convex position).

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

Keywords and phrases Dominating set, k -center, geometric set cover, independent set, clique, vertex cover, unit-disk graphs, convex position, dispersion, maximally separated sets

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.73

Related Version *Full Version:* <http://arxiv.org/abs/2501.00207>



© Anastasiia Tkachenko and Haitao Wang;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 73; pp. 73:1–73:20



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Funding This research was supported in part by NSF under Grant CCF-2300356.

1 Introduction

Let P be a set of n points in the plane. The *unit-disk graph* of P , denoted by $G(P)$, is the graph with P as its vertex set such that two points are connected by an edge if their (Euclidean) distance is at most 1. Equivalently, $G(P)$ is the intersection graph of congruent disks with radius $1/2$ and centered at the points in P (i.e., two disks have an edge if they intersect). This model is particularly useful in applications such as wireless sensor networks, where connectivity is determined by signal ranges, represented by unit disks [6, 19, 42, 43].

1.1 Our results

We consider several classical problems on $G(P)$. These problems are all NP-hard. However, little attention has been given to special configurations of points, such as when the points are in convex position, despite the potential for significant algorithmic simplifications in such cases. In this paper, we systematically study these problems under the condition that the points of P are in convex position (i.e., every point of P appears as a vertex in the convex hull of P) and present efficient algorithms. We hope our results can lead to efficient solutions to other problems in this setting.

Dominating set. A *dominating set* of $G(P)$ is a subset S of vertices of $G(P)$ such that each vertex of $G(P)$ is either in S or adjacent to a vertex in S . The dominating set problem, which seeks a dominating set of the smallest size, is a classical NP-hard problem [19, 29, 31]. In the weighted case, each point of P has a weight and the problem is to find a dominating set of minimum total weight. The dominating set problem has been widely studied, with various approximation algorithms proposed [23, 27, 41, 55].

To the best of our knowledge, we are not aware of any previous work under the convex position assumption. For the unweighted case, we present an algorithm of $O(kn \log n)$ time, where k is the smallest dominating set size of $G(P)$. For the weighted case, we derive an algorithm of $O(n^3 \log^2 n)$ time. In particular, given any k , our algorithm can compute in $O(kn^2 \log^2 n)$ time a minimum-weight dominating set of size at most k .

Discrete k -center. A closely related problem is the *discrete k -center* problem. Given a number k , the problem is to compute a subset of k points in P (called *centers*) such that the maximum distance between any point in P and its nearest center is minimized. The problem, which is NP-hard [49], is also a classical problem with applications in clustering, facility locations, and network design. An algorithm for the dominating set problem can be used to solve the *decision version* of the discrete k -center problem: Given a value r and k , decide whether there exists a subset of k centers such that the distance from any point in P to its nearest center is at most r . Indeed, if we define the unit-disk graph of P with respect to r , then a dominating set of size k in the graph is a discrete k -center of P for r , and vice versa.

For the convex position case, we are not aware of any previous work. We propose an algorithm of $O(\min\{n^{4/3} \log n + kn \log^2 n, k^2 n \log^2 n\})$ time.

Independent set. An *independent set* of $G(P)$ is a subset of vertices such that no two vertices have an edge. The *maximum independent set* problem is to find an independent set of the largest cardinality. The problem of finding a maximum independent set in $G(P)$ is NP-hard [19]. Many approximation algorithms for the problem have been developed in the literature, e.g., [21, 22, 37, 38].

Under the convex position assumption, using the technique of Singireddy, Basappa, and Mitchell [47] for a dispersion problem (more details to be discussed later), one can find a maximum independent set in $G(P)$ in $O(n^6 \log n)$ time. We give a new algorithm of $O(n^{7/2})$ time,¹ and another randomized algorithm of $O(n^{37/11})$ expected time using the recent randomized result of Agarwal, Ezra, and Sharir [1]. Furthermore, our algorithm can be extended to compute a maximum-weight independent set of $G(P)$ within the same time complexity when points of P have weights; specifically, a *maximum-weight independent set* is an independent set whose total vertex weight is maximized. Since the vertices of a graph excluding an independent set form a vertex cover, our algorithm also computes a minimum-weight vertex cover of $G(P)$ in $O(n^{7/2})$ time or in randomized $O(n^{37/11})$ expected time.

Furthermore, we consider a small-size case that is to find an (unweighted) independent set of size 3 in $G(P)$. If P is not necessarily in convex position, the problem has been studied by Agarwal, Overmars, and Sharir [2], who presented an $O(n^{4/3} \log^2 n)$ time algorithm. We consider the convex position case and derive an algorithm of $O(n \log n)$ time. Note that finding an independent set of size 2 is equivalent to computing a farthest pair of points of P , which can be done in $O(n \log n)$ time using the farthest Voronoi diagram [45].

In addition, we consider a more general small-size case that is to find a maximum-weight independent set of size 3 in $G(P)$ when points of P have weights and are not necessarily in convex position. Our algorithm runs in $O(n^{5/3+\delta})$ time; δ refers to an arbitrarily small positive constant. Our technique can also be used to find a maximum-weight clique of size 3 in $G(P)$ within the same time complexity. In addition, we show that a maximum-weight independent set or clique of size 2 can be found in $n^{4/3} 2^{O(\log^* n)}$ time. All these algorithms also work for computing the minimum-weight independent set or clique. We are not aware of any previous work on these weighted problems. As mentioned above, the problem of finding an (unweighted) independent set of size 3 can be solved in $O(n^{4/3} \log^2 n)$ time [2]. It is also known that finding an (unweighted) clique of size 3 in a disk graph (not necessarily unit-disk graph) can be done in $O(n \log n)$ time [30].

The dispersion problem. A related problem is the dispersion problem (also called *maximally separated set problem* [2]). Given P and a number k , we wish to find a subset of k points from P so that their minimum pairwise distance is maximized. The problem is NP-hard [50]. An algorithm for the independent set problem of $G(P)$ can be used as a decision algorithm for the dispersion problem: Given a value r , we can decide whether P has a subset of k points whose minimum pairwise distance is larger than r using the independent set algorithm (i.e., by defining an edge for two points in the graph if their distance is at most r).

Under the convex position assumption, Singireddy, Basappa, and Mitchell [47] previously gave an $O(n^4 k^2)$ time algorithm for the problem. Using our independent set algorithm as a decision procedure and doing binary search among the interpoint distances of P , we present a new algorithm that can solve the problem in $O(n^{7/2} \log n)$ time, or in randomized $O(n^{37/11} \log n)$ expected time. For a special case where $k = 3$, the algorithm of [2] solves the problem in $O(n^{4/3} \log^3 n)$ time even if the points of P are not in convex position. Our new algorithm, which works on the convex-position case only, runs in $O(n \log^2 n)$ time. This is achieved using parametric search [20, 39] with our independent set algorithm as a decision algorithm. In addition, with our decision algorithm and Chan's randomized technique [11], we can obtain a randomized algorithm of $O(n \log n)$ expected time. We note that a recent work [35] proposed another algorithm of $O(n^2)$ time, apparently unaware of the result in [2].

¹ Throughout the paper, the algorithm runtime is deterministic unless otherwise stated.

1.2 Related work

Unit-disk graphs are a fundamental model in wireless networks, particularly where coverage and connectivity are governed by proximity [6, 19, 42, 43]. However, many classical graph problems, including coloring, vertex cover, independent set, and dominating set, remain NP-hard even when restricted to unit-disk graphs [19]. One exception is that finding a maximum clique in a unit-disk graph can be done in polynomial time [19, 25, 26] and the current best algorithm runs in $O(n^{2.5} \log n)$ time [26] (see [34] for a comment about improving the runtime to $O(n^{7/3+o(1)})$).

The assumption that points are in convex position can simplify certain problems that are otherwise NP-hard for general point sets in the plane. This has motivated the exploration of other computational problems under similar assumptions. For example, the *continuous k -center* problem where centers can be anywhere in the plane is NP-hard for arbitrary points but become polynomial time solvable under the convex position assumption [18]. The convex position constraint was even considered for classical problems that are already polynomial time solvable in the general case. For instance, Aggarwal, Guibas, Saxe, and Shor [4] gave a renowned linear time algorithm for computing the Voronoi diagram for a set of planar points in convex position. Refer to [15, 36, 44] for more work for points in convex position.

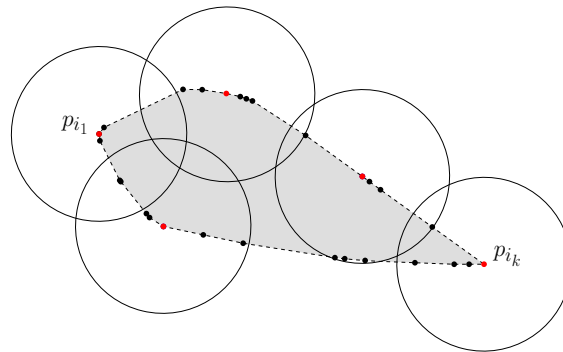
The k -center problem under a variety of constraints has received much attention. Particularly, when k , the number of centers, is two and the centers can be anywhere in the plane (referred to as the *continuous 2-center problem*), several near-linear time algorithms have been developed [12, 24, 46, 51], culminating in an optimal $O(n \log n)$ time [17]. The problem variations under other constraints were also considered. For example, the k -center problem can be solved in $O(n \log n)$ time if centers are required to lie on the same line [16, 53] or two lines [10]. The continuous one-center problem is the classical smallest enclosing circle problem and can be solved in linear time [40].

For the convex position case of the continuous k -center problem, Choi, Lee, and Ahn [18] proposed an $O(\min\{k, \log n\} \cdot n^2 \log n + k^2 n \log n)$ time algorithm. For comparison, the worst-case runtime of their algorithm is cubic, while our discrete k -center algorithm runs in near quadratic time.

The discrete 2-center problem also gets considerable attention. Agarwal, Sharir, and Welzl gave the first subquadratic $O(n^{4/3} \log^5 n)$ time algorithm [3]; the logarithmic factor was slightly improved by Wang [52]. As the continuous two-center problem can be solved in $O(n \log n)$ time [17] while the current best discrete two-center algorithm runs in $\Omega(n^{4/3})$ time [3, 52], the discrete problem appears more challenging than the continuous counterpart. This makes our discrete k -center algorithm even more interesting because it is almost a linear factor faster than the continuous k -center algorithm in [18]. Therefore, it is an intriguing question whether the algorithm in [18] can be further improved.

Other variations of the discrete k -center problem for small k were recently studied by Chan, He, and Yu [13], improving over previous results [8, 9, 32].

The dispersion problem and some of its variants have also been studied before. The general planar dispersion problem can be solved by an exact algorithm in $n^{O(\sqrt{k})}$ time [2]. If all points of P lie on a single line, Araki and Nakano [5] gave an algorithm of $O((2k^2)^k \cdot n)$ time (assuming that the points are not given sorted), which is $O(n)$ for a constant k . For a circular case where all points of P lie on a circle and the distance between two points is measured by their distance along the circle, the problem is solvable in $O(n)$ time [48], provided that the points are given sorted along the circle. We note that this implies that the line case problem, which can be viewed as a special case of the circular case, is also solvable in $O(n)$ time after the points are sorted on the line.



■ **Figure 1** Illustrating the ordering property of S (the centers of the disks).

1.3 Our approach

The weighted dominating set problem reduces to the following problem: Given any k , find a minimum weight dominating set of size at most k . This is equivalent to finding a minimum weight subset of at most k points of P such that the union of the unit disks centered at these points covers P . Let S be an optimal solution for the problem (points of S are called *centers*). If we consider P as a cyclic list of points along the convex hull of P , then for each center $p \in S$, its unit disk D_p may cover multiple maximal contiguous subsequences (called *sublists*) of P . We prove that it is possible to assign at most two such sublists to each center $p \in S$ such that (1) p belongs to at least one of these sublists; (2) the union of the sublists assigned to all centers is P ; (3) for every two centers $p_i, p_j \in S$, the sublists of the points assigned to p_i can be separated by a line from the sublists assigned to p_j . Using these properties, we further obtain the following structural property (called *ordering property*; see Figure 1) about the optimal solution S : There exists an ordering of the centers of S as $p_{i_1}, p_{i_2}, \dots, p_{i_k}$ such that (1) p_{i_1} (resp., p_{i_k}) is only assigned one sublist; (2) if a center p_{i_j} , $1 < j < k$, is assigned two sublists, then one of them is on P_1 , the portion of P from p_{i_1} to p_{i_k} clockwise, and the other is on P_2 , the portion of P from p_{i_1} to p_{i_k} counterclockwise; (3) the order of the centers of the sublists along P_1 (resp., P_2) from p_{i_1} to p_{i_k} is a (not necessarily contiguous) subsequence of the above ordering.

The above ordering property is crucial to the success of our method. Using the property, we develop a dynamic programming algorithm of $O(kn^2 \log^2 n)$ time. Setting $k = n$ leads to an $O(n^3 \log^2 n)$ time algorithm for the original weighted dominating set problem. These properties are also applicable to the unweighted case, which is essentially a special case of the weighted problem. Using an additional greedy strategy, the runtime of the algorithm can be improved by roughly a linear factor for the unweighted case.

To solve the discrete k -center problem, we use the algorithm for the unweighted dominating set problem as the decision problem: Given any value r , determine whether $r \geq r^*$, where r^* is the optimal objective value, i.e., the minimum value for which there exist k centers such that the maximum distance from any point of P to its closest center is at most r^* . Observe that r^* must be equal to the distance of two points of P . As such, by doing binary search on the pairwise distances of points of P and applying the distance selection framework in [54] with our unweighted dominating set algorithm, we can compute r^* in $O(n^{4/3} \log n + kn \log^2 n)$ time. Furthermore, using parametric search [20, 39], we develop another algorithm of $O(k^2 n \log^2 n)$ time, which is faster than the first algorithm when $k = o(n^{1/6} / \sqrt{\log n})$.

For the independent set problem, our algorithm is a dynamic program, which is in turn based on the observation that the Voronoi diagram of a set of points in convex position forms a tree [4]. The (unweighted) size-3 case is solved by new observations and developing

efficient data structures. As discussed above, we tackle the dispersion problem by using the independent set algorithm as a decision procedure. For computing a maximum-weight independent set of size 3 for points in arbitrary position, our algorithm relies on certain interesting observations and a *tree-structured biclique partition* of P . Biclique partition has been studied before, e.g., [33, 54]. However, to our best knowledge, tree-structured biclique partitions have never been introduced before. Our result may find applications elsewhere.

Outline. The rest of the paper is organized as follows. After introducing notation in Section 2, we present our algorithms for the dominating set, the discrete k -center, the independent set, and the dispersion problems for points in convex position in Sections 3, 4, 5 and 6, respectively. As the only problem for points in arbitrary position studied in this paper, the size-3 weighted independent set problem is discussed in Section 7. Due to the space limit, many details and proofs are omitted but can be found in the full paper.

2 Preliminaries

We introduce some notations that will be used throughout the paper, in addition to those already defined in Section 1, e.g., P , n , $G(P)$.

A *unit disk* refers to a disk with radius 1; the boundary of a unit disk is a *unit circle*. For any point p in the plane, we use D_p to denote the unit disk centered at p . For any two points p and q in the plane, we use $|pq|$ to denote their (Euclidean) distance and use \overline{pq} to denote the line segment connecting them. Let \vec{pq} to denote the directed segment from p to q .

Let $\mathcal{H}(P)$ be the convex hull of P . If the points in P are in convex position, then we can consider P as a cyclic sequence. Specifically, let $P = \langle p_1, p_2, \dots, p_n \rangle$ represent a cyclic list of the points ordered counterclockwise along $\mathcal{H}(P)$. We use a *sublist* to refer to a contiguous subsequence of P . Multiple sublists are said to be *consecutive* if their concatenation is also a sublist. For any two points p_i and p_j in P , we define $P[i, j]$ as the sublist of P from p_i counterclockwise to p_j , inclusive, i.e., if $i \leq j$, then $P[i, j] = \langle p_i, p_{i+1}, \dots, p_j \rangle$; otherwise, $P[i, j] = \langle p_i, p_{i+1}, \dots, p_n, p_1, \dots, p_j \rangle$. We also denote by $P(i, j)$ the sublist $P[i, j]$ excluding p_i , and similarly for other variations, e.g., $P(i, j)$ and $P(i, j)$.

For simplicity of the discussion, we make a general position assumption that no three points of P are collinear and no four points lie on the same circle. This assumption is made without loss of generality as degenerate cases can be handled through perturbations.

3 The dominating set problem

In this section, we present our algorithms for the dominating set problem on a set P of n points in convex position. The weighted and unweighted cases are discussed in Sections 3.2 and 3.3, respectively. We first prove in Section 3.1 the structural properties that our algorithms rely on and then present these algorithms.

3.1 Structural properties

We examine the structural properties of the dominating sets in the unit-disk graph $G(P)$ for the weighted case, which are also applicable to the unweighted case.

Let \mathcal{A} represent a partition of P into consecutive, nonempty, and disjoint sublists. Suppose $S \subseteq P$ is a dominating set of $G(P)$; points of S are called *centers*. It is not difficult to see that the union of the collection \mathcal{D} of unit disks centered at the points in S covers P .

We say that a collection \mathcal{D} of unit disks *covers* \mathcal{A} if every sublist $\alpha \in \mathcal{A}$ is covered by at least one disk from \mathcal{D} . An *assignment* $\phi : \mathcal{A} \rightarrow S$ is a mapping from sublists in \mathcal{A} to points in S , such that each sublist α is assigned to exactly one center $p_i \in S$ with $\alpha \subseteq D_{p_i}$. For each $p_i \in S$, we define G_{p_i} as the set of points in the sublists $\alpha \in \mathcal{A}$ that are assigned to p_i ; G_{p_i} is called the *group* of p_i . Depending on the context, G_{p_i} may also represent the collection of sublists assigned to p_i . By definition, the groups of two centers of S are disjoint.

An assignment ϕ is said to be *line separable* if, for every two groups of ϕ , there exists a line ℓ that separates the points from the two groups, that is, the points of one group lie on one side of ℓ or on ℓ while those of the other group lie strictly on the other side of ℓ .

As discussed in Section 1.3, our main target is to prove the ordering property. This is achieved by proving a series of lemmas. We start with the lemma that proves a line separable property.

► **Lemma 1.** *Let S be a dominating set of $G(P)$. There exist a partition \mathcal{A} of P and a line-separable assignment $\phi : \mathcal{A} \rightarrow S$ such that for any center $p_i \in S$, $p_i \in G_{p_i}$, meaning that a sublist of p_i contains p_i .*

For the assignment ϕ from Lemma 1, for each center $p_i \in S$, we refer to the sublist of p_i that contains p_i as the *main sublist* of p_i .

► **Lemma 2.** *Let S be a dominating set for $G(P)$. Then there exist a partition \mathcal{A} of P and an assignment $\phi : \mathcal{A} \rightarrow S$ with the following properties: (1) ϕ is line separable; (2) each center of S is assigned at most two sublists, one of which is a main sublist.*

► **Lemma 3.** *Let S be an optimal dominating set and $\phi : \mathcal{A} \rightarrow S$ be the assignment given by Lemma 2. There exists a pair of centers (p_i, p_j) in S , called a decoupling pair, such that the following hold: (1) each of p_i and p_j has only one sublist; (2) for any center S that has two sublists, one sublist is in $P(i, j)$ while the other is in $P(j, i)$.*

Let S be an optimal dominating set and $\phi : \mathcal{A} \rightarrow S$ be the assignment given by Lemma 2. Let (p_i, p_j) be a decoupling pair from Lemma 3. For any center of $S \setminus \{p_i, p_j\}$, each of its sublist must be either entirely in $P(i, j)$ or in $P(j, i)$, and by Lemma 3, the center has at most one sublist in $P(i, j)$ and at most one sublist in $P(j, i)$. We finally prove in the following lemma the ordering property discussed in Section 1.3.

► **Lemma 4.** (The ordering property) *Let S be an optimal dominating set and $\phi : \mathcal{A} \rightarrow S$ be the assignment given by Lemma 2. Let (p_i, p_j) be a decoupling pair from Lemma 3. Then, there exists an ordering of all centers of S as $p_{i_1}, p_{i_2}, \dots, p_{i_k}$ with $k = |S|$ such that (see Figure 1)*

1. $p_i = p_{i_1}$ and $p_j = p_{i_k}$, i.e., p_{i_1} and p_{i_k} are the first and last points in the ordering, respectively.
2. The sequence of the centers of S that have at least one sublist in $P[i, j]$ (resp., $P[j, i]$) ordered by the points of their sublists appearing in $P[i, j]$ (resp., $P[j, i]$) from p_i to p_j is a (not necessarily contiguous) subsequence of the ordering.
3. For any t , $1 \leq t \leq k$, the sublists of the first t centers in the ordering are consecutive (i.e., their union, which is $\bigcup_{l=1}^t G_{i_l}$, is a sublist of P).
4. For any t , $2 \leq t \leq k$, $\bigcup_{l=1}^{t-1} G_{i_l} \subseteq \bigcup_{l=1}^t G_{i_l}$.
5. For any t , $1 \leq t \leq k$, each sublist of p_{i_t} appears at one end of $\bigcup_{l=1}^t G_{i_l}$ (if $t = k$, then $\bigcup_{l=1}^t G_{i_l}$ becomes the cyclic list of P ; for convenience, we view $\bigcup_{l=1}^t G_{i_l}$ as a list by cutting it right after the clockwise endpoint of G_{i_k}).

Proof. First of all, notice that the first two properties imply the last three. Therefore, it suffices to prove the first two properties.

Let S_1 (resp., S_2) be the subset of centers of $S \setminus \{p_i, p_j\}$ that have a sublist in $P(i, j)$ (resp., $P(j, i)$). By Lemma 3, each center of S_1 has at most one sublist in $P(i, j)$ and each center of S_2 has at most one sublist in $P(j, i)$. We add p_i and p_j to both S_1 and S_2 . We sort all centers of S_1 as a sequence (called *the sorted sequence* of S_1) by the points of their sublists appearing in $P[i, j]$ from p_i to p_j (and thus p_i is the first center and p_j is the last one in the sequence). Similarly, we sort all centers of S_2 as a sequence (called *the sorted sequence* of S_2) by the points of their sublists appearing in $P[j, i]$ from p_i to p_j (and thus p_i is the first center and p_j is the last one in the sequence). To prove the first two properties of the lemma, it suffices to show the following *statement*: There exists an ordering of S such that (1) p_i is the first one in the ordering and p_j is the last one; (2) the sorted sequence of S_1 (resp., S_2) is a subsequence of the ordering.

We say that two centers $p_{j_1}, p_{j_2} \in S$ are *conflicting* if p_{j_1} appears in front of p_{j_2} in the sorted sequence of S_1 while p_{j_2} appears in front of p_{j_1} in the sorted sequence of S_2 . It is not difficult to see that if no two centers of S are conflicting then the above statement holds. Assume to the contrary that there exist two centers $p_{j_1}, p_{j_2} \in S$ that are conflicting. Then, since the two centers are in both S_1 and S_2 , each of them has two sublists. Since they are conflicting, by the definition of the sorted sequences of S_1 and S_2 and due to the convexity of P , the group of p_{j_1} cannot be separated from the group of p_{j_2} by a line, a contradiction with the line separable property of ϕ . \blacktriangleleft

3.2 The weighted dominating set problem

For each point $p_i \in P$, let w_i denote its weight. We assume that each $w_i > 0$, since otherwise p_i could always be included in the solution. For any subset $S \subseteq P$, let $w(S)$ denote the total weight of all points of S . We mainly consider the following *bounded size problem*: Given a number k , compute a dominating set S of minimum total weight with $|S| \leq k$ in the unit-disk graph $G(P)$. If we have an algorithm for this problem, then applying the algorithm with $k = n$ can compute a minimum weight dominating set for $G(P)$. Let S^* denote an optimal dominating set for the above bounded size problem. Define $W^* = w(S^*)$.

In what follows, we first describe the algorithm and then discuss how to implement the algorithm efficiently.

Algorithm description. We begin by introducing the following definition.

► **Definition 5.** For two points $p_i, p_j \in P$ ($p_i = p_j$ is possible), define a_i^j as the index of the first point p of P counterclockwise from p_j such that $|p_i p| > 1$, and b_i^j the index of the first point p of P clockwise from p_j such that $|p_i p| > 1$ (if $|p_i p_j| > 1$, then $a_i^j = b_i^j = j$). If $|p_i p| \leq 1$ for all points $p \in P$, then let $a_i^j = b_i^j = 0$.

For a subset $P' \subseteq P$, let $\mathcal{D}(P')$ denote the union of the unit disks centered at the points of P' . Note that a subset $S \subseteq P$ is a dominating set if and only if $P \subseteq \mathcal{D}(S)$.

Our algorithm has k iterations. In each t -th iteration with $1 \leq t \leq k$, we compute a set \mathcal{L}_t of $O(n^2)$ sublists of P , and each sublist $L \in \mathcal{L}_t$ is associated with a weight $w'(L)$ and a set $S_L \subseteq P$ of at most t points. Our algorithm maintains the following invariant: For each sublist $L \in \mathcal{L}_t$, $w(S_L) \leq w'(L)$ and points of L are all covered by $\mathcal{D}(S_L)$. Suppose that there exists a set $S \subseteq P$ of k points such that $P \subseteq \mathcal{D}(S)$. Then we will show that \mathcal{L}_k contains a sublist L that is P and $w'(L) \leq W^*$. As such, after k iterations, we only need to find all sublists of \mathcal{L}_k that are P and then return the one with the minimum weight.

In the first iteration, for each point $p_i \in P$, we compute the two indices a_i^i and b_i^i ; we show in Lemma 7 that this can be done in $O(\log n)$ time after $O(n \log n)$ time preprocessing. Then, let $\mathcal{L}_1 = \{P(b_i^i, a_i^i) \mid p_i \in P\}$. For each sublist $L = P(b_i^i, a_i^i)$ of \mathcal{L}_1 , we set $S_L = \{p_i\}$ and $w'(L) = w_i$. Clearly, the algorithm invariant holds on all sublists of \mathcal{L}_1 . This finishes the first iteration. Although $|\mathcal{L}_1| = O(n)$, as will be seen next, $|\mathcal{L}_t| = O(n^2)$ for all $t \geq 2$.

In general, suppose that we have a set \mathcal{L}_{t-1} of $O(n^2)$ sublists and each sublist $L \in \mathcal{L}_{t-1}$ is associated with a weight $w'(L)$ and a set $S_L \subseteq P$ of at most $t-1$ points such that the algorithm invariant holds, i.e., $w(S_L) \leq w'(L)$ and points of L are all covered by $\mathcal{D}(S_L)$. We now describe the t -th iteration of the algorithm.

For each point $p_i \in P$, we perform a *counterclockwise processing procedure* as follows. For each point $p_j \in P$, we do the following. Compute the minimum weight sublist from \mathcal{L}_{t-1} that contains $P[a_i^i, j]$; we call this step a *minimum-weight enclosing sublist query*. We show later that each such query can be answered in $O(\log^2 n)$ time after $O(n^2 \log n)$ time preprocessing on the sublists of \mathcal{L}_{t-1} . Let $P[j_{i1}, j_{i2}]$ be the sublist computed above. Then, we compute the index $a_i^{j_{i2}+1}$. By definition, the union of the following three sublists is a sublist of P : $P(b_i^i, a_i^i)$, $P[j_{i1}, j_{i2}]$, and $P(j_{i2}, a_i^{j_{i2}+1})$; denote by L the sublist. We set $S_L = S_{L'} \cup \{p_i\}$ and $w'(L) = w'(L') + w_i$, where $L' = P[j_{i1}, j_{i2}]$. We add L to \mathcal{L}_t . We next argue that the algorithm invariant holds for L , i.e., points of L are covered by $\mathcal{D}(S_L)$, $|S_L| \leq t$, and $w(S_L) \leq w'(L)$. Indeed, by definition, all the points of $P(b_i^i, a_i^i) \cup P(j_{i2}, a_i^{j_{i2}+1})$ are covered by the disk D_{p_i} . Since the sublist L' is from \mathcal{L}_{t-1} , by our algorithm invariant, L' is covered by $\mathcal{D}(S_{L'})$, $|S_{L'}| \leq t-1$, and $w(S_{L'}) \leq w'(L')$. Therefore, L is covered by $\mathcal{D}(S_{L'} \cup \{p_i\})$ and $|S_L| \leq t$. In addition, we have $w(S_L) \leq w(S_{L'}) + w_i \leq w'(L') + w_i = w'(L)$. As such, the algorithm invariant holds on L .

The above counterclockwise processing procedure for p_i will add $O(n)$ sublists to \mathcal{L}_t . Symmetrically, we perform a *clockwise processing procedure* for p_i , which will also add $O(n)$ sublists to \mathcal{L}_t . We briefly discuss it. Given $p_i \in P$, for each point $p_j \in P$, we compute the minimum weight sublist from \mathcal{L}_{t-1} that contains $P[j, b_i^i]$. Let $P[j_{i3}, j_{i4}]$ be the sublist computed above. Then, we compute the index $b_i^{j_{i3}-1}$. Let L be the sublist that is the union of the following three sublists: $P(b_i^i, a_i^i)$, $P[j_{i3}, j_{i4}]$, and $P(b_i^{j_{i3}-1}, j_{i3})$. We let $S_L = S_{L'} \cup \{p_i\}$ and $w'(L) = w'(L') + w_i$, where $L' = P[j_{i3}, j_{i4}]$. As above, the algorithm invariant holds on L . We add L to \mathcal{L}_t . In this way, the t -th iteration computes $O(n^2)$ sublists in \mathcal{L}_t .

After the k -th iteration, we find from all sublists of \mathcal{L}_k that are P the one L^* whose weight $w'(L^*)$ is the minimum. Based on the ordering property in Lemma 4, the next lemma shows that S_{L^*} is an optimal dominating set.

► **Lemma 6.** S_{L^*} is an optimal dominating set and $W^* = w'(L^*)$.

Time analysis. In each iteration, we perform $O(n^2)$ operations for computing indices a_i^j and b_i^j , and perform $O(n^2)$ minimum-weight enclosing sublist queries. We show later that each query takes $O(\log^2 n)$ time after $O(n^2 \log n)$ time preprocessing. As such, each iteration of the algorithm takes $O(n^2 \log^2 n)$ time and the total time of the algorithm is thus $O(kn^2 \log^2 n)$.

Algorithm implementation. The following lemma provides a data structure for computing the indices a_i^j and b_i^j .

► **Lemma 7.** We can construct a data structure for P in $O(n \log n)$ time such that the indices a_i^j and b_i^j can be computed in $O(\log n)$ time for any two points $p_i, p_j \in P$.

Given a set \mathcal{L} of m sublists of P , each sublist has a weight. We wish to build a data structure to answer the following minimum-weight enclosing sublist queries: Given a sublist L , compute the minimum weight sublist of \mathcal{L} that contains L . We have the following lemma.

► **Lemma 8.** *We can construct a data structure for \mathcal{L} in $O(m \log m)$ time, with $m = |\mathcal{L}|$, so that each minimum-weight enclosing sublist query can be answered in $O(\log^2 m)$ time.*

Theorem 9 summarizes our result. Applying Theorem 9 with $k = n$ leads to Corollary 10.

► **Theorem 9.** *Given a number k and a set P of n weighted points in convex position in the plane, we can find in $O(kn^2 \log^2 n)$ time a minimum-weight dominating set of size at most k in the unit-disk graph $G(P)$, or report no such dominating set exists.*

► **Corollary 10.** *Given a set P of n weighted points in convex position in the plane, we can compute a minimum-weight dominating set in the unit-disk graph $G(P)$ in $O(n^3 \log^2 n)$ time.*

3.3 The unweighted case

In this section, we consider the unweighted dominating set problem. The goal is to compute the smallest dominating set in the unit-disk graph $G(P)$. Note that all properties for the weighted case are also applicable here. In particular, by setting all point weights to 1 and applying Theorem 9, one can solve the unweighted problem in $O(n^3 \log^2 n)$ time. We provide an improved algorithm of $O(kn \log n)$ time, where k is the smallest dominating set size.

Algorithm description. We follow the iterative algorithmic scheme of the weighted case, but incorporate a greedy strategy using the property that all points of P have the same weight.

In each t -th iteration of the algorithm, $t \geq 1$, we compute a set \mathcal{L}_t of $O(n)$ sublists and each list $L \in \mathcal{L}_t$ is associated with a set $S_L \subseteq P$ of at most t points. Our algorithm maintains the following invariant: For each sublist $L \in \mathcal{L}_t$, all points of L are covered by $\mathcal{D}(S_L)$, i.e., the union of the unit disks centered at the points of S_L . If k is the smallest dominating set size, we show in Lemma 11 that after k iterations, \mathcal{L}_k is guaranteed to contain a sublist that is P . Thus, we can stop the algorithm as soon as the first sublist that is P is computed.

Initially, we compute the indices a_i^i and b_i^i for all points $p_i \in P$. By Lemma 7, this takes $O(\log n)$ time after $O(n \log n)$ time preprocessing. In the first iteration, we have $\mathcal{L}_1 = \{P(b_i^i, a_i^i) \mid p_i \in P\}$. For each sublist $L = P(b_i^i, a_i^i) \in \mathcal{L}_1$, we set $S_L = \{p_i\}$. Clearly, the algorithm invariant holds.

In general, suppose that we have a set \mathcal{L}_{t-1} of $O(n)$ sublists such that the algorithm invariant holds. We assume that no sublist of \mathcal{L}_{t-1} is P . Then, the t -th iteration of the algorithm works as follows. For each point $p_i \in P$, we perform the following *counterclockwise processing procedure*. We first compute the sublist of \mathcal{L}_{t-1} that contains p_i and has the most counterclockwise endpoint. This is done by a *counterclockwise farthest enclosing sublist query*. We show later in Section 3.3 that each such query takes $O(\log n)$ time after $O(n \log n)$ time preprocessing for \mathcal{L}_{t-1} . Let $P[j_{i1}, j_{i2}]$ be the sublist computed above. Then, we compute the index $a_i^{j_{i2}+1}$ in $O(\log n)$ time by Lemma 7. Note that the union of the following three sublists is a sublist L of P : $P(b_i^i, a_i^i)$, $P[j_{i1}, j_{i2}]$, and $P(j_{i2}, a_i^{j_{i2}+1})$. We add L to \mathcal{L}_t and set $S_L = S_{L'} \cup \{p_i\}$ with $L' = P[j_{i1}, j_{i2}]$. By our algorithm invariant, points of L' are covered by $\mathcal{D}(S_{L'})$. By definition, points of $P(b_i^i, a_i^i) \cup P(j_{i2}, a_i^{j_{i2}+1})$ are covered by D_{p_i} . Therefore, all points of L are covered by $\mathcal{D}(S_L)$. Hence, the algorithm invariant holds for L . In addition, if L is P , we stop the algorithm and return S_L as an optimal dominating set.

Symmetrically, we perform a *clockwise processing procedure* for p_i . We compute the sublist from \mathcal{L}_{t-1} that contains p_i and has the most clockwise endpoint; this is done by a *clockwise farthest enclosing sublist query*. Let $P[j_{i3}, j_{i4}]$ be the sublist computed above. Then, we compute the index $b_i^{j_{i3}-1}$. Let L be the sublist that is the union of the following three

sublists: $P(b_i^i, a_i^i)$, $P[j_{i3}, j_{i4}]$, and $P(j_{i3}, b_i^{j_{i3}-1})$. Let $S_L = S_{L'} \cup \{p_i\}$ with $L' = P[j_{i3}, j_{i4}]$. As above, the algorithm invariant holds on L . We add L to \mathcal{L}_t . If L is P , then we stop the algorithm and return S_L as an optimal dominating set.

The following lemma proves the correctness of the algorithm.

► **Lemma 11.** *If the algorithm first time computes a sublist L that is P , then S_L is the smallest dominating set of $G(P)$.*

Time analysis. In each iteration, we perform $O(n)$ operations for computing indices a_i^j and b_i^j and $O(n)$ counterclockwise/clockwise farthest enclosing sublist queries. Computing indices a_i^j and b_i^j takes $O(\log n)$ time by Lemma 7. We show in Lemma 12 that each counterclockwise/clockwise farthest enclosing sublist query can be answered in $O(\log n)$ time after $O(n \log n)$ time preprocessing. As such, each iteration runs in $O(n \log n)$ time and the total time of the algorithm is $O(kn \log n)$, where k is the smallest dominating set size.

Algorithm implementation. It remains to describe the data structure for answering counterclockwise/clockwise farthest enclosing sublist queries. We only discuss the counterclockwise case as the clockwise case can be handled analogously. Given a set \mathcal{L} consisting of n sublists of P , the goal is to build a data structure to answer the following counterclockwise farthest enclosing sublist queries: Given a point $p \in P$, find a sublist in \mathcal{L} that contains p with the farthest counterclockwise endpoint from p . We have the following lemma.

► **Lemma 12.** *We can construct a data structure for \mathcal{L} in $O(n \log n)$ time such that each counterclockwise farthest enclosing sublist query can be answered in $O(\log n)$ time.*

We conclude with the following theorem and corollary, which will be used in Section 4 to solve the discrete k -center problem.

► **Theorem 13.** *Given a set P of n points in convex position in the plane, the smallest dominating set of the unit-disk graph $G(P)$ can be computed in $O(kn \log n)$ time, where k is the size of the smallest dominating set.*

► **Corollary 14.** *Given k , r , and a set P of n points in convex position in the plane, one can do the following in $O(kn \log n)$ time: determine whether there exists a subset $S \subseteq P$ of at most k points such that the distance from any point of P to its closest point in S is at most r , and if so, find such a subset S .*

Proof. We redefine the unit-disk graph of P with a parameter r , where two points in P are connected by an edge if their distance is at most r . We then apply the algorithm of Theorem 13. If the algorithm finds a sublist L that is P within k iterations, then we return $S = S_L$; otherwise such a subset S as in the lemma statement does not exist. Since we run the algorithm for at most k iterations, the total time of the algorithm is $O(kn \log n)$. ◀

4 The discrete k -center problem

In this section, we present our algorithm for the discrete k -center problem. Let P be a set of n points in convex position in the plane. Given a number k , the goal is to compute a subset $S \subseteq P$ of at most k points (called *centers*) so that the maximum distance between any point in P and its nearest center is minimized. Let r^* denote the optimal objective value.

Given a value r , the *decision problem* is to determine whether $r \geq r^*$, or equivalently, whether there exist a set of k centers in P such that the distance from any point of P to its closest center is at most r . By Corollary 14, the problem can be solved in $O(kn \log n)$ time. Clearly, r^* is equal to the distance of two points of P , that is $r^* \in R$, where R is defined as the set of all pairwise distances between points in P . If we explicitly compute R and then perform a binary search on R using the algorithm of Corollary 14 as a *decision algorithm*, then r^* can be computed in $O(n^2 + kn \log^2 n)$ time. We can improve the algorithm by using the distance selection algorithms, which can find the k -th smallest value in R in $O(n^{4/3} \log n)$ time for any given k [33, 54]. In fact, by applying the algorithmic framework of Wang and Zhao [54] with our decision algorithm, r^* can be computed in $O(n^{4/3} \log n + nk \log^2 n)$ time.

In the following, we present another algorithm of $O(k^2 n \log^2 n)$ time using the parametric search [20, 39]. This algorithm is faster than the above one when $k = o(n^{1/6}/\sqrt{\log n})$.

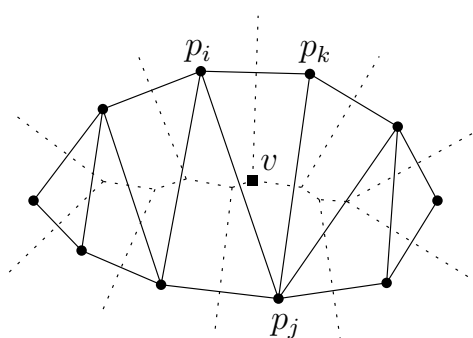
We simulate the decision algorithm over the unknown optimal value r^* . The algorithm maintains an interval $(r_1, r_2]$ that contains r^* . Initially, $r_1 = -\infty$ and $r_2 = \infty$. During the algorithm, the decision algorithm is invoked on certain *critical values* r to determine whether $r \geq r^*$; based on the outcome, the interval $(r_1, r_2]$ is shrunk accordingly so that the new interval still contains r^* . Upon completion, we can show that $r^* = r_2$ must hold.

Algorithm overview. For any r , certain variables in our decision algorithm are now defined with respect to r as the radius of unit disks and therefore may be considered as functions of r . For example, we use $a_i^j(r)$ to represent a_i^j when the unit disk radius is r . The algorithm has k iterations. We wish to compute the sublist set $\mathcal{L}_t(r^*)$ in each t -th iteration, $1 \leq t \leq k$. Specifically, the set $\mathcal{L}_1(r^*)$ relies on $a_i^i(r^*)$ and $b_i^i(r^*)$ for all points $p_i \in P$. As such, in the first iteration, we will compute $a_i^i(r^*)$ and $b_i^i(r^*)$ for all $p_i \in P$. The computation process will generate certain critical values r , call the decision algorithm on these values, and shrink the interval $(r_1, r_2]$ accordingly. After that, $\mathcal{L}_1(r^*)$ can be computed.

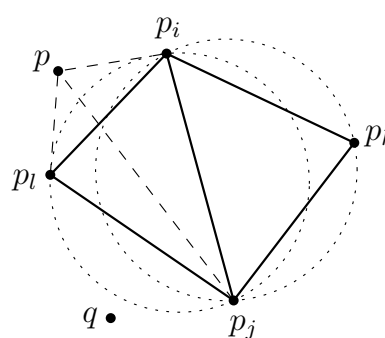
In a general t -th iteration, our goal is to compute the set $\mathcal{L}_t(r^*)$. We assume that the set $\mathcal{L}_{t-1}(r^*)$ is already available with an interval $(r_1, r_2]$ containing r^* . Then, for each $p_i \in P$, we perform a counterclockwise processing procedure. We first compute the sublist of $\mathcal{L}_{t-1}(r^*)$ with the farthest counterclockwise endpoint and containing $a_i^i(r^*)$. This procedure depends solely on $a_i^i(r^*)$ and $\mathcal{L}_{t-1}(r^*)$, which are already available, and thus no critical values are generated. Suppose that $P[j_{i1}(r^*), j_{i2}(r^*)]$ is the sublist computed above. The next step is to compute $a_i^{j_{i2}(r^*)+1}(r^*)$. This step will again generate critical values and shrink the interval $(r_1, r_2]$. After that, we add to $\mathcal{L}_t(r^*)$ the sublist that is the union of the following three sublists: $P(b_i^i(r^*), a_i^i(r^*))$, $P(j_{i1}(r^*), j_{i2}(r^*))$, and $P(j_{i2}(r^*), a_i^{j_{i2}(r^*)+1}(r^*))$. Similarly, we perform a clockwise processing procedure for each point $p_i \in P$. After that, the set $\mathcal{L}_t(r^*)$ is computed. The details can be found in the full version.

In summary, the algorithm can compute r^* in $O(k^2 n \log^2 n)$ time. Combining with the $O(n^{4/3} \log n + kn \log^2 n)$ time algorithm discussed earlier, we obtain the following result.

► **Theorem 15.** *Given a set P of n points in convex position in the plane and a number k , we can compute in $O(\min\{n^{4/3} \log n + kn \log^2 n, k^2 n \log^2 n\})$ time a subset $S \subseteq P$ of size at most k , such that the maximum distance from any point of P to its nearest point in S is minimized.*



■ **Figure 2** Illustrating $\mathcal{DT}(S)$, the solid segments, and $\mathcal{VD}(S)$, the dotted segments.



■ **Figure 3** Illustrating Lemma 18.

5 The independent set problem

In this section, we present our algorithms for the independent set problem, assuming that the points of P are in convex position. In Section 5.1, we present the algorithm for computing a maximum-weight independent set. Section 5.2 gives the algorithm for computing an (unweighted) independent set of size 3.

5.1 The maximum-weight independent set problem

Recall that $P = \langle p_1, p_2, \dots, p_n \rangle$ is a cyclic list ordered along $\mathcal{H}(P)$ in counterclockwise order. For each point $p_i \in P$, let w_i denote its weight. We assume that each $w_i > 0$ since otherwise p_i can be simply ignored, which would not affect the optimal solution. For any subset $P' \subseteq P$, let $w(P')$ denote the total weight of all points of P' .

For any three points p_1, p_2, p_3 , let $D(p_1, p_2, p_3)$ denote the disk whose boundary contains them. Thus, $\partial D(p_1, p_2, p_3)$ is the unique circle through these points. For any compact region B in the plane, we use ∂B to denote its boundary and use \bar{B} to denote the complement region of B in the plane. In particular, for a disk D in the plane, ∂D is its bounding circle, and \bar{D} refers to the region of the plane outside D .

In what follows, we first describe the algorithm and explain why it is correct, and then discuss how to implement the algorithm efficiently.

5.1.1 Algorithm description and correctness

To motivate our algorithm and demonstrate its correctness, we first examine the optimal solution structure and develop a recursive relation on which our dynamic program is based.

Let S be a maximum-weight independent set of $G(P)$, or equivalently, S is a maximum-weight subset of P such that the minimum pairwise distance of the points of S is larger than 1. Let $\mathcal{DT}(S)$ denote the Delaunay triangulation of S . If (p, q) is the closest pair of points of S , then \overline{pq} must be an edge of $\mathcal{DT}(S)$ and in fact, the shortest edge of $\mathcal{DT}(S)$ [45]. As such, finding a maximum-weight independent set of $G(P)$ is equivalent to finding a maximum-weight subset $S \subseteq P$ such that the shortest edge of $\mathcal{DT}(S)$ has length larger than 1. The algorithm in [47] is based on this observation, which also inspires our algorithm.

Consider a triangle $\triangle p_i p_j p_k$ of $\mathcal{DT}(S)$ such that the points p_i, p_j, p_k are in the counterclockwise order of P (i.e., ordered counterclockwise on $\mathcal{H}(P)$). Due to the property of Delaunay triangulation, the disk $D(p_i, p_j, p_k)$ does not contain any point of $S \setminus \{p_i, p_j, p_k\}$ [45]. Since the points of S are in convex position, we have the following observation.

► **Observation 16.** $\mathcal{DT}(S)$ does not contain an edge connecting two points from any two different subsets of $\{P(i, j), P(j, k), P(k, i)\}$; see Figure 2.

Observation 16 implies the following: To find an optimal solution S , if we know that $\triangle p_i p_j p_k$ is a triangle in $\mathcal{DT}(S)$, since no point of $S \setminus \{p_i, p_j, p_k\}$ lies in the disk $D(p_i, p_j, p_k)$, we can independently search $P(i, j) \cap \overline{D(p_i, p_j, p_k)}$, $P(j, k) \cap \overline{D(p_i, p_j, p_k)}$, and $P(k, i) \cap \overline{D(p_i, p_j, p_k)}$, respectively. This idea forms the basis of our dynamic program.

Let W^* denote the total weight of a maximum-weight independent set of $G(P)$.

For any pair of indices (i, j) with $|p_i p_j| > 1$, we call (i, j) a *canonical pair* and define $f(i, j)$ as the total weight of a maximum-weight subset P' of $P(i, j)$ such that $P' \cup \{p_i, p_j\}$ forms an independent set of $G(P)$; if no such subset P' exists, then $f(i, j) = 0$. Computing $f(i, j)$ is a subproblem in our dynamic program. For simplicity, we let $f(i, j) = -(w_i + w_j)$ if (i, j) is not canonical, i.e., $|p_i p_j| \leq 1$. Lemma 17 explains why we are interested in $f(i, j)$.

► **Lemma 17.** $W^* = \max_{1 \leq i, j \leq n} (f(i, j) + w_i + w_j)$.

By Lemma 17, to compute W^* , it suffices to compute $f(i, j)$ for all pairs of indices $1 \leq i, j \leq n$ and the one with the largest $f(i, j) + w_i + w_j$ leads to the optimal solution. To compute $f(i, j)$, we define another type of subproblems that will be used in our algorithm.

For any three points p_i, p_j, p_k such that they are ordered counterclockwise in P and their minimum pairwise distance is larger than 1, we call (i, j, k) a *canonical triple*.

For a canonical triple (i, j, k) , by slightly abusing the notation, we define $f(i, j, k)$ as the total weight of a maximum-weight subset P' of $P(i, j) \cap \overline{D(p_i, p_j, p_k)}$ such that $P' \cup \{p_i, p_j\}$ is an independent set; if no such subset P' exists, then $f(i, j, k) = 0$. For any canonical pair (i, j) , if we consider p_0 a dummy point to the left of $\overrightarrow{p_i p_j}$ and infinitely far from the supporting line of $\overrightarrow{p_i p_j}$ so that $D(p_i, p_j, p_0)$ becomes the left halfplane of $\overrightarrow{p_i p_j}$, then $f(i, j, 0)$ following the above definition is exactly $f(i, j)$; for convenience, we also consider $(i, j, 0)$ a canonical triple. To make the discussion concise, we often use $f(i, j, 0)$ instead of $f(i, j)$ since the way we compute $f(i, j, 0)$ is consistent with the way we compute $f(i, j, k)$ for $k \neq 0$.

For any canonical triple (i, j, k) , define $P_k(i, j) = \{p \mid p \in P(i, j), p \notin D(p_i, p_j, p_k), |pp_i| > 1, |pp_j| > 1\}$. For any canonical pair (i, j) , define $P_0(i, j) = \{p \mid p \in P(i, j), |pp_i| > 1, |pp_j| > 1\}$. Note that $P_0(i, j)$ is consistent with $P_k(i, j)$ if we consider p_0 a dummy point as defined above. Observe also that $P_k(i, j) = P_0(i, j) \cap \overline{D(p_i, p_j, p_k)}$ for any canonical triple (i, j, k) . By definition, $f(i, j, k)$ (including the case $k = 0$) is the total weight of a maximum-weight independent set $P' \subseteq P_k(i, j)$; this is the reason we introduce the notation $P_k(i, j)$.

The following lemma gives the recursive relation of our dynamic programming algorithm.

► **Lemma 18.** For any canonical triple (i, j, k) , including the case $k = 0$, the following holds (see Figure 3):

$$f(i, j, k) = \begin{cases} \max_{p_l \in P_k(p_i, p_j)} (f(i, l, j) + f(l, j, i) + w_l), & \text{if } P_k(i, j) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

With Lemma 18, it remains to find an order to solve the subproblems so that when computing $f(i, j, k)$, the values $f(i, l, j)$ and $f(l, j, i)$ for all $p_l \in P_k(p_i, p_j)$ are available.

For any two points $p_i, p_j \in P$, we call $\overline{p_i p_j}$ a *diagonal*.

We process the diagonals $\overline{p_i p_j}$ for all $1 \leq i, j \leq n$ in the following way. For each $j = 2, \dots, n$ in this order, we enumerate $i = j - 1, j - 2, \dots, 1$ to process $\overline{p_i p_j}$ as follows. If $|p_i p_j| \leq 1$, then we set $f(i, j) = -(w_i + w_j)$. Otherwise, (i, j) is a canonical pair, and we compute $f(i, j)$, i.e., $f(i, j, 0)$, by Equation (1); one can check that the values $f(i, l, j)$ and $f(l, j, i)$ for all $p_l \in P_0(p_i, p_j)$ have already been computed. Next, for each point $p_k \in P(j, i)$

with $|p_i p_k| > 1$ and $|p_j p_k| > 1$, (i, j, k) is a canonical triple and we compute $f(i, j, k)$ by Equation (1); again, the values $f(i, l, j)$ and $f(l, j, i)$ for all $p_l \in P_k(p_i, p_j)$ have already been computed. Finally, by Lemma 17, we can return the largest $f(i, j) + w_i + w_j$ among all canonical pairs (i, j) as W^* . The algorithm only computes the value W^* , but by the standard back-tracking technique a maximum-weight independent set can also be obtained.

5.1.2 Algorithm implementation

We can easily implement the algorithm in $O(n^4)$ time. Indeed, there are $O(n^3)$ subproblems $f(i, j, k)$. Each subproblem can be computed in $O(n)$ time by checking every point $p_l \in P_k(i, j)$. As such, the total time is $O(n^4)$. We give a better algorithm below.

Specifically, we show that for each canonical pair (i, j) we can compute the subproblems $f(i, j, k)$ for all $p_k \in P(j, i)$ in a total of $O(n^{3/2})$ time. To this end, we reduce the problem to an *offline outside-disk range max-cost query* problem. For each point $p_l \in P_k(i, j)$, we define the *cost* of p_l as $\text{cost}(p_l) = f(i, l, j) + f(l, j, i) + w_l$. Recall that $P_0(i, j) = \{p \mid p \in P(i, j), |pp_i| > 1, |pp_j| > 1\}$ and $P_k(i, j) = P_0(i, j) \cap \overline{D(p_i, p_j, p_k)}$. As such, computing $f(i, j, k)$ is equivalent to finding the maximum-cost point of $P_0(i, j)$ outside the query disk $D(p_i, p_j, p_k)$. Our goal is to answer all such disk queries for all $p_k \in P(j, i)$. We note that this problem can be solved in $O(n^{15/11})$ expected time by applying the recent randomized algorithm of Agarwal, Ezra, and Sharir [1]. In the full version, we present a deterministic algorithm of $O(n^{3/2})$ time by using cuttings [14]. We thus have the following result.

► **Theorem 19.** *Given a set P of n weighted points in convex position in the plane, a maximum-weight independent set in the unit-disk graph of P can be computed in $O(n^{7/2})$ deterministic time, or in $O(n^{37/11})$ randomized expected time.*

Using the randomized result of [1], the problem can be solved in $O(n^{37/11})$ expected time.

The following corollary will be used in Section 6 to solve the dispersion problem.

► **Corollary 20.** *Given a set P of n points in convex position in the plane and a number $r > 0$, one can find in $O(n^{7/2})$ deterministic time or in $O(n^{37/11})$ randomized expected time a maximum subset of P such that the distance of every two points of the subset is larger than r .*

5.2 Computing an independent set of size 3

To facilitate the discussion in Section 6 for the dispersion problem, we consider the following problem: Given a set P of n points in convex position and a number $r > 0$, find three points from P whose minimum pairwise distance is larger than or equal to r .

We follow the notation in Section 2. In particular, $P = \langle p_1, p_2, \dots, p_n \rangle$ is a cyclic list ordered along $\mathcal{H}(P)$ counterclockwise. In the following definition, for each point $p_i \in P$, we define a_i similarly to $a_i^>$ in Definition 5 with respect to r (i.e., change “ > 1 ” to “ $\geq r$ ”).

► **Definition 21.** *For each point $p_i \in P$, define a_i as the index of the first point p of P counterclockwise from p_i such that $|p_i p| \geq r$; similarly, define $b_i \in P$ as the index of the first point p clockwise from P such that $|p_i p| \geq r$. If $|p_i p| < r$ for all points $p \in P$, then let $a_i = b_i = 0$.*

We will make use of the following lemma, which has been proved previously in [35].

► **Lemma 22** ([35]). *P has three points whose minimum pairwise distance is at least r if and only if there exists a point $p_i \in P$ such that $P[a_i, b_i]$ has two points whose distance is at least r .*

If p_i is a point of P such that $P[a_i, b_i]$ has two points whose distance is at least r , we say that p_i is a *feasible point*. By Lemma 22, it suffices to find a feasible point (if it exists). Our algorithm comprises two procedures. In the first procedure, we compute a_i and b_i for all points $p_i \in P$. This can be done in $O(n \log n)$ time by slightly changing the algorithm of Lemma 7. The second procedure finds a feasible point. In the following, we present an $O(n \log n)$ time algorithm. We start with the following easy but crucial observation.

► **Observation 23.** *A point $p_i \in P$ is a feasible point if and only if there is a point $p_k \in P[a_i, b_i]$ such that p_{a_k} is also in $P[a_i, b_i]$ and (p_i, p_k, p_{a_k}) is in counterclockwise order in P .*

For each point $p_i \in P$, note that $a_i \neq i$ must hold; we define $a'_i = \begin{cases} a_i, & \text{if } i < a_i \\ a_i + n, & \text{otherwise.} \end{cases}$

By definition, $i < a'_i$ always holds and $a'_i = a_i$ if $a'_i \leq n$. Note that if $a_i = b_i$, then $P[a_i, b_i]$ has only one point, and therefore p_i cannot be a feasible point. As such, we only need to focus on the points p_i with $a_i \neq b_i$. Our algorithm is based on the following lemma, which in turn relies on Observation 23.

► **Lemma 24.** *For each $p_i \in P$, we have the following.*

1. *If $a_i < b_i$, then p_i is a feasible point if and only if $\min_{k \in [a_i, b_i]} a'_k \leq b_i$.*
2. *If $a_i > b_i$, then p_i is a feasible point if and only if $\min_{k \in [a_i, n]} a'_k \leq b_i + n$ or $\min_{k \in [1, b_i]} a'_k \leq b_i$.*

Define an array $A[1 \cdots n]$ such that $A[k] = a'_k$ for each $1 \leq k \leq n$. In light of Lemma 24, for each point $p_i \in P$, we can determine whether p_i is a feasible point using at most two *range-minima* queries of the following type: Given a range $[i, j]$ with $i \leq j$, find the minimum number in the subarray $A[i \cdots j]$. It is possible to answer each range-minima query in $O(1)$ time after $O(n)$ time preprocessing on A [7, 28]. For our problem, since it suffices to have $O(\log n)$ query time and $O(n \log n)$ preprocessing time, we can use a simple solution by constructing an augmented binary search tree. As such, in $O(n \log n)$ time we can find a feasible point or report that no such point exists.

In summary, in $O(n \log n)$ time we can determine whether P has three points whose minimum pairwise distance is at least r . If the answer is yes, then these three points can also be found within the same time complexity according to the proofs of Lemmas 22 and 24. We conclude with the following theorem.

► **Theorem 25.** *Given a set P of n points in convex position in the plane and a number r , in $O(n \log n)$ time one can find three points of P whose minimum pairwise distance is at least r or report that no such three points exist.*

6 The dispersion problem

Given a set P of n points in convex position in the plane and a number k , the dispersion problem is to find a subset of k points from P so that the minimum pairwise distance of the points of the subset is maximized.

Let r^* be the minimum pairwise distance of the points in an optimal solution subset. The value $r^* \in R$, where R is the set of pairwise distances of the points of P . Given a value r , the *decision problem* is to determine whether $r < r^*$, or equivalently, whether P has a subset of k points whose minimum pairwise distance is larger than r . By Corollary 20, the decision problem can be solved in $O(n^{7/2})$ time. Using the decision algorithm and doing binary search on the sorted list of R , r^* can be computed in $O(n^{7/2} \log n)$ time.

► **Theorem 26.** *Given a set of n points in convex position in the plane and a number k , one can find a subset of k points whose minimum pairwise distance is maximized in $O(n^{7/2} \log n)$ deterministic time, or in $O(n^{37/11} \log n)$ randomized expected time.*

The size-3 case. We now consider the case where $k = 3$. Given a number r , the *decision problem* is to determine whether $r \leq r^*$, that is, whether P has three points whose minimum pairwise distance is at least r . With Theorem 25 as our *decision algorithm*, the decision problem is solvable in $O(n \log n)$ time. To compute r^* , we follow the standard parametric search framework [39] and simulate the decision algorithm on the unknown optimal value r^* with an interval $[r_1, r_2]$ that contains r^* . In fact, Cole’s technique [20] can be applied here. In addition, we observe that Chan’s randomized technique [11] is applicable here. Consequently, applying the technique with our $O(n \log n)$ time decision algorithm leads to a randomized algorithm of $O(n \log n)$ expected time for the problem.

► **Theorem 27.** *Given a set of n points in convex position in the plane, one can find three points whose minimum pairwise distance is maximized in $O(n \log^2 n)$ deterministic time or in $O(n \log n)$ randomized expected time.*

7 The size-3 weighted independent set for points in arbitrary position

Given a set P of n weighted points in the plane in arbitrary position, the problem is to find a maximum-weight independent set of size 3 in $G(P)$. As the size of our target independent set is fixed, we allow points to have negative weights.

Define $\overline{G(P)}$ as the complement graph of $G(P)$. The problem is equivalent to finding a maximum-weight clique of size 3 in $\overline{G(P)}$. We want to partition $\overline{G(P)}$ into bicliques, i.e., complete bipartite graphs. We give the formal definition below.

► **Definition 28** (Biclique partition). *Define a biclique partition of $\overline{G(P)}$ as a collection of edge-disjoint bicliques $\Gamma(P) = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ such that the following are satisfied:*

1. *For each pair $(a, b) \in A_t \times B_t \in \Gamma$, $|ab| > 1$.*
2. *For any points $a, b \in P$ with $|ab| > 1$, Γ has a unique biclique $A_t \times B_t$ that contains (a, b) .*

In our problem, we need a stronger version of the partition, called a *tree-structured biclique partition*, and Lemma 30 explains why we introduce the concept.

► **Definition 29** (Tree-structured biclique partition). *A biclique partition $\Gamma(P) = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ is tree-structured if all the subsets A_t ’s form a tree \mathcal{T}_A such that for each internal node A_t , all its children subsets form a partition of A_t .*

► **Lemma 30.** *Let $\Gamma(P) = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ be a tree-structured biclique partition of $\overline{G(P)}$ and \mathcal{T}_A is the tree formed by the subsets A_t ’s. Then, the triple $a, b, c \in P$ forms an independent set in $G(P)$ if and only if $\Gamma(P)$ has a biclique (A_t, B_t) that contains a pair (a, b) , and A_t has an ancestor subset $A_{t'}$ in \mathcal{T}_A such that $c \in B_{t'}$ and $|bc| > 1$.*

With this, we obtain the following result; see the full version for the details.

► **Theorem 31.** *Given a set P of n weighted points in the plane, one can find a maximum-weight (or minimum-weight) independent set (or clique) of size 3 in the unit-disk graph $G(P)$ in $O(n^{5/3+\delta})$ time, for any arbitrarily small constant $\delta > 0$.*

References

- 1 Pankaj K. Agarwal, Esther Ezra, and Micha Sharir. Semi-algebraic off-line range searching and biclique partitions in the plane. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, pages 4:1–4:15, 2024. doi:10.4230/LIPIcs.SoCG.2024.4.
- 2 Pankaj K. Agarwal, Mark H. Overmars, and Micha Sharir. Computing maximally separated sets in the plane. *SIAM Journal on Computing*, 36(3):815–834, 2006. doi:10.1137/S00975397044446591.
- 3 Pankaj K. Agarwal, Micha Sharir, and Emo Welzl. The discrete 2-center problem. *Discrete and Computational Geometry*, 20:287–305, 1998. doi:10.1007/PL00009387.
- 4 Alok Aggarwal, Leonidas J. Guibas, James Saxe, and Peter W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete and Computational Geometry*, 4:591–604, 1989. doi:10.1007/BF02187749.
- 5 Tetsuya Araki and Shin-ichi Nakano. Max–min dispersion on a line. *Journal of Combinatorial Optimization*, 44(3):1824–1830, 2022. doi:10.1007/s10878-020-00549-5.
- 6 Paul Balister, Béla Bollobás, Amites Sarkar, and Mark Walters. Connectivity of random k -nearest-neighbour graphs. *Advances in Applied Probability*, 37(1):1–24, 2005. doi:10.1239/aap/1113402397.
- 7 Michael A. Bender and Martín Farach-Colton. The LCA problem revisited. In *Proceedings of the 4th Latin American Symposium on Theoretical Informatics*, pages 88–94, 2000. doi:10.1007/10719839_9.
- 8 Sergei Bespamyatnikh and David G. Kirkpatrick. Rectilinear 2-center problems. In *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG)*, 1999. URL: <http://www.cccg.ca/proceedings/1999/fp55.pdf>.
- 9 Sergei Bespamyatnikh and Michael Segal. Rectilinear static and dynamic discrete 2-center problems. In *Proceedings of the 6th Workshop on Algorithms and Data Structures (WADS)*, pages 276–287, 1999. doi:10.1007/3-540-48447-7_28.
- 10 Binay Bhattacharya, Ante Ćustić, Sandip Das, Yuya Higashikawa, Tsunehiko Kameda, and Naoki Katoh. Geometric p -center problems with centers constrained to two lines. In *Proceedings of the 18th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCGG)*, pages 24–36, 2015. doi:10.1007/978-3-319-48532-4_3.
- 11 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete and Computational Geometry*, 22(4):547–567, 1999. doi:10.1007/PL00009478.
- 12 Timothy M. Chan. More planar two-center algorithms. *Computational Geometry: Theory and Applications*, 13:189–198, 1999. doi:10.1016/S0925-7721(99)00019-X.
- 13 Timothy M. Chan, Qizheng He, and Yuancheng Yu. On the fine-grained complexity of small-size geometric set cover and discrete k -center for small k . In *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 34:1–34:19, 2023. doi:10.4230/LIPIcs.ICALP.2023.34.
- 14 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
- 15 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, Micha Sharir, and Emo Welzl. Improved bounds on weak ϵ -nets for convex sets. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 495–504, 1993. doi:10.1145/167088.167222.
- 16 Danny Z. Chen, Jian Li, and Haitao Wang. Efficient algorithms for the one-dimensional k -center problem. *Theoretical Computer Science*, 592:135–142, 2015. doi:10.1016/j.tcs.2015.05.028.
- 17 Kyungjin Cho, Eunjin Oh, Haitao Wang, and Jie Xue. Optimal algorithm for the planar two-center problem. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, pages 40:1–40:15, 2024. doi:10.4230/LIPIcs.SoCG.2024.40.
- 18 Jongmin Choi, Jaegun Lee, and Hee-Kap Ahn. Efficient k -center algorithms for planar points in convex position. In *Proceedings of the 18th Algorithms and Data Structures Symposium (WADS)*, pages 262–274, 2023. doi:10.1007/978-3-031-38906-1_18.

- 19 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990. doi:10.1016/0012-365X(90)90358-0.
- 20 Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34:200–208, 1987. doi:10.1145/7531.7537.
- 21 Gautam K. Das, Guilherme D. da Fonseca, and Ramesh K. Jallu. Efficient independent set approximation in unit disk graphs. *Discrete Applied Mathematics*, 280:63–70, 2020. doi:10.1016/j.dam.2018.05.049.
- 22 Gautam K. Das, Minati De, Sudeshna Kolay, Subhas C. Nandy, and Susmita Sur-Kolay. Approximation algorithms for maximum independent set of a unit disk graph. *Information Processing Letters*, 115:439–446, 2015. doi:10.1016/j.ipl.2014.11.002.
- 23 Minati De and Abhiruk Lahiri. Geometric dominating-set and set-cover via local-search. *Computational Geometry*, 113(C):102007, 2023. doi:10.1016/j.comgeo.2023.102007.
- 24 David Eppstein. Faster construction of planar two-centers. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 131–138, 1997. URL: <http://dl.acm.org/citation.cfm?id=314161.314198>.
- 25 David Eppstein. Graph-theoretic solutions to computational geometry problems. In *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.
- 26 Jared Espenant, J. Mark Keil, and Debajyoti Mondal. Finding a maximum clique in a disk graph. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, pages 30:1–30:17, 2023. doi:10.4230/LIPIcs.SoCG.2023.30.
- 27 Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA)*, pages 243–254, 2010. doi:10.1007/978-3-642-15775-2_21.
- 28 Dov Harel and Robert E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13:338–355, 1984. doi:10.1137/0213024.
- 29 Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979. doi:10.1016/0166-218X(79)90044-1.
- 30 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Triangles and girth in disk graphs and transmission graphs. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA)*, pages 64:1–64:14, 2019. doi:10.4230/LIPIcs.ESA.2019.64.
- 31 Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972. doi:10.1007/978-3-540-68279-0_8.
- 32 Matthew J. Katz, Klara Kedem, and Michael Segal. Discrete rectilinear 2-center problems. *Computational Geometry*, 15(4):203–214, 2000. doi:10.1016/S0925-7721(99)00052-8.
- 33 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.
- 34 J. Mark Keil and Debajyoti Mondal. The maximum clique problem in a disk graph made easy. *arXiv:2404.03751*, 2024. URL: <https://arxiv.org/abs/2404.03751>, doi:10.48550/arXiv.2404.03751.
- 35 Yasuaki Kobayashi, Shin-ichi Nakano, Kei Uchizawa, Takeaki Uno, Yutaro Yamaguchi, and Katsuhisa Yamanaka. An $O(n^2)$ -time algorithm for computing a max-min 3-dispersion on a point set in convex position. *IEICE Transactions on Information and Systems*, 105(3):503–507, 2022. doi:10.1587/transinf.2021FCP0013.
- 36 Andrzej Lingas. On approximation behavior and implementation of the greedy triangulation for convex planar point sets. In *Proceedings of the 2nd Annual Symposium on Computational Geometry (SoSG)*, pages 72–79, 1986. doi:10.1145/10515.10523.
- 37 Madhav V. Marathe, Heinz Breu, Harry B. Hunt III, Sekharipuram S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995. doi:10.1002/net.3230250205.

- 38 Tomomi Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Proceedings of the 2nd Japanese Conference on Discrete and Computational Geometry (JCDCG)*, pages 194–200, 1998. doi:10.1007/978-3-540-46515-7_16.
- 39 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983. doi:10.1145/2157.322410.
- 40 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983. doi:10.1137/0212052.
- 41 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete and Computational Geometry*, 44:883–895, 2010. doi:10.1007/s00454-010-9285-9.
- 42 Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsv) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM)*, pages 234–244, 1994. doi:10.1145/190809.190336.
- 43 Charles E. Perkins and Pravin Bhagwat. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999. doi:10.1109/MCSA.1999.749281.
- 44 Dana S. Richards and Jeffrey S. Salowe. A rectilinear steiner minimal tree algorithm for convex point sets. In *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 447, pages 201–212, 1990. doi:10.1007/3-540-52846-6_90.
- 45 Michael I. Shamos and Dan Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 151–162, 1975. doi:10.1109/SFCS.1975.8.
- 46 Micha Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete and Computational Geometry*, 18:125–134, 1997. doi:10.1007/PL00009311.
- 47 Vishwanath R. Singireddy, Manjanna Basappa, and Joseph S.B. Mitchell. Algorithms for k -dispersion for points in convex position in the plane. In *Proceedings of the 9th International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 59–70, 2023. doi:10.1007/978-3-031-25211-2_5.
- 48 Kuo-Hui Tsai and Da-Wei Wang. Optimal algorithms for circle partitioning. In *Proceedings of the Third Annual International Computing and Combinatorics Conference (COCOON)*, pages 304–310, 1997. doi:10.1007/BFb0045097.
- 49 Vijay V. Vazirani. *Approximation Algorithms*. Springer, Berlin Heidelberg, 1st edition, 2001.
- 50 Da-Wei Wang and Yue-Sun Kuo. A study on two geometric location problems. *Information processing letters*, 28(6):281–286, 1988. doi:10.1016/0020-0190(88)90174-3.
- 51 Haitao Wang. On the planar two-center problem and circular hulls. *Discrete and Computational Geometry*, 68:1175–1226, 2022. doi:10.1007/S00454-021-00358-5.
- 52 Haitao Wang. Unit-disk range searching and applications. *Journal of Computational Geometry*, 14(1):343–394, 2023. doi:10.20382/jocg.v14i1a13.
- 53 Haitao Wang and Jingru Zhang. Line-constrained k -median, k -means, and k -center problems in the plane. *International Journal of Computational Geometry and Application*, 26(3):185–210, 2016. doi:10.1142/S0218195916600049.
- 54 Haitao Wang and Yiming Zhao. Improved algorithms for distance selection and related problems. In *Proceedings of the 31st Annual European Symposium on Algorithms (ESA)*, pages 101:1–101:14, 2023. doi:10.4230/LIPIcs.ESA.2023.101.
- 55 Frank Ángel Hernández Mira, Ernesto Parra Inza, José María Sigarreta Almira, and Nodari Vakhania. A polynomial-time approximation to a minimum dominating set in a graph. *Theoretical Computer Science*, 930:142–156, 2022. doi:10.1016/j.tcs.2022.07.020.

Nearly-Optimal Algorithm for Non-Clairvoyant Service with Delay

Noam Touitou  

Unaffiliated, Tel Aviv, Israel

Abstract

We consider the online service with delay problem, in which a server traverses a metric space to serve requests that arrive over time. Requests gather individual delay cost while awaiting service, penalizing service latency; an algorithm seeks to minimize both its movement cost and the total delay cost. Algorithms for the problem (on general metric spaces) are only known for the clairvoyant model, where the algorithm knows future delay cost in advance (e.g., Azar et al., STOC'17; Azar and Touitou, FOCS'19; Touitou, STOC'23). However, in the non-clairvoyant setting, only negative results are known: where n is the size of the metric space and m is the number of requests, there are lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ on competitiveness (Azar et al., STOC'17), that hold even for randomized algorithms (Le et al., SODA'23).

In this paper, we present the first algorithm for non-clairvoyant online service with delay. Our algorithm is deterministic and $O(\min\{\sqrt{n} \log n, \sqrt{m} \log m\})$ -competitive; combined with the lower bounds of Azar et al., this settles the correct competitive ratio for the problem up to logarithmic factors, in terms of both n and m .

2012 ACM Subject Classification Theory of computation \rightarrow K-server algorithms; Theory of computation \rightarrow Online algorithms

Keywords and phrases Online, Delay, Deadlines, k -server, Non-clairvoyant

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.74

1 Introduction

In online service with delay, or OSD, a server exists on a metric space of n points. Requests arrive over time, where every request is associated with a point in the metric space which the server must visit to satisfy the request. The algorithm may move the server at any moment in time, at a cost which is the distance traveled by the server in the metric space (the movement is instantaneous). In addition, requests accumulate delay cost while pending, motivating the algorithm to serve requests promptly. The goal of the algorithm is to minimize the sum of total movement cost and total delay cost over the given input.

In this model, a crucial choice is whether the online algorithm becomes aware of a request's future delay cost upon its release (*clairvoyant* model) or only knows delay cost accumulated in the past (*non-clairvoyant* model). Azar et al. [4], who introduced the problem of online service with delay, presented an algorithm for the clairvoyant model of polylogarithmic competitiveness in the size of the metric space n ; specifically, $O(\log^4 n)$ -competitiveness. This was later improved to $O(\log^2 n)$ -competitiveness [6], and then to $O(\log(\min\{n, m\}))$ -competitiveness [35], where m is the number of requests in the online input.

In the non-clairvoyant model, however, there are no known positive results. Azar et al. [4] presented $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ lower bounds on the competitiveness of any deterministic algorithm; this bound also holds for randomized algorithms [31]. (Note that the lower bound in [31] is stated for the joint replenishment problem, a special case of online service with delay.)



© Noam Touitou;

licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 74; pp. 74:1–74:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1.1 Our Results

In this paper, we present the first non-clairvoyant algorithm for online service with delay. Define k to be the number of locations on which requests are released in the input; we prove the following theorem.

► **Theorem 1.** *There exists a polynomial-time, deterministic, non-clairvoyant algorithm for online service with delay which is $O(\sqrt{k} \log k)$ -competitive.*

In particular, note that $k \leq \min(n, m)$; thus, Theorem 1 also yields an $O(\min\{\sqrt{n} \log n, \sqrt{m} \log m\})$ competitiveness bound. Recalling the $\Omega(\sqrt{n}), \Omega(\sqrt{m})$ lower bounds on competitiveness of [4] for non-clairvoyant algorithms for OSD, we conclude that our competitive ratio is tight up to a logarithmic factor.

Another problem of interest is online service with *deadlines*. In this problem, instead of accumulating delay cost, every request has an associated deadline by which it must be served. Online service with deadlines is a special case of online service with delay; intuitively, a request with a deadline corresponds to a request with delay that goes from zero to infinity at the time of the deadline. Yet, online service with deadlines has been studied explicitly in the past (e.g., [24, 25, 35]). Theorem 1 for online service with delay also yields Theorem 2 for online service with deadlines as a corollary, providing the first non-clairvoyant algorithm for online service with deadlines.

► **Theorem 2.** *There exists a polynomial-time, deterministic, non-clairvoyant algorithm for online service with deadlines which is $O(\sqrt{k} \log k)$ -competitive.*

1.2 Related Work

A class of problems related to online service with delay is online network design with delay, where connectivity requests are handled by transmitting items. Such problems include TCP Acknowledgement [21, 29, 17, 28], joint replenishment [18, 15, 11, 19], multilevel aggregation [9, 16, 31, 6, 8], facility location [6, 7, 10], and set cover with delay [2, 33, 31]. In [7], a general framework for such problems in the clairvoyant model was introduced, yielding logarithmic competitiveness.

While some problems, such as set cover with delay, retain polylogarithmic competitiveness in the non-clairvoyant setting [2], others become much harder. This is the case for joint replenishment, facility location and multilevel aggregation with delay, which have $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ lower bounds on the competitiveness of any randomized algorithm [4, 31]. As online service with delay is a generalization of the joint replenishment and multilevel aggregation problems, these lower bounds extend also to non-clairvoyant service with delay. An $O(\min(\sqrt{n \log n}, \sqrt{m \log m}))$ -competitive framework for non-clairvoyant network design was introduced in [34], nearly matching these lower bounds.

Online service with delay in the clairvoyant setting has also been considered with k servers (rather than a single server). Azar et al. [4] presented a $O(k \cdot \text{polylog}(n))$ -competitive randomized algorithm for the problem. The algorithm involved randomly embedding the metric space into a tree, then solving deterministically on that tree; as the algorithm did not use randomization in server allocation, its dependence on k is linear. Subsequently, for the special case of a metric space that is a weighted star, Gupta et al. [26] presented an $O(\log n \log k)$ -competitive randomized algorithm. For a general metric space, Gupta et al. [25] presented a $O(\text{polylog}(\Delta, n))$ -competitive algorithm, where Δ is the aspect ratio of the metric space (largest-to-smallest pairwise-distance ratio). For the special case of a uniform

metric space, Krnetic et al. [30] settled the exact competitive ratio admitted by the problem; interestingly, the upper bound was achieved by a *non-clairvoyant* algorithm, showing that clairvoyance is not needed on a uniform space.

Another noteworthy line of work in online algorithms with delay has been on online matching, where matching requests arrive over time and accumulate delay cost while unmatched; see, e.g., [1, 22, 3, 12, 13, 5, 32, 27, 20]. In most of these works, a main assumption is that delay accumulation is identical for all requests, which nullifies the need for clairvoyance. (A notable exception is [20], that considers more general delay models.)

1.3 Our Techniques

Consider the algorithm for clairvoyant online service with delay in [35]. This algorithm performs *services*; a service is a sequence of server movements that occurs instantaneously at a given time. Each service is triggered by a set of requests gathering sufficient delay cost. In each service, the server moves within a ball of a certain radius around its location, serving some pending requests subject to a given budget; the radius of the ball and the service budget are determined by a property called the service’s *level*.

The prioritization of requests within the ball is according to the future delay accumulation of requests. Intuitively, this prioritization ensures that if a request “survived” a service λ at time t , but then gathers delay cost that triggers another service λ' at time t' , then the requests that *were* served by λ would have gathered large delay during $[t, t']$ had they been left pending; this is used to justify a doubling argument, e.g. allowing λ' to have twice the budget as λ . Services that use this doubling mechanism are called “secondary”, while other services are called “primary”.

However, in the non-clairvoyant setting considered in this paper, we cannot predict future delay, and thus cannot prioritize requests. Instead, we are relegated to spending budget “blindly”. In our algorithm, this is expressed by solving a prize-collecting Steiner tree problem in which we aim to visit some locations within a certain ball, where the penalty function is uniform; i.e., visiting every location is equally important. This results in our algorithm visiting the most locations subject to a certain budget per location. Conversely – and crucially – we maintain the property that locations *not* visited by our solution are expensive to visit; in fact, every *bundle* of these requests is expensive. The construction of a poly-time prize-collecting algorithm with this property is based on the construction in [34], and hinges on prize-collecting Steiner tree admitting a Lagrangian approximation (as shown, e.g., by Goemans and Williamson [23]).

Then, we wait for these unvisited locations to gather delay equal to their budget; once enough locations accumulate this delay, we can charge it to the optimal solution, as both the incurred delay and the cost of serving these requests are large. However, when unvisited locations gather large delay, the algorithm must also serve them; it does so greedily, moving to the request and back. We call these greedy services “tertiary”, and they exist alongside primary and secondary services.

An additional, more technical component of the algorithm is that of *domes*. As in [35], requests have levels in our algorithm, and a service starts when requests of level at most ℓ gather enough delay in a radius- 2^ℓ ball, for some level ℓ . However, unlike in [35], inside an inner ball of radius $2^{\ell-1}$, we only consider delay of requests *exactly* ℓ , forming a dome-like structure. The benefit of this structure is in eliminating the slack in service levels that existed in the doubling argument of [35]. The use of this property in the proof is related to the investment counters that are maintained per location, rather than per request (as in [35]), as non-clairvoyance prohibits us from knowing a request’s future delay cost.

2 Preliminaries

In online service with delay, a metric space G of n points is given; we denote by $\delta(u, v)$ the distance between two points $u, v \in G$. Requests are released over time in an online manner, where every request q is associated with a point $V(q) \in G$. At any point in time, the algorithm can move the server from its current location a to a new location a' , paying a cost of $\delta(a, a')$. Then, every request pending q where $V(q) = a'$ becomes served. We denote by r_q the release time of request q . Every request q also has some continuous, non-decreasing delay function d_q , which maps from a time $t \geq r_q$ to the delay cost incurred by q if pending until time t ¹; we assume that every request must eventually be served, and thus d_q tends to infinity as time advances. Without loss of generality, we also assume that $d_q(r_q) = 0$ (assuming otherwise would simply add a constant additive term to the cost of all solutions). We expand this notation to multiple requests: for every subset of requests $Q' \subseteq Q$ and time t , we define $d_{Q'}(t) := \sum_{q \in Q'} d_q(t)$. We also define $m := |Q|$ to be the total number of requests in the input sequence.

Our algorithms perform *services*, which are a set of server movements that take place instantaneously. Where λ is a service, we denote by t_λ the time in which the service occurs. We define $a(t), a^*(t)$ to be the locations of the algorithm's server and the optimum's server at time t , respectively. Throughout the paper, when we refer to the value of some variable at t_λ for some service λ , we refer to the value immediately *before* the service. For example, $a(t_\lambda)$ is the location of the server immediately before λ . As a shorthand, we also define $a_\lambda := a(t_\lambda)$.

We define $B(v, r)$ to be the closed ball centered at v of radius r (i.e., the set of points of distance at most r from v). We also sometimes equate the points of the metric space G with the nodes of a clique graph, where the edge connecting nodes u, v has weight $\delta(u, v)$. This graph representation is useful when discussing charging cylinders, a useful tool in our analysis.

For ease of notation, we use the superscript $+$ to indicate the positive part of a number. That is, $x^+ := \max(0, x)$.

3 Online Service with Delay

This section introduces an algorithm for non-clairvoyant online service with delay, proving Theorem 1. First, we define k to be the number of locations on which requests are released in the input. Note that $k \leq \min(n, m)$; we thus focus on proving $O(\sqrt{k} \log k)$ -competitiveness, which implies the competitiveness bound of Theorem 1.

3.1 The Algorithm

Before describing the algorithm, we first introduce some of its components.

Steiner tree. As part of our algorithm, we formulate and solve a Steiner tree problem. In Steiner tree, one is given a graph with edge costs and a set of terminal nodes. The goal is to buy an edge subset of minimal cost that connects the terminals. With terminals V , and assuming that the graph is known from context, we use $ST^*(V)$ to denote the optimal cost of a solution. We also sometimes refer to the (equivalent) rooted version, in which the terminals must be connected to a designated root node r ; here, we use $ST^*(V; r)$ to denote the optimal cost.

¹ The continuity assumption is without loss of generality: relaxing this assumption, one could simulate continuous delay growth upon observing a delay "spike".

Prize-collecting Steiner tree and Lagrangian approximation. In the prize-collecting variant of the (rooted) Steiner tree problem, we are also given a penalty function π mapping from each terminal to a non-negative penalty; we write the prize-collecting input as $(Q, \pi; r)$. The algorithm must connect terminals to the root by buying edges, and must pay the penalty for unconnected terminals. The goal of the algorithm is to minimize the sum of edge costs and penalty costs; we write $\text{PCST}^*(Q, \pi; r)$ to refer to the optimal solution to the input $(Q, \pi; r)$. A *Lagrangian* prize-collecting algorithm is an algorithm that has different approximation ratios w.r.t. these two costs, such that it is 1-approximate on penalty costs. Specifically, Goemans and Williamson [23] presented the following algorithm for prize-collecting Steiner tree.

► **Theorem 3** (due to [23]). *There exists an approximation algorithm PCST for prize-collecting Steiner tree such that, fixing any input $(Q, \pi; r)$, it holds that*

$$\text{PCST}^b(Q, \pi; r) + 2 \cdot \text{PCST}^p(Q, \pi; r) \leq 2 \cdot \text{PCST}^*(Q, \pi; r),$$

where $\text{PCST}^b, \text{PCST}^p$ are the buying and penalty costs of PCST, respectively.

In [34], a simple procedure is presented that converts a Lagrangian prize-collecting procedure into a procedure which identifies a solution to a maximal subset of requests subject to a budget per request. Combining this result with the algorithm of [23] yields Lemma 4, which introduces the procedure PCSOLVE used in our algorithm.

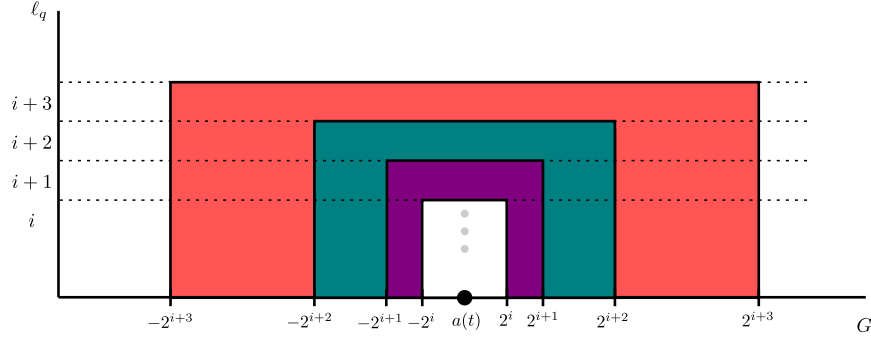
► **Lemma 4** (due to [23] and Proposition 21 from [34]). *There exists a procedure PCSOLVE which, given a prize-collecting Steiner tree input $(Q, \pi; r)$, outputs a solution T for a request subset $Q' \subseteq Q$ such that:*

- $c(T) \leq 2 \cdot \sum_{q \in Q'} \pi(q)$.
- For every $Q'' \subseteq Q \setminus Q'$, it holds that $\text{ST}^*(Q''; r) \geq \sum_{q \in Q''} \pi(q)$.

Levels and pointers. The algorithm maintains for every pending request q a level ℓ_q , which is initially $-\infty$. A service λ also has a level ℓ_λ , which determines its budget for serving requests. As services occur, they may increase the levels of requests. The algorithm also maintains for request q a pointer μ_q to the last service that increased the level of q . (A fine point, as seen in the algorithm, is that a service may also appear in the pointer μ_q only for “attempting” to increase ℓ_q .) Finally, overloading notation, each service λ also has a pointer μ_λ that points to a prior service; this pointer is equal to the pointer μ_q for some request q considered by λ .

Residual delay counters and domes. The algorithm maintains a *residual delay* counter $g_{v,\ell}$ for every location v and level ℓ ; for time t , we define $g_{v,\ell}(t)$ to be the value of $g_{v,\ell}$ at time t . This counter grows with the delay cost incurred by level- ℓ requests on v . In addition, the counters are occasionally decreased by services made by the algorithm; this can be interpreted as the service “paying” for incurred delay out of its budget. Positive counters correspond to incurred delay that has not yet been handled; such counters can trigger a service, in the manner we now describe.

At any time t , and for every level ℓ , the algorithm considers positive residual delay counters of levels at most ℓ inside $B(a(t), 2^\ell)$. Specifically, it considers the total unhandled residual delay of requests of level at most ℓ inside the ring $B(a(t), 2^\ell) \setminus B(a(t), 2^{\ell-1})$, plus unhandled residual delay of requests of level *exactly* ℓ inside the internal ball $B(a(t), 2^{\ell-1})$. This forms a dome-like structure, as shown in Figure 1. The algorithm waits until a dome corresponding to some level ℓ becomes critical, i.e., has a lot of unhandled residual delay, and then starts a service. These notions are formalized in Definition 5.



This visualization illustrates the structure of domes. The metric space visualized is a line, shown on the horizontal axis. The vertical axis shows counter levels. The shapes in color correspond to the different domes at the current time, which include different levels at different points, according to their distance from the server.

■ **Figure 1** Visualization of domes.

► **Definition 5** (domes). *For every time t , level ℓ , and $v \in B(a(t), 2^\ell)$, define*

$$y_\ell(v, t) := \begin{cases} (g_{v, \ell}(t))^+ & v \in B(a(t), 2^{\ell-1}) \\ \sum_{\ell' \leq \ell} (g_{v, \ell'}(t))^+ & v \in B(a(t), 2^\ell) \setminus B(a(t), 2^{\ell-1}) \end{cases}$$

For time t and level ℓ , define $Y_\ell(t) := \sum_{v \in B(a(t), 2^\ell)} y_\ell(v, t)$; where t is known from context, we also write Y_ℓ . If $Y_\ell \geq 2^\ell$, we say that dome ℓ is critical.

Algorithm's Description. Whenever dome ℓ becomes critical, we start a service λ of level $\ell + 4$. (If more than one dome is critical at the same time, we handle the dome of the largest level first.) The service first considers whether the dome has any positive residual delay from the inner ball $B(a_\lambda, 2^{\ell-1}) = B(a_\lambda, 2^{\ell-5})$. If not, the service λ is called *primary*. Otherwise, consider a location v of the inner ball that contributes positive residual delay to the dome; it must be that $g_{v, \ell} > 0$, which implies (through Proposition 9) that there exists a level- ℓ pending request on v . The service arbitrarily chooses such a request σ_λ , and observes the last service to modify the level of the request, i.e., μ_{σ_λ} ; denote this service by λ' . Depending on λ' , the algorithm decides if the service will invest for the future (“secondary” service) or act greedily (“tertiary” service). Specifically, the algorithm maintains a counter $\beta(\lambda')$ for the number of times λ' has triggered tertiary services; if the counter is low, λ becomes tertiary and increments this counter; otherwise, λ will be secondary. After deciding if a service is primary, secondary, or tertiary, the algorithm zeroes all positive residual delay on locations in $B(a_\lambda, 2^{\ell_\lambda})$ of levels up to ℓ_λ ; in particular, this zeroes the residual delay that triggered λ .

Next, if the service is tertiary, it simply moves the server to σ_λ and back, concluding the service. Otherwise, λ is primary or secondary, and thus invests in serving pending requests, through the method SERVEELIGIBLE. In SERVEELIGIBLE, the service considers the subset of locations V_λ inside $B(a_\lambda, 2^{\ell_\lambda})$ on which there are pending requests. To each location in V_λ , the algorithm assigns a budget of $x_\lambda = 2^{\ell_\lambda} / \sqrt{|V_\lambda|}$ to visiting each such location; the locations V_λ , together with their budgets as penalties, are transferred to PCSOLVE as a prize-collecting Steiner tree input (where the root is the current location of the server a_λ). PCSOLVE outputs a tree connecting some locations $Q_\lambda^\circ \subseteq V_\lambda$ to a_λ ; this tree is then traversed by the server in a DFS fashion, visiting Q_λ° and ending at a_λ . For the remaining locations $V_\lambda \setminus Q_\lambda^\circ$, where

PCSOLVE pays the penalty, the algorithm decreases the delay counters for those locations by x_λ . On those locations, the algorithm makes pending requests of level at most ℓ_λ point to λ , and upgrades their level to ℓ_λ .

Finally, for primary services, the algorithm may move the server to a new location a'_λ . Recall that λ starts upon dome $\ell_\lambda - 4$ becoming critical; if the majority of the residual delay making dome $\ell_\lambda - 4$ critical occurs inside a small-radius ball (specifically, radius $2^{\ell_\lambda - 8}$), then the center of this ball becomes the new location a'_λ , at which the server rests at the end of the service.

The non-clairvoyant algorithm for online service with delay is given in Algorithm 1, and the SERVEELIGIBLE method appears in Algorithm 2. We henceforth focus on analyzing Algorithm 1 and proving Theorem 1.

■ **Algorithm 1** Non-clairvoyant algorithm for online service with delay.

```

1 Function UPONCRITICAL ( $\ell$ )
2   start a service  $\lambda$ , and set  $\ell_\lambda \leftarrow \ell + 4$ .
3   denote by  $t$  the current time, and by  $a_\lambda$  the current location of the server.
4   let  $V_\lambda^\dagger \subseteq B(a_\lambda, 2^{\ell_\lambda - 4})$  be the subset of locations  $v$  where  $y_{\ell_\lambda - 4}(v, t) > 0$ .
5   if  $V_\lambda^\dagger \cap B(a_\lambda, 2^{\ell_\lambda - 5}) = \emptyset$  then
6     | say  $\lambda$  is primary.
7   else
8     | let  $\sigma_\lambda$  be an arbitrary request of level  $\ell_\lambda - 4$  on a point in  $V_\lambda^\dagger \cap B(a_\lambda, 2^{\ell_\lambda - 5})$ .
9     | define  $\mu_\lambda \leftarrow \mu_{\sigma_\lambda}$ .
10    | if  $\beta(\mu_\lambda) \geq 2\sqrt{|V_{\mu_\lambda}|}$  then
11    | | say  $\lambda$  is secondary.
12    | else
13    | | say  $\lambda$  is tertiary.

    // if primary and most residual delay is in small-radius ball, mark its center for future
    // movement.
14   if  $\lambda$  is primary and  $\exists a' \in G: y_{\ell_\lambda - 4}(B(a', 2^{\ell_\lambda - 8}) \cap V_\lambda^\dagger, t) > 2^{\ell_\lambda - 5}$  then define  $a'_\lambda := a'$ .
15   foreach  $v \in B(a_\lambda, 2^{\ell_\lambda})$  do // zero residual delay inside service ball, of levels at most  $\ell_\lambda$ .
16     | foreach  $\ell' \leq \ell_\lambda$  do
17     | | set  $g_{v, \ell'} \leftarrow \min(g_{v, \ell'}, 0)$ 
18   if  $\lambda$  is tertiary then
19     | visit  $\sigma_\lambda$  with the server, and return to  $a_\lambda$  afterwards†.
20     | set  $\beta(\mu_\lambda) \leftarrow \beta(\mu_\lambda) + 1$ .
21     | return
22   call SERVEELIGIBLE( $\lambda$ ).
23   if  $\lambda$  is primary then move the server from  $a_\lambda$  to  $a'_\lambda$ .
24   set  $\beta(\lambda) \leftarrow 0$ .

```

25 [†] In Line 19 of this algorithm, and in Line 5 of Algorithm 2, to avoid serving requests without accounting for their delay, the algorithm does not consider requests of level greater than ℓ_λ as served by the movements in λ . thus, the algorithm might consider some requests as still pending when they are, in fact, served.

■ **Algorithm 2** The ServeEligible method.

```

1 Function SERVEELIGIBLE( $\lambda$ )
2   let  $V_\lambda$  be the subset of locations in  $B(a_\lambda, 2^{\ell_\lambda})$  on which there are pending requests.
3   define  $x_\lambda \leftarrow \frac{2^{\ell_\lambda}}{\sqrt{|V_\lambda|}}$ .
4   run PCSOLVE( $V_\lambda, x_\lambda; a_\lambda$ ) to obtain a solution  $T$  to the subset  $Q_\lambda^\circ \subseteq V_\lambda$  of locations.
5   traverse  $T$  with the server using DFS, returning to  $a_\lambda$  at the end; let  $Q_\lambda$  be the set of
      requests of types in  $Q_\lambda^\circ$  that are served†.
      // upgrade surviving eligible requests, and invest in eligible location counters.
6   foreach  $v \in V_\lambda$  do
7     foreach request  $q$  on  $v$  of level at most  $\ell_\lambda$  do
8       set  $\ell_q \leftarrow \ell_\lambda$ .
9       set  $\mu_q \leftarrow \lambda$ .
10    decrease  $g_{v, \ell_\lambda}$  by  $x_\lambda$ .

```

3.2 Definitions and Properties

► **Definition 6.** We define the following.

1. Let Λ denote the set of services in the algorithm. We partition Λ into $\Lambda^1, \Lambda^2, \Lambda^3$, the sets of primary, secondary, and tertiary services, respectively.
2. For two services $\lambda_1, \lambda_2 \in \Lambda$ where λ_2 occurs after λ_1 , if $\mu_{\lambda_2} = \lambda_1$ we say that λ_2 is assigned to λ_1 . Note that in this case $\lambda_1 \in \Lambda^1 \cup \Lambda^2$, $\lambda_2 \in \Lambda^2 \cup \Lambda^3$.
3. Let $\lambda \in \Lambda^1 \cup \Lambda^2$ be a service. If there exists a secondary service $\lambda' \in \Lambda^2$ which is assigned to λ , then we call λ a certified service, and say that λ' certified λ . We denote the set of certified services by $\Lambda^c \subseteq \Lambda^1 \cup \Lambda^2$.

► **Proposition 7.** For a certified service $\lambda \in \Lambda^c$, we have the following:

1. The service λ is certified by exactly one service $\lambda' \in \Lambda^2$.
2. It holds that $\ell_{\lambda'} = \ell_\lambda + 4$.
3. It holds that $\delta(a_\lambda, a_{\lambda'}) \leq 2^{\ell_\lambda + 1}$.

Proof. First, observing Line 9 of Algorithm 2, we note that if for request q we have $\mu_q = \lambda$, then it must be the case that $\ell_q = \ell_\lambda$ at that time. Note that λ is only certified by some $\lambda' \in \Lambda^2$ if $\mu_{\sigma_{\lambda'}} = \lambda$, and thus $\ell_{\sigma_{\lambda'}} = \ell_\lambda$ at $t_{\lambda'}$; but, $\ell_{\lambda'} = \ell_{\sigma_{\lambda'}} + 4$ at $t_{\lambda'}$, proving the second item. Moreover, note that $\sigma_{\lambda'} \in V_\lambda$, and thus $\delta(a_\lambda, \sigma_{\lambda'}) \leq 2^{\ell_\lambda}$. In addition, since $\sigma_{\lambda'} \in B(a_{\lambda'}, 2^{\ell_{\lambda'} - 5})$, we have $\delta(a_{\lambda'}, \sigma_{\lambda'}) \leq 2^{\ell_{\lambda'} - 5} = 2^{\ell_\lambda - 1}$. The triangle inequality thus implies $\delta(a_\lambda, a_{\lambda'}) \leq 2^{\ell_\lambda} + 2^{\ell_\lambda - 1} \leq 2^{\ell_\lambda + 1}$, proving the third item.

We now prove the first item, i.e., the uniqueness of λ' . Suppose, for contradiction, that a certified service $\lambda \in \Lambda^c$ is certified by two services $\lambda', \lambda'' \in \Lambda^2$, and assume without loss of generality that λ' occurs before λ'' . We prove that after λ' , there remains no pending request q with $\mu_q = \lambda$, and thus λ'' cannot certify λ later on. Indeed, consider such a request q immediately before λ' ; it holds that $q \in V_\lambda \subseteq B(a_\lambda, 2^{\ell_\lambda})$, and it must also be the case that $\ell_q = \ell_\lambda$ at that time. But, we've shown that $\delta(a_\lambda, a_{\lambda'}) \leq 2^{\ell_\lambda + 1}$, and thus $B(a_\lambda, 2^{\ell_\lambda}) \subseteq B(a_{\lambda'}, 2^{\ell_\lambda + 4}) = B(a_{\lambda'}, 2^{\ell_{\lambda'}})$. Thus, $q \in V_{\lambda'}$, and will either be served by λ' or have μ_q be set to λ' , in contradiction to having $\mu_q = \lambda$ at $t_{\lambda''}$. ◀

► **Proposition 8.** For every level i it holds that $Y_i \leq 2^{i+2}$ at every point in time.

The proof of Proposition 8 is in Section B.

► **Proposition 9.** *For every point $v \in G$, level ℓ and time t , it holds that $g_{v,\ell}(t) \leq d_{Q'}(t)$, where Q' is the set of pending requests of level exactly ℓ on v at time t . (In particular, when $Q' = \emptyset$, it holds that $g_{v,\ell}(t) \leq 0$.)*

Proof. We prove this by induction as time advances, noting that the proposition holds at time 0 (before any requests are released). Note that $g_{v,\ell}(t)$ grows at the same rate as the delay of pending requests of level ℓ on v , and thus delay growth cannot break the invariant. In addition, counter decreases in services cannot break the invariant. The only risky event is when a request of level ℓ is upgraded or completed. But, this only happens in a service λ such that $\ell_\lambda \geq \ell$ and $v \in B(a_\lambda, 2^{\ell_\lambda})$. Note that in such services, Line 17 of Algorithm 1 ensures that $g_{v,\ell}$ is non-positive after the service, maintaining the invariant. ◀

► **Proposition 10.** *During a service λ , if the algorithm moves its server from a_λ to a'_λ in Line 23 of Algorithm 1, then $2^{\ell_\lambda-5} - 2^{\ell_\lambda-8} \leq \delta(a_\lambda, a'_\lambda) \leq 2^{\ell_\lambda-4} + 2^{\ell_\lambda-8}$.*

Proof. Note that the server only moves to a'_λ when λ is primary. In this case, it holds that $V_\lambda^\dagger \cap B(a_\lambda, 2^{\ell_\lambda-5}) = \emptyset$. But, due to Proposition 9, this implies that $g_{v,\ell_\lambda-4}(t) \leq 0$ for every $v \in B(a_\lambda, 2^{\ell_\lambda-5})$; thus, for such v we have $y_{\ell_\lambda-4}(v, t) = 0$, and thus V_λ^\dagger is contained in the ring $B(a_\lambda, 2^{\ell_\lambda-4}) \setminus B(a_\lambda, 2^{\ell_\lambda-5})$. But, from the definition of a'_λ , $B(a'_\lambda, 2^{\ell_\lambda-8})$ must contain a point from V_λ^\dagger , which completes the proof. ◀

For a service λ , define the cost of the service, denoted $c(\lambda)$, to be the total movement of the server during λ plus the total amount by which the service decreases counters $g_{v,\ell}$.

► **Proposition 11.** $\text{ALG} \leq \sum_{\lambda \in \Lambda} c(\lambda)$.

Proof. To prove the proposition, we just have to prove that the total delay incurred by the algorithm is upper-bounded by the total amount by which counters are decreased in the algorithm. First, recall the assumption that the delay of requests tends to infinity as time advances; thus, every request is eventually served by the algorithm. Thus, once all requests are served, Proposition 9 implies that $g_{v,\ell} \leq 0$ for every $v \in G$ and level ℓ , which completes the proof. ◀

The following observation results from the fact that every counter $g_{v,\ell}$ is counted in exactly one dome (and can also easily be seen in Figure 1).

► **Observation 12.** *At any time t , and for every level ℓ , it holds that*

$$\sum_{\ell' \leq \ell} Y_{\ell'} = \sum_{v \in B(a(t), 2^\ell)} \sum_{\ell' \leq \ell} (g_{v,\ell'})^+.$$

► **Lemma 13.** $\text{ALG} \leq O(\sqrt{k}) \cdot \sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda} + O(\sqrt{k}) \cdot \sum_{\lambda \in \Lambda^c} 2^{\ell_\lambda}$.

Proof. Applying Proposition 11, it is enough to bound $\sum_{\lambda \in \Lambda} c(\lambda)$. We first bound the cost of tertiary services $\sum_{\lambda \in \Lambda^3} c(\lambda)$. Consider a tertiary service $\lambda \in \Lambda^3$; it incurs the following costs:

1. It zeroes any positive counters $g_{v,\ell}$ for every $v \in B(a_\lambda, 2^{\ell_\lambda})$ and $\ell \leq \ell_\lambda$. The cost of this is at most

$$\sum_{v \in B(a_\lambda, 2^{\ell_\lambda})} \sum_{\ell \leq \ell_\lambda} (g_{v,\ell})^+ = \sum_{\ell \leq \ell_\lambda} Y_\ell \leq 2^{\ell_\lambda+3},$$

where the equality is due to Observation 12, and the inequality is due to Proposition 8 and summing a geometric sequence.

74:10 Nearly-Optimal Algorithm for Non-Clairvoyant Service with Delay

2. It moves the server from a_λ to σ_λ and back (Line 19 of Algorithm 1). Since $\sigma_\lambda \in B(a_\lambda, 2^{\ell_\lambda-4})$, the cost of this is at most $2^{\ell_\lambda-3}$.

Overall, the cost of a tertiary service λ is $O(2^{\ell_\lambda})$.

Now, consider a *non-tertiary* service $\lambda \in \Lambda^1 \cup \Lambda^2$; there are at most $2 \cdot \sqrt{|V_\lambda|} + 1$ tertiary services assigned to λ' , as each such service raises $\beta(\lambda)$ by 1 and $\beta(\lambda)$ does not exceed $2 \cdot \sqrt{|V_\lambda|} + 1$; note that $2 \cdot \sqrt{|V_\lambda|} + 1 = O(\sqrt{k})$. Moreover, for a tertiary service λ' assigned to λ we have $\ell_{\lambda'} = \ell_\lambda + 4$. Overall, we have that the cost of tertiary services is $O(\sqrt{k}) \cdot \sum_{\lambda \in \Lambda^1 \cup \Lambda^2} 2^{\ell_\lambda}$.

Next, we bound the cost of non-tertiary services. The cost of a service $\lambda \in \Lambda^1 \cup \Lambda^2$ consists of the following terms:

1. It zeroes positive counters $g_{v,\ell}$ for every $v \in B(a_\lambda, 2^{\ell_\lambda})$ and $\ell \leq \ell_\lambda$. This cost can be bounded by $O(1) \cdot 2^{\ell_\lambda}$, using the same argument as for tertiary services.
2. In Algorithm 2, the algorithm moves the server (Line 5) and decreases counters (Line 10). From the guarantee of PCSOLVE in Lemma 4, the cost of moving the server is at most $2 \cdot 2 \cdot x_\lambda \cdot |Q_\lambda^\circ|$; the cost of decreasing counters is $(|V_\lambda| - |Q_\lambda^\circ|) \cdot x_\lambda$. Overall, the cost of this item is at most $O(1) \cdot |V_\lambda| \cdot x_\lambda$, which is at most $O(1) \cdot 2^{\ell_\lambda} \cdot \sqrt{|V_\lambda|}$. Using the fact that $|V_\lambda| \leq k$, the cost of this item is at most $O(\sqrt{k}) \cdot 2^{\ell_\lambda}$.
3. In Line 23 of Algorithm 1, the server might move to a new location a'_λ . However, $\delta(a_\lambda, a'_\lambda) \leq 2^{\ell_\lambda-4} + 2^{\ell_\lambda-8}$ due to Proposition 10. Thus, the cost of this movement is also $O(1) \cdot 2^{\ell_\lambda}$.

Overall, it holds that $c(\lambda) \leq O(\sqrt{k}) \cdot 2^{\ell_\lambda}$; combining this with the bound for tertiary services, we get

$$\text{ALG} \leq O(\sqrt{k}) \cdot \sum_{\lambda \in \Lambda^1 \cup \Lambda^2} 2^{\ell_\lambda}. \quad (1)$$

Finally, we bound $\sum_{\lambda \in \Lambda^2} 2^{\ell_\lambda} \leq O(1) \sum_{\lambda \in \Lambda^c} 2^{\ell_\lambda}$, which completes the proof. Consider a secondary service $\lambda \in \Lambda^2$, which certifies some prior service $\lambda' = \mu_\lambda$. Through Proposition 7, it holds that $\ell_{\lambda'} = \ell_\lambda - 4$, and thus $2^{\ell_\lambda} \leq O(1) \cdot 2^{\ell_{\lambda'}}$. Again through Proposition 7, it holds that λ' is only certified once (by λ); thus, summing over secondary services yields

$$\sum_{\lambda \in \Lambda^2} 2^{\ell_\lambda} \leq O(1) \cdot \sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda},$$

which combined with Equation (1) completes the proof. \blacktriangleleft

3.3 Charging Cylinders

It remains to bound the terms $\sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda}$ and $\sum_{\lambda \in \Lambda^c} 2^{\ell_\lambda}$. To do so, we restate the notions of charging balls and charging cylinders introduced in [35].

First, consider our goal. The optimal solution takes some tour through the graph over time, and we would like to map its resulting cost to the services of the algorithm. To this end, we give each service λ temporary ‘‘ownership’’ of a set of edges and edge parts, for some interval in time. We then charge the cost associated with λ (i.e. 2^{ℓ_λ}) to the total cost incurred by the optimal solution in traversing these edges and edge parts during the given interval. In doing so, one must ensure that no movement of OPT is charged to twice; this is ensured by having *disjointness*, which is the property that no edge part is owned by more than one service at any point in time.

Charging shapes and cylinders. A *charging shape* is a shape in the metric space that contains some edges and “parts” of edges. (For the sake of this definition, an edge e can be partitioned into parts whose weights sum up to $c(e)$.)

A *charging ball* centered at v of radius r – which, overloading notation, we denote $B(v, r)$ – is a charging shape such that:

- If both u, w are in $B(v, r)$ then $B(v, r)$ contains the entire edge $\{u, w\}$.
- If $u \in B(v, r)$ but $w \notin B(v, r)$, then $B(v, r)$ contains the part of $\{u, w\}$ connected to u of weight $r - \delta(v, u)$.

A *charging cylinder* is a pair (B, I) where I is a time interval and B is a charging shape.

Disjointness. We say that a set of charging shapes is disjoint if the parts of edges that appear in different charging shapes do not intersect. We say that a set of charging cylinders is disjoint if for every time t it holds that the charging shapes of the cylinders whose time intervals contain t are disjoint.

Intersections. For a set of edges $E' \subseteq E$ and charging shape B , we define $c(E' \cap B)$ to be the total weight of edges in E' that appear in B . For a cylinder $\gamma = (B, I)$, we define $c(\text{OPT} \cap \gamma)$ (called the *intersection* of OPT with γ) to be the total weight of edges in E' that appear in B , where E' is the set of edges traversed by OPT at least once during I .

Theorem 14 is due to [35], and is used to relate the total intersection of the cylinder set we construct to the cost of the optimal solution. The intuition for the theorem is the following: suppose we are given a set of cylinders Γ whose shapes are balls centered at N points. One can replace every such charging ball of radius r with a *perforated* charging ball, from which balls of small radius $\Theta(r/N^2)$ are removed around each of the N points; let Γ' be the set of cylinders after this perforation process. In Γ' , two cylinders whose shapes are (perforated) charging balls with radii r_1, r_2 where $r_1 \leq O(r_2/N^2)$ are guaranteed to be disjoint. Suppose that in the original set Γ , every two cylinders with radii within a constant factor from each other are also disjoint; after the perforation, we can partition Γ' into $\Theta(\log N)$ disjoint classes. This bounds the intersection of Γ' with OPT by at most $\Theta(\log N) \cdot \text{OPT}$. Moreover, one can observe that this perforation of a cylinder’s shape decreases its intersection with OPT by at most a constant times the charging ball’s radius; this relates the intersection of Γ with OPT to the intersection of Γ' with OPT . For more intuition regarding cylinders and Theorem 14, see Figure 2.

► **Theorem 14** ([35]). *Let Γ be a set of cylinders such that for every cylinder $\gamma \in \Gamma$ its charging shape $B(v_\gamma, r_\gamma)$ is a ball. Partition Γ into $\{\Gamma_i\}$ where for every $\gamma \in \Gamma_i$ it holds that $\lceil \log r_\gamma \rceil = i$. If for every i the set of cylinders Γ_i is disjoint, then for every constant $c > 0$ it holds that*

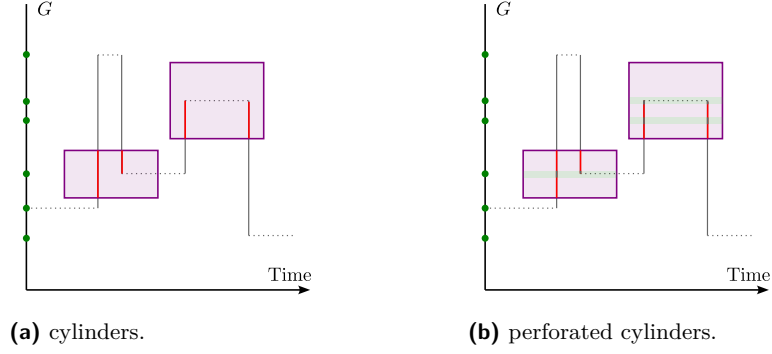
$$\sum_{\gamma \in \Gamma} c(\text{OPT} \cap \gamma) \leq O(\log k) \cdot \text{OPT} + c \cdot \sum_{\gamma \in \Gamma} r_\gamma,$$

where k is the number of centers used by cylinders in Γ .

Through constructing and analyzing such cylinders, we prove the following lemmas.

► **Lemma 15.** *It holds that $\sum_{\lambda \in \Lambda^t} 2^{\ell_\lambda} \leq O(\log k) \cdot \text{OPT}$.*

► **Lemma 16.** *It holds that $\sum_{\lambda \in \Lambda^c} 2^{\ell_\lambda} \leq O(\log k) \cdot \text{OPT}$.*



This figure illustrates the concept of cylinders used in this paper and in the proof of Theorem 14. Figure 2a shows a metric space G which is a line, the points of which appear on the vertical axis. The tour of the optimal server appears as an axis-aligned curve traversing space over time; the length of the solid parts of the curve correspond to the movement cost of the optimal solution. Two charging cylinders appear in the figure, whose charging shapes are balls; those balls appear as intervals in the one-dimensional metric space, and thus the cylinder appears as a rectangle in the figure. For each cylinder γ , the part of the optimal tour counted towards $c(\text{OPT} \cap \gamma)$ is shown in red. Note that as the shown cylinders are disjoint, the sum of $\sum_{\gamma} c(\text{OPT} \cap \gamma)$ lower-bounds OPT . Figure 2b shows the main tool used in proving Theorem 14 in [35], which is perforation; in essence, lower-radii charging balls are removed from each cylinder around each point in the metric space, enabling immediate disjointness with all cylinders of much lower radii.

■ **Figure 2** Illustration of cylinders.

The proof of Lemma 15 appears in Section A, while the proof of Lemma 16 appears in the remainder of this section. We now note that given these lemmas, the proof of the main theorem is complete.

Proof of Theorem 1. The proof is immediate from Lemma 13, Lemma 15 and Lemma 16. ◀

The remainder of this section proves Lemma 16. For every service $\lambda \in \Lambda^c$, we denote by Λ_λ the set of tertiary services assigned to λ . We also define $\Sigma_\lambda := \{\sigma_{\lambda'} | \lambda' \in \Lambda_\lambda\}$; note that $\Sigma_\lambda \subseteq E_\lambda$. Finally, we define $\Sigma_\lambda^\circ := V(\Sigma_\lambda)$.

First, we study the size of the set Σ_λ° .

► **Proposition 17.** *For every $\lambda \in \Lambda^c$, it holds that $|\Sigma_\lambda^\circ| = \lceil 2\sqrt{|V_\lambda|} \rceil$.*

Proof. First, note that $|\Sigma_\lambda|$ is exactly the final value of the counter $\beta(\lambda)$, as each tertiary service assigned to λ increases $\beta(\lambda)$ by 1 (Line 20 of Algorithm 1). A service assigned to λ will only be tertiary if $\beta(\lambda) < 2\sqrt{|V_\lambda|}$; otherwise, it would be secondary. Thus, the final value of the counter is at most $\lceil 2\sqrt{|V_\lambda|} \rceil$. However, we know that a secondary service is assigned to λ , as $\lambda \in \Lambda^c$. Thus, the final value of $\beta(\lambda)$ is exactly $\lceil 2\sqrt{|V_\lambda|} \rceil$, completing the proof. ◀

► **Definition 18.** *For a service $\lambda \in \Lambda^c$, define:*

- E_λ to be the set of pending requests of level at most ℓ_λ on points in Σ_λ° at t_λ .
- The certified time interval $I_c(\lambda) := (\min_{q \in E_\lambda} r_q, \max_{\lambda' \in \Lambda_\lambda} t_{\lambda'}]$.
- The certified cylinder $\gamma_c(\lambda) := (B(a_\lambda, 3 \cdot 2^{\ell_\lambda}), I_c(\lambda))$.

We also define the justified residual delay of λ , which is the amount $D_{c,\lambda}^* := \sum_{v \in V_\lambda^*} x_\lambda$, where $V_\lambda^* \subseteq \Sigma_\lambda^\circ$ is the subset of locations which were not visited by the optimal solution during $I_c(\lambda)$.

To prove that the intersection of a certified cylinder with the optimal solution is sufficiently large, we first restate a proposition from [35]. Intuitively, it states that if the set of requested terminals in a Steiner tree input is concentrated in a ball, a constant fraction of the cost required for connecting them must be incurred within a padded version of the ball, whose radius is larger by a constant factor.

► **Proposition 19** (restatement of Proposition 3.22 from [35]). *Let V be a set of locations that are contained in $B(v, r)$, for some location v and radius r , and let $G' \subseteq G$ be any subgraph connecting V . Then it holds that $c(G' \cap \mathcal{B}(v, 3r)) \geq \text{ST}(V)/2$.*

► **Proposition 20.** *It holds that $\sum_{\lambda \in \Lambda^c} D_{c,\lambda}^* \leq \text{OPT}$.*

Proof. We observe charging triplets of the form (v, ℓ, I) , where v is a location, ℓ is a level, and I is a time interval. Every certified service $\lambda \in \Lambda^c$ owns triplets. Specifically, for every service $\lambda \in \Lambda^c$, location $v \in V_\lambda^*$, and λ_v the tertiary service assigned to λ where $\sigma_{\lambda_v} \in v$, we say that λ owns the triplet (v, ℓ_λ, I_v) , where $I_v = [t_\lambda, t_{\lambda_v}]$ be the time interval between λ and the service λ_v . We now claim the following:

1. If λ owns triplet v, ℓ, I , then a delay of at least x_λ is incurred by requests of level ℓ on v during I . Moreover, these requests are pending in the optimal solution during the interval I .
2. No two triplets intersect. That is, there are no two services $\lambda_1, \lambda_2 \in \Lambda^c$ such that λ_1 owns (v, ℓ, I_1) , λ_2 owns (v, ℓ, I_2) and $I_1 \cap I_2 \neq \emptyset$.

The first claim implies that the optimal solution incurs a delay cost of $D_{c,\lambda}^*$ per service $\lambda \in \Lambda^c$, and the second claim implies that no delay cost is charged twice. Together, these two claims complete the proof.

Claim 1. Fix service $\lambda \in \Lambda^c$, location $v \in V_\lambda^*$, and tertiary service λ_v assigned to λ where $\sigma_{\lambda_v} \in v$; let $I_v = [t_\lambda, t_{\lambda_v}]$, and set $\ell := \ell_\lambda$. As $v \in \Sigma_\lambda^o \subseteq V_\lambda$, it holds immediately after λ that $g_{v,\ell_\lambda} \leq -x_\lambda$ (due to Line 17 of Algorithm 1 and Line 10 of Algorithm 2). However, at t_{λ_v} , it holds that $y_{\ell_{\lambda_v}-4}(v) > 0$; moreover, it holds that $v \in B(a_{\lambda_v}, 2^{\ell_{\lambda_v}-5})$, and thus $y_{\ell_{\lambda_v}-4}(v, t_{\lambda_v}) = (g_{v,\ell_{\lambda_v}-4}(t_{\lambda_v}))^+ = (g_{v,\ell_\lambda}(t_{\lambda_v}))^+$. Thus, the counter g_{v,ℓ_λ} has increased by at least x_λ during the time interval $I_v := (t_\lambda, t_{\lambda_v}]$.

Next, we claim that inside I_v , there was no non-tertiary service λ' of level at least ℓ such that $v \in B(a_{\lambda'}, 2^{\ell_\lambda})$. Assume otherwise, and note that $\sigma_{\lambda'}$ is pending on v , and is of level exactly ℓ ; thus, after λ' , it would either be served or have $\mu_{\sigma_{\lambda'}}$ be set to λ' , in contradiction to λ_v being assigned to λ .

As a result, we claim that every request on v of level ℓ_λ during I_v belongs to E_λ . First, note that immediately after λ , all pending level- ℓ requests on v are in E_λ . Suppose for contradiction that this is broken at some point; that is, a new level- ℓ request joins. This could only happen if its level is upgraded by some non-tertiary service during I_v , but there is no such service due to the previous claim.

Finally, note that the requests of E_λ are pending in the optimal solution during I_v , as the optimal server did not visit v during $I_c(\lambda)$; thus, the optimal solution incurs the same delay cost of at least x_λ .

Claim 2. Assume that two triplets $(v, \ell, I_1), (v, \ell, I_2)$ that are owned by services $\lambda_1, \lambda_2 \in \Lambda^c$, are such that $I_1 \cap I_2 \neq \emptyset$; note that $\ell_{\lambda_1} = \ell_{\lambda_2} = \ell$. Assume without loss of generality that λ_1 is the earlier service; then, $t_{\lambda_2} \in I_1$. Note that $v \in B(a_{\lambda_2}, 2^{\ell_{\lambda_2}})$; however, this cannot be, as no non-tertiary service λ of level at least ℓ can occur during I_1 such that $v \in B(a_\lambda, 2^{\ell_\lambda})$, as seen in the proof of the first claim. ◀

► **Proposition 21.** *For every $\lambda \in \Lambda^c$, it holds that $2^{\ell_\lambda-1} \leq c(\text{OPT} \cap \gamma_c(\lambda)) + D_{c,\lambda}^*$.*

Proof. Consider the set of locations $W := \Sigma_\lambda^\circ \setminus V_\lambda^*$, which is a subset of V_λ . These locations were not visited during λ , as the solution computed by PCSOLVE in Line 4 of Algorithm 2 does not include them. However, due to the guarantee of PCSOLVE in Lemma 4, this implies that

$$\text{ST}^*(W; a_\lambda) \geq x_\lambda \cdot |W| = 2^{\ell_\lambda} / \sqrt{|V_\lambda|} \cdot |W|.$$

Noting that $W \subseteq B(a_\lambda, 2^{\ell_\lambda})$, we have $\text{ST}^*(W) \geq \text{ST}^*(W; a_\lambda) - 2^{\ell_\lambda}$. Applying Proposition 19, denoting by G^* the subgraph traversed by OPT during $I_c(\lambda)$, it holds that $c(G^* \cap B(a_\lambda, 3 \cdot 2^{\ell_\lambda})) \geq \text{ST}^*(W)/2$. Combining the above, we get:

$$\begin{aligned} c(\text{OPT} \cap \gamma_c(\lambda)) + D_{c,\lambda}^* &\geq \text{ST}^*(W)/2 + (|\Sigma_\lambda^\circ| - |W|)x_\lambda \\ &\geq \frac{2^{\ell_\lambda-1}}{\sqrt{|V_\lambda|}} \cdot |W| - 2^{\ell_\lambda-1} + (|\Sigma_\lambda^\circ| - |W|) \frac{2^{\ell_\lambda}}{\sqrt{|V_\lambda|}} \\ &\geq 2^{\ell_\lambda-1} \cdot \left(\frac{|W|}{\sqrt{|V_\lambda|}} + \frac{|\Sigma_\lambda^\circ| - |W|}{\sqrt{|V_\lambda|}} - 1 \right) = 2^{\ell_\lambda-1} \cdot \left(\frac{|\Sigma_\lambda^\circ|}{\sqrt{|V_\lambda|}} - 1 \right) \geq 2^{\ell_\lambda-1} \end{aligned}$$

where the final inequality is due to the fact that $|\Sigma_\lambda^\circ| = \lceil 2\sqrt{|V_\lambda|} \rceil$, due to Proposition 17. ◀

► **Proposition 22.** *For every ℓ , the set of cylinders $\{\gamma_c(\lambda) \mid \lambda \in \Lambda^c \wedge \ell_\lambda = \ell\}$ is disjoint.*

The proof of Proposition 22 appears in Section B.

Proof of Lemma 16. It holds that

$$\sum_{\lambda \in \Lambda^c} 2^{\ell_\lambda} \leq 2 \sum_{\lambda \in \Lambda^c} (c(\text{OPT} \cap \gamma_c(\lambda)) + D_{c,\lambda}^*) \leq O(\log k) \cdot \text{OPT} + O(1) \cdot \text{OPT} = O(\log k) \cdot \text{OPT}$$

where the first inequality is due to Proposition 21, and the second inequality is due to Proposition 22 and Proposition 20. ◀

4 Conclusions

In this paper, we presented a $O(\min\{\sqrt{n} \log n, \sqrt{m} \log m\})$ -competitive algorithm for non-clairvoyant online service with delay, that is deterministic and runs in polynomial time. This nearly matches the lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt{m})$ on competitiveness that appear in [4].

However, the choice of metric space for the problem greatly affects the achievable competitive ratio for non-clairvoyant service with delay. The square-root lower bounds in [4] are obtained for a uniform metric space; however, for a metric space which is a line, an $O(\log n)$ -competitive non-clairvoyant algorithm exists [14]. Given this stark difference in competitive ratio, it would be interesting to identify a salient property of the metric space, and study competitiveness as a function of this property.

References

- 1 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul M. Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 1:1–1:20, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.1.

- 2 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Tuitou. Set cover with delay - clairvoyance is not required. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.8.
- 3 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. In *Approximation and Online Algorithms - 16th International Workshop, WAOA 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, pages 21–35, 2018. doi:10.1007/978-3-030-04693-4_2.
- 4 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalaya Panigrahi. Online service with delay. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 551–563, 2017. doi:10.1145/3055399.3055475.
- 5 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 301–320. SIAM, 2021. doi:10.1137/1.9781611976465.20.
- 6 Yossi Azar and Noam Tuitou. General framework for metric optimization problems with delay or with deadlines. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 60–71, 2019. doi:10.1109/FOCS.2019.00013.
- 7 Yossi Azar and Noam Tuitou. Beyond tree embeddings - a deterministic framework for network design with deadlines or delay. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1368–1379. IEEE, 2020. doi:10.1109/FOCS46700.2020.00129.
- 8 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jiri Sgall, Kim Thang Nguyen, and Pavel Veselý. New results on multi-level aggregation. *Theor. Comput. Sci.*, 861:133–143, 2021. doi:10.1016/j.tcs.2021.02.016.
- 9 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jiri Sgall, Nguyen Kim Thang, and Pavel Veselý. Online algorithms for multi-level aggregation. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 12:1–12:17, 2016. doi:10.4230/LIPICs.ESA.2016.12.
- 10 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, and Jan Marcinkowski. Online facility location with linear delay. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPICs*, pages 45:1–45:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.APPROX/RANDOM.2022.45.
- 11 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, Dorian Nogneng, and Jiri Sgall. Better approximation bounds for the joint replenishment problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 42–54, 2014. doi:10.1137/1.9781611973402.4.
- 12 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In *Approximation and Online Algorithms - 16th International Workshop, WAOA 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, pages 51–68, 2018. doi:10.1007/978-3-030-04693-4_4.
- 13 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. A match in time saves nine: Deterministic online matching with delays. In *Approximation and Online Algorithms - 15th International Workshop, WAOA 2017, Vienna, Austria, September 7-8, 2017, Revised Selected Papers*, pages 132–146, 2017. doi:10.1007/978-3-319-89441-6_11.

- 14 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. Online service with delay on a line. In *Structural Information and Communication Complexity - 25th International Colloquium, SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers*, pages 237–248, 2018. doi:10.1007/978-3-030-01325-7_22.
- 15 Carlos Fisch Brito, Elias Koutsoupias, and Shailesh Vaya. Competitive analysis of organization networks or multicast acknowledgment: How much to wait? *Algorithmica*, 64(4):584–605, 2012. doi:10.1007/s00453-011-9567-5.
- 16 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon. $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1235–1244, 2017. doi:10.1137/1.9781611974782.80.
- 17 Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 253–264, 2007. doi:10.1007/978-3-540-75520-3_24.
- 18 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: primal dual competitive algorithms. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 952–961, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347186>.
- 19 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.40.
- 20 Lindsey Deryckere and Seeun William Umboh. Online matching with set and concave delays. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPICs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.APPROX/RANDOM.2023.17.
- 21 Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. TCP dynamic acknowledgment delay: Theory and practice (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 389–398, 1998. doi:10.1145/276698.276792.
- 22 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. In *Algorithms and Complexity - 10th International Conference, CIAC 2017, Athens, Greece, May 24-26, 2017, Proceedings*, pages 209–221, 2017. doi:10.1007/978-3-319-57586-5_18.
- 23 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '92*, pages 307–316, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=139404.139468>.
- 24 Anupam Gupta, Amit Kumar, and Debmalaya Panigrahi. Caching with time windows. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.
- 25 Anupam Gupta, Amit Kumar, and Debmalaya Panigrahi. A hitting set relaxation for k -server and an extension to time-windows. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 504–515. IEEE, 2021. doi:10.1109/FOCS52979.2021.00057.
- 26 Anupam Gupta, Amit Kumar, and Debmalaya Panigrahi. Caching with time windows and delays. *SIAM J. Comput.*, 51(4):975–1017, 2022. doi:10.1137/20m1346286.

- 27 Kun He, Sizhe Li, Enze Sun, Yuyi Wang, Roger Wattenhofer, and Weihao Zhu. Randomized algorithm for MPMMD on two sources. In Jugal Garg, Max Klimm, and Yuqing Kong, editors, *Web and Internet Economics - 19th International Conference, WINE 2023, Shanghai, China, December 4-8, 2023, Proceedings*, volume 14413 of *Lecture Notes in Computer Science*, pages 348–365. Springer, 2023. doi:10.1007/978-3-031-48974-7_20.
- 28 Sungjin Im, Benjamin Moseley, Chenyang Xu, and Ruilong Zhang. Online dynamic acknowledgement with learned predictions. *CoRR*, abs/2305.18227, 2023. doi:10.48550/arXiv.2305.18227.
- 29 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgment and other stories about $e/(e-1)$. *Algorithmica*, 36(3):209–224, 2003.
- 30 Predrag Krnetic, Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. The k-server problem with delays on the uniform metric space. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPICs*, pages 61:1–61:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ISAAC.2020.61.
- 31 Ngoc Mai Le, Seoun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1594–1610. SIAM, 2023. doi:10.1137/1.9781611977554.ch59.
- 32 Mathieu Mari, Michal Pawlowski, Runtian Ren, and Piotr Sankowski. Online matching with delays and stochastic arrival times. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh, editors, *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pages 976–984. ACM, 2023. doi:10.5555/3545946.3598737.
- 33 Noam Touitou. Nearly-tight lower bounds for set cover and network design with deadlines/delay. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPICs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ISAAC.2021.53.
- 34 Noam Touitou. Frameworks for nonclairvoyant network design with deadlines or delay. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 105:1–105:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.105.
- 35 Noam Touitou. Improved and deterministic online service with deadlines or delay. 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 761–774. ACM, 2023. doi:10.1145/3564246.3585107.

A Proof of Lemma 15

To bound the cost of primary services, we first identify and focus on two subsets of services in Λ^1 , namely, $\Lambda^{1,s}$ and $\Lambda^{1,f}$.

► **Definition 23.** We define $\Lambda^{1,s} \subseteq \Lambda^1$ to be the subset of services in which the server did not move to a new location a'_λ (that is, Line 23 of Algorithm 1 did not run).

We define $\Lambda^{1,f} \subseteq \Lambda^1 \setminus \Lambda^{1,s}$ to be the subset of services λ in which $\delta(a^*(t_\lambda), a'_\lambda) \geq 2^{\ell_\lambda - 7}$; that is, in λ the server moved to a new location that is of distance at least $2^{\ell_\lambda - 7}$ from the optimal solution.

74:18 Nearly-Optimal Algorithm for Non-Clairvoyant Service with Delay

► **Proposition 24.** $\sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda} \leq O(1) \cdot \text{OPT} + O(1) \cdot (\sum_{\lambda \in \Lambda^{1,f}} 2^{\ell_\lambda} + O(1) \cdot \sum_{\lambda \in \Lambda^{1,s}} 2^{\ell_\lambda})$.

The proof of Proposition 24 is similar to that of Proposition A.11 in [35]. Intuitively, we again define a potential function proportional to the distance between the algorithm's server and optimal solution's server. For a service $\lambda \in \Lambda^1 \setminus (\Lambda^{1,f} \cup \Lambda^{1,s})$, the final movement in Line 23 of Algorithm 1 shrinks the distance between the algorithm and the optimal solution such that the resulting decrease in potential is at least 2^{ℓ_λ} .

Proof of Proposition 24. Consider the potential function $\phi(t) := 2^7 \cdot \delta(a(t), a^*(t))$. Note that $\phi(t)$ only takes non-negative values, and is initially 0. The potential function can change upon a movement of the algorithm's server (Line 23 of Algorithm 1) or the optimal solution's server. Note that the total increase in ϕ due to movements in OPT is at most $2^7 \cdot \text{OPT}$; also denote by Δ_λ the change to ϕ due to service $\lambda \in \Lambda^1$ in the algorithm. (Note that in non-primary services, the final location of the server is the same as its initial location, and thus they do not affect the potential function.) Thus, denoting by $\phi(\infty)$ the final value of the potential function, it holds that

$$0 \leq \phi(\infty) \leq \sum_{\lambda \in \Lambda^1} \Delta_\lambda + 2^7 \cdot \text{OPT}.$$

Adding $\sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda}$ to both sides yields

$$\sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda} \leq \sum_{\lambda \in \Lambda^1} (2^{\ell_\lambda} + \Delta_\lambda) + 2^7 \cdot \text{OPT}. \quad (2)$$

First, consider a service $\lambda \in \Lambda^1 \setminus (\Lambda^{1,f} \cup \Lambda^{1,s})$, and let $t := t_\lambda$. In λ , the server moves from a_λ to a'_λ , and it is guaranteed that $a^*(t) \in B(a'_\lambda, 2^{\ell_\lambda-7})$. Using Proposition 10, it holds $\delta(a_\lambda, a'_\lambda) \geq 2^{\ell_\lambda-5} - 2^{\ell_\lambda-8}$; thus, $\delta(a_\lambda, a^*(t)) \geq 2^{\ell_\lambda-5} - 2^{\ell_\lambda-7} - 2^{\ell_\lambda-8} \geq 2^{\ell_\lambda-6}$. However, after the movement of the algorithm's server to a'_λ , the distance between the algorithm's server and the optimum's server is at most $2^{\ell_\lambda-7}$; thus, the distance between the servers decreased by at least $2^{\ell_\lambda-7}$, and thus $\Delta_\lambda \leq -2^{\ell_\lambda}$. This implies that $2^{\ell_\lambda} + \Delta_\lambda \leq 0$ for every $\lambda \in \Lambda^1 \setminus \Lambda^{1,f} \cup \Lambda^{1,s}$; plugging into Equation (2), we get

$$\sum_{\lambda \in \Lambda^1} 2^{\ell_\lambda} \leq 2^7 \cdot \text{OPT} + \sum_{\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}} (2^{\ell_\lambda} + \Delta_\lambda). \quad (3)$$

Again using Proposition 10, for services $\lambda \in \Lambda^1$ where the server moves, it holds that $\delta(a_\lambda, a'_\lambda) \leq 2^{\ell_\lambda-4} + 2^{\ell_\lambda-8} \leq 2^{\ell_\lambda-3}$, and thus $\Delta_\lambda \leq \cdot 2^{\ell_\lambda+4}$. (When the server does not move, $\Delta_\lambda = 0$.) Thus, it holds that for every $\lambda \in \Lambda^1$, $2^{\ell_\lambda} + \Delta_\lambda \leq 17 \cdot 2^{\ell_\lambda}$. Plugging into Equation (3) completes the proof. ◀

► **Definition 25.** For a service $\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}$, define:

- R_λ to be the set of pending requests of level at most $\ell_\lambda - 4$ on points in V_λ^\dagger at t_λ .
- The primary time interval $I_p(\lambda) := (\min_{q \in R_\lambda} r_q, t_\lambda]$.
- The primary cylinder $\gamma_p(\lambda) := (B(a_\lambda, 2^{\ell_\lambda-3}), I_p(\lambda))$.

Further define $V_\lambda^* \subseteq V_\lambda^\dagger$ be the set of locations in V_λ^\dagger not visited by the optimal solution during $I_p(\lambda)$. Finally, define the justified residual delay of λ , which is the amount $D_{p,\lambda}^* := \sum_{v \in V_\lambda^*} y_{\ell_\lambda-4}(v, t_\lambda)$.

The proofs of the following three propositions appear in Section B.

► **Proposition 26.** $\sum_{\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}} D_{p,\lambda}^* \leq \text{OPT}$

Proof of Proposition 26. First a service $\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}$, and define $t := t_\lambda$. We first note that the delay cost in $D_{p,\lambda}^*$ is indeed incurred by the optimal solution. Fix a location $v \in V_\lambda^*$; by definition, $y_{\ell_\lambda-4}(v) = \sum_{\ell \leq \ell_\lambda-4} (g_{v,\ell}(t))^+ \leq \sum_{\ell \leq \ell_\lambda-4} \sum_{q \in R'_\ell} d_q(t)$, where $R'_\ell \subseteq R_\lambda$ is the subset of requests on v of level exactly ℓ (this inequality is due to Proposition 9). But, from the definition of $I_p(\lambda)$, the requests of R_λ on v are not served in the optimal solution at t , and thus the optimal solution incurred $\sum_{\ell \leq \ell_\lambda-4} \sum_{q \in R'_\ell} d_q(t)$ delay cost for those requests. Next, note that this delay cost is not charged to the optimal solution for more than one service; this is ensured by the zeroing of the delay counters in Line 17 of Algorithm 1. ◀

► **Proposition 27.** *For every $\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}$, it holds that $2^{\ell_\lambda-8} \leq c(\text{OPT} \cap \gamma_p(\lambda)) + D_{p,\lambda}^*$.*

Proof of Proposition 27. Consider a service $\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}$, and define $t := t_\lambda$, $B := B(a_\lambda, 2^{\ell_\lambda-4})$, $B' := B(a_\lambda, 2^{\ell_\lambda-3})$. Note that B is the ball that contains all requests in R_λ , and that B' is the shape used for the charging cylinder $\gamma_p(\lambda)$. Let a, a^* denote the locations of the algorithm's and optimal solution's servers at t , respectively.

Case 1: $\lambda \in \Lambda^{1,f}$. Define $a' := a'_\lambda$. Define $B^* := B(a', 2^{\ell_\lambda-8})$, $B^{**} := B(a', 2^{\ell_\lambda-7})$; note that $B^* \subseteq B^{**}$. From the choice of a' , it holds that $y_{\ell_\lambda-4}(B^*) \geq 2^{\ell_\lambda-5}$. However, $a^* \notin B^{**}$, from the fact that $\lambda \in \Lambda^{1,f}$. Thus, one of the following holds:

- The optimal solution did not visit B^* during $I_p(\lambda)$. In this case, it holds that $B^* \subseteq V_\lambda^*$, and thus $D_{p,\lambda}^* \geq y_{\ell_\lambda-4}(B^*) \geq 2^{\ell_\lambda-5}$, which completes the proof for this case.
- The optimal solution visited B^* during $I_p(\lambda)$; in this case, the optimal server moved a distance of at least $2^{\ell_\lambda-7} - 2^{\ell_\lambda-8} = 2^{\ell_\lambda-8}$ inside $B^{**} \setminus B^*$ during $I_p(\lambda)$ (since it was in a^* at the end of $I_p(\lambda)$). Now, note that $B^{**} \subseteq B'$, as Proposition 10 states that $\delta(a', a) \leq 2^{\ell_\lambda-4} + 2^{\ell_\lambda-8}$; this yields $c(\text{OPT} \cap \gamma_p(\lambda)) \geq 2^{\ell_\lambda-7} - 2^{\ell_\lambda-8} \geq 2^{\ell_\lambda-8}$, completing the proof.

Case 2: $\lambda \in \Lambda^{1,s}$. Denote $B^* := B(a^*, 2^{\ell_\lambda-8})$. From the definition of $\Lambda^{1,s}$, it holds that $y_{\ell_\lambda-4}(B \setminus B^*) \geq 2^{\ell_\lambda-5}$. Suppose the optimal solution's server does not visit $B \setminus B^*$ during $I_p(\lambda)$; then, it holds that $B \setminus B^* \subseteq V_\lambda^*$, and thus $D_{p,\lambda}^* \geq 2^{\ell_\lambda-5}$, completing the proof.

Otherwise, the optimal solution's server visited $B \setminus B^*$ during $I_p(\lambda)$. There are two subcases to consider:

- Suppose $\delta(a^*, a) \leq 2^{\ell_\lambda-3} - 2^{\ell_\lambda-8}$. In this case, $B^* \subseteq B'$. The optimal solution visited $B \setminus B^*$ during $I_p(\lambda)$, but was at a^* at t ; thus, it moved a distance of at least $2^{\ell_\lambda-8}$ inside B^* , and thus inside B' , during $I_p(\lambda)$. This implies that $c(\text{OPT} \cap \gamma_p(\lambda)) \geq 2^{\ell_\lambda-8}$, completing the proof.
- Suppose $\delta(a^*, a) > 2^{\ell_\lambda-3} - 2^{\ell_\lambda-8}$, and thus the distance of a^* from B is at least $2^{\ell_\lambda-4} - 2^{\ell_\lambda-8} \geq 2^{\ell_\lambda-8}$. Using a similar argument to the previous item, the optimal solution moved a distance of at least $2^{\ell_\lambda-8}$ inside the ring $B(a, 2^{\ell_\lambda-4} + 2^{\ell_\lambda-8}) \setminus B(a, 2^{\ell_\lambda-4})$ during $I_p(\lambda)$. Observing that this ring is contained in B' yields that $c(\text{OPT} \cap \gamma_p(\lambda)) \geq 2^{\ell_\lambda-8}$, completing the proof. ◀

► **Proposition 28.** *For every ℓ , the set of cylinders $\{\gamma_p(\lambda) \mid \lambda \in \Lambda^{1,f} \cup \Lambda^{1,s} \wedge \ell_\lambda = \ell\}$ is disjoint.*

Proof of Proposition 28. Suppose for contradiction that there exist $\lambda_1, \lambda_2 \in \Lambda^{1,f} \cup \Lambda^{1,s}$ where $\ell_{\lambda_1} = \ell_{\lambda_2} = \ell$ such that $I_p(\lambda_1) \cap I_p(\lambda_2) \neq \emptyset$ and $B(a_{\lambda_1}, 2^{\ell-3}) \cap B(a_{\lambda_2}, 2^{\ell-3}) \neq \emptyset$. Assume without loss of generality that $t_{\lambda_1} \leq t_{\lambda_2}$, and thus $t_{\lambda_1} \in I_p(\lambda_2)$. Then, observe that after λ_1 , all pending requests of level at most ℓ in $B(a_{\lambda_1}, 2^\ell)$ are upgraded to level ℓ . However, in λ_2 , the requests in R_{λ_2} are of level at most $\ell - 4$; moreover, the requests in R_{λ_2} are in $B(a_{\lambda_2}, 2^{\ell-4}) \subseteq B(a_{\lambda_1}, 2^\ell)$, where the containment is due to the fact that $\delta(a_{\lambda_1}, a_{\lambda_2}) \leq 2^{\ell-2}$ (otherwise, we contradict $B(a_{\lambda_1}, 2^{\ell-3}) \cap B(a_{\lambda_2}, 2^{\ell-3}) \neq \emptyset$). This implies that all requests in R_{λ_2} arrived after t_{λ_1} , yielding a contradiction to $t_{\lambda_1} \in I_p(\lambda_2)$. ◀

Proof of Lemma 15. It holds that

$$\begin{aligned} \sum_{\lambda \in \Lambda^{1,f}} 2^{\ell_\lambda} + \sum_{\lambda \in \Lambda^{1,s}} 2^{\ell_\lambda} &\leq \sum_{\lambda \in \Lambda^{1,f} \cup \Lambda^{1,s}} 2^8 \cdot (c(\text{OPT} \cap \gamma_p(\lambda)) + D_{p,\lambda}^*) \\ &\leq O(\log k) \cdot \text{OPT} + O(1) \cdot \text{OPT} = O(\log k) \cdot \text{OPT} \end{aligned}$$

where the first inequality is due to Proposition 27, and the second inequality is due to Proposition 28 and Proposition 20. \blacktriangleleft

B Omitted Proofs

Proof of Proposition 8. First, suppose a service λ occurs, and consider the point immediately after the loop containing Line 17 of Algorithm 1. At that time, we have $Y_i \leq \sum_{v \in B(a_\lambda, 2^i)} \sum_{i' \leq i} (g_{v,i'})^+ = 0$ for every $i \leq \ell_\lambda$. Moreover, for $i > \ell_\lambda$, it holds that $Y_i \leq 2^i$ (otherwise, dome i would be critical, and λ would have a higher level as a result).

We now prove the proposition as an invariant over time, noting that at time 0 it trivially holds. Note that the invariant cannot be broken by continuous delay increase; indeed, when Y_i exceeds 2^i , a service is immediately started which zeroes Y_i . Thus, the only possible event that could break the invariant is a movement by the server (Line 23 of Algorithm 1).

Suppose for contradiction that this happens after some (primary) service λ , at a time t^+ immediately after λ , as the server moves from a_λ to a'_λ . As before, consider the time during λ immediately after the loop containing Line 17 of Algorithm 1, and denote it by t^- . We also apply the superscripts $-$, $+$ to Y_i and $g_{v,i}$ to refer to their values at times t^- , t^+ respectively.

Consider any class i . If $i \leq \ell_\lambda - 1$, then note that $\delta(a'_\lambda, a_\lambda) \leq 2^{\ell_\lambda - 4} + 2^{\ell_\lambda - 8} < 2^{\ell_\lambda - 1}$, due to Proposition 10. Thus, $B(a'_\lambda, 2^i) \subseteq B(a_\lambda, 2^{\ell_\lambda})$. But, for every $v \in B(a_\lambda, 2^{\ell_\lambda})$ and $i' \leq \ell_\lambda$, it holds that $g_{v,i'}^+ = g_{v,i'}^- \leq 0$, and thus $Y_i^+ = 0$. Otherwise, $i \geq \ell_\lambda$; in this case, note that $B(a'_\lambda, 2^i) \subseteq B(a_\lambda, 2^{i+1})$. Thus:

$$\begin{aligned} Y_i^+ &\leq \sum_{i' \leq i} Y_{i'}^+ \\ &= \sum_{i' \leq i} \sum_{v \in B(a'_\lambda, 2^{i'})} (g_{v,i'})^+ \\ &\leq \sum_{i' \leq i} \sum_{v \in B(a_\lambda, 2^{i+1})} (g_{v,i'})^+ \\ &\leq \sum_{i' \leq i+1} \sum_{v \in B(a_\lambda, 2^{i+1})} (g_{v,i'})^+ \\ &= \sum_{i' \leq i+1} Y_{i'}^- \leq \sum_{i' \leq i+1} 2^{i'} \leq 2^{i+2}, \end{aligned}$$

where the first inequality stems from $\{Y_{i'}\}$ being non-negative and the two equalities use the definition of Y_i . The penultimate inequality uses the fact that $Y_{i'}^- \leq 2^{i'}$ for every i' , as noted in the beginning of the proof. \blacktriangleleft

Proof of Proposition 22. Fixing any ℓ , assume otherwise that there exist two cylinders $\gamma_c(\lambda_1), \gamma_c(\lambda_2) \in \{\gamma_c(\lambda) \mid \lambda \in \Lambda^c \wedge \ell_\lambda = \ell\}$ that are not disjoint. That is, it holds that $\delta(a_{\lambda_1}, a_{\lambda_2}) < 6 \cdot 2^\ell$, and also $I_c(\lambda_1) \cap I_c(\lambda_2) \neq \emptyset$. Let λ'_1, λ'_2 be the level- $(\ell + 4)$ secondary services that certify λ_1, λ_2 , respectively; without loss of generality, assume λ'_1 occurs before λ'_2 .

First, we claim that $\delta(a_{\lambda_1}, a_{\lambda'_1}) \leq 1.5 \cdot 2^\ell$. To prove this, consider $q_1 := \sigma_{\lambda'_1}$; it must be that $q_1 \in B(a_{\lambda_1}, 2^{\ell_{\lambda_1}}) \cap B(a_{\lambda'_1}, 2^{\ell_{\lambda'_1} - 5})$. Since $\ell_{\lambda_1} = \ell$ and $\ell_{\lambda'_1} = \ell + 4$, the claim holds.

Next, we claim that λ'_1 occurs before λ_2 . Assume otherwise, and consider the request $q_2 := \sigma_{\lambda_2}$. Since λ'_2 certifies λ_2 ; moreover, it holds that $q \in B(a_{\lambda_2}, 2^{\ell_{\lambda_2}})$. Thus,

$$\begin{aligned} \delta(q_2, a_{\lambda'_1}) &\leq \delta(q_2, a_{\lambda_2}) + \delta(a_{\lambda_2}, a_{\lambda_1}) + \delta(a_{\lambda_1}, a_{\lambda'_1}) \\ &\leq 2^\ell + 6 \cdot 2^\ell + 1.5 \cdot 2^\ell \leq 8.5 \cdot 2^\ell \leq 2^{\ell_{\lambda'_1}}. \end{aligned}$$

Thus, $q_2 \in V_{\lambda'_1}$; moreover, as q was pending at both t_{λ_2} and $t_{\lambda'_2}$, it is pending at $t_{\lambda'_1}$. Due to Line 9 of Algorithm 2, after λ'_1 , q_2 is either served or of level $\ell_{\lambda'_1} = \ell + 4$. But this is a contradiction to having level $\ell_{\lambda'_2} - 4 = \ell$ later on, at $t_{\lambda'_2}$; thus, λ'_1 must occur before λ_1 .

Thus, λ'_1 occurs between λ_1 and λ_2 . Note that λ'_1 occurs after all the tertiary services assigned to λ_1 , and thus λ'_1 occurs after $I_c(\lambda_1)$. We claim that λ'_1 also occurs before the release of every request in E_{λ_2} , and thus before $I_c(\lambda_2)$. This claim would yield that $I_c(\lambda_1), I_c(\lambda_2)$ are disjoint, contradicting the assumption that $\gamma_c(\lambda_1), \gamma_c(\lambda_2)$ are disjoint, and would thus conclude the proof of the proposition.

To prove the claim, suppose a request $q \in E_{\lambda_2}$ on some $v \in V_{\lambda_2}$ is released before $t_{\lambda'_1}$. Then, note that $\delta(a_{\lambda_2}, a_{\lambda'_1}) \leq \delta(a_{\lambda_2}, a_{\lambda_1}) + \delta(a_{\lambda_1}, a_{\lambda'_1}) \leq 6 \cdot 2^\ell + 1.5 \cdot 2^\ell = 7.5 \cdot 2^\ell$. Thus, $B(a_{\lambda_2}, 2^{\ell_{\lambda_2}}) \subseteq B(a_{\lambda'_1}, 2^{\ell_{\lambda'_1}})$, and thus $v \in V_{\lambda'_1}$. As q is pending at $t_{\lambda'_1}$, it is of level at least $\ell_{\lambda'_1} = \ell + 4$ after λ'_1 ; this is in contradiction to $\ell_q \leq \ell$ at λ_2 , completing the proof. ◀

Canonical Labeling of Sparse Random Graphs

Oleg Verbitsky 

Institut für Informatik, Humboldt-Universität zu Berlin, Germany

Maksim Zhukovskii 

School of Computer Science, University of Sheffield, UK

Abstract

We show that if $p = O(1/n)$, then the Erdős-Rényi random graph $G(n, p)$ with high probability admits a canonical labeling computable in time $O(n \log n)$. Combined with the previous results on the canonization of random graphs, this implies that $G(n, p)$ with high probability admits a polynomial-time canonical labeling whatever the edge probability function p . Our algorithm combines the standard color refinement routine with simple post-processing based on the classical linear-time tree canonization. Noteworthy, our analysis of how well color refinement performs in this setting allows us to complete the description of the automorphism group of the 2-core of $G(n, p)$.

2012 ACM Subject Classification Mathematics of computing → Random graphs; Mathematics of computing → Graph algorithms

Keywords and phrases Graph isomorphism, random graphs, canonical labeling, color refinement

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.75

Related Version *Full Version*: <https://arxiv.org/abs/2409.18109> [33]

Funding *Oleg Verbitsky*: Supported by DFG grant KO 1053/8–2. On leave from the IAPMM, Lviv, Ukraine.

Acknowledgements The authors would like to thank Michael Krivelevich for helpful discussions.

1 Introduction

On an n -vertex input graph G , a *canonical labeling algorithm* computes a bijection $\lambda_G : V(G) \rightarrow \{1, \dots, n\}$ such that if another graph G' is isomorphic to G , then the isomorphic images of G and G' under respective permutations λ_G and $\lambda_{G'}$ are equal. Given the labelings λ_G and $\lambda_{G'}$, it takes linear time to check whether G and G' are isomorphic. The existence of polynomial-time algorithms for testing isomorphism of two given graphs and, in particular, for producing a canonical labeling remain open. Babai's breakthrough quasi-polynomial algorithm for testing graph isomorphism [7] was subsequently extended to a canonical labeling algorithm of the same time complexity [8]. In the present paper, we address the canonical labeling problem for the Erdős-Rényi (or binomial) random graph $G(n, p)$. Recall that the vertex set of $G(n, p)$ is $\{1, \dots, n\}$, and each pair of vertices is adjacent with probability $p = p(n)$, independently of the other pairs.

Babai, Erdős, and Selkow [5] proved that the simple algorithmic routine known as *color refinement* (*CR* for brevity) with high probability produces a *discrete* coloring of the vertices of $G(n, 1/2)$, that is, a coloring where the vertex colors are pairwise different. Since the vertex colors are isomorphism-invariant, this yields a canonical labeling of $G(n, 1/2)$ by numbering the color names in the lexicographic order. Here and throughout, we say that an event happens for $G(n, p)$ *with high probability* (*whp* for brevity) if the probability of this event tends to 1 as $n \rightarrow \infty$. The result of [5] has a fundamental meaning: almost all graphs admit an easily computable canonical labeling and, hence, the graph isomorphism problem has low average-case complexity.



The argument of [5] can be extended to show [14, Theorem 3.17] that the CR coloring of $G(n, p)$ is whp discrete for all $n^{-1/5} \ln n \ll p \leq 1/2$. Note that it is enough to consider the case of $p \leq 1/2$ since $G(n, 1-p)$ has the same distribution as the complement of $G(n, p)$. Remarkably, the algorithm of Babai, Erdős, and Selkow performs only a bounded number of color refinement steps and, due to this, works in linear time.

A different algorithm suggested by Bollobás [12] works in polynomial time and whp produces a canonical labeling of $G(n, p)$ in a much sparser regime, namely when $\frac{(1+\delta) \ln n}{n} \leq p \leq 2n^{-11/12}$ for any positive constant δ . The next improvement was obtained by Czapka and Pandurangan [16] who proved that, in a bounded number of rounds, CR yields a discrete coloring of $G(n, p)$ whp when $\frac{\ln^4 n}{n \ln \ln n} \ll p \leq \frac{1}{2}$. Finally, Linial and Mosheiff [27] designed a polynomial time algorithm for canonical labeling of $G(n, p)$ when $\frac{1}{n} \ll p \leq \frac{1}{2}$. As shown by Gaudio, Rácz, and Sridhar [21], in the subdiapason $p \geq \frac{(1+\delta) \ln n}{n}$ for any fixed $\delta > 0$, a canonical labeling can still be provided by CR in a bounded number of rounds.

The decades-long line of research summarized above leaves open the question whether a random graph $G(n, p)$ admits efficient canonization in the regime $p = O(1/n)$. Note that the case of $p = o(1/n)$ is easy. Indeed, as long as $pn = 1 - \omega(n^{-1/3})$, whp $G(n, p)$ is a vertex-disjoint union of trees and *unicyclic* graphs (i.e., connected graphs containing exactly one cycle). Canonization of such graphs is tractable due to the classical linear-time canonical labeling algorithms for trees [1] and even planar graphs (see [6] for a survey of the early work on graph isomorphism covering these graph classes). Thus, efficient canonization remains unknown for all $p = p(n)$ such that, for some $C > 0$ and all n , $1 - Cn^{-1/3} \leq pn \leq C$ (even though $G(n, p)$ stays planar with a non-negligible probability as long as $pn = 1 + O(n^{-1/3})$; see [32]). Our first result closes this gap.

► **Theorem 1.** *If $p = O(1/n)$, then $G(n, p)$ whp admits a canonical labeling computable in time $O(n \log n)$.*

The development of canonical labeling algorithms for $G(n, p)$ is summarized in Table 1. The sources marked by * show that canonical labeling in the corresponding range can be obtained by CR in a constant number of refinement rounds. An inspection of the other algorithms reveals that all of them can be implemented as a combination of the 2-WL

■ **Table 1** Canonical labeling algorithms for random graphs in the full-scale Erdős-Rényi evolutionary model $G(n, p)$.

Edge probability	Algorithm
$p = \frac{1}{2}$	Babai, Erdős, and Selkow [5]*
$n^{-1/5} \ln n \ll p \leq 1/2$	Bollobás [14]*
$\frac{(1+\delta) \ln n}{n} \leq p \leq 2n^{-11/12}$	Bollobás [12]
$\frac{\ln^4 n}{n \ln \ln n} \ll p \leq \frac{1}{2}$	Czapka and Pandurangan [16]*
$\frac{(1+\delta) \ln n}{n} \leq p \ll n^{-5/6}$	Gaudio, Rácz, and Sridhar [21]*
$\frac{1}{n} \ll p \leq \frac{1}{2}$	Linial and Mosheiff [27]
$p = O(\frac{1}{n})$	this paper

(2-dimensional Weisfeiler-Leman) algorithm [34] with tree canonization. The 2-WL is an extension of CR which computes a canonical coloring of *pairs* of vertices. Thus, in these cases, canonical labeling can be obtained in time $O(n^3 \log n)$, matching the running time bound for 2-WL (see [23]). If $p = O(1/n)$, the running time is actually $O(n \log n)$ because our algorithm, as discussed below, uses CR along with simple pre- and post-processing.

A simple argument shows that Theorem 1, combined with the previous results, implies that the Erdős-Rényi random graph $G(n, p)$ whp admits an efficiently computable canonical labeling whatever the edge probability function $p(n)$.

► **Corollary 2.** *There exists a polynomial time algorithm that, for any function $p = p(n)$ with values in $[0, 1]$, whp produces a canonical labeling of $G(n, p)$.*

We now recall some highlights of the evolution of the random graph. Erdős and Rényi [19] proved their spectacular result that when p passes a certain threshold around $1/n$, then the size of the largest connected component in $G(n, p)$ rapidly grows from $\Theta(\log n)$ to $\Theta(n)$. A systematic study of the structure of connected components in the random graph when p is around the critical value $1/n$ was initiated in the influential paper of Bollobás [13]. For more details about the phase transition, see, e.g., [24, Chapter 5].

A connected graph is called *complex*, if it has more than one cycle. The union of all complex components of a graph G will be called the *complex part* of G , and the union of the other components will be referred to as the *simple part*. As we already mentioned, if $pn = 1 - \omega(n^{-1/3})$ then the complex part of $G(n, p)$ is whp empty. This is the so-called *subcritical phase*. In the *critical phase*, when $pn = 1 \pm O(n^{-1/3})$, the complex part of $G(n, p)$ whp has size $O_P(n^{2/3})$ and its structure is thoroughly described in [29, 30]. Here and below, for a sequence of random variables ξ_n and a sequence of reals a_n we write $\xi_n = O_P(a_n)$ if the sequence ξ_n/a_n is stochastically bounded¹. Finally, in the *supercritical phase*, when $pn = 1 + \omega(n^{-1/3})$, whp $G(n, p)$ contains a unique complex connected component and this component has size $\Theta(n^2(p - 1/n))$. In particular, when $p = O(1/n)$ and $p > (1 + \delta)/n$ for a constant $\delta > 0$ (we refer to this case as *strictly supercritical regime*), whp this component has linear size $\Omega(n)$. It is called the *giant component* as all the other connected components have size $O(\log n)$.

In general, the simple part of a graph G is easily canonizable by the known techniques, which reduces our problem to finding a canonical labeling for the complex part of G . Furthermore, recall that the 2-core of a graph H , which we will for brevity call just *core* and denote by $\text{core}(H)$, is the maximal subgraph of H that does not have vertices of degree 1. Equivalently, $\text{core}(H)$ can be defined as the subgraph of H obtained by iteratively pruning all vertices in H that have degree at most 1 until there are no more such vertices. Thus, if H is the (non-empty) complex part of G , then H consists of $\text{core}(H)$ and some (possibly empty) rooted trees planted at the vertices of the core. It follows that if we manage to canonically label the vertices of $\text{core}(H)$, then this labeling easily extends to a canonical labeling of the entire graph G .

Suppose that CR is run on H . In the most favorable case, it would output a vertex coloring discrete on $\text{core}(H)$. It turns out that, though not exactly true, this is indeed the case to a very large extent.

¹ I.e. for every $\varepsilon > 0$, there exists $C > 0$ and n_0 such that $\mathbb{P}(|\xi_n/a_n| > C) < \varepsilon$ for all $n \geq n_0$.

► **Theorem 3.** Let $G_n = G(n, p)$ and assume that $p = O(1/n)$. Let H_n denote the complex part of G_n and $C_n = \text{core}(H_n)$. When CR is run on H_n , then

1. CR assigns individual colors to all but $O_P(1)$ vertices in C_n ;
2. the other color classes whp have size 2;
3. whp, every such color class is an orbit of the automorphism group $\text{Aut}(H_n)$ consisting of two vertices with degree 2 in C_n .

► **Remark 4.** From our proofs it is easy to derive that, when $np = 1 + o(1)$, then CR distinguishes between all vertices of C_n whp.

Theorem 3 allows us to obtain an efficient canonical labeling algorithm for $G(n, p)$, as stated in Theorem 1, by combining CR with simple post-processing whose most essential part is invoking the linear-time tree canonization. Another consequence of Theorem 3 is that CR alone is powerful enough to solve the standard version of the graph isomorphism problem for the complex part of $G(n, p)$. Specifically, we say that a graph H is *identifiable* by CR if CR distinguishes H from any non-isomorphic graph H' (in the sense that CR outputs different multisets of vertex colors on inputs H and H'). It is not hard to see that H is identifiable by CR whenever the CR coloring of H is discrete. Fortunately, the properties of the CR coloring ensured by Theorem 3 are still sufficient for CR-identifiability.

► **Corollary 5.** Under the assumption of Theorem 3,

1. H_n is whp identifiable by CR and, consequently,
2. whp, G_n is identifiable by CR exactly when the simple part of G_n is identifiable.

The CR-identifiability of the simple part of a graph admits an explicit, efficiently verifiable characterization, which we give in Theorem 14. This characterization can be used to show that the random graph G_n is identifiable by CR with probability asymptotically bounded away from 0 and 1.

Our techniques for proving Theorems 1 and 3 can also be used for deriving a structural information about the automorphisms of a random graph. As proved by Erdős and Rényi [20] and by Wright [35], $G(n, p)$ for $p \leq 1/2$ is asymmetric, i.e., has no non-identity automorphism, if $np - \ln n \rightarrow \infty$ as $n \rightarrow \infty$. This result is best possible because if, $np - \ln n \leq \gamma$ for some constant $\gamma > 0$, then the random graph has at least 2 isolated vertices with non-vanishing probability. It is noteworthy that the asymmetry of $G(n, p)$ in the regime $p \geq \frac{(1+\delta)\ln n}{n}$ can be certified by the fact that CR coloring of $G(n, p)$ is discrete due to the aforementioned result of Gaudio, Rácz, and Sridhar [21]. In the diapason of p forcing $G(n, p)$ to be disconnected, the action of the automorphism group can be easily understood on the simple part and on the tree-like pieces of the complex part, and full attention should actually be given to the core of the complex part. Theorem 3 provides a pretty much precise information about the action of $\text{Aut}(G_n)$ on C_n . More subtle questions arise if, instead of considering the action of $\text{Aut}(G_n)$ on C_n , we want to understand the automorphisms of C_n itself. It is quite remarkable that the CR algorithm, applied to C_n rather than to H_n , can serve as a sharp instrument for tackling this problem (and, in fact, the proof of Theorem 3 is based on an analysis of the output of CR on C_n).

We begin with describing simple types of potential automorphisms of C_n (with the intention of showing that, whp, all automorphism of C_n are actually of this kind). If a vertex x has degree 2 in C_n , then it belongs to a (unique) path from a vertex s of degree at least 3 to a vertex t of degree at least 3 with all intermediate vertices having degree 2. We call such a path in C_n *pendant*. It is possible that $s = t$, and in this case we speak of a *pendant cycle*. Clearly, the reflection of a pendant cycle fixing its unique vertex of degree more than 2 is an automorphism of C_n . Furthermore, call two pendant paths *transposable* if they have the

same length and share the endvertices. Note that C_n has an automorphism transposing such paths (and fixing their endvertices). Let A_1 denote the set of the automorphisms provided by pendant cycles, and let A_2 be the set of the automorphisms provided by transposable pairs of pendant paths. Moreover, C_n can have a connected component consisting of two vertices of degree 3 and three pendant paths of pairwise different lengths between these vertices. Such a component has a single non-trivial automorphism, which contributes in $\text{Aut}(C_n)$. The set of such automorphisms of C_n will be denoted by A_3 .

Recall that an elementary abelian 2-group is a group in which all non-identity elements have order 2 or, equivalently, a group isomorphic to the group $(\mathbb{Z}_2)^k$ for some k .

► **Theorem 6.** *Let $G_n = G(n, p)$ and assume that $p = O(1/n)$. Let C_n be the core of the complex part of G_n .*

1. *The order of $\text{Aut}(C_n)$ is stochastically bounded, i.e., $|\text{Aut}(C_n)| = O_P(1)$.*
2. *Whp, $\text{Aut}(C_n)$ is an elementary abelian 2-group. Moreover, $A_1 \cup A_2 \cup A_3$ is a minimum generating set of $\text{Aut}(C_n)$.²*
3. *In addition,*
 - (a) *if $pn \geq 1 + \delta$ for a constant $\delta > 0$, then both A_1 and A_2 are non-empty with non-negligible probability, while $A_3 = \emptyset$ whp.*
 - (b) *If $pn = 1 + o(1)$ and $pn = 1 + \omega(n^{-1/3})$, then $A_1 \neq \emptyset$ with non-negligible probability, while $A_2 = A_3 = \emptyset$ whp.*
 - (c) *If $pn = 1 \pm O(n^{-1/3})$, then both A_1 and A_3 are non-empty with non-negligible probability, while $A_2 = \emptyset$ whp.*

This theorem makes a final step in the study of the automorphisms group of a random graph. Recall that H_n is whp empty when $np = 1 - \omega(n^{-1/3})$ and that $G(n, p)$ is connected and asymmetric when $np = \ln n + \omega(1)$. We, therefore, focus on the intermediate diapason. If $np \rightarrow \infty$ as $n \rightarrow \infty$, then the core of the giant component of $G(n, p)$ is whp still asymmetric, as proved independently by Łuczak [28] and Linial and Mosheiff [27]. Moreover, Łuczak described the automorphisms group of the core of the giant component of $G(n, p)$ when $np > \gamma$ for a large enough constant γ , and obtained an analogue of Theorem 6 for this case; see [28, Theorem 4]. Our Theorem 6 not only extends [28, Theorem 4] to the full range of $p = O(1/n)$ but also refines this result even for $np > \gamma$ by showing that $\text{Aut}(C_n)$ is actually an elementary 2-group. Another interesting observation is that some automorphisms of the core do not extend to automorphisms of the entire $G(n, p)$. Indeed, if $np = 1 + o(1)$, then whp $\text{Aut}(G(n, p))$ acts trivially on the core; see Remark 4.

Related work. As we already mentioned, Theorem 1 combined with the previous results on canonical labeling of $G(n, p)$ for $1/n \ll p \leq 1/2$ implies the existence of a polynomial-time canonical labeling algorithm succeeding on $G(n, p)$ whp for an *arbitrary* edge probability function $p = p(n)$. In this form, this result has been independently obtained by Michael Anastos, Matthew Kwan, and Benjamin Moore [2]. Another result in their paper describes the action of $\text{Aut}(G(n, p))$ on the core of $G(n, p)$, which follows also from our Theorem 3 and the results of Łuczak [28] and Linial and Mosheiff [27]. The techniques used in [2] and in our paper are completely different, though both proofs rely on color refinement. The two approaches have their own advantages. The method developed in [2] is used there also

² Consequently, whp C_n contains neither a triple of pairwise transposable paths, nor two isomorphic components with an automorphism in A_3 , nor a cyclic component with a single chord between diametrically opposite vertices. Moreover, whp no two pendant cycles in C_n share a vertex.

to show that color refinement is helpful for canonical labeling of the random graph when $p \gg 1/n$ and to study the smoothed complexity of graph isomorphism. Our method allows obtaining precise results on the automorphism group of the core (Theorem 6).

Immerman and Lander [23] showed a tight connection between CR-identifiability and definability of a graph in first-order logic with counting quantifiers. Corollary 5 can, therefore, be recast in logical terms as follows. If $p = O(1/n)$, then H_n is whp definable in this logic with using only two first-order variables (where the definability of a graph H means the existence of a formula which is true on H and false on any graph non-isomorphic to H). Definability of the giant component of $G(n, p)$ in the standard first-order logic (without counting quantifiers) was studied by Bohman et al. [11].

The rest of the paper and proof strategy. Section 2 begins with formal description of the color refinement algorithm in Subsection 2.1 and then, in Subsection 2.2, presents a useful criterion of CR-distinguishability in terms of universal covers. The concept of a universal cover appeared in algebraic and topological graph theory [10, 15, 31], and its tight connection to CR was observed in [3]. Subsection 2.3 pays special attention to the CR-identifiability of unicyclic graphs, which in Subsection 2.4 allows us to obtain an explicit criterion of CR-identifiability for general graphs in terms of the complex and the simple part of a graph. Finally, in Subsection 2.5 we use the relationship between CR and universal covers to prove useful properties of the CR-coloring of the core of an arbitrary graph.

Theorem 1 and Corollaries 2 and 5 are proved in Section 3. The proofs of Theorem 1 and Corollary 5 are based on Theorem 3. A crucial ingredient of the proof of Theorem 3 is our Main Lemma (Lemma 20). This lemma says that CR is unable to distinguish between two vertices in the core only if they lie either on pendant paths (with the same endvertices) transposable by an automorphism of the graph or on a pendant cycle admitting a reflection by an automorphism. Note that this statement alone, which is a part of the information provided by Theorem 3, is enough to derive Theorem 1 and Corollary 5.

The proof of Main Lemma is outlined in Section 4. It heavily relies on the notion of a kernel. The *kernel* $K(G)$ of a graph G is a multigraph obtained from $\text{core}(G)$ by contracting all pendant paths. That is, $K(G)$ is obtained via the following iterative procedure: at every step if there exists a vertex u with only two neighbors v_1, v_2 , we remove u with both incident edges and add the edge $\{v_1, v_2\}$ instead. Note that this transformation can lead to appearance of multiple edges and loops.

To prove that CR colors vertices of the core in the described manner, we use the contiguous models due to Ding, Lubetzky, and Peres [18] in strictly supercritical regime and due to Ding, Kim, Lubetzky, and Peres in critical regime [17]. They allow to transfer properties of random multigraphs with given degree sequences to the kernel of the giant component in the random graph. Another important ingredient in our proofs is the fact that in the kernel of the supercritical random graph there are whp no small complex subgraphs. We consider separately two types of vertices: first, we prove that CR colors differently all vertices such that their neighborhoods induce trees. This is done in Sections 4.1 and 4.2 for $p = 1 + \omega(n^{-1/3})$ and $p = 1 + O(n^{-1/3})$ respectively. Then, in Section 4.3, we prove that these vertices are helpful to distinguish between all the remaining vertices.

A complete proof of Theorem 3 and the proof of Theorem 6 are omitted due to the space constraints and can be found in the full version of the paper [33].

2 Color refinement: From identifiability to canonical labeling

2.1 Description of the CR algorithm

We now give a formal description of the *color refinement* algorithm (*CR* for short). CR operates on vertex-colored graphs but applies also to uncolored graphs by assuming that their vertices are colored uniformly. An input to the algorithm consists either of a single graph or a pair of graphs. Consider the former case first. For an input graph G with initial coloring C_0 , CR iteratively computes new colorings

$$C_i(x) = \left(C_{i-1}(x), \{ \{ C_{i-1}(y) \}_{y \in N(x)} \} \right), \quad (1)$$

where $\{ \{ \}$ denotes a multiset and $N(x)$ is the neighborhood of a vertex x . Denote the partition of $V(G)$ into the color classes of C_i by \mathcal{P}_i . Note that each subsequent partition \mathcal{P}_{i+1} is either finer than or equal to \mathcal{P}_i . If $\mathcal{P}_{i+1} = \mathcal{P}_i$, then $\mathcal{P}_j = \mathcal{P}_i$ for all $j \geq i$. Suppose that the color partition stabilizes in the t -th round, that is, t is the minimum number such that $\mathcal{P}_t = \mathcal{P}_{t-1}$. CR terminates at this point and outputs the coloring $C = C_t$. Note that if the colors are computed exactly as defined by (1), they will require exponentially long color names. To prevent this, the algorithm renames the colors after each refinement step, using the same set of no more than n color names.

If an input consists of two graphs G and H , then it is convenient to assume that their vertex sets $V(G)$ and $V(H)$ are disjoint. The vertex colorings of G and H define an initial coloring C_0 of the union $V(G) \cup V(H)$, which is iteratively refined according to (1). The color partition \mathcal{P}_i is defined exactly as above but now on the whole set $V(G) \cup V(H)$. As soon as the color partition of $V(G) \cup V(H)$ stabilizes, CR terminates and outputs the current coloring $C = C_t$ of $V(G) \cup V(H)$. The color names are renamed for both graphs synchronously.

We say that CR *distinguishes* G and H if $\{ \{ C(x) \}_{x \in V(G)} \} \neq \{ \{ C(x) \}_{x \in V(H)} \}$. If CR fails to distinguish G and H , then we call these graphs *CR-equivalent* and write $G \equiv_{\text{CR}} H$. A graph G is called *CR-identifiable* if $G \equiv_{\text{CR}} H$ always implies $G \cong H$.

2.2 Covering maps and universal covers

A surjective homomorphism from a graph K onto a graph G is a *covering map* if its restriction to the neighborhood of each vertex in K is bijective. We suppose that G is a finite graph, while K can be an infinite graph. If there is a covering map from K to G (in other terms, K covers G), then K is called a *covering graph* of G . Let G be connected. We say that a graph U is a *universal cover* of a graph G if U covers every connected covering graph of G . A universal cover $U = U^G$ of G is unique up to isomorphism. Alternatively, U^G can be defined as a tree that covers G . If G is itself a tree, then $U^G \cong G$; otherwise the tree U^G is infinite.

A straightforward inductive argument shows that a covering map α preserves the coloring produced by CR, that is, $C_i(u) = C_i(\alpha(u))$ for all i , where C_i is defined by (1). It follows that, if two connected graphs G and H have a common universal cover, i.e., $U^G \cong U^H$, then $\{ C(u) : u \in V(G) \} = \{ C(v) : v \in V(H) \}$. The converse implication is also true, as a consequence of the following lemma.

► **Lemma 7** (cf. Lemmas 2.3 and 2.4 in [26]). *Let U^G and U^H be universal covers of connected graphs G and H respectively. Furthermore, let α be a covering map from U^G to G and β be a covering map from U^H to H . For a vertex x of U^G and a vertex y of U^H , let U_x^G and U_y^H be the rooted versions of U^G and U^H with roots at x and y respectively. Then $U_x^G \cong U_y^H$ (isomorphism of rooted trees) if and only if $C(\alpha(x)) = C(\beta(y))$.*

The union of CR-identifiable graphs does not need be CR-identifiable. However, the concept of a universal cover allows us to state the following criterion, which is an extension of [4, Thm. 5.4] (see [4, p. 649] for details).

► **Lemma 8.** *Let G_1, \dots, G_k be connected CR-identifiable graphs and G be their vertex-disjoint union. Then G is CR-identifiable if and only if, for every pair of distinct i and j such that neither G_i nor G_j is a tree, the universal covers of G_i and G_j are non-isomorphic.*

2.3 Unicyclic graphs

2.3.1 Universal covers of unicyclic graphs

For a unicyclic graph G , its $\text{core}(G)$ is the set of vertices lying on the unique cycle of G . We use the notation $c(G) = |\text{core}(G)|$ for the length of this cycle. For a vertex x in $\text{core}(G)$, let G_x denote the subgraph of G induced by the vertices reachable from x along a path avoiding the other vertices in $\text{core}(G)$. This is obviously a tree. Moreover, we define G_x as a rooted tree with root at x . Let $t(x)$ denote the isomorphism class of the rooted tree G_x . We treat t as a coloring of $\text{core}(G)$ and write $R(G)$ to denote the cycle of G endowed with this coloring. Thus, $R(G)$ is defined as a vertex-colored cycle graph. It will also be useful to see $R(G)$ as a *circular word* over the alphabet $\{t(x) : x \in \text{core}(G)\}$; see, e.g., [22] and the references therein for more details on this concept in combinatorics on words. In fact, $R(G)$ is associated with two circular words, depending on one of the two directions in which we go along $R(G)$. However, the choice of one of the two words is immaterial in what follows.

Speaking about a *word*, we mean a standard, non-circular word. Two words are conjugated if they are obtainable from one another by cyclic shifts. A circular word is formally defined as the conjugacy class of a word. A word u is a period of a word v if $v = u^k$ for some $k \geq 1$. A word u is a period of a circular word w if u is a period of some representative in the conjugacy class of w . Note that if u is a period of a word v , then any conjugate of u is a period of some conjugate of v . This allows us to consider periods of circular words themselves being circular words. We define the *periodicity* $p(w)$ of a circular word w to be the minimum length of a period of w . It may be useful to keep in mind that a period of length $p(w)$ is also a period of every period of w ; cf. [22, Proposition 1] (note that our terminology is different from [22]).

For a unicyclic graph G , we define its periodicity by $p(G) = p(R(G))$, where $R(G)$ is seen as a circular word as explained above. Note that $p(G)$ is a divisor of $c(G)$ and that $1 \leq |\{t(x)\}_{x \in \text{core}(G)}| \leq p(G) \leq c(G)$.

Like trees, unicyclic graphs are also characterizable in terms of universal covers.

▷ **Claim 9.** A connected graph G is unicyclic if and only if U^G has a unique infinite path.

The unique infinite path subgraph of U^G will be denoted by $P(U^G)$. The structure of U^G is clear: The cycle of G is unfolded into the infinite path $P(U^G)$. Moreover, let α be a covering map from U^G to G . Then U^G is obtained by planting a copy of the rooted tree $G_{\alpha(x)}$ at each vertex x on $P(U^G)$. The path $P(U^G)$ will be considered being a vertex colored graph, with each vertex x colored by $t(\alpha(x))$.

The following observation is quite useful in what follows. Let α be a covering map from U^G to G . The restriction of α to $P(U^G)$ is a covering map from the vertex-colored path $P(U^G)$ to the vertex-colored cycle $R(G)$. Note that a covering map must preserve vertex colors.

► **Lemma 10.** *Let G and H be connected unicyclic graphs. Then $U^G \cong U^H$ if and only if the circular words $R(G)$ and $R(H)$ have a common period. Moreover, if $U^G \cong U^H$, then $p(G) = p(H)$.*

Proof. In one direction the statement is clear: if $R(G)$ and $R(H)$ have a common period, then $U^G \cong U^H$ by the definition. Let $U \cong U^G \cong U^H$ be a common universal cover of G and H . We can naturally see $P(U)$ as an infinite word. An arbitrary subword of length $p(G)$ of $P(U)$ is a period of $P(U)$, and the same is true for an arbitrary subword of length $p(H)$ of $P(U)$. It follows that $P(U)$ has a period $u = u_1 \dots u_q$ of length $q = \gcd(p(G), p(H))$. Indeed, it is sufficient to note that there exist integers β_1, β_2 such that $q = \beta_1 p(G) - \beta_2 p(H)$. Thus, for any u_0, u_q at distance q in $P(U)$, we get that $u_0 = u_{\beta_1 p(G)} = u_{q + \beta_2 p(H)} = u_q$.

If α and β are covering maps from U to G and H respectively, then $\alpha(u_1) \dots \alpha(u_q)$ is a period of $R(G)$ and $\beta(u_1) \dots \beta(u_q)$ is a period of $R(H)$. Since α and β preserve the vertex colors, we have the equality $\alpha(u_1) \dots \alpha(u_q) = \beta(u_1) \dots \beta(u_q)$. This also implies that, in fact, $q = p(G) = p(H)$. ◀

2.3.2 CR-identifiability of unicyclic graphs

▷ **Claim 11.** Let G be a connected unicyclic graph. Suppose that $G \equiv_{\text{CR}} H$ and H consists of connected components H_1, \dots, H_m . Then

1. $U^{H_i} \cong U^G$ for all i ,
2. every H_i is unicyclic, and
3. $c(G) = c(H_1) + \dots + c(H_m)$.

Proof. 1. Fix $i \in [m]$. Let U^G and $W = U^{H_i}$ and α^U, α^W be the respective covering maps. Let y be a vertex in U^{H_i} . Since G and H are CR-equivalent, U^G must contain a vertex x such that $C(\alpha^U(x)) = C(\alpha^W(y))$. By Lemma 7, $U_x \cong W_y$. It follows that $U \cong W$.

2. Immediately by Part 1 and Claim 9.

3. Fix a period of $R(G)$ and set $T = \sum_x |V(G_x)|$ where the summation goes over all x in this period (this definition obviously does not depend on the choice of the period). Let $p = p(G)$. Note that $|V(G)| = \frac{c(G)}{p} T$. By Part 1 and Lemma 10, we similarly have $|V(H_i)| = \frac{c(H_i)}{p} T$. The required equality now follows from the trivial equality $|V(G)| = |V(H_1)| + \dots + |V(H_m)|$. ◀

► **Lemma 12.** *A connected unicyclic graph G is CR-identifiable if and only if one of the following conditions is true:*

- $p(G) = 1$ and $c(G) \in \{3, 4, 5\}$,
- $p(G) = 2$ and $c(G) \in \{4, 6\}$,
- $p(G) = c(G)$.

Proof. (\Leftarrow) Suppose that a connected unicyclic graph G satisfies one of the three conditions and show that it is CR-identifiable. Assuming that H is CR-equivalent to G , we have to check that G and H are actually isomorphic.

Assume first that H is connected. If H is not unicyclic, then G and H have equal number of vertices but different number of edges. This implies that G and H have different degree sequences, contradicting the assumptions that $G \equiv_{\text{CR}} H$. Therefore, H must be unicyclic. By Part 3 of Claim 11, we have $c(G) = c(H)$. Along with Lemma 10, which is applicable because $U^G \cong U^H$ whenever $G \equiv_{\text{CR}} H$, this implies that $R(G) \cong R(H)$. The last relation, in its turn, implies that $G \cong H$.

Assume now that H is disconnected. Let H_1, \dots, H_m be the connected components of H . Combining Claim 11 and Lemma 10, we see that $p(G) = p(H_1) \leq c(H_1) < c(G)$. It follows that G satisfies one of the first two conditions. The restrictions on $c(G)$, however, rule out the equality in Part 3 of Claim 11. In particular, if $c(G) = 6$, then the only possible case is $m = 2$ and $c(H_1) = c(H_2) = 3$. However, it contradicts the equality $p(H_1) = p(H_2) = p(G) = 2$. Thus, the case of disconnected H is actually impossible, that is, all such H are distinguishable from G by CR. \blacktriangleleft

(\Rightarrow) Suppose that all three conditions are false. That is, either $p(G) = 1$ and $c(G) \geq 6$, or $p(G) = 2$ and $c(G) \geq 8$ (note that $c(G)$ is even in this case), or $3 \leq p(G) < c(G)$ (in the last case, $p(G)$ is a proper divisor of $c(G)$). In each case, $R(G)$ is CR-equivalent to a disjoint union of two shorter vertex-colored cycles R_1 and R_2 , both sharing the same period of length $p(G)$ with $R(G)$. Taking the connected unicyclic graphs H_1 and H_2 such that $R(H_1) \cong R_1$ and $R(H_2) \cong R_2$, we see that G is CR-equivalent to the disjoint union of H_1 and H_2 and is, therefore, not CR-identifiable. \blacktriangleleft

► **Lemma 13.** *Let G and H be connected unicyclic graphs with $c(H) \leq c(G)$. Assume that both G and H are CR-identifiable. Then $U^G \cong U^H$ if and only if $\{t(x) : x \in \text{core}(G)\} = \{t(x) : x \in \text{core}(H)\}$ and one of the following conditions is true:*

- $G \cong H$,
- $p(G) = p(H) = 1$ and $3 \leq c(H) < c(G) \leq 5$,
- $p(G) = p(H) = 2$ and $c(H) = 4$ while $c(G) = 6$.

Proof. (\Leftarrow) By Lemma 10.

(\Rightarrow) Let G and H be CR-identifiable connected unicyclic graphs with $c(H) \leq c(G)$. Assume that $U^G \cong U^H$. The equality $\{t(x) : x \in \text{core}(G)\} = \{t(x) : x \in \text{core}(H)\}$ immediately follows from Lemma 10. By the same corollary, $p(G) = p(H) = p$. If $p \geq 3$, then Lemma 12 yields the equality $c(G) = p(G) = p(H) = c(H)$, which readily implies $G \cong H$ by using Lemma 10 once again. If $p \leq 2$, then either $c(G) = c(H)$ and $G \cong H$ or $c(H) < c(G)$ and then, by Lemma 12, $c(H)$ and $c(G)$ are as claimed. \blacktriangleleft

2.4 A general criterion of CR-identifiability

Deciding whether a given graph is CR-identifiable is an efficiently solvable problem [4, 25]. For our purposes, it is beneficial to have a more explicit description of CR-identifiable graphs in terms of the complex and the simple part of a graph. We now derive such a description from the facts obtained for unicyclic graphs in the preceding subsection.

► **Theorem 14.**

1. *A graph G is CR-identifiable if and only if both the complex and the simple parts of G are CR-identifiable.*
2. *The simple part of G is CR-identifiable if and only if both of the following two conditions are true:*
 - (a) *every unicyclic component of G is CR-identifiable, i.e., is as described in Lemma 12;*
 - (b) *every two unicyclic components of G have non-isomorphic universal covers, i.e., there is no pair of connected components as described in Lemma 13.*

Proof.

1. If G is CR-identifiable, then its complex and simple parts are both CR-identifiable as a consequence a more general fact: The vertex-disjoint union of any set of connected components of G is CR-identifiable. This fact is easy to see directly, and it also immediately follows from Lemma 8.

In the other direction, assuming that the complex and the simple parts of G are CR-identifiable, we have to conclude that G is CR-identifiable. Lemma 8 reduces our task to verification that if H is a complex connected component of G and S is a simple connected component of G (a tree or a unicyclic graph), then the universal covers of H and S are non-isomorphic. The last condition follows from the fact that the universal cover of a tree is the tree itself and from Claim 9.

2. The second part of the theorem follows immediately from Lemma 8 due to the well-known fact [23] that every tree is CR-identifiable. \blacktriangleleft

2.5 Coloring the cores of general graphs

We conclude this section by collecting useful general facts about the CR-colors of vertices in the core of a graph. Let G be an arbitrary graph. If x is a vertex in $\text{core}(G)$, then in G we have a tree growing from the root x that shares with $\text{core}(G)$ only the vertex x . We denote this rooted tree by T_x .

\triangleright **Claim 15.** Let G and H be graphs. Let x be a vertex in $\text{core}(G)$ and y be a vertex in $\text{core}(H)$. If $T_x \not\cong T_y$, then $C(x) \neq C(y)$.

Proof. Clearly, it suffices to prove this for connected G and H . The condition $T_x \not\cong T_y$ readily implies that $U_x^G \not\cong U_y^H$, and the claim follows from Lemma 7. \blacktriangleleft

\triangleright **Claim 16.** Let G and H be two graphs (it is not excluded that $G = H$). For vertices $u \in V(G)$ and $v \in V(H)$ assume that $C(u) = C(v)$. Then $u \in \text{core}(G)$ if and only if $v \in \text{core}(H)$.

Proof. Assume that G and H are connected (the general case will easily follow). Let α_G be a covering map from U^G to G , and α_H be a covering map from U^H to H . Consider $x \in V(U^G)$ and $y \in V(U^H)$ such that $\alpha_G(x) = u$ and $\alpha_H(y) = v$. Note that $u \in \text{core}(G)$ if and only if there is a cycle in U^G containing x , and the same is true about v any y . This proves the claim because $U_x^G \cong U_y^H$ by Lemma 7. \blacktriangleleft

In our proofs, we will deal with cores that locally have a tree structure, that is, the balls of sufficiently large radii around most of its vertices induce trees. In this case, CR distinguishes vertices that have non-isomorphic neighborhoods.

\triangleright **Claim 17.** Let $B_r(v)$ denote the set of vertices at distance at most r from a vertex v . Let $v_1, v_2 \in V(G)$. If, for some r , the r -neighborhoods $B_r(v_1)$ and $B_r(v_2)$ induce non-isomorphic trees rooted in v_1 and v_2 respectively, then $C(v_1) \neq C(v_2)$.

Proof. This is a direct consequence of Lemma 7. \blacktriangleleft

Throughout the paper, we identify the vertex set of the kernel of G with the set of vertices of $\text{core}(G)$ having degrees at least 3 in the core. We now state another consequence of Lemma 7.

\triangleright **Claim 18.** Let G be a graph with minimum degree at least 2 and let K be its kernel. Let r be a positive integer. For $v \in V(K)$, let $\mathcal{B}_r^K(v)$ be the subgraph of K induced by the set of vertices at distance at most r from v in K . Let $\mathcal{B}_r(v) \subset G$ be the subdivided version of $\mathcal{B}_r^K(v)$. Let v_1, v_2 be vertices of K such that, for some r , graphs $\mathcal{B}_r(v_1), \mathcal{B}_r(v_2) \subset G$ are non-isomorphic trees rooted in v_1, v_2 . Then $C^G(v_1) \neq C^G(v_2)$.

Finally, we need the fact that the partition produced by CR on a graph refines the partition produced by CR on its core.

▷ **Claim 19.** Let u and v be vertices in $\text{core}(G)$. Let C and C' be the colorings produced by CR run on G and $\text{core}(G)$ respectively. If $C'(u) \neq C'(v)$, then also $C(u) \neq C(v)$.

Proof. Clearly, it is enough to prove the claim for a connected graph G . Let us assume towards a contradiction that $C(u) = C(v)$. Let α be a covering map from U^G to G . Let $x, y \in V(U^G)$ be such that $\alpha(x) = u$ and $\alpha(y) = v$. Due to Lemma 7, $U_x^G \cong U_y^G$. Therefore, $U_x^{\text{core}(G)} = \text{core}(U_x^G) \cong \text{core}(U_y^G) = U_y^{\text{core}(G)}$. But then, again by Lemma 7, $C'(u) = C'(v)$, a contradiction. \triangleleft

3 Proofs of main results

3.1 Derivation of Corollary 5 from Theorem 3

Part 1. Any isomorphism of graphs obviously respects their cores; cf. Claim 16. Note that the CR-color of any vertex x in the core C_n contains a complete information about the isomorphism type of the rooted tree T_x “growing” from this vertex (cf. Claim 15). This has the following consequence. Let C'_n denote the colored version of C_n where each vertex x is colored by the isomorphism type of T_x . Then H_n is CR-identifiable if and only if C'_n is CR-identifiable. In order to show that C'_n is CR-identifiable it suffices to show that C'_n is reconstructible up to isomorphism from the multiset of the vertex colors produced by CR on input C'_n . The CR-color partition of C'_n is equal to the restriction of the CR-color partition of H_n to C_n (recall Claim 16). Theorem 3, therefore, provides us with the following information (whp):³

- (a) every CR-color class of C'_n has size either 1 or 2,
- (b) every two equally colored vertices have degree 2,
- (c) every two equally colored vertices are transposable by an automorphism of C'_n .

Moreover, our Main Lemma (Lemma 20) ensures that H_n whp has no involutory automorphism of type A_3 described in Section 1. Along with this fact, the above conditions readily imply that the color classes of size 2 occur either “along” a pair of transposable pendant paths between two vertices of degree at least 3 or correspond to the reflectional symmetry of a pendant cycle. Here we use the notions introduced in Section 1 in the context of $\text{Aut}(C_n)$, which should now be refined by taking into account the coloring of C'_n .

If $\{u\}$ and $\{v\}$ are two color classes of size 1, then the colors $C(u)$ and $C(v)$ yield the information on whether the vertices u and v are adjacent or not. For color classes $\{u\}$ and $\{v, v'\}$, note that u and v are adjacent if and only if u and v' are adjacent. This adjacency pattern is as well reconstructible from the colors $C(u)$ and $C(v) = C(v')$. If $\{u, u'\}$ and $\{v, v'\}$ are two color classes of size 2, then they span either a complete or empty bipartite graph or a matching (for example, u is adjacent to v , u' is adjacent to v' , and there is no other edges between these color classes). Each of these three possible adjacency patterns is reconstructible from the colors $C(u) = C(u')$ and $C(v) = C(v')$. A crucial observation, completing the proof, is that all ways to put a matching between $\{u, u'\}$ and $\{v, v'\}$ lead to isomorphic graphs.

Part 2. This follows from part 1 by part 1 of Theorem 14.

³ Note that Conditions (b) and (c) are provided by Main Lemma. Condition (a) is not essential for the argument in Subsections 3.1 and 3.2, which easily extends to the case of more than two mutually transposable pendant paths.

3.2 Derivation of Theorem 1 from Theorem 3

Before proceeding to the proof, we remark that when we say that a canonical labeling algorithm succeeds on a random graph G_n , we mean that the algorithm works correctly on a certain efficiently recognizable (closed under isomorphisms) class of graphs \mathcal{C} such that G_n belongs to \mathcal{C} whp. Though not explicitly stated in the argument below, it will be clear that, in our case, \mathcal{C} is the class of all graphs satisfying the conditions of Theorem 3. Note that these conditions are easy to check after running CR on a graph.

First of all, we distinguish the complex and the simple parts of G_n and compute a canonical labeling of the simple part separately. This is doable in linear time. It remains to handle the complex part H_n .

It is enough to compute a suitable injective coloring of H_n and subsequently to rename the colors in their lexicographic order by using the labels that were not used for the simple part. To this end, we run CR on H_n . This takes time $O(n \log n)$ as CR can be implemented [9] in time $O((n+m) \log n)$, where m denotes the number of edges (which is linear for the sparse random graph under consideration). Then we begin with coloring the vertices of the core C_n . Theorem 3 along with Claim 16 ensures that the vertices of degree at least 3 already received individual colors. The duplex colors occur along transposable pendant paths and pendant cycle (like in Section 3.1, these notions are understood with respect to H_n rather than to C_n alone). To make such vertex colors unique, we keep the original colors along one of two transposable paths and concatenate their counterparts in the other path with a special symbol. We proceed similarly with symmetric pendant cycles. In this way, every vertex x in the core C_n receives an individual color $\ell(x)$. In the last phase, we compute a canonical labeling for each tree part T_x of H_n , regarding T_x as a tree rooted at x . This coloring is not injective yet because some T_x and T_y can be isomorphic. This is rectified by concatenating all vertex colors in T_x with $\ell(x)$.

3.3 Proof of Corollary 2

As well known, if $1/n \ll p \leq 1/2$ then the core of the giant component of $G(n, p)$ coincides with the core of the entire graph. Due to the classical linear-time algorithms for canonical labeling of trees, this observation reduces canonical labeling of $G(n, p)$ with $1/n \ll p \leq 1/2$ to canonical labeling of its core.

Linial and Mosheiff [27] suggested an algorithm A_1 that, for any p with $\frac{1}{n} \ll p(n) < n^{-2/3}$, whp labels canonically $G(n, p)$ in time $O(n^4)$ by distinguishing between all vertices of the core. In [16], it was proved that, if $\frac{\ln^4 n}{n} \leq p \leq \frac{1}{2}$, then CR whp distinguishes between all vertices of the entire $G(n, p)$. Finally, our Theorem 1 provides an algorithm A_2 that, for any $p = O(1/n)$, whp labels canonically $G(n, p)$ in time $O(n \ln n)$. Now, consider the following algorithm A:

1. Run CR. If it colors differently all vertices, then halt and output the canonical labeling produced by CR.
2. If the algorithm does not halt in Step 1, then run A_1 . If it succeeds (i.e., colors differently all vertices in the core of the input graph), then halt and output the labeling produced by A_1 .
3. If the algorithm does not halt in Steps 1 and 2, then run A_2 and output the labeling it produces (or give up if A_2 fails).

Let us show that the algorithm A succeeds whp for any p with $p(n) \leq 1/2$. Assume, to the contrary, that there exist a constant $\varepsilon > 0$ and a sequence $(n_k)_{k \in \mathbb{N}}$ such that

$$\mathbb{P}(\text{A fails on } G(n_k, p(n_k))) > \varepsilon$$

for all k . If there is a subsequence $(n_{k_i})_{i \in \mathbb{N}}$ and a constant $C > 0$ such that $p(n_{k_i}) < C/n_{k_i}$ for all i , then we get a contradiction with the performance of the algorithm A_2 . Therefore, $p(n_k) \gg \frac{1}{n_k}$. If there is a subsequence $(n_{k_i})_{i \in \mathbb{N}}$ such that $p(n_{k_i}) < n_{k_i}^{-2/3}$ for all i , then we get a contradiction with the performance of the algorithm A_1 . It follows that $p(n_k) \geq n_k^{-2/3}$ for all k . This, however, contradicts the result of [16] that CR in this regime produces a discrete coloring of $G(n, p)$ whp.

In order to obtain canonical labeling, whp, for all p with $p(n) \in [0, 1]$, we run the algorithm A on input G and if it fails, then we run A once again on the complement of G .

4 CR-coloring of the random graph

In this section, we state and prove our Main Lemma that describes the output of CR on the random graph. Given a graph G , we call vertices u and v in $\text{core}(G)$ *interchangeable*, if

- they both have degree 2 in $\text{core}(G)$,
- u and v belong to a cycle $F \subset \text{core}(G)$ with the following property: there exists a vertex w on the cycle such that w has degree at least 3 in $\text{core}(G)$, $d_F(u, w) = d_F(v, w)$, and all the other vertices on the cycle, but the vertex opposite to w when $|V(F)|$ is even, have degree 2 in $\text{core}(G)$. In other words, u and v either belong to a pendant cycle or to two transposable pendant paths, and the respective transposition replaces u and v .

► **Lemma 20 (Main Lemma).** *Let $\gamma > 1$ be a constant, $pn \leq \gamma$, and $G_n = G(n, p)$. Let H_n be the union of complex components in G_n , and C_n be its core. If CR is run on H_n , then whp any pair of vertices in C_n receiving the same color is interchangeable. Under the condition $pn = 1 + \omega(n^{-1/3})$, this is true also if CR is run on C_n .*

The proof of Main Lemma is given in Sections 4.1–4.3. We consider separately large p (supercritical phase) and small p (critical phase). In both cases, we specify good sets of vertices and show that all vertices from good sets are distinguished by CR. This is done in Section 4.1 for large p and Section 4.2 for small p . Finally, in Section 4.3 we complete the proof: we show that distinguishing between good vertices in the core is sufficient to distinguish between all pairs in the core that are not interchangeable.

For a graph G , $d_G(u, v)$ is the shortest-path distance between u and v in G . Sometimes, when the graph is clear from the context, we omit the subscript G . For a vertex v and a real number r , we denote by $\mathcal{B}_r^G(v)$ the ball of radius r around v in G , i.e., the graph induced on the set of all vertices at distance at most r from v in G . For a non-negative integer r , we denote by $\mathcal{S}_r^G(v) \subset \mathcal{B}_r^G(v)$ the sphere of radius r around v in G , i.e., the graph induced on the set of all vertices at distance exactly r from v in G .

For a connected graph G , its *excess* is the difference between the number of edges and the number of vertices. In particular, a tree has excess -1 . We call ℓ -*complex* a connected graph with excess ℓ . The *total excess* of a graph without unicyclic components is the sum of excesses of all its components.

4.1 Distinguishing good vertices in the core in the supercritical and strictly supercritical phases

In this subsection, we let $p = p(n)$ be such that $\gamma \geq np = 1 + \omega(n^{-1/3})$ for some constant $\gamma > 1$. Denote $\delta_n := np - 1$. We denote the kernel and the core of the giant component of $G_n \sim G(n, p)$ by K_n and C_n . Let C^{core} be the coloring produced by CR on C_n .

We assign to every edge e of C_n the weight $1/\ell$, where $\ell - 1$ is the number of vertices that subdivide the edge of the kernel e belongs to. The weight of a path is the sum of weights of its edges. For $u, v \in V(C_n)$, let $d^f(u, v)$ be the *fractional distance* between u and v , i.e. the minimum weight of a path between u and v . We denote the respective metric space by \mathcal{M}_n .

Fix a positive real s . Let D_s be the set of all $v \in V(C_n)$ such that the ball around v in \mathcal{M}_n of radius s induces an acyclic graph. For every vertex $v \in D_s$ and integer $r < s$, let $\mathcal{P}_r(v)$ be the multiset of lengths of edge-disjoint paths from C_n that are produced by subdividing edges $\{x, y\} \in E(K_n)$, where $d^f(x, v) \leq r$ while $d^f(y, v) > r$. We will need the following facts.

▷ **Claim 21.** Let $s \geq 0.6(\ln(\delta_n^3 n))^{2/3}$. Whp for any two different $u, v \in D_s \cap V(K_n)$, there exists an integer $r \leq 0.5(\ln(\delta_n^3 n))^{2/3}$ such that the multisets $\mathcal{P}_r(u)$ and $\mathcal{P}_r(v)$ are different.

Proof. We fix $s \geq 0.6(\ln(\delta_n^3 n))^{2/3}$ and let $D := D_s$. We prove this claim in the contiguous models \tilde{G}_n , defined in [17, Thm. 2] and [18, Thm. 1] and then use these theorems to conclude that it also holds in G_n . So, in what follows, $\tilde{K}_n = K(\tilde{G}_n)$, $\tilde{C}_n = C(\tilde{G}_n)$, and $\tilde{D} = D(\tilde{K}_n)$.

Let us expose \tilde{K}_n and let $u, v \in \tilde{D} \cap V(\tilde{K}_n)$. As proved in [17, 18], whp $N = |V(\tilde{K}_n)| = \Theta(\delta_n^3 n)$. Assume first that the distance between u and v is at most $0.4(\ln(\delta_n^3 n))^{2/3}$ in \tilde{K}_n . Let P be the shortest path between u and v – it is unique due to the definition of \tilde{D} . Let v' be a neighbor of v in \tilde{K}_n that does not belong to P . Then, by the definition of \tilde{D} , $|\mathcal{S}_r^{\tilde{K}_n}(v') \setminus \mathcal{B}_r^{\tilde{K}_n}(v)| \geq 2^r$ for all $r \in [0.4(\ln(\delta_n^3 n))^{2/3}, 0.5(\ln(\delta_n^3 n))^{2/3}]$. Since $u, v \in \tilde{D}$, we have that $\mathcal{B}_s^{\tilde{K}_n}(u)$ and $\mathcal{B}_s^{\tilde{K}_n}(v)$ are trees. It immediately implies, that for every such r , $|\mathcal{S}_{r+1}^{\tilde{K}_n}(v) \setminus \mathcal{B}_{r+1}^{\tilde{K}_n}(u)| \geq 2^r$.

We then generate subdivisions of the edges of the kernel from the definition of \tilde{G}_n in the following order: for every $r = \lceil 0.4(\ln(\delta_n^3 n))^{2/3} \rceil, \dots, \lfloor 0.5(\ln(\delta_n^3 n))^{2/3} \rfloor$, we, first, subdivide all edges growing from $\mathcal{B}_{r+1}^{\tilde{K}_n}(u)$ outside of the ball, and then all edges growing from $\mathcal{S}_{r+1}^{\tilde{K}_n}(v)$ outside of $\mathcal{B}_{r+1}^{\tilde{K}_n}(v)$. Notice that all sets $\mathcal{S}_{r+1}^{\tilde{K}_n}(v)$ are disjoint for different r . For every r , as soon as the edges that correspond to the vertex u are subdivided, the event that $\mathcal{P}_{r+1}(u) = \mathcal{P}_{r+1}(v)$ immediately implies that the *random* multiset of lengths of paths from \tilde{C}_n , that are produced by subdividing edges from \tilde{K}_n that grow from $\mathcal{B}_{r+1}^{\tilde{K}_n}(v)$ outside, should be equal to a predefined value. This multiset has size at least 2^r . Since the geometric random variables considered in [17, 18] do not have atoms with probability measure $1 - o(1)$, the latter event has probability at most $2^{-\Theta(r)}$ due to the de Moivre–Laplace local limit theorem. Eventually,

$$\begin{aligned} \mathbb{P}\left(\mathcal{P}_{r+1}(u) = \mathcal{P}_{r+1}(v) \text{ for all } r \in \left[0.4(\ln(\delta_n^3 n))^{2/3}, 0.5(\ln(\delta_n^3 n))^{2/3}\right]\right) &\leq \\ &\leq \exp\left(-\Theta((\log(\delta_n^3 n))^{4/3})\right). \end{aligned}$$

Assume now that the distance between u and v is bigger than $0.4(\ln(\delta_n^3 n))^{2/3}$ in \tilde{K}_n . Then, by the definition of \tilde{D} , sets $\mathcal{B}_{0.2(\ln(\delta_n^3 n))^{2/3}}^{\tilde{K}_n}(v)$ and $\mathcal{B}_{0.2(\ln(\delta_n^3 n))^{2/3}}^{\tilde{K}_n}$ are disjoint and sets $\mathcal{S}_r^{\tilde{K}_n}(v)$ have size at least 2^r for all $r \in [0.15(\ln(\delta_n^3 n))^{2/3}, 0.2(\ln(\delta_n^3 n))^{2/3} - 1]$. As above, we get that $\mathcal{P}_r(u) = \mathcal{P}_r(v)$ for all $r \in [0.15(\ln(\delta_n^3 n))^{2/3}, 0.2(\ln(\delta_n^3 n))^{2/3} - 1]$ with probability at most $\exp(-\Theta((\log(\delta_n^3 n))^{4/3}))$.

The union bound over all pairs $u, v \in \tilde{D}$ along with [17, Thm. 2] and [18, Thm. 1] completes the proof. ◁

▷ **Claim 22.** Let $s^* := \lfloor (\ln(\delta_n^3 n))^{2/3} \rfloor$ and $D = D_{s^*}$. Whp, $C^{\text{core}}(u) \neq C^{\text{core}}(v)$ for any distinct $u, v \in D$.

Proof. Assume that the assertion of Claim 21 holds for $s = s^*$ and $s = s^* - 1$ deterministically. Let $u, v \in D \cap V(K_n)$. Let B_u and B_v be the subdivided versions of $\mathcal{B}_{s^*}^{K_n}(u)$ and $\mathcal{B}_{s^*}^{K_n}(v)$ in C_n . Since $B_u \not\cong B_v$ due to the conclusion of Claim 21, we get that $C^{\text{core}}(u) \neq C^{\text{core}}(v)$ due to Claim 18. It remains to consider the case $v \in D \setminus V(K_n)$ and $v \neq u \in D$. Assume towards contradiction that $C^{\text{core}}(u) = C^{\text{core}}(v)$. Then, both u and v have degree 2 in C_n . In particular, $u \notin V(K_n)$. Consider the edges e_u, e_v of K_n that u and v subdivide. Let P_u, P_v be the subdivided versions of e_u, e_v . Due to the assertion of Claim 21 applied to $s = s^* - 1$, we get that all vertices of K_n from $e_u \cup e_v$ have different colors. On the other hand, by the definition of CR, the neighbors of u should have exactly the same color as the neighbors of v . Thus, by induction, we get that the entire paths P_u, P_v are colored identically. It may only happen if the endpoints of P_u coincide with the endpoints of P_v . By the definition of D , it means that $P_u = P_v =: P$. Since the endpoints of P are colored in different colors, it can be easily shown by induction that all vertices in P are also colored in different colors. Thus, $u = v$, yielding a contradiction. \triangleleft

4.2 Distinguishing good vertices in the core in the critical regime

Let A be a large positive number. Let $1 - n^{-1/3} \ln n \leq pn = 1 + o(1)$. Whp any complex component in $G_n \sim G(n, p)$ has size at least $100A \ln n$ due to the following well-known fact.

\triangleright **Claim 23.** Let $\gamma > 1$, $np \leq \gamma$, and $G_n \sim G(n, p)$. There exists $\varepsilon = \varepsilon(\gamma)$ such that $\varepsilon \rightarrow \infty$ as $\gamma \rightarrow 1$ and whp any connected subgraph of G_n of size at most $\varepsilon \ln n$ is not complex.

Let us say that a path $u_1 \dots u_k$ *extends* the path $v_1 \dots v_k$ if, for some $i \in \{2, \dots, k\}$ the sets $\{v_1, \dots, v_{i-1}\}$ and $\{u_{k-i+2}, \dots, u_k\}$ are disjoint and $u_1 = v_i, \dots, u_{k-i+1} = v_k$. For convenience, we assume that this notion is closed under rotations of paths, i.e. if $u_1 \dots u_k$ extends $v_1 \dots v_k$, then it also extends $v_k \dots v_1$ and we also say that $u_k \dots u_1$ extends both $v_1 \dots v_k$ and $v_k \dots v_1$ in this case. We call two paths $v_1 \dots v_k$ and $u_1 \dots u_k$ *weakly disjoint*, if they are either vertex-disjoint or one path extends the other one.

\triangleright **Claim 24.** Whp in G_n there are no two weakly disjoint paths $v_1 \dots v_k$ and $u_1 \dots u_k$ of length $k = \lfloor A \ln n \rfloor$ such that, for every $i \in \{2, \dots, k-1\}$, v_i has degree 2 if and only if u_i has degree 2.

Proof. Due to Claim 23, whp in G_n there are no complex subgraphs with at most $2k$ vertices.

For a path $P = v_1 \dots v_k$ in G_n , let us consider a binary word $w(P) = (w_2, \dots, w_{k-1})$ defined as follows: $w_i = 1$ if and only if v_i has degree 2 in G_n . Notice that, if a path $u_1 \dots u_k$ extends the path $v_1 \dots v_k$ so that $u_1 = v_i, \dots, u_{k-i+1} = v_k$ and $w(v_1 \dots v_k) = w(u_1 \dots u_k)$, then $w(u_1 \dots u_k)$ is periodic and defined by $w(v_1 \dots v_{i+1}) = (w_2, \dots, w_i)$.

Let X be the number of pairs of paths as in the statement of the claim and such that there are at most 2 edges between the paths (we are allowed to assume this since there are no complex subgraphs of size at most $2k$). Fix two weakly disjoint path $\mathbf{v} = v_1 \dots v_k$ and $\mathbf{u} = u_1 \dots u_k$ and assume without loss of generality that either \mathbf{u} extends \mathbf{v} , or they are disjoint. Let i be such that $u_1 = v_i$. If there is no such i , i.e. the paths are disjoint, set $i = k+1$. Then, expose edges from all v_j , $j \leq i$, and assume that they send at most 2 edges to u_2, \dots, u_{k-1} , other than the edge $\{u_1, u_2\}$. Then, probability that for every $j \in \{2, \dots, k-1\}$, v_j has degree 2 if and only if u_j has degree 2, is at most

$$\max \left\{ (1-p)^{n-2k}, (1-(1-p)^n) \right\}^{k-4} \leq \left(1 - e^{-(1+o(1))} \right)^{k-4} = o \left(\left(\frac{2}{3} \right)^k \right).$$

We then get

$$\mathbb{E}X \leq \sum_{i=2}^{k+1} n^{k+i-1} p^{k+i-3} \left(\frac{2}{3}\right)^k \leq kn^2(1+o(1))^{2k} \left(\frac{2}{3}\right)^k = o(1),$$

for an appropriate choice of A . Due to Markov's inequality, $\mathbb{P}(X \geq 1) \leq \mathbb{E}X = o(1)$, completing the proof. \triangleleft

Let D be the set of all vertices v in $C_n = \text{core}(G_n)$ that belong to a complex component of G_n and such that $B_v := \mathcal{B}_{3A \ln n}^{G_n}(v)$ is a tree. Let C be the coloring produced by CR on G_n .

\triangleright **Claim 25.** Whp, $C(u) \neq C(v)$ for any $u, v \in D$.

Proof. Assume that the statement of Claim 24 holds deterministically in G_n . Fix two different $u, v \in D$. Let us show towards contradiction that trees B_v and B_u are not isomorphic.

Take an arbitrary path $v_1 \dots v_k$ of length $k := \lfloor 1.9A \ln n \rfloor$, where $v_1 = v$. Since, by assumption, $B_v \cong B_u$, there exists a path $\mathbf{u} = u_1 \dots u_k$ such that $u_1 = u$ and, for every $i \in \{2, \dots, k-1\}$, v_i has degree 2 if and only if u_i has degree 2. In the same way, since $v \in \text{core}(G_n)$ and B_v is a tree, we may consider a path $v'_1 \dots v'_k$ that shares only the vertex $v'_1 = v = v_1$ with $v_1 \dots v_k$. Since $B_u \cong B_v$, there should be a path $\mathbf{u}' = u'_1 \dots u'_k$ that shares with $u_1 \dots u_k$ the only vertex $u'_1 = u = u_1$ and such that, for every $i \in \{2, \dots, k-1\}$, v'_i has degree 2 if and only if u'_i has degree 2. Since B_v is acyclic and since pairs of paths $v_1 \dots v_k$, $u_1 \dots u_k$ and $v'_1 \dots v'_k$, $u'_1 \dots u'_k$ cannot be disjoint due to Claim 24, u must lie on the path $P := v_k \dots v_1 \dots v'_k$. Moreover, since B_u is acyclic, once \mathbf{u} or \mathbf{u}' leave P , they never meet with P again. Thus, the path $u_k \dots u_1 \dots u'_k$ is divided by P in at most 3 parts: the first part does not have common vertices with P , the second part is a subpath of P , and the third part does not have common vertices with P again. Let Q be the longest part of the three. Then Q has length $\ell \geq \frac{1}{3}(2k-1) > A \ln n$. Moreover, since $\deg_{G_n} u = \deg_{G_n} v$ by assumption and $u \neq v$, there should be a subpath $P' \subset P$ such that P' and Q are weakly disjoint, and the degrees of internal vertices in P' and Q are aligned in the sense that the i -th inner vertex of P' have degree 2 if and only if the i -th vertex of Q has degree 2. This is impossible due to Claim 24. Thus, $B_v \not\cong B_u$ implying $C(u) \neq C(v)$ due to Claim 17. \triangleleft

4.3 Completing the proof of Main Lemma (Lemma 20)

Due to Claims 22, 25, and 19, it remains to prove the following:

1. If $1 + \omega(n^{-1/3}) = pn \leq \gamma$, then
 - $C^{\text{core}}(u) \neq C^{\text{core}}(v)$ for every $v \in V(C_n) \setminus D$ and $u \in D$;
 - $C^{\text{core}}(u) \neq C^{\text{core}}(v)$ for any non-interchangeable pair $u, v \in V(C_n) \setminus D$;
2. If $1 - n^{-1/3} \ln n \leq pn = 1 + o(1)$, then
 - $C(u) \neq C(v)$ for every $v \in V(C_n) \setminus D$ and $u \in D$;
 - $C(u) \neq C(v)$ for any non-interchangeable pair $u, v \in V(C_n) \setminus D$.

We will use the following technical fact, which follows from [17, Thm. 2] and [18, Thm. 1].

\triangleright **Claim 26.** Let $\delta > 0$ be a constant, $n^{-1/3} \ll \delta_n := pn - 1 \leq \delta$, and $G_n \sim G(n, p)$. Then whp in $K(G_n)$ there are no complex subgraphs of size at most $(\ln(n\delta_n^3))^{3/4}$.

For the sake of brevity, below we prove both statements in two different regimes simultaneously. Thus, with some abuse of notation, in the supercritical phase (i.e., $1 + \omega(n^{-1/3}) = pn \leq \gamma$), we let $G_n := C_n$ and $C := C^{\text{core}}$ as we only consider CR on C_n . We also assume that when $1 + \omega(n^{-1/3}) = pn \leq \gamma$, the core is equipped with the fractional distance d^f ,

constituting the metric space \mathcal{M}_n . If $1 - n^{-1/3} \ln n \leq pn \leq 1 + o(1)$, then G_n is equipped with the usual shortest-path distance, that we denote by d^f as well. We also use the following notation: $d = \lfloor (\ln(\delta_n^3 n))^{2/3} \rfloor$ when we prove the assertion for $1 + \omega(n^{-1/3}) = pn \leq \gamma$ and $d = \lfloor 3A \ln n \rfloor$ when we prove it for $1 - n^{-1/3} \ln n \leq pn = 1 + o(1)$. In what follows, we assume that the assertions of Claims 22, 23, 25, and 26 hold deterministically in G_n .

1. Assume that some $v \notin D$ and $u \in D$ have $C(u) = C(v)$. We know that v is d -close to a cycle F of length at most $2d$. If $v \in V(F)$, then let v' be the closest to v vertex on F that has degree more than 2 in the core. Otherwise, let $v' = v$. Let P be the shortest path from v' to F . Let us extend this path by a path P' of length $10d$ beyond v' . Due to Claim 23 and Claim 26, it has a subpath $w \dots w'$ of length $5d$ consisting of vertices from D only such that $d^f(w, v') \leq d$. We know that all elements of the vector $\mathbf{c} := (C(w), \dots, C(w'))$ are different. Then, due to our assumption, u must have a vertex z at distance at most $d^f(w, v)$ such that z is the first vertex of a path $z \dots z'$ with $C(w) = C(z), \dots, C(w') = C(z')$.

We now consider separately two cases: $w = z$ and $w \neq z$. In the first case, we have that the distance from u to the closest cycle (which is F , the same as for v) is at most

$$d^f(w, u) + d^f(w, v') + d^f(v', F) \leq \text{length of } F + 2(d^f(w, v') + d^f(v', F)) \leq 6d.$$

Let P be the shortest path between u and v . Due to Claim 23 and Claim 26, there exists a path $u\tilde{w} \dots \tilde{w}'$ of length $5d + 1$ that does not meet P and consists of vertices from D only. Due to Claim 22 and Claim 25 all elements of the vector $\tilde{\mathbf{c}} := (C(\tilde{w}), \dots, C(\tilde{w}'))$ are different and no element of $\tilde{\mathbf{c}}$ equals to any element of \mathbf{c} . Moreover, by construction, $d^f(v, \tilde{w}) > d^f(u, \tilde{w})$. Then, due to our assumption, v must have a neighbor $\tilde{z} \neq \tilde{w}$ such that \tilde{z} is the first vertex of a path $\tilde{z} \dots \tilde{z}'$ with $C(\tilde{w}) = C(\tilde{z}), \dots, C(\tilde{w}') = C(\tilde{z}')$. Note that $\tilde{w} \neq \tilde{z}, \dots, \tilde{w}' \neq \tilde{z}'$ due to Claim 23 and Claim 26. Since all vertices in D are distinguished by $C(\cdot)$, we conclude that all vertices $\tilde{z}, \dots, \tilde{z}'$ must be outside D . Due to Claim 23, Claim 26, and the definition of D , they constitute a (self-avoiding) path and are d -close to a cycle of length at most $2d$. Since the path has length $5d$, we get a contradiction with Claim 23 or Claim 26.

We then assume $w \neq z$. It may only happen when $z \notin D$. Moreover, all z, \dots, z' are not in D . Indeed, otherwise, different paths $w \dots w'$ and $z \dots z'$ have common vertices. Then the path from z to F that goes through w has length greater than d . However, due to Claim 23 and Claim 26, there are no two different paths from z to F , both of length at most $12d$ and, also, there is no other cycle F' of length at most $2d$ such that a path from z to F' has at most d vertices. This contradicts the fact that $z \notin D$. Thus, we again get a (self-avoiding) path consisting of vertices z, \dots, z' that are d -close to a cycle of length at most $2d$. This contradicts Claim 23 or Claim 26 again, since the path has length $5d$. We conclude that every vertex $u \in D$ has $C(u)$ that does not equal to the color of any other vertex in the core.

2. It remains to prove that, for any two distinct $u, v \notin D$ that are not interchangeable, $C(u) \neq C(v)$. Fix two such vertices u and v . We may assume that $T_u \cong T_v$ since otherwise $C(u) \neq C(v)$ due to Claim 15. Let F_u and F_v be two cycles of length at most $2d$ that are closest to u and v respectively (both are at distance at most d from the respective vertices). If $F_u \neq F_v$, then set $F := F_u$. In this case, we let $u' = u$ when $u \notin V(F)$ and let u' be the closest vertex of degree 3 in F to u otherwise. If $F_u = F_v =: F$, then, without loss of generality we assume that either u is not in F or both u, v are in F . Let $u' = u$ when $u \notin V(F)$ and let u' be a vertex of F that has degree at least 3 and such that $d^f(u, u') \neq d^f(v, u')$ otherwise. Note that such a vertex exists due to the definition

of an interchangeable pair. Consider a path P of length $10d$ that starts at u' and does not meet F . Due to Claim 23 and Claim 26, this path has a vertex w in D such that $d(w, u') \leq d$. If $F_u \neq F_v$, then

$$d^f(v, w) \geq d^f(F_u, F_v) - d^f(v, F_v) - d^f(w, F_u) > d^f(u, w)$$

due to Claim 23 and Claim 26. Finally, let $F_u = F_v$. Assume, in addition, $u \notin V(F)$. Then the only possibility for $C(u)$ to be equal to $C(v)$ is to have a path P' between v and w of length $d^f(u, w)$. Let us extend P' by a path of length $10d$ beyond v . Due to Claim 23 and Claim 26 this path has a vertex w' from D such that $d^f(w', v) \leq d$. But then $d^f(u, w') > d^f(v, w')$, implying $C(u) \neq C(v)$. If $u, v \in V(F)$, then

$$d^f(v, w) = d^f(v, u') + d^f(u', w) \neq d^f(u, u') + d^f(u', w) = d^f(u, w).$$

In either case, we get $d^f(v, w) \neq d^f(u, w)$ or $C(u) \neq C(v)$. Recalling that w has a unique color, we readily conclude that $C(u) \neq C(v)$, completing the proof.

References



- 1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullmann. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., 1974.
- 2 Michael Anastos, Matthew Kwan, and Benjamin Moore. Smoothed analysis for graph isomorphism, 2024. [arXiv:2410.06095](https://arxiv.org/abs/2410.06095).
- 3 D. Angluin. Local and global properties in networks of processors. In *The 12th Annual ACM Symposium on Theory of Computing*, pages 82–93, 1980.
- 4 Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. Graph isomorphism, color refinement, and compactness. *Comput. Complex.*, 26(3):627–685, 2017. doi:10.1007/s00037-016-0147-6.
- 5 L. Babai, P. Erdős, and S. M. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980. doi:10.1137/0209047.
- 6 László Babai. Moderately exponential bound for Graph Isomorphism. In *Proc. of the 3rd Int. Conf. on Fundamentals of Computation Theory (FCT'81)*, volume 117 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 1981. doi:10.1007/3-540-10854-8_4.
- 7 László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC'16)*, pages 684–697, 2016. doi:10.1145/2897518.2897542.
- 8 László Babai. Canonical form for graphs in quasipolynomial time: preliminary report. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*, pages 1237–1246, 2019. doi:10.1145/3313276.3316356.
- 9 Cristoph Berkholz, Paul Bonsma, and Martin Grohe. Tight lower and upper bounds for the complexity of canonical colour refinement. *Theory of Computing Systems*, 60:581–614, 2017. doi:10.1007/S00224-016-9686-0.
- 10 N. Biggs. *Algebraic graph theory*. Cambridge University Press, 2nd edition, 1994.
- 11 Tom Bohman, Alan M. Frieze, Tomasz Łuczak, Oleg Pikhurko, Clifford D. Smyth, Joel Spencer, and Oleg Verbitsky. First-order definability of trees and sparse random graphs. *Comb. Probab. Comput.*, 16(3):375–400, 2007. doi:10.1017/S0963548306008376.
- 12 Bela Bollobás. Distinguishing vertices of random graphs. *Ann. Discrete Math.*, 13:33–50, 1982.
- 13 Bela Bollobás. The evolution of random graphs. *Transactions of the American Mathematical Society*, 286(1):257–274, 1984.
- 14 Bela Bollobás. *Random graphs*. Cambridge University Press, 2001.
- 15 D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs. Theory and applications*. Leipzig: J. A. Barth Verlag, 3rd edition, 1995.

- 16 Tomek Czajka and Gopal Pandurangan. Improved random graph isomorphism. *Journal of Discrete Algorithms*, 6:85–92, 2008. doi:10.1016/J.JDA.2007.01.002.
- 17 Jian Ding, Jeong Han Kim, Eyal Lubetzky, and Yuval Peres. Anatomy of young giant component in the random graph. *Random Structures & Algorithms*, 39(2):139–178, 2011. doi:10.1002/RSA.20342.
- 18 Jian Ding, Eyal Lubetzky, and Yuval Peres. Anatomy of the giant component: The strictly supercritical regime. *European Journal of Combinatorics*, 35:155–168, 2014. doi:10.1016/J.EJC.2013.06.004.
- 19 Paul Erdős and Alfred Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.
- 20 Paul Erdős and Alfred Rényi. Asymmetric graphs. *Acta Math Acad Sci Hung*, 14:295–315, 1963.
- 21 Julia Gaudio, Miklós Z. Rácz, and Anirudh Sridhar. Average-case and smoothed analysis of graph isomorphism, 2023. arXiv:2211.16454.
- 22 László Hegedüs and Benedek Nagy. On periodic properties of circular words. *Discrete Mathematics*, 339(3):1189–1197, 2016. doi:10.1016/j.disc.2015.10.043.
- 23 Neil Immerman and Eric Lander. *Describing Graphs: A First-Order Approach to Graph Canonization*, pages 59–81. Springer New York, 1990.
- 24 Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random graphs*. John Wiley & Sons, 2000.
- 25 Sandra Kiefer, Pascal Schweitzer, and Erkal Selman. Graphs identified by logics with counting. *ACM Trans. Comput. Log.*, 23(1):1:1–1:31, 2022. doi:10.1145/3417515.
- 26 Andreas Krebs and Oleg Verbitsky. Universal covers, color refinement, and two-variable counting logic: Lower bounds for the depth. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'15)*, pages 689–700. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.69.
- 27 N. Linial and J. Mosheiff. On the rigidity of sparse random graphs. *Journal of Graph Theory*, 85(2):466–480, 2017. doi:10.1002/JGT.22073.
- 28 T. Łuczak. The automorphism group of random graphs with a given number of edges. *Math Proc Camb Phil Soc*, 104:441–449, 1988.
- 29 Tomasz Łuczak. The phase transition in a random graph. In D. Miklós, V.T. Sós, and T. Szónyi, editors, *Combinatorics, Paul Erdős is Eighty*, volume 2, pages 399–422. Bolyai Soc. Math. Stud. 2, J. Bolyai Math. Soc., Budapest, 1996.
- 30 Tomasz Łuczak, Boris Pittel, and John C. Wierman. The structure of a random graph at the point of the phase transition. *Transactions of the American Mathematical Society*, 341(2):721–748, 1994.
- 31 W. S. Massey. *Algebraic topology: An introduction*, volume 56 of *Graduate Texts in Mathematics*. Springer, 5th edition, 1981.
- 32 Marc Noy, Vlady Ravelomanana, and Juanjo Rué. On the probability of planarity of a random graph near the critical point. *Proc. Am. Math. Soc.*, 143(3):925–936, 2015. doi:10.1090/S0002-9939-2014-12141-1.
- 33 Oleg Verbitsky and Maksim Zhukovskii. Canonical labeling of sparse random graphs, 2024. arXiv:2409.18109.
- 34 B. Yu. Weisfeiler and A.A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Ser. 2*, 9:12–16, 1968. In Russian. English translation is available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.
- 35 E. M. Wright. Asymmetric and symmetric graphs. *Glasgow Math J*, 15:69–73, 1974.

Dynamic Unit-Disk Range Reporting

Haitao Wang  

Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

Yiming Zhao  

Department of Computer Sciences, Metropolitan State University of Denver, CO, USA

Abstract

For a set P of n points in the plane and a value $r > 0$, the *unit-disk range reporting* problem is to construct a data structure so that given any query disk of radius r , all points of P in the disk can be reported efficiently. We consider the dynamic version of the problem where point insertions and deletions of P are allowed. The previous best method provides a data structure of $O(n \log n)$ space that supports $O(\log^{3+\epsilon} n)$ amortized insertion time, $O(\log^{5+\epsilon} n)$ amortized deletion time, and $O(\log^2 n / \log \log n + k)$ query time, where ϵ is an arbitrarily small positive constant and k is the output size. In this paper, we improve the query time to $O(\log n + k)$ while keeping other complexities the same as before. A key ingredient of our approach is a shallow cutting algorithm for circular arcs, which may be interesting in its own right. A related problem that can also be solved by our techniques is the dynamic unit-disk range emptiness queries: Given a query unit disk, we wish to determine whether the disk contains a point of P . The best previous work can maintain P in a data structure of $O(n)$ space that supports $O(\log^2 n)$ amortized insertion time, $O(\log^4 n)$ amortized deletion time, and $O(\log^2 n)$ query time. Our new data structure also uses $O(n)$ space but can support each update in $O(\log^{1+\epsilon} n)$ amortized time and support each query in $O(\log n)$ time.

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

Keywords and phrases Unit disks, range reporting, range emptiness, alpha-hulls, dynamic data structures, shallow cuttings

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.76

Related Version *Full Version*: <https://arxiv.org/pdf/2501.00120>

Funding This research was supported in part by NSF under Grant CCF-2300356.

1 Introduction

Range searching is a fundamental problem and has been studied extensively in computational geometry [2, 3, 26]. In this paper, we consider a dynamic range reporting problem regarding disks of fixed radius, called *unit disks*.

Given a set P of n points in the plane, the *unit-disk range reporting* problem (or UDRR for short) is to construct a data structure to report all points of P in any query unit disk. The problem is also known as the *fixed-radius neighbor problem* in the literature [4, 14, 16, 17]. Chazelle and Edelsbrunner [17] constructed a data structure of $O(n)$ space that can answer each query in $O(\log n + k)$ time, where k is the output size; their data structure can be constructed in $O(n^2)$ time. By a standard lifting transformation [5], the problem can be reduced to the half-space range reporting queries in 3D; this reduction also works if the radius of the query disk is arbitrary. Using Afshani and Chan's 3D half-space range reporting data structure [1], one can construct a data structure of $O(n)$ space with $O(\log n + k)$ query time, while the preprocessing takes $O(n \log n)$ expected time since it invokes Ramos' algorithm [27] to construct shallow cuttings for a set of planes in 3D. Chan and Tsakalidis [13] later presented an $O(n \log n)$ -time deterministic shallow cutting algorithm. Combining the framework in [1] with the shallow cutting algorithm [13], one can build a data structure of $O(n)$ space in $O(n \log n)$ deterministic time that can answer each UDRR query in $O(\log n + k)$ time.



© Haitao Wang and Yiming Zhao;
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).

Editors: Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyễn Kim Thăng;

Article No. 76; pp. 76:1–76:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



We consider the dynamic UDRR problem in which point insertions and deletions of P are allowed. By the lifting transformation, the problem can be reduced to dynamic halfspace range reporting in 3D [7, 10, 11], which also works for query disks of arbitrary radii. Using the currently best result of dynamic halfspace range reporting [6], one can obtain a data structure of $O(n \log n)$ space that supports $O(\log^{3+\epsilon} n)$ amortized insertion time, $O(\log^{5+\epsilon} n)$ amortized deletion time, and $O(\log^2 n / \log \log n + k)$ query time, where ϵ is an arbitrarily small positive constant and k is the output size.

Our result. In this paper, we achieve the optimal $O(\log n + k)$ query time, while the space of the data structure and the update time complexities are the same as above.

A byproduct of our techniques is a static data structure of $O(n)$ space that can be built in $O(n \log n)$ time and support $O(\log n + k)$ query time. This matches the above result of [1, 13]. But our method is much simpler. Indeed, the algorithm of [1, 13] involves relatively advanced geometric techniques like computing shallow cuttings for the planes in 3D, planar graph separators, etc. Our algorithm, on the contrary, relies only on elementary techniques (the most complicated one might be a fractional cascading data structure [18, 19]). One may consider our algorithm a generalization of the classical 2D half-plane range reporting algorithm of Chazelle, Guibas, and Lee [20].

Our techniques may also be useful for solving other problems related to unit disks. In particular, we can obtain an efficient algorithm for the dynamic *unit-disk range emptiness queries*. For a dynamic set P of points in the plane, we wish to determine whether a query unit disk contains any point of P (and if so, return such a point as an “evidence”). The previous best solution is to use a dynamic nearest neighbor search data structure [11]. Specifically, we can have a data structure of $O(n)$ space that supports $O(\log^2 n)$ amortized insertion time, $O(\log^4 n)$ amortized deletion time, and $O(\log^2 n)$ query time. Using our techniques, we obtain an improved data structure of $O(n)$ space that supports both insertions and deletions in $O(\log^{1+\epsilon} n)$ amortized time and supports queries in $O(\log n)$ time.

Our approach. We first discuss our static data structure. We use a set of $O(n)$ grid cells (each of which is an axis-parallel rectangle) to capture the proximity information for the points of P , such that the distance between any two points in the same cell is at most 1. For a query unit disk D_q whose center is q , points of P in the cell C that contains q can be reported immediately. The critical part is handling other cells that contain points of $P \cap D_q$. The number of such cells is constant and each of them is separated from C (and thus from q) by an axis-parallel line. The problem thus boils down to the following subproblem: Given a set Q of points in a grid cell C' above a horizontal line ℓ , report the points of Q in any query unit disk whose center is below ℓ . A point $p \in Q$ is in D_q if and only if q lies in the unit disk D_p centered at p , or equivalently, q is above the arc of the boundary of D_p below ℓ . Let Γ denote the set of all such arcs for all points $p \in Q$. To find the points of Q in D_q , it suffices to report the arcs of Γ below q . To tackle this problem, we follow the same framework as that for the 2D half-plane range reporting algorithm [20], by constructing lower envelope layers of Γ and building a fractional cascading data structure on them [18, 19].

To make the data structure dynamic, we first derive a data structure to maintain the grid cells dynamically so that each update (point insertions/deletions) can be handled in $O(\log n)$ amortized time. This dynamic data structure could be of independent interest. Next, we develop a data structure to dynamically maintain arcs of Γ to support the arc-reporting queries (i.e., given a query point, report all arcs of Γ below it). To this end, we cannot use the fractional cascading data structure anymore because it is not amenable to dynamic

changes. Instead, we adapt the techniques for dynamically maintaining a set of lines to answer line-reporting queries (i.e., given a query point, report all lines below the query point; its dual problem is the halfplane range reporting queries) [6, 9–11, 23]. To make these techniques work for the arcs of Γ in our problem, we need an efficient shallow cutting algorithm for Γ . For this, we adapt the algorithm in [13] for lines and derive an $O(|\Gamma| \log |\Gamma|)$ time shallow cutting algorithm for Γ . As shallow cuttings have many applications, our algorithm may be interesting in its own right.

Outline. The rest of the paper is organized as follows. In Section 2, we introduce notation and a *conforming coverage set* of grid cells to capture the proximity information for points of P . Section 3 discusses our dynamic UDRR data structure. A main subproblem of it is solved in Section 4. A key ingredient of our method is an efficient algorithm for computing shallow cuttings for arcs; this algorithm is presented in Section 5. Section 6 demonstrates that our techniques may be used to solve other related problems, such as dynamic unit-disk range emptiness queries. Due to the space limit, some details and proofs are omitted but can be found in the full paper.

2 Preliminaries

We define some notation that will be used throughout the paper. A *unit disk* refers to a disk of radius 1. A *unit circle* is defined similarly. Unless otherwise stated, a circular arc or an arc refers to a circular arc of radius 1. For a circular arc γ , we call the circle that contains γ the *underlying circle* of γ and call the disk whose boundary contains γ the *underlying disk*.

For any point q , let D_q denote the unit disk centered at q . For any region R and any set P of points in the plane, let $P(R)$ denote the subset of points of P inside R , i.e., $P(R) = P \cap R$. For any region R in the plane, we use ∂R to denote its boundary, e.g., if R is a disk, then ∂R is its bounding circle.

Unless otherwise stated, ϵ refers to an arbitrarily small positive constant. Depending on the context, we often use k to denote the output size of a reporting query. For any point p in the plane, we use $x(p)$ and $y(p)$ to denote the x and y -coordinates of p , respectively.

2.1 Conforming coverage of P

Let P be a set of n points in the plane. We wish to have a data structure for P to answer the following *unit-disk range reporting queries*: Given a query unit disk D , report $P(D)$, i.e., the points of P in D .

As discussed in Section 1, our method (for both static and dynamic problems) relies on a set of grid cells to capture the proximity information for the points of P . The technique of using grids has been widely used in various algorithms for solving problems in unit-disk graphs [12, 28–32]. However, the difference here is that we need to handle updates to P and therefore our grid cells will be dynamically changed as well. To resolve the issue, our definition of grid cells is slightly different from the previous work. Specifically, we define a *conforming coverage set* of cells for P in the following.

► **Definition 1** (Conforming Coverage). *A set \mathcal{C} of cells in the plane is called a conforming coverage for P if the following conditions hold.*

1. *Each cell of \mathcal{C} is an axis-parallel rectangle of side lengths at most $1/2$. This implies that the distance of every two points in each cell is at most 1.*
2. *The union of all cells of \mathcal{C} covers all the points of P .*

3. Every two cells are separated by an axis-parallel line.
4. Each cell $C \in \mathcal{C}$ is associated with a subset $N(C) \subseteq \mathcal{C}$ of $O(1)$ cells (called neighboring cells of C) such that for any point $q \in C$, $P(D_q) \subseteq \bigcup_{C' \in N(C)} P(C')$.
5. For any point q , if q is not in any cell of \mathcal{C} , then $P \cap D_q = \emptyset$.

To solve the static problem, after computing a conforming coverage set of cells for P , we never need to change it in the future. As such, the following lemma from [28] suffices.

► **Lemma 2** ([28]).

1. A conforming coverage set \mathcal{C} of size $O(n)$, along with $P(C)$ and $N(C)$ for all cells $C \in \mathcal{C}$, can be computed in $O(n \log n)$ time and $O(n)$ space.
2. With $O(n \log n)$ time and $O(n)$ space preprocessing, given any point q , we can do the following in $O(\log n)$ time: Determine whether q is in a cell C of \mathcal{C} , and if so, return C and $N(C)$.

However, for the dynamic problem, due to the updates of P , the conforming coverage set also needs to be maintained dynamically. For this, we have the following lemma.

► **Lemma 3.** A conforming coverage set \mathcal{C} of $O(n)$ cells for P can be maintained in $O(n)$ space (where n is the size of the current set P) such that each point insertion of P can be handled in $O(\log n)$ worst-case time, each point deletion can be handled in $O(\log n)$ amortized time, and the following query can be answered in $O(\log n)$ time: Given a query point q , determine whether q is in a cell C of \mathcal{C} , and if so, return C and $N(C)$.

The proof of Lemma 3, which is lengthy and technical, is in the full paper. Roughly speaking, if a point p is inserted to P , then at most $O(1)$ cells will be added to \mathcal{C} and p will eventually be inserted into $P(C)$ for the cell $C \in \mathcal{C}$ containing p . If a point p is deleted from P , the deletion boils down to the deletion of p from $P(C)$ for the cell $C \in \mathcal{C}$ containing p . We do not remove cells from \mathcal{C} . Instead, we reconstruct the entire data structure after $n/2$ deletions; this guarantees that the size of \mathcal{C} is always $O(n)$. See the full paper for the details.

3 Dynamic range reporting

Let P be a set of points in the plane. We wish to maintain a data structure for P to answer unit-disk range reporting queries subject to point insertions and deletions of P . Let n denote the size of the current set P .

Using Lemma 3, we maintain a conforming coverage set \mathcal{C} of $O(n)$ cells for P . To insert a point p to P , our insertion algorithm for Lemma 3 boils down to inserting p to $P(C)$ for a cell $C \in \mathcal{C}$ that contains p . To delete a point p from P , our deletion algorithm for Lemma 3 boils down to deleting p from $P(C)$ for a cell $C \in \mathcal{C}$ that contains p .

Consider a query unit disk D_q whose center is q . If q is not in a cell of \mathcal{C} , then by Definition 1(5), $P(D_q) = \emptyset$ and thus we simply return null. Otherwise, to report $P(D_q)$, it suffices to report $P(C') \cap D_q$ for all cells $C' \in N(C)$. In the case of $C' = C$, since the distance between two points in C is at most 1 by Definition 1(1), we can simply report all points of $P(C)$. If $C' \neq C$, C and C' are separated by an axis-parallel line by Definition 1(3). Without loss of generality, we assume that C and C' are separated by a horizontal line ℓ with C' above ℓ and C below ℓ . As $q \in C$, q is below ℓ , i.e., q is separated from C' by ℓ . Our goal is to report points of $P(C') \cap D_q$. Due to the point updates of $P(C')$, our problem is reduced to the following subproblem, called *dynamic line-separable UDRR problem*.

► **Problem 1** (Dynamic line-separable UDRR). *For a set Q of m points above a horizontal line ℓ , maintain Q in a data structure to support the following operations. (1) Insertion: insert a point to Q ; (2) deletion: delete a point from Q ; (3) unit-disk range reporting query: given a point q below ℓ , report the points of Q in the unit disk D_q .*

We have the following Lemma 4 for the dynamic line-separable UDRR problem.

► **Lemma 4.** *For the dynamic line-separable UDRR problem, we can have a data structure of $O(m \log m)$ space to maintain Q to support insertions in $O(\log^{3+\epsilon} m)$ amortized time, deletions in $O(\log^{5+\epsilon} m)$ amortized time, and unit-disk range reporting queries in $O(k + \log m)$ time, where k is the output size, ϵ is an arbitrarily small positive constant, and m is the size of the current set Q .*

We will prove Lemma 4 in Section 4. With Lemmas 3 and 4, we can obtain the following main result for our original dynamic UDRR problem.

► **Theorem 5.** *We can maintain a set P of points in the plane in a data structure of $O(n \log n)$ space to support insertions in $O(\log^{3+\epsilon} n)$ amortized time, deletions in $O(\log^{5+\epsilon} n)$ amortized time, and unit-disk range reporting queries in $O(k + \log n)$ time, where k is the output size, ϵ is an arbitrarily small positive constant, and n is the size of the current set P .*

Proof. We build the data structure \mathcal{D} in Lemma 3 to maintain a conforming coverage set \mathcal{C} of $O(n)$ cells for P . For each cell $C \in \mathcal{C}$ that contains at least one point of P , we maintain a data structure $\mathcal{D}_e(C)$ for $P(C)$ with respect to the supporting line of each edge e of C . Since the space of each $\mathcal{D}_e(C)$ is $O(|P(C)| \log |P(C)|)$, each cell of \mathcal{C} has four edges, and $\sum_{C \in \mathcal{C}} |P(C)| = n$, the total space of the overall data structure is $O(n \log n)$.

Insertions. To insert a point p to P , we first update \mathcal{D} by Lemma 3, which takes $O(\log n)$ worst-case time. The insertion algorithm of Lemma 3 eventually inserts p to $P(C)$ for a cell $C \in \mathcal{C}$ that contains p . We insert p to $\mathcal{D}_e(C)$ for each edge e of C , which takes $O(\log^{3+\epsilon} n)$ amortized time by Lemma 4. As such, each insertion takes $O(\log^{3+\epsilon} n)$ amortized time.

Deletions. To delete a point p from P , we first update \mathcal{D} by Lemma 3, which takes $O(\log n)$ amortized time. The deletion algorithm of Lemma 3 eventually deletes p from $P(C)$ for a cell $C \in \mathcal{C}$ that contains p . We delete p from $\mathcal{D}_e(C)$ for each edge e of C , which takes $O(\log^{5+\epsilon} n)$ amortized time by Lemma 4. As such, each deletion takes $O(\log^{5+\epsilon} n)$ amortized time.

Queries. Given a query unit disk D_q with center q , we first check whether q is in a cell of \mathcal{C} , and if so, find such a cell; this takes $O(\log n)$ time by Lemma 3. If no cell of \mathcal{C} contains q , then $P \cap D_q = \emptyset$ and we simply return null. Otherwise, let C be the cell of \mathcal{C} that contains q . We first report all points of $P(C)$. Next, for each $C' \in N(C)$, by Definition 1(3), C and C' are separated by an axis-parallel line ℓ . Since each edge of C and C' is axis-parallel, C' must have an edge e whose supporting line is parallel to ℓ and separates C and C' . Using $\mathcal{D}_e(C')$, we report all points of $P(C')$ inside D_q . As $|N(C)| = O(1)$, the total query time is $O(\log n + k)$ by Lemma 4. ◀

4 Proving Lemma 4: Dynamic line-separable UDRR

We now prove Lemma 4. For notational convenience, instead of m , we use n to denote the size of Q .

Consider a query unit disk D_q with center q below ℓ . The goal is to report $Q(D_q)$. Observe that a point $p \in Q$ is in D_q if and only if q is in the unit disk D_p . The portion of ∂D_p below ℓ is a circular arc, denoted by γ_p . Since p is above ℓ , γ_p is on the lower half

circle of ∂D_p and thus is x -monotone. As such, p is in D_q if and only if q is above the arc γ_p . Define Γ to be the set of arcs γ_p for all points $p \in Q$. Therefore, reporting the points of Q in D_q becomes reporting the arcs of Γ that are below q , which we call *arcs reporting queries*.

In what follows, an arc of Γ always refers to the portion below ℓ of a unit circle with center above ℓ . Our problem thus becomes dynamically maintaining a set Γ of arcs to report the arcs of Γ below a query point q . The arcs reporting queries can be reduced to the following *k -lowest-arcs queries*: Given a query vertical line ℓ^* and a number $k \geq 1$, report the k lowest arcs of Γ intersecting ℓ^* . We have the following observation, which follows the proof of Chan [7] for lines.

► **Observation 6** ([7]). *Suppose that we can answer each k -lowest-arcs query in $O(\log n + k)$ time. Then, the arcs of Γ below a query point q can be reported in $O(\log n + k)$ time, where k is the output size.*

In light of the above observation, we now focus on the k -lowest-arcs queries. We adapt a technique for a similar problem on lines (which is the dual problem of the dynamic halfplane range reporting problem): Dynamically maintain a set of lines (subject to insertions and deletions) to report the k -lowest lines at a query vertical line. For this problem, Chan [10] gave a data structure of $O(n \log n)$ space that supports $O(\log^{6+\epsilon} n)$ amortized update time and $O(k + \log n)$ query time. De Berg and Staals [6] improved the result of [10] for dynamically maintaining a set of planes in 3D. They gave a data structure of $O(n \log n)$ space that supports $O(\log^{3+\epsilon} n)$ amortized insertion time, $O(\log^{5+\epsilon} n)$ amortized deletion time, and $O(\log^2 n / \log \log n + k)$ query time. Their approach is based on the techniques for dynamically maintaining planes for answering lowest point queries [9, 11, 23] and these techniques in turn relies on computing shallow cuttings on the planes in 3D [13]. In the following, we will extend these techniques to the arcs of Γ and prove the following result.

► **Lemma 7.** *For the set Γ of arcs, we can have a data structure of $O(n \log n)$ space to support insertions in $O(\log^{3+\epsilon} n)$ amortized time, deletions in $O(\log^{5+\epsilon} n)$ amortized time, and k -lowest-arcs queries in $O(k + \log n)$ time, where n is the size of the current set Γ .*

Combining Lemma 7 and Observation 6 immediately leads to Lemma 4.

In what follows, we first develop a shallow cutting algorithm for arcs of Γ in Section 4.1 and then using the algorithm to prove Lemma 7 in Section 4.2.

4.1 Shallow cuttings

Without loss of generality, we assume that ℓ is the x -axis. Let \mathbb{R}^- (resp., \mathbb{R}^+) be the half-plane below (resp., above) ℓ . Note that each arc of Γ is x -monotone, every arc has both endpoints on ℓ , and every two arcs cross each other at most once.

We use \mathbb{R}^+ -constrained unit disk to refer to a unit disk with center in \mathbb{R}^+ and use \mathbb{R}^+ -constrained arc to refer to a portion of the arc $C \cap \mathbb{R}^-$ for a unit circle C with center in \mathbb{R}^+ . For any point $q \in \mathbb{R}^-$, let $\rho(q)$ to denote the vertical downward ray from q . We say that an arc γ of Γ is *below* q if it intersects $\rho(q)$. As the center of γ is in \mathbb{R}^+ and $\rho_q \in \mathbb{R}^-$, γ intersects ρ_q at most once.

For a parameter $r \leq n$ and a region R of the plane, a $(1/r)$ -cutting covering R for the arcs of Γ is a set of interior-disjoint cells such that the union of all cells covers R and each cell intersects at most n/r arcs of Γ . For each cell Δ , its *conflict list* Γ_Δ is the set of arcs of Γ that intersect Δ . The size of the cutting is the number of its cells.

For a point $p \in \mathbb{R}^-$, the *level* of p in Γ is the number of arcs of Γ below p . For any integer $k \in [1, n]$, the $(\leq k)$ -*level* of Γ , denoted by $L_{\leq k}(\Gamma)$, is defined as the region consisting of all points of \mathbb{R}^- with level at most k . Given parameters $r, k \in [1, n]$, a k -*shallow* $(1/r)$ -*cutting* is a $(1/r)$ -cutting for Γ that covers $L_{\leq k}(\Gamma)$.

We use *pseudo-trapezoid* to refer to a region that has two vertical line segments as left and right edges, an \mathbb{R}^+ -constrained arc or a line segment on ℓ as a top edge, and an \mathbb{R}^+ -constrained arc as a bottom edge. In particular, if a pseudo-trapezoid does not have a bottom edge, i.e., the bottom side is unbounded, then we call it a *bottom-open* pseudo-trapezoid.

We say that a shallow cutting is in the *bottom-open pseudo-trapezoid form* if every cell of it is a bottom-open pseudo-trapezoid. Our main result about the shallow cuttings for Γ is given in the following theorem.

► **Theorem 8.** *There exist constants B, C , and C' , such that for a parameter $k \in [1, n]$, we can compute a $(B^i k)$ -shallow $(CB^i k/n)$ -cutting of size at most $C' \frac{n}{B^i k}$ in the bottom-open pseudo-trapezoid form, along with conflict lists of all its cells, for all $i = 0, 1, \dots, \log_B \frac{n}{k}$, in $O(n \log \frac{n}{k})$ total time. In particular, we can compute a k -shallow (Ck/n) -cutting of size $O(n/k)$, along with its conflict lists, in $O(n \log \frac{n}{k})$ time.*

Since the proof of Theorem 8 is technical and lengthy (and is one of our main results in this paper), we devote the entire Section 5 to it.

4.2 Proving Lemma 7

We now prove Lemma 7. With the shallow cutting algorithm in Theorem 8, we generalize the techniques of [6, 10] for lines to the arcs of Γ . We first give two deletion-only data structures, which will be needed in our fully dynamic data structure for Lemma 7.

4.2.1 Deletion-only data structure

Our first deletion-only data structure is given in Lemma 9 (see the full paper for the proof).

► **Lemma 9.** *There is a data structure of $O(n)$ size to maintain a set Γ of n arcs to support $O(\log n)$ amortized time deletions and $O(\sqrt{n} \log^{O(1)} n + k)$ time k -lowest-arcs queries. If a set Γ of n arcs is given initially, the data structure can be constructed in $O(n \log n)$ time.*

We have the following lemma for another deletion-only data structure, obtained by following the same algorithmic scheme as [6, Lemma 6] and replacing their shallow cutting algorithm for lines with our shallow cutting algorithm for arcs of Γ in Theorem 8.

► **Lemma 10.** [6, Lemma 6] *For any fixed r , there is a data structure of $O(n \log r)$ size to maintain a set Γ of n arcs to support $O(r \log n)$ amortized time deletions and $O(\log r + n/r + k)$ time k -lowest-arcs queries. If a set Γ of n arcs is given initially, the data structure can be constructed in $O(n \log n)$ time.*

4.2.2 Fully-dynamic data structure for Lemma 7

With the two deletion-only data structures in Lemmas 9 and 10, we are now in a position to describe our fully dynamic data structure for Lemma 7.

Overview. To achieve our result in Lemma 7, roughly speaking, we can simply plug our shallow cutting algorithm for Γ in Theorem 8 and Lemma 9 into the algorithmic scheme of [6] or [10]. The algorithms of [6] and [10] are similar. For the method of [10], we can just replace their shallow cutting algorithm for lines [13] with our shallow cutting algorithm for Γ in Theorem 8 and replace their deletion-only data structure [24] with a combination of Lemmas 9 and 10. In addition, a general technique of querying multiple structures simultaneously from [6, Theorem 1] is also needed. For the method of [6], it was described for the plane problem in 3D with a query time $O(\log^2 n / \log \log n + k)$. We can follow the same algorithmic scheme but using our shallow cutting algorithm for Γ in Theorem 8 and Lemma 9 in the corresponding places. In addition, since our problem is a 2D problem, the technique of dynamic interval trees in [13] can be used to reduce the query time component from $O(\log^2 n / \log \log n)$ to $O(\log n)$. In the following, we adapt the method from Chan [10].

The data structure is an adaptation of the one for dynamic 3D convex hulls [9, 11, 23] (the idea was originally given in [9] and subsequent improvements were made in [11, 23]). We first give the following lemma. The lemma is similar to [10, Theorem 3.1], which is based on the result in [9], but Lemma 11 provides slightly better complexities than [10, Theorem 3.1] by using the recently improved result of [11] (see the full paper for more details).

► **Lemma 11** ([9–11, 23]). *Let Γ be a set of arcs, which initially is \emptyset and undergoes n updates (insertions and deletions). For any $b \geq 2$, we can maintain a collection of shallow cuttings T_i^j in the bottom-open pseudo-trapezoid form, $i = 1, 2, \dots, \lceil \log n \rceil$, $j = 1, 2, \dots, O(\log_b n)$, in $b^{O(1)} \log^5 n$ amortized time per update such that the following properties hold.*

1. *Each cutting T_i^j is of size $O(2^i)$ and never changes until it is replaced by a new one created from scratch. The total size of all cuttings created over time is $b^{O(1)} \log^3 n$.*
2. *Each cell $\Delta \in T_i^j$ is associated with a list L_Δ of $O(n/2^i)$ arcs of Γ . Each list L_Δ undergoes deletions only after its creation. The total size of all such lists created over time is $b^{O(1)} n \log^4 n$.*
3. *For any $k \geq 1$, let $i_k = \lceil \log(n/Ck) \rceil$ for a sufficiently large constant C . For any vertical line ℓ^* , if an arc $\gamma \in \Gamma$ is among the k lowest arcs at ℓ^* , then there exists some j such that γ is in the list L_{Δ^j} of the cell $\Delta^j \in T_{i_k}^j$ intersecting ℓ^* .*
4. *At any moment, for each i , the number of cells of the current cuttings T_i^j for all j is $O(2^i)$. This implies that the total size of the lists L_Δ of all cells Δ of all current cuttings T_i^j at any moment is $O(n \log n)$.*

With Lemma 11, we can answer a k -lowest-arcs query as follows. Consider a query vertical line ℓ^* . By Lemma 11(3), for each j , we compute the cell Δ^j of $T_{i_k}^j$ intersecting ℓ^* , which takes $O(\log n)$ time by binary search as the x -projections of $T_{i_k}^j$ partition the x -axis into intervals. Then, we use “brute-force” to find the k lowest arcs among all arcs in L_{Δ^j} in $O(k)$ time as $|L_{\Delta^j}| = O(k)$. Finally, among all arcs found above, we return the k lowest arcs, which takes $O(k \log_b n)$ time. As such, the total query time is $O((\log n + k) \log_b n)$.

An improved query algorithm. We now improve the query time. We store each list L_Δ by a deletion-only data structure that supports k -lowest-arcs queries. Suppose that such a deletion-only data structure is of space $S_0(|L_\Delta|)$, supports each k -lowest-arcs query in $O(Q_0(|L_\Delta|) + k)$ time and $D_0(|L_\Delta|)$ deletion time, and can be built in $O(P_0(|L_\Delta|))$ time. Then, by Lemma 11(2), each update causes at most $b^{O(1)} \log^4 n$ amortized number of deletions to the lists L_Δ , and thus the amortized update time is

$$U(n) = b^{O(1)} \log^5 n + \max_{\Delta \in T_i^j} D_0(|L_\Delta|) \cdot b^{O(1)} \log^4 n + P_0(b^{O(1)} n \log^4 n) / n. \quad (1)$$

Note that the last term is obtained due to the following. After every n updates, we reconstruct the entire data structure and thus the reconstruction time is on the order of $\sum_{\Delta \in T_i^j} P_0(|L_\Delta|)$, which is bounded by $P_0(b^{O(1)}n \log^4 n)$ since $\sum_{\Delta \in T_i^j} |L_\Delta| = b^{O(1)}n \log^4 n$ by Lemma 11(2) (assuming that $P_0(n) = \Omega(n)$).

By Lemma 11(4), the total space is

$$S(n) = O\left(\sum_{i=1}^{\log n} 2^i \cdot S_0(n/2^i)\right). \quad (2)$$

For each query, there are two tasks: (1) Compute the cell Δ^j for every j ; (2) find the k lowest arcs of all lists L_{Δ^j} for all j . In the following, we solve the first task in $O(\log n + \log_b n)$ time and solve the second task in $O(\log n + k)$ time.

For the first task, following the method in [10], for each i , we use a dynamic interval tree [5] to store the intervals of the x -projections of the cuttings T_i^j for all j in an interval tree T_i . Using T_i , all t intervals intersecting ℓ^* can be computed in $O(\log n + t)$ time. In our problem, $t = O(\log_b n)$. Insertions and deletions of intervals on T_i can be supported in $O(\log n)$ amortized time. We can thus maintain all interval trees T_i in additional amortized $\log n \cdot b^{O(1)} \log^3 n$ time per update as the total size of all cuttings over time is $b^{O(1)} \log^3 n$ by Lemma 11(1). This additional update time is subsumed by the first term in (1). In this way, the first task can be solved in $O(\log n + \log_b n)$ time.

For the second task, instead of brute-force, we use the deletion-only data structures for the lists L_{Δ^j} and resort to a technique of querying multiple k -lowest-arcs data structures simultaneously in [6, Theorem 1], which is based on an adaption of the heap selection algorithm of Frederickson [22]. Applying [6, Theorem 1], the second task can be accomplished in $O(k + \log_b n \cdot \max_j Q_0(|L_{\Delta^j}|))$ time, which is $O(k + \log_b n \cdot Q_0(O(k)))$ since the size of each $|L_{\Delta^j}|$ is $O(k)$.

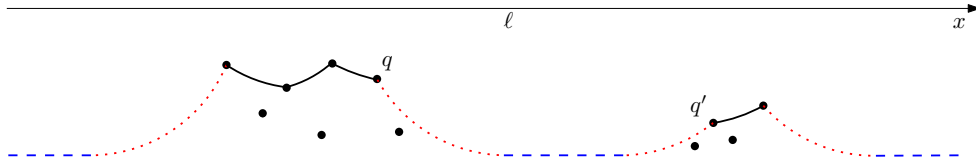
Combining the complexities of the first and second tasks, the overall query time is

$$Q(n) = O(\log n + \log_b n + k) + Q_0(O(k)) \cdot \log_b n. \quad (3)$$

Let $m = |L_\Delta|$. Depending on m , we use different deletion-only data structures for L_Δ .

1. If $m \geq \log^3 n$, then we use Lemma 9 to handle L_Δ with $P_0(m) = O(m \log m)$, $S_0(m) = O(m)$, $D_0(m) = O(\log m)$, $Q_0(m) = m^{1/2} \log^{O(1)} m$. Plugging them into (1), (2), and (3) and setting $b = \log^\epsilon n$, we obtain $U(n) = O(\log^{5+\epsilon} n)$, $S(n) = O(n \log n)$, and $Q(n) = O(\log n + k + k^{1/2} \log^{O(1)} k \cdot \log n / \log \log n)$. Since $m = O(k)$, we have $k = \Omega(m)$. As $m \geq \log^3 n$, we have $k = \Omega(\log^3 n)$. Therefore, $Q(n) = O(\log n + k)$.
2. If $m < \log^3 n$, then we use Lemma 10 to handle L_Δ by setting $r = \log n / \log \log n$. This results in $P_0(m) = O(m \log m)$, $S_0(m) = O(m \log \log n)$, $D_0(m) = O(\log n \log m / \log \log n)$, $Q_0(m) = O(\log \log n + m \log \log n / \log n)$. Since $m < \log^3 m$, we have $D_0(m) = O(\log n)$. Plugging them into (1) and (3) and setting $b = \log^\epsilon n$, we obtain $U(n) = O(\log^{5+\epsilon} n)$ and $Q(n) = O(\log n + k + (\log \log n + k \log \log n / \log n) \cdot \log n / \log \log n)$, which is $O(\log n + k)$. For the space, since $m < \log^3 n$, if we plug $S_0(m) = O(m \log \log n)$ into (2), we only need to consider those i 's such that $n/2^i < \log^3 n$. There are only $O(\log \log n)$ such i 's. Therefore, we obtain $S(n) = O(n \log^2 \log n)$ for all such m 's in this case.

Combining the above two cases leads to $U(n) = O(\log^{5+\epsilon} n)$, $S(n) = O(n \log n)$, and $Q(n) = O(\log n + k)$. We can actually obtain a better bound for the insertion time. If $P'(n)$ is the preprocessing time for constructing the data structure for a set of n arcs, then the amortized insertion time $I(n)$ is bounded by $I(n) = O(b \log_b n \cdot P'(n)/n)$ [6, 11]. According



to our above discussion and Lemma 11(4), constructing the deletion-only data structures for all lists L_Δ is $O(n \log^2 n)$. The shallow cuttings in Lemma 11 can be built in $O(n \log^2 n)$ following the method in [11] and using our shallow cutting algorithm in Theorem 8. In this way, we can bound the amortized insertion time by $O(b \log_b n \log^2 n)$, which is $O(\log^{3+\epsilon} n)$ with $b = \log^\epsilon n$. This proves Lemma 7.

5 Algorithm for shallow cuttings

In this section, we prove Theorem 8. We follow the notation in Section 4.1.

As in [13], we use parameter $K = n/r$ instead of r . A k -shallow $(1/r)$ -cutting becomes a k -shallow (K/n) -cutting and we use a (k, K) -shallow cutting to represent it. It has the following properties: (1) $|\Gamma_\Delta| \leq K$ for each cell Δ ; (2) the union of all cells covers $L_{\leq k}(\Gamma)$. Theorem 8 says that a $(k, O(k))$ -shallow cutting of size $O(n/k)$ in the bottom-open pseudo-trapezoid form can be computed in $O(n \log \frac{n}{k})$ time.

Following the analysis of Matoušek [25], we start with the following lemma (see the full paper for the proof), which shows the existence of the shallow cutting in the pseudo-trapezoid form, i.e., each cell is a pseudo-trapezoid but not necessarily bottom-open.

► **Lemma 12.** *For any $k \in [1, n]$, there exists a $(k, O(k))$ -shallow cutting of size $O(n/k)$ in the pseudo-trapezoid form.*

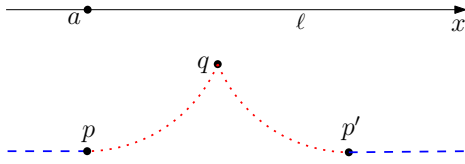
Chan and Tsakalidis [13] introduced a vertex form of the shallow cutting for lines. Here for arcs of Γ , using vertices is not sufficient. We will introduce a vertex-segment form in Section 5.2. The definition requires a concept, which we call *line-separated α -hulls* and is discussed in Section 5.1. In Section 5.3, we present our algorithm to compute shallow cuttings in the vertex-segment form.

5.1 Line-separated α -hulls

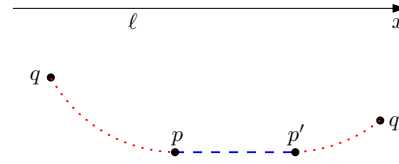
The line-separated α -hull is an extension of the α -hull introduced in [21]. In [21], α -hull is considered for all values $\alpha \in (-\infty, \infty)$. For our problem, we only consider the value $\alpha = -1$.

Let Q be a set of points in \mathbb{R}^- . We define the *line-separated α -hull* $H_\ell(Q)$ of Q with respect to the x -axis ℓ as the complement of the union of all unit disks with centers in \mathbb{R}^+ that do not contain any point of Q (so the disk centers and the points of Q are separated by ℓ , which is why we use “line-separated”; see Fig. 1).

Many of the properties of the α -hulls [21] can be extended to the line-separated case. We list some of these in the following observation; the proof is a straightforward extension of that in [21] by adding the “line-separated” constraint.



■ **Figure 2** Illustration the wings of the a point q . The two (red) dotted curves are wing arcs and the two (blue) dashed segments are wing half-lines. p and p' are the left and right wing vertices, respectively.



■ **Figure 3** Illustration two points q and q' that are in far-away position. The two (red) dotted arcs and the (blue) dashed segments in between constitute $\beta(q, q')$.

► **Observation 13.**

1. $Q \subseteq H_\ell(Q)$, and for any subset $Q' \subseteq Q$, $H_\ell(Q') \subseteq H_\ell(Q)$.
2. A point $q \in Q$ is a vertex of $H_\ell(Q)$ if and only if there exists a unit disk with center in \mathbb{R}^+ and its boundary containing q such that the interior of the disk does not contain any point of Q .
3. If there is an \mathbb{R}^+ -constrained arc connecting two points of Q such that the interior of the underlying disk of the arc does not contain any point of Q , then the arc is an edge of $H_\ell(Q)$.

For any two points $q, q' \in \mathbb{R}^-$ that can be covered by a \mathbb{R}^+ -constrained unit disk, there exists a unique \mathbb{R}^+ -constrained arc that connects q and q' ; we use $\gamma(q, q')$ to denote that arc.

5.1.1 Algorithm for computing $H_\ell(Q)$

By slightly modifying the algorithm of [21], $H_\ell(Q)$ can be computed in $O(m \log m)$ time, where $m = |Q|$. The algorithm also suggests that $\partial H_\ell(Q)$ is x -monotone. In the following, assuming that the points of Q are already sorted from left to right as q_1, q_2, \dots, q_m , we give a linear time algorithm to compute $H_\ell(Q)$, which is similar in spirit to Graham’s scan for computing convex hulls.

Irrelevant points. Note that if a point $q \in Q$ whose y -coordinate is smaller than or equal to -1 , then q must be in $H_\ell(Q)$ because every \mathbb{R}^+ -constrained disk does not contain q in the interior. Hence, in that case q is *irrelevant* for computing $H_\ell(Q)$ and thus can be ignored. If all points of Q are irrelevant, then $H_\ell(Q)$ is simply the region below the horizontal line whose y -coordinate is -1 . In the following, we assume that every point of Q is relevant.

Wings. Consider a point $q \in Q$. Let a be a point on the x -axis ℓ with $x(a) < x(q)$ such that q is on the unit circle C_a centered at a . Let p be the lowest point of C_a . We define the *left wing* of q to be the concatenation of the following two parts (see Fig. 2): (1) the arc of $C_a \cap \mathbb{R}^-$ between q and p , called the *left wing arc*, and the horizontal half-line with p as the right endpoint, called the *left wing half-line*. The point p is called the *left wing vertex* of q . We define the *right wing* of q and the corresponding concepts symmetrically. The left and right wings together actually form the boundary of the line-separated α -hull of $\{q\}$. In Fig. 1, the four (red) dotted arcs are wing arcs and the three (blue) dashed segments are on wing half-lines.

Far-away position. Consider two points $q, q' \in \mathbb{R}^-$ such that $x(q) < x(q')$. We say that (q, q') are in *far-away* position if $x(p) < x(p')$ holds, where p is the right wing vertex of q and p' is the left wing vertex of q' (see Fig. 3). In this case, q' is above the right wing of q

and there is no \mathbb{R}^+ -constrained unit disk covering both q and q' . We use $\beta(q, q')$ to denote the concatenation of the right wing arc of q , the segment $\overline{pp'}$, and the left wing arc of q' . In fact, the left wing of q , $\beta(q, q')$, and the right wing of q' together form the boundary of the line-separated α -hull of $\{q, q'\}$. In Fig. 1, q and q' are also in far-away position.

The algorithm. Define $Q_i = \{q_1, \dots, q_i\}$ for each $1 \leq i \leq m$. Our algorithm handles the points of Q incrementally from q_1 to q_m . For each q_i , the algorithm computes $H_\ell(Q_i)$ by updating $H_\ell(Q_{i-1})$ with q_i . Suppose that $q_{i_1}, q_{i_2}, \dots, q_{i_t}$ are the points of Q_i that are the vertices of $H_\ell(Q_i)$ sorted from left to right. Then, our algorithm maintains the following invariant: the boundary $\partial H_\ell(Q_i)$ is x -monotone and consists of the following parts from left to right: the left wing of q_{i_1} , $\gamma(q_{i_j}, q_{i_{j+1}})$ or $\beta(q_{i_j}, q_{i_{j+1}})$, for each $j = 1, 2, \dots, t-1$ in order, and the right wing of q_{i_t} . $H_\ell(Q_i)$ is the region below $\partial H_\ell(Q_i)$.

Initially, for q_1 , we set $\partial H_\ell(Q_1)$ to the concatenation of the left wing and the right wing of q_1 . In general, suppose we already have $\partial H_\ell(Q_{i-1})$. We compute $\partial H_\ell(Q_i)$ as follows. We process the points of $q_{i_1}, q_{i_2}, \dots, q_{i_t}$ in the backward order. For ease of exposition, we assume that $t > 1$; the special case $t = 1$ can be easily handled.

We first process the point q_{i_t} . If q_{i_t} and q_i are in the far-away position, then we delete the right wing of q_{i_t} from $H_\ell(Q_{i-1})$ and add $\beta(q_{i_t}, q_i)$ and the right wing of q_i . This finishes computing $H_\ell(Q_i)$. Below, we assume that q_{i_t} and q_i are not in the far-away position.

If q_i is below the right wing of q_{i_t} , then q_i is inside $H_\ell(Q_{i-1})$. In this case, $H_\ell(Q_i)$ is $H_\ell(Q_{i-1})$ and we are done. If q_i is above the right wing of q_{i_t} , then we further check whether the arc $\gamma(q_{i_t}, q_i)$ exists (which is true if and only if there exists an \mathbb{R}^+ -constrained unit disk covering both q_{i_t} and q_i).

- If $\gamma(q_{i_t}, q_i)$ does not exist (in this case q_{i_t} must be below the left wing of q_i and thus q_{i_t} does not contribute to $H_\ell(Q_i)$ because it is “dominated” by q_i), then we “prune” q_{i_t} from $\partial H_\ell(Q_{i-1})$, i.e., delete the right wing of q_{i_t} and also delete $\gamma(q_{i_{t-1}}, q_{i_t})$ or $\beta(q_{i_{t-1}}, q_{i_t})$ whichever exists in $H_\ell(Q_{i-1})$. Next, we process $q_{i_{t-1}}$ following the same algorithm.
- If $\gamma(q_{i_t}, q_i)$ exists, then we further check whether D contains $q_{i_{t-1}}$, where D is the underlying disk of $\gamma(q_{i_t}, q_i)$. If $q_{i_{t-1}} \in D$, then q_{i_t} must be in $H_\ell(\{q_{i_{t-1}}, q_i\})$ and thus does not contribute to $H_\ell(Q_i)$. In this case, we prune q_{i_t} as above and continue processing $q_{i_{t-1}}$. If $q_{i_{t-1}} \notin D$, we delete the right wing of q_{i_t} from $\partial H_\ell(Q_{i-1})$ and add the arc $\gamma(q_{i_t}, q_i)$ and the right wing of q_i ; this finishes computing $H_\ell(Q_i)$.

Clearly, the runtime for computing $H_\ell(Q_i)$ is $O(1 + t')$, where t' is the number of points of $q_{i_1}, q_{i_2}, \dots, q_{i_t}$ pruned from $H_\ell(Q_{i-1})$. The overall algorithm for computing $H_\ell(Q)$ takes $O(m)$ time since once a point is pruned it will never appear on the hull again, which resembles Graham’s scan for computing convex hulls.

5.1.2 Vertical decompositions

According to the above discussion, $H_\ell(Q)$ has at most $5m$ vertices with $m = |Q|$, including all wing vertices. The vertical downward rays from all vertices partition $H_\ell(Q)$ into at most $5m$ bottom-open pseudo-trapezoids and rectangles. We call this partition the *vertical decomposition* of $H_\ell(Q)$, denoted by $\text{VD}(Q)$.

In our later discussion, we need to combine Q with a set S of pairwise disjoint segments on ℓ whose endpoints are all in Q . For each segment $s \in S$, we draw a vertical downward ray from each endpoint of s ; let $R(s)$ denote the bottom-open rectangular region bounded by the two rays and s . The regions $R(s)$ for all segments $s \in S$ form the *vertical decomposition* of S , denoted by $\text{VD}(S)$.

We combine $\text{VD}(Q)$ and $\text{VD}(S)$ to form a vertical decomposition of (Q, S) , denoted by $\text{VD}(Q \cup S)$ as follows. Let U be the upper envelope of $H_\ell(Q)$ and S . We draw a vertical downward ray from each vertex of U . These rays divide the region below U into cells, each of which is bounded by two vertical rays from left and right, and bounded from above by a line segment or an \mathbb{R}^+ -constrained arc. These cells together form the decomposition $\text{VD}(Q \cup S)$. In the following, depending on the context, $\text{VD}(Q)$ may refer to the region covered by all cells of it; the same applies to $\text{VD}(S)$ and $\text{VD}(Q \cup S)$. As such, we have $\text{VD}(Q \cup S) = \text{VD}(Q) \cup \text{VD}(S)$. Note that since the endpoints of all segments of S are in Q and on ℓ , the boundary $\partial\text{VD}(Q \cup S)$ is x -monotone.

5.2 Shallow cuttings in the vertex-segment form

We introduce a vertex-segment form of the shallow cutting. Given parameters $k, K \in [1, n]$ with $k \leq K$, a (k, K) -shallow cutting for the arcs of Γ in the *vertex-segment* form is a set Q of points in \mathbb{R}^- along with a set S of interior pairwise-disjoint segments on ℓ such that the following conditions hold:

1. The endpoints of all segments of S are in Q .
2. Every point of Q has level at most K in Γ .
3. Every segment of S intersects at most K arcs of Γ .
4. $\text{VD}(Q \cup S)$ covers $L_{\leq k}(\Gamma)$.

The *conflict list* of a point $q \in Q$, denoted by Γ_q , is the set of arcs of Γ below q . Note that $|\Gamma_q| \leq K$ as the level of q is at most K . The *conflict list* of a segment $s \in S$, denoted by Γ_s , is the set of arcs intersecting s . The conflict lists of (Q, S) refer to the conflict lists of all points of Q and all segments of S . The *size* of the cutting is defined to be $|Q|$. Observe that since the endpoints of all segments of S are in Q and the segments of S are interior pairwise-disjoint, we have $|S| < |Q|$. Therefore, $\text{VD}(Q \cup S)$ has $O(|Q|)$ cells, and more specifically, at most $5|Q|$ cells. Further, we have following observation.

► **Observation 14.** *Suppose that (Q, S) is a (k, K) -shallow cutting for Γ in the vertex-segment form. Then every cell of $\text{VD}(Q \cup S)$ intersects at most $3K$ arcs of Γ .*

Proof. Consider a cell Δ of $\text{VD}(Q \cup S)$. Let e be the top edge of Δ . By the definition of $\text{VD}(Q \cup S)$, e is one of the following: a segment of S , an arc $\gamma(q, q')$ for two points $q, q' \in Q$, a wing arc of a point $q \in Q$, and a segment of a wing half-line of a point $q \in Q$. Below, we argue $|\Gamma_\Delta| \leq 3K$ for each of these cases, where Γ_Δ is the set of arcs of Γ intersecting Δ .

1. If e is a segment of S , then recall that $|\Gamma_e| \leq K$. Also, the size of the conflict list of each endpoint of e is at most K . For any arc $\gamma \in \Gamma$ intersecting Δ , since the center of γ is in \mathbb{R}^+ , γ must either intersect e or in the conflict list of at least one endpoint of e . Therefore, $|\Gamma_\Delta| \leq 3K$ holds.
2. If e is an arc $\gamma(q, q')$ for two points $q, q' \in Q$, then any arc $\gamma \in \Gamma$ intersecting Δ must be in the conflict list of one of q and q' . Hence, we have $|\Gamma_\Delta| \leq 2K$.
3. If e is a wing arc γ of a point $q \in Q$, then q is an endpoint of γ . Let p be the other endpoint of γ . By definition, the y -coordinate of p is -1 . Thus, no arc of Γ is below p . By definition, the radius of γ is 1 and the center of γ is on ℓ . Hence, any arc of Γ intersecting Δ must be in the conflict list of q and thus $|\Gamma_\Delta| \leq |\Gamma_q| \leq K$.
4. If e is a segment s of a wing half-line of a point $q \in Q$, then by definition e is horizontal and has y -coordinate equal to -1 . As centers of all arcs of Γ are in \mathbb{R}^+ , no arc of Γ can intersect Δ . Hence, $|\Gamma_\Delta| = 0$.

Combining all the above cases leads to $|\Gamma_\Delta| \leq 3K$. ◀

In the next two lemmas, we show that shallow cuttings in the vertex-segment form and in the pseudo-trapezoid form can be transformed to each other.

► **Lemma 15.** *A (k, K) -shallow cutting of size t in the pseudo-trapezoid form can be transformed into a $(k, k + K)$ -shallow cutting in the vertex-segment form of size $O(t)$.*

Proof. Let Ξ be a (k, K) -shallow cutting of size t in the pseudo-trapezoid form. Without loss of generality, we assume that all cells of Ξ intersect $L_{\leq k}(\Gamma)$. Define Q to be the set of vertices of all cells of Ξ . Define S to be the top edges of all cells of Ξ that are segments of ℓ . Since the interiors of cells of Ξ are pairwise disjoint, the segments of S are also interior pairwise-disjoint. As Ξ has t cells, we have $|Q| = O(t)$. In the following, we argue that (Q, S) is a $(k, k + K)$ -shallow cutting in the vertex-segment form.

First of all, by definition, endpoints of all segments of S are in Q . Consider a point $q \in Q$, which is a vertex of a cell $\Delta \in \Xi$. As Δ intersects $L_{\leq k}(\Gamma)$ and $|\Gamma_\Delta| \leq K$, there are at most $k + K$ arcs of Γ below q . Hence, q has level at most $k + K$ in Γ . For each segment $s \in S$, since it is a top edge of a cell $\Delta \in \Xi$ and $|\Gamma_\Delta| \leq K$, we obtain $|\Gamma_s| \leq K$.

It remains to argue that $\text{VD}(Q \cup S)$ covers $L_{\leq k}(\Gamma)$. By definition, the union of all cells of Ξ covers $L_{\leq k}(\Gamma)$. Consider a cell $\Delta \in \Xi$, which is a pseudo-trapezoid. We show that $\Delta \subseteq \text{VD}(Q \cup S)$, which will prove that $\text{VD}(Q \cup S)$ covers $L_{\leq k}(\Gamma)$.

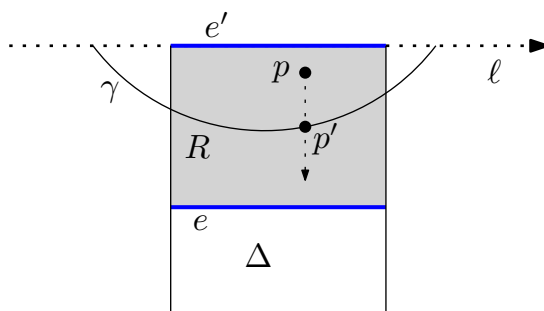
Let e be the top edge of Δ . As Δ is a pseudo-trapezoid, e is either a segment on ℓ or an \mathbb{R}^+ -constrained arc. If e is a segment of ℓ , then $e \in S$ and thus Δ is a cell of $\text{VD}(S)$. Hence, $\Delta \subseteq \text{VD}(S) \subseteq \text{VD}(Q \cup S)$. If e is an \mathbb{R}^+ -constrained arc, then let q_1 and q_2 be its two endpoints; thus e is the arc $\gamma(q_1, q_2)$. Since Δ is a pseudo-trapezoid with $\gamma(q_1, q_2)$ as the top edge, Δ must be contained in the line-separated α -hull $H_\ell(\{q_1, q_2\})$, which is a subset of $H_\ell(Q)$ by Observation 13(1) as $q_1, q_2 \in Q$. Recall that $\text{VD}(Q)$ is the vertical decomposition of $H_\ell(Q)$. Hence, we have $\Delta \subseteq \text{VD}(Q) \subseteq \text{VD}(Q \cup S)$.

This proves $\Delta \subseteq \text{VD}(Q \cup S)$ and therefore $\text{VD}(Q \cup S)$ covers $L_{\leq k}(\Gamma)$. ◀

► **Lemma 16.** *A (k, K) -shallow cutting of size t in the vertex-segment form can be transformed into a $(k, 3K)$ -shallow cutting of size $O(t)$ in the bottom-open pseudo-trapezoid form.*

Proof. Let (Q, S) be a (k, K) -shallow cutting of size t in the vertex-segment form. We intend to take the vertical decomposition $\text{VD}(Q, S)$ as the $(k, 3K)$ -shallow cutting Ξ of size $O(t)$ in the bottom-open pseudo-trapezoid form. However, a subtle issue is that some cells of $\text{VD}(Q, S)$ might not be pseudo-trapezoids. More specifically, consider a cell $\Delta \in \text{VD}(Q, S)$. Let e be the top edge of Δ . According to the definition of $\text{VD}(Q, S)$, e belongs to one of the three cases: (1) e is an \mathbb{R}^+ -constrained arc; (2) e is a segment of ℓ ; (3) e is a segment of a wing half-line of a point of Q . In the first two cases, Δ is a bottom-open pseudo-trapezoid and we include Δ in Ξ . In the third case, Δ is not a pseudo-trapezoid by our definition since e is a line segment but not on ℓ . In this case, we extend Δ by moving e upwards until ℓ to obtain an extended cell Δ' , which is a bottom-open pseudo-trapezoid; we add Δ' to Ξ . We call Δ' a *special cell* of Ξ .

We claim that $\Gamma_{\Delta'} = \emptyset$, i.e., Δ' does not intersect any arcs of Γ . Indeed, let e' be the top edge of Δ' . Let R be the rectangular region of Δ' between e and e' (see Fig. 4). Then, $\Delta' = \Delta \cup R$. Consider any point $p \in R$. We argue that no arc of Γ is below p , which will prove the claim. Assume to the contradiction that there is an arc $\gamma \in \Gamma$ below p . Without loss of generality, we assume that γ is the lowest arc of Γ intersecting the vertical downward ray $\rho(p)$. Let p' be the intersection of γ and $\rho(p)$. By definition, $p' \in L_{\leq 0}(\Gamma)$. Since (Q, S) is a (k, K) -shallow cutting, $\text{VD}(Q, S)$ covers $L_{\leq k}(\Gamma)$ and thus covers $L_{\leq 0}(\Gamma)$ as $k \geq 0$. Therefore, $\text{VD}(Q, S)$ covers p' . On the other hand, since e is on a wing half-line of a point of Q , the



■ **Figure 4** Illustrating the proof of $\Gamma_{\Delta'} = \emptyset$, with $\Delta' = R \cup \Delta$, where R is the gray rectangle and Δ is the region below e .

y -coordinate of e is -1 and thus no arcs of Γ intersect e . Hence, p' must be above e . But since e is the top edge of the cell $\Delta \in \text{VD}(Q, S)$, p' cannot be covered by $\text{VD}(Q, S)$, a contradiction. This proves that $\Gamma_{\Delta'} = \emptyset$.

Since $|Q| = t$, $\text{VD}(Q \cup S)$ has $O(t)$ cells. By definition, the size of Ξ is $O(t)$. We next show that Ξ is a $(k, 3K)$ -shallow cutting in the bottom-open pseudo-trapezoid form. First of all, by definition, each cell of $\text{VD}(Q, S)$ is a bottom-open pseudo-trapezoid. Also, since $\text{VD}(Q, S)$ covers $L_{\leq k}(\Gamma)$ and each cell of $\text{VD}(Q, S)$ is either in Ξ or contained in a cell of Ξ , $\text{VD}(Q, S)$ is a subset of Ξ . Therefore, Ξ covers $L_{\leq k}(\Gamma)$. For each Δ of Ξ , if it is a special cell, then $|\Gamma_{\Delta}| = 0$ as proved above; otherwise Δ is also a cell in $\text{VD}(Q, S)$ and we have $|\Gamma_{\Delta}| \leq 3K$ by Observation 14. Therefore, Ξ is a $(k, 3K)$ -shallow cutting in the bottom-open pseudo-trapezoid form. ◀

Combining Lemmas 12 and 15 leads to the following.

► **Corollary 17.** *Given $k \in [1, n]$, there exists a $(k, O(k))$ -shallow cutting of size $O(n/k)$ in the vertex-segment form.*

5.3 Computing shallow cuttings in the vertex-segment form

In what follows, by extending the algorithm of [13], we present an algorithm to compute shallow cuttings for Γ in the vertex-segment form.

We say that a (standard) cutting is in the *pseudo-trapezoid form* if every cell of the cutting is a pseudo-trapezoid. The following result was known previously [15, 28].

► **Lemma 18** ([15, 28]). *Given any constant $\epsilon > 0$, an ϵ -cutting for Γ in the pseudo-trapezoid form of $O(1)$ size covering the plane, along with its conflict lists, can be computed in $O(n)$ time.*

We say that a shallow cutting (Q, S) in the vertex-segment form is *sorted* if points of Q are sorted by their x -coordinates. The following is the main theorem about our algorithm.

► **Theorem 19.** *There exist constants B, C, C' , such that for any parameter $k \in [1, n]$, given a (Bk, CBk) -shallow cutting (Q_{IN}, S_{IN}) in the sorted vertex-segment form for Γ of size at most $C' \frac{n}{Bk}$ along with its conflict lists, we can compute a (k, Ck) -shallow cutting (Q_{OUT}, S_{OUT}) in the sorted vertex-segment form for Γ of size at most $C' \frac{n}{k}$ along with its conflict lists in $O(n)$ time.*

Proof. Let ϵ be a constant to be set later. We begin by computing the decomposition $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$. Since Q_{IN} is sorted, $H_\ell(Q_{\text{IN}})$ can be computed in $O(|Q_{\text{IN}}|)$ time using the algorithm from Section 5.1. As the endpoints of all segments of S_{IN} are in Q_{IN} and segments of S_{IN} are interior pairwise-disjoint on ℓ , the segments of S_{IN} can also be sorted from left to right in $O(|Q_{\text{IN}}|)$ time. As such, computing $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ can be done in $O(|Q_{\text{IN}}|)$ time, which is $O(n/k)$ as $|Q_{\text{IN}}| \leq C' \frac{n}{Bk}$.

Next, for each cell $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$, we perform the following two steps.

1. Compute an ϵ -cutting Ξ_Δ of size $O(1)$ for Γ_Δ . We clip the cells of Ξ_Δ to lie within Δ (and redivide each new cell into pseudo-trapezoids if needed). Let Q_Δ denote the set of vertices of all cells of Ξ_Δ and S_Δ the set of top edges of the cells of Ξ_Δ that are segments of ℓ .

Since $\epsilon = O(1)$, Ξ_Δ has $O(1)$ cells and computing Ξ_Δ takes $O(|\Gamma_\Delta|)$ time by Lemma 18. Hence, both $|Q_\Delta|$ and $|S_\Delta|$ are $O(1)$. As $\sum_{\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})} |\Gamma_\Delta| = O(n)$, the total time of this step for all cells $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ is $O(n)$.

2. Compute by brute force a smallest subset $Q'_\Delta \subseteq Q_\Delta$, along with a subset $S'_\Delta \subseteq S_\Delta$, such that the following conditions are satisfied.
 - a. The endpoints of all segments of S'_Δ are in Q'_Δ .
 - b. Every vertex in Q'_Δ has level in Γ_Δ at most Ck .
 - c. Every segment in S'_Δ intersects at most Ck arcs of Γ_Δ .
 - d. For each cell $\sigma \in \Xi_\Delta$ whose vertices are all in $L_{\leq 2k}(\Gamma_\Delta)$, σ is covered by $\text{VD}(Q'_\Delta, S'_\Delta)$.

As both $|Q_\Delta|$ and $|S_\Delta|$ are $O(1)$, there are $O(1)$ different pairs of Q'_Δ and S'_Δ . For each such pair (Q'_Δ, S'_Δ) , we can check whether the four conditions are satisfied in $O(|\Gamma_\Delta|)$ time, because $|Q'_\Delta|$, $|S'_\Delta|$, and the size of Ξ_Δ are all $O(1)$. Hence, finding a smallest subset Q'_Δ with S'_Δ takes $O(|\Gamma_\Delta|)$ time. After that, for each point $q \in Q'_\Delta$, its conflict list in Γ_Δ , which is also its conflict list in Γ , can be found in $O(|\Gamma_\Delta|)$ time. Similarly, for each segment of S'_Δ , its conflict list can be found in $O(|\Gamma_\Delta|)$ time. As both $|Q'_\Delta|$ and $|S'_\Delta|$ are $O(1)$, finding the conflict lists of (Q'_Δ, S'_Δ) takes $O(|\Gamma_\Delta|)$ time. Since $\sum_{\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})} |\Gamma_\Delta| = O(n)$, the runtime of this step for all $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ is $O(n)$.

We define $Q_{\text{OUT}} = \bigcup_{\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})} Q'_\Delta$ and $S_{\text{OUT}} = \bigcup_{\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})} S'_\Delta$. We can sort the points of Q_{OUT} in $O(n/k)$ time as follows. For each $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$, we sort the points of Q'_Δ , which takes $O(1)$ time as $|Q'_\Delta| = O(1)$. Then, for all cells $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ in order from left to right, we concatenate the sorted lists of Q'_Δ , and the resulting list is a sorted list of Q_{OUT} . This takes $O(n/k)$ time in total as $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ has $O(n/k)$ cells.

The total runtime of the above algorithm is $O(n)$.

Correctness. In the following, we argue the correctness, i.e., prove that $(Q_{\text{OUT}}, S_{\text{OUT}})$ is a (k, Ck) -shallow cutting for Γ of size at most $C' \frac{n}{k}$. We first show that $(Q_{\text{OUT}}, S_{\text{OUT}})$ is a (k, Ck) -shallow cutting and then bound its size.

Consider a cell $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$. For each point $q \in Q'_\Delta$, according to our algorithm, q has level in Γ_Δ at most Ck . In light of the definition of $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$, Δ is a bottom-open cell bounded by two vertical rays. Hence, the level of q in Γ_Δ is also its level in Γ . Therefore, q has level in Γ at most Ck .

For each segment $s \in S'_\Delta$, by definition, s is a top edge of a cell $\Delta \in \text{VD}(Q_{\text{IN}}, S_{\text{IN}})$. According to our algorithm, s intersects at most Ck arcs of Γ_Δ . Since $s \subseteq \Delta$, any arc of Γ intersecting s must intersect Δ and thus is in Γ_Δ . Therefore, s intersects at most Ck arcs of Γ . In addition, according to our algorithm, the endpoints of s are in Q'_Δ .

To show that $(Q_{\text{OUT}}, S_{\text{OUT}})$ is a (k, Ck) -shallow cutting, it remains to prove that $\text{VD}(Q_{\text{OUT}}, S_{\text{OUT}})$ covers $L_{\leq k}(\Gamma)$. Consider a point $p \in L_{\leq k}(\Gamma)$. By definition, $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ covers $L_{\leq Bk}(\Gamma)$. By setting $B > 1$, $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$ covers $L_{\leq k}(\Gamma)$ and thus p must be in a cell Δ of $\text{VD}(Q_{\text{IN}}, S_{\text{IN}})$. In the following, we argue that p is covered by $\text{VD}(Q'_\Delta \cup S'_\Delta)$, which will prove that $\text{VD}(Q_{\text{OUT}}, S_{\text{OUT}})$ covers $L_{\leq k}(\Gamma)$.

Let σ be the cell of the cutting Ξ_Δ that contains p . Since p has level in Γ_Δ at most k and the number of arcs of Γ_Δ intersecting σ is at most $\epsilon \cdot |\Gamma_\Delta|$, every vertex of σ has level at most $k + \epsilon \cdot |\Gamma_\Delta|$. Because the size of the conflict list of each point of Q_{IN} is at most CBk , $|\Gamma_\Delta| \leq 3CBk$ by Observation 14. Therefore, $k + \epsilon \cdot |\Gamma_\Delta| \leq 2k$ by setting the constant $\epsilon = 1/(3CB)$. Hence, all vertices of σ are in $L_{\leq 2k}(\Gamma_\Delta)$ and thus σ is covered by $\text{VD}(Q'_\Delta \cup S'_\Delta)$ according to our algorithm. As $p \in \sigma$, we obtain that p is covered by $\text{VD}(Q'_\Delta \cup S'_\Delta)$.

Bounding the size of $\text{VD}(Q_{\text{OUT}}, S_{\text{OUT}})$, i.e., $|Q_{\text{OUT}}|$. We now prove $|Q_{\text{OUT}}| \leq C'n/k$. To this end, we compare it against a $(5k, 5c_0k)$ -shallow cutting (Q^*, S^*) of size at most $c'_0 n/(5k)$ for some constant c'_0 , whose existence is guaranteed by Corollary 17. The details can be found in the full paper. This proves the theorem. \blacktriangleleft

► **Corollary 20.** *There exist constants B , C , and C' , such that for any parameter $k \in [1, n]$, we can compute a $(B^i k, CB^i k)$ -shallow cutting in the sorted vertex-segment form of size at most $C' \frac{n}{B^i k}$, along with its conflict lists, for all $i = 0, 1, \dots, \log_B \frac{n}{k}$ in $O(n \log \frac{n}{k})$ total time. In particular, we can compute a (k, Ck) -shallow cutting of size $O(n/k)$ in the sorted vertex-segment form, along with its conflict lists, in $O(n \log \frac{n}{k})$ time.*

Proof. By Theorem 19, the runtime $T(n, k)$ satisfies the recurrence $T(n, k) = T(n, Bk) + O(n)$ with the trivial base case $T(n, n) = O(n)$. The recurrence solves to $T(n, k) = O(n \log \frac{n}{k})$. \blacktriangleleft

Proving Theorem 8. We first apply Corollary 20 to compute the shallow cuttings in the sorted vertex-segment form. Then, we transform them to shallow cuttings in the bottom-open pseudo-trapezoid form by Lemma 16, which can be done in additional $O(n \log \frac{n}{k})$ time (i.e., linear time for each cutting). This proves Theorem 8.

6 Dynamic unit-disk range emptiness queries

Our techniques may be extended to solve other related problems about unit disks. In the following, we demonstrate one exemplary problem: the dynamic unit-disk range emptiness queries (see the full paper for a simple solution to the static problem using our techniques).

For a set P of points in the plane, we wish to maintain P in a dynamic data structure for points insertions and deletions to answer *unit-disk range emptiness queries*: Given a unit disk D , determine whether D contains a point of P , and if so, return such a point. One can solve the problem by using a dynamic nearest neighbor search data structure (i.e., given a query disk D , using a nearest neighbor query we find a point $p \in P$ nearest to the center of D ; D contains a point of P if and only if $p \in D$). The current best dynamic nearest neighbor search data structure is given by Chan [11]; with that, we can obtain a data structure of $O(n)$ space in $O(n \log n)$ time that supports $O(\log^2 n)$ amortized insertion time, $O(\log^4 n)$ amortized deletion time, and $O(\log^2 n)$ time for unit-disk range emptiness queries. In the following, using our techniques, we propose a better result.

As in Section 3 for the dynamic reporting problem, we can use Lemma 3 to reduce the problem to the following line-separated problem.

► **Problem 2 (Dynamic line-separable unit-disk range emptiness queries).** *Given a set Q of m points above a horizontal line ℓ , build a data structure to maintain Q to support the following operations. (1) Insertion: insert a point to Q ; (2) deletion: delete a point from Q ; (3) unit-disk range emptiness query: given a unit disk D whose center is below ℓ , determine whether D contains a point of Q , and if so, return such a point.*

To solve the line-separable problem, we define the set Γ of arcs using Q in the same way as before. Let D_q be a unit disk with center q below ℓ . Note that $D_q \cap Q \neq \emptyset$ if and only if q is above the lower envelope of Γ . Further, q is above the lower envelope of Γ if and only if the lowest arc of Γ intersecting ℓ_q is below q , where ℓ_q is the vertical line through q . Therefore, our problem reduces to the following *vertical line queries* subject to arcs insertions and deletions for Γ : Given a vertical line ℓ^* , find the lowest arc of Γ that intersects ℓ^* .

To solve the dynamic vertical line query problem among arcs of Γ , we apply Chan's framework [8] for the dynamic vertical line query problem among lines (in the dual plane, a vertical line query is dual to: Finding an extreme point on the convex hull of all dual points along a query direction). To this end, we need the following two components: (1) a dynamic data structure of $O(m)$ space with $O(\log m)$ query time and $m^{O(1)}$ update time; (2) a deletion-only data structure of $O(m)$ space that can be built in $O(m \log m)$ time, supporting $O(\log m)$ query time and $O(\log m)$ amortized deletion time. For (1), we can use our static data structure as discussed above, i.e., whenever there is an update, we simply rebuild the data structure. For (2), Wang and Zhao [30] already provided such a data structure. Using these two components, we can apply exactly the same framework of Chan [8]. Indeed, the framework still works for the arcs of Γ because every arc is x -monotone. With the framework and the above two components, we can obtain a data structure of $O(m)$ space that allows insertions and deletions of arcs of Γ in $O(\log^{1+\epsilon} m)$ amortized update time and answers a vertical line query in $O(\log m)$ time, where m is the size of the current set Γ . Consequently, we can solve Problem 2 with the same time complexities. Finally, with Lemma 3 and our problem reduction, we can have a data structure of $O(n)$ space that allows insertions and deletions of points of P in $O(\log^{1+\epsilon} n)$ amortized time and answers a unit-disk range emptiness query in $O(\log n)$ time, where n is the size of the current set P .

References

- 1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009. doi:10.1137/1.9781611973068.21.
- 2 Pankaj K. Agarwal. Range searching, in *Handbook of Discrete and Computational Geometry*, C.D. Tóth, J. O'Rourke, and J.E. Goodman (eds.), pages 1057–1092. CRC Press, 3rd edition, 2017.
- 3 Pankaj K. Agarwal. Simplex range searching and its variants: a review. In *A Journey Through Discrete Mathematics*, pages 1–30. Springer, 2017. doi:10.1007/978-3-319-44479-6_1.
- 4 Jon L. Bentley and Hermann A. Maurer. A note on Euclidean near neighbor searching in the plane. *Information Processing Letters*, 8:133–136, 1979.
- 5 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry — Algorithms and Applications*. Springer-Verlag, Berlin, 3rd edition, 2008.
- 6 Sarita de Berg and Frank Staals. Dynamic data structures for k -nearest neighbor queries. *Computational Geometry: Theory and Applications*, 111(101976), 2023. doi:10.1016/j.comgeo.2022.101976.
- 7 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$ -levels in three dimensions. *SIAM Journal on Computing*, 20:561–575, 2000. doi:10.1137/S0097539798349188.
- 8 Timothy M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *Journal of the ACM*, 48:1–12, 2001. doi:10.1145/363647.363652.
- 9 Timothy M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *Journal of the ACM*, 57:16:1–16:15, 2010. doi:10.1145/1706591.1706596.
- 10 Timothy M. Chan. Three problems about dynamic convex hulls. *International Journal of Computational Geometry and Applications*, 22:341–364, 2012. doi:10.1142/S0218195912600096.
- 11 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discrete and Computational Geometry*, 64:1235–1252, 2020. doi:10.1007/s00454-020-00229-5.

- 12 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.
- 13 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete and Computational Geometry*, 56:866–881, 2016. doi:10.1007/s00454-016-9784-4.
- 14 Bernard Chazelle. An improved algorithm for the fixed-radius neighbor problem. *Information Processing Letters*, 16:193–198, 1983. doi:10.1016/0020-0190(83)90123-0.
- 15 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
- 16 Bernard Chazelle, Richard Cole, Franco P. Preparata, and Chee-Keng Yap. New upper bounds for neighbor searching. *Information and Control*, 68:105–124, 1986. doi:10.1016/S0019-9958(86)80030-4.
- 17 Bernard Chazelle and Herbert Edelsbrunner. Optimal solutions for a class of point retrieval problems. *Journal of Symbolic Computation*, 1:47–56, 1985. doi:10.1016/S0747-7171(85)80028-6.
- 18 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986. doi:10.1007/BF01840440.
- 19 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1:163–191, 1986. doi:10.1007/BF01840441.
- 20 Bernard Chazelle, Leonidas J. Guibas, and D.T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985. doi:10.1007/BF01934990.
- 21 Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29:551–559, 1983. doi:10.1109/TIT.1983.1056714.
- 22 G.N. Frederickson. An optimal algorithm for selection in a min-heap. *Information and Computation*, 104:197–214, 1993. doi:10.1006/inco.1993.1030.
- 23 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete and Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.
- 24 Jiří Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992. doi:10.1007/BF02293051.
- 25 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.
- 26 Jiří Matoušek. Geometric range searching. *ACM Computing Survey*, 26:421–461, 1994. doi:10.1145/197405.197408.
- 27 Edgar A. Ramos. On range reporting, ray shooting and k -level construction. In *Proceedings of the 15th Annual Symposium on Computational Geometry (SoCG)*, pages 390–399, 1999. doi:10.1145/304893.304993.
- 28 Haitao Wang. Unit-disk range searching and applications. *Journal of Computational Geometry*, 14:343–394, 2023. doi:10.20382/jocg.v14i1a13.
- 29 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete and Computational Geometry*, 64:1141–1166, 2020. doi:10.1007/s00454-020-00219-7.
- 30 Haitao Wang and Yiming Zhao. Computing the minimum bottleneck moving spanning tree. In *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 82:1–82:15, 2022. doi:10.4230/LIPIcs.MFCS.2022.82.
- 31 Haitao Wang and Yiming Zhao. An optimal algorithm for L_1 shortest paths in unit-disk graphs. *Computational Geometry: Theory and Applications*, 110:101960: 1–9, 2023. doi:10.1016/j.comgeo.2022.101960.
- 32 Haitao Wang and Yiming Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14:14–47, 2023. doi:10.20382/jocg.v14i1a2.

